Nonconvex Piecewise Linear Knapsack Problems

S. Kameshwaran and Y. Narahari
Electronic Commerce Laboratory
Department of Computer Science and Automation
Indian Institute of Science
Bangalore - India
{ kameshn,hari }@csa.iisc.ernet.in

Abstract

This paper considers the minimization version of a class of nonconvex knapsack problems with piecewise linear cost structure. The items to be included in the knapsack have a divisible quantity and a cost function. An item can be included partially in a given quantity range and the cost is a nonconvex piecewise linear function of quantity. Given a demand, the optimization problem is to choose an optimal quantity for each item such that the demand is satisfied and the total cost is minimized. This problem is encountered in many situations in manufacturing, logistics, and supply chain design. In particular, we are motivated by winner determination in *volume discount* procurement auctions in electronic commerce. The piecewise linearity of the cost function gives the problem two different knapsack structures related to precedence constrained and multiple choice knapsack problems. Two mixed integer linear programming formulations are proposed based on the above structures. Motivated by the unique nature of the varying demands on the problem in different scenarios, the following algorithms are developed: (1) a fast polynomial time, near optimal heuristic using convex envelopes; (2) two exact, pseudo polynomial time dynamic programming algorithms; (3) a polynomial time 2-approximation algorithm; and (4) a fully polynomial time approximation scheme. A test suite was developed to generate representative problem instances with different characteristics. Extensive computational experiments conducted on these problem instances show that the proposed formulation and algorithms are faster than the existing techniques.

Keywords

Combinatorial optimization, piecewise linear knapsack problems, precedence constrained knapsack model, multiple choice knapsack model, LP relaxation, dynamic programming, convex envelopes, approximation algorithms, fully polynomial time approximation schemes.

All correspondence is to be addressed to: Y. Narahari, Department of Computer Science and Automation, Indian Institute of Science, Bangalore - 560 012, India. Email: hari@csa.iisc.ernet.in.

Acronyms

LP	Linear Programming
DP	Dynamic Programming
IP	Integer Program
MILP	Mixed Integer Linear Program
NKP	Nonlinear Knapsack Problem
PLKP	Piecewise Linear Knapsack Problem
IM	Incremental Model
MCM	Multiple Choice Model
CCM	Convex Combination Model
PCKM	Precedence Constrained Knapsack Model
LP:PCKM	LP relaxation of PCKM
CE:PCKM	PCKM with Convex Envelope of original cost function
MCKM	Multiple Choice Knapsack Model
MKP	Multiple Choice Knapsack Problem
FPTAS	Fully Polynomial Time Approximation Scheme

1 Introduction

Knapsack problems are among the the most intensively studied combinatorial optimization problems [31, 28, 22]. The knapsack problem considers a set of items, each having an associated cost and a weight. The problem (minimization version) is then to choose a subset of the given items such that the corresponding total cost is minimized while the total weight satisfies the specified demand. Well studied variants of the knapsack problems [31, 28] include: *bounded/unbounded knapsack problems*, *subset-sum problems*, *change-making problems*, *multiple knapsack problems*, *multiple choice knapsack problems*, and *bin packing problems*. The applications of these problems vary from industrial applications and financial management to personal health-care.

1.1 Nonlinear Knapsack Problems

A generic *nonlinear knapsack problem* (also called *nonlinear resource allocation problem*) has a demand B and a set J of N items. Each item $j \in J$ has a nonlinear cost function $Q_j(b)$ defined over the quantity $b \in [\underline{a}_j, \overline{a}_j]$. The optimization problem is to choose quantity $q_j \in \{0, [\underline{a}_j, \overline{a}_j]\}$ for each item j to meet the demand B ($\sum_{j \in J} q_j \geq B$) such that the total cost $\sum_{j \in J} Q_j(q_j)$ of accumulated demand is minimized. The problem can be formulated as the following nonlinear integer programming problem:

$$\text{(NKP)}: \qquad \min \sum_{j \in J} Q_j(q_j) \tag{1}$$

subject to

$$\sum_{j \in J} q_j \geq B$$

$$\underline{a}_j x_j \leq q_j \leq \overline{a}_j x_j \qquad \forall j \in J$$

$$x_j \in \{0,1\},\ q_j \geq 0, \text{ integer} \quad \forall j \in J$$

The decision variable q_j can take either zero value or an integer value in the range $[\underline{a}_j, \overline{a}_j]$. This is implemented as linear inequalities using binary decision variables x_j. There are several variations to the above problem: the objective function is an arbitrary function $Q(q_1, \ldots, q_{|J|})$; each item has a weight w_j and the demand constraint is $\sum_{j \in J} w_j q_j \geq B$; a generic demand constraint $\sum_{j \in J} g_j(q_j) \geq B$; and the variables q_j are continuous. The problem considered in this paper is as in NKP above, with Q_j as a nonconvex piecewise linear cost function.

The NKP and its variants are encountered either directly, or as a subproblem, in a variety of applications, including production planning, financial modeling, stratified sampling, and in capacity planning in manufacturing, health-care, and computer networks [5]. Different available algorithms for the problem were developed with varying assumptions on the cost function Q_j and the demand constraint. The book by Ibaraki and Katoh [17] provides an excellent collection of algorithms for demand constraint $\sum_{j \in J} q_j = B$. The paper by Bretthauer and Shetty [5] reviews the algorithms and applications for NKP with generic demand constraint $\sum_{j \in J} g_j(q_j) \geq B$.

The objective functions considered in most of the literature for convex knapsack problems are *separable* like in NKP. If the decision variables are continuous, the differentiability and optimality property of the convex functions are used in designing algorithms [34, 2]. For problems with more general separable nonlinear convex constraint of the form $\sum_{j \in J} g_j(q_j) \geq B$, multiplier search methods [3] and variable pegging methods [23, 4] are the general techniques. The algorithms for problems with discrete variables are based on Lagrangian and linear relaxations. For convex demand constraints, the solution methodologies include branch-and-bound [3], linearization, and dynamic programming [29, 14].

Unlike convex knapsack problems, very little work has been done on nonconvex knapsack problems. Dynamic programming (DP) is the predominant solution technique for this problem. A pseudo-polynomial time DP algorithm was developed in [17]. Approximation algorithms based on dynamic programming were proposed in [26] (the problem was called as capacitated plant allocation problem). Concave cost functions were considered in [30], but the solution technique used local minimizers for obtaining a local optimal solution. Use of branch-and-bound algorithm was proposed as a promising technique in [5].

1.2 Contributions and Outline

In this paper, we consider a nonlinear knapsack problem called as *nonconvex piecewise linear knapsack problem* (PLKP). As the name suggests, the cost function Q_j is a nonconvex piecewise linear cost function. The demand constraints are the same as in NKP. The problem is motivated due to its application in the *winner determination* of volume discount procurement auctions in electronic commerce. The cost function and the associated knapsack problem are introduced in Section 2. We study the problem from different perspectives owing to its diverse demands in varying scenarios.

- First as an optimization problem, Section 3 investigates various mathematical programming formulations. Two new formulations are proposed based on the precedence constraint and multiple choice structure of the problem. Computational experiments that compare the solution time of the proposed formulations with that of the existing textbook formulations, when solved with a commercial optimization solver, are presented. This is useful for practitioners to choose the formulation that best suits their needs.

- For applications that demand a *good* solution in relatively short time, a fast linear programming based heuristic is developed in Section 4. The approach uses the convex envelopes of the cost function to solve the relaxation. Computational experiments were performed on problem instances with different features to study the trade-off in savings in computational time with the optimality gap of heuristic solution.

- Two exact algorithms based on dynamic programming are developed in Section 5. The purpose of these algorithms is two fold: one of the algorithms is useful in solving PLKPs which appear as subproblems in multi-attribute procurement and the other algorithm is used to develop a fully polynomial time approximation scheme.

- The PLKP is $\mathcal{NP}$-hard and it is of theoretical interest to investigate the possibilities of approximation. Section 6 presents a 2-approximation algorithm and a fully polynomial time approximation scheme. Section 7 concludes the paper.

2 The Nonconvex Piecewise Linear Knapsack Problem

The nonconvex piecewise linear knapsack problem (PLKP) considered in this paper is the same as NKP above, with the cost function Q_j as a piecewise linear function. The piecewise linear cost function Q_j defined over the quantity range $[\underline{a}_j, \overline{a}_j]$ is shown in Figure 1. Table 1 provides the notation.

The cost function Q_j can be represented by tuples of break points, slopes, and costs at break points: $Q_j \equiv ((\underline{a}_j = \tilde{\delta}_j^0, \ldots, \overline{a}_j = \tilde{\delta}_j^{l_j}), (\beta_j^1, \ldots, \beta_j^{l_j}), (\tilde{n}_j^0, \ldots, \tilde{n}_j^{l_j}))$. For notational convenience, define $\delta_j^s \equiv \tilde{\delta}_j^s - \tilde{\delta}_j^{s-1}$ and n_j^s as the fixed cost associated with segment s. Note that, by this definition, $n_j^0 = \tilde{n}_j^0$. The function is assumed to be strictly increasing, but it need not be marginally decreasing as shown in the figure. The assumed cost structure is generic enough to include various special cases: concave, convex, continuous, and $\underline{a}_j = 0$. The PLKP with the generic cost structure was shown to be $\mathcal{NP}$-hard upon reduction from the knapsack problem in [20].

2.1 Motivation for PLKP

The discontinuous piecewise linear cost structure is common in telecommunication and electricity pricing, and in manufacturing and logistics. This cost structure for network flow problems with application is supply chain management was studied in [6]. Our motivation for considering this cost structure for knapsack problems is driven by its applications in electronic commerce, in

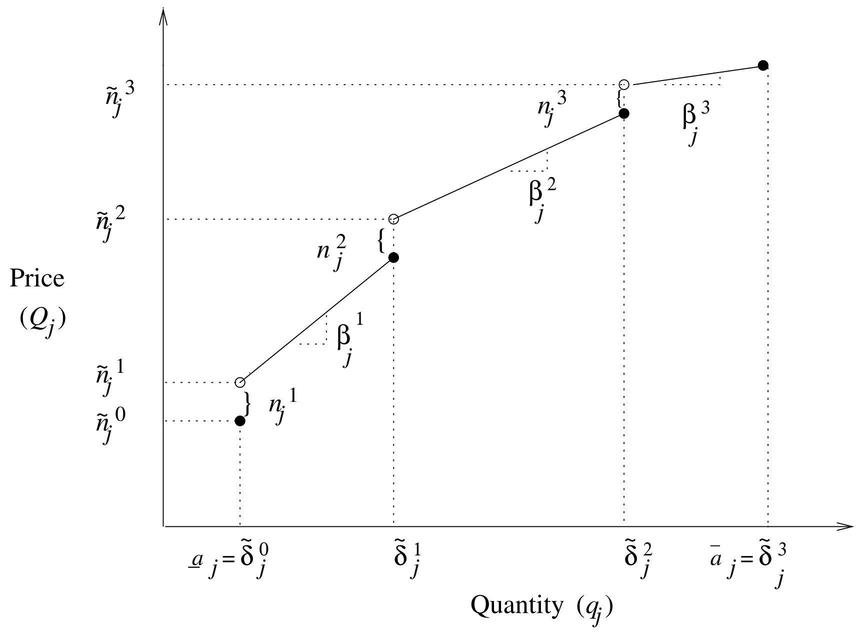


Figure 1: Piecewise linear cost function Q_j

particular, electronic procurement. For example, consider an organization interested in procuring multiple units of a certain good. Using its private electronic marketplace, the organization would invite bids from its suppliers, quoting the demand and the deadline. The suppliers, who wish to sell the goods, submit sealed bids, in which the price is given as a piecewise linear function of the quantity. The bid evaluation problem faced by the buyer is to choose a set of winning sellers and to determine the quantity to be bought from each winning seller such that the total cost is minimized while satisfying the demand and supply constraints. This is exactly the PLKP. The cost structure enables the suppliers to express their *volume discount* strategy or *economies of scale* and/or the production and logistics constraints. The volume discount strategy, which is *buy more and pay less* can be expressed with marginally decreasing cost functions. The discontinuities in the cost structure can capture the production and transportation constraints. Procurement auctions with piecewise linear cost curves are common in industry for long-term strategic sourcing [8]. A procurement scenario with piecewise linear cost function was considered in [24] and approximation algorithms based on dynamic programming were developed. The nonconvex piecewise linear knapsack problems also arise as subproblems in bid evaluation of complex procurement scenarios like multi-unit procurement of heterogeneous goods [10] and multiattribute procurement [21].

Another potential application of the PLKP is the *capacitated plant allocation problem.* Many industrial organizations operate several plants with different production cost structures. One of the operational decisions is to determine the production level of each plant in order to minimize the cost of the requested total production. More precisely, consider the situation where B units of a single product have to be manufactured and N plants with capacities in range $[\underline{a}_j, \overline{a}_j]$ for plant j are available. The cost of production (and possibly transportation) by plant j is given by a nonconvex

$[\underline{a}_j, \overline{a}_j]$	$\underline{a}_j$ is the minimum quantity to be included and $\overline{a}_j$ is the upper limit
Q_j	Piecewise linear cost function for item j defined over $[\underline{a}_j, \overline{a}_j]$
l_j	Number of piecewise linear segments in Q_j
β_j^s	Slope of Q_j on $(\tilde{\delta}_j^{s-1}, \tilde{\delta}_j^s)$
δ_j^s	$\equiv \tilde{\delta}_j^s - \tilde{\delta}_j^{s-1}$
$\tilde{n}_j^s$	$Q_j(\tilde{\delta}_j^s) + n_j^s$
n_j^s	Fixed cost associated with segment s of item j
Q_j	$\equiv ((\underline{a}_j = \tilde{\delta}_j^0, \ldots, \overline{a}_j = \tilde{\delta}_j^{l_j}), (\beta_j^1, \ldots, \beta_j^{l_j}), (\tilde{n}_j^0, \ldots, \tilde{n}_j^{l_j}))$

Table 1: Notation for the nonconvex piecewise linear knapsack problem

piecewise linear cost function Q_j. Then the decision of choosing the plant capacities such that the total cost is minimized is PLKP. A similar version of the problem was considered in [26], with a more generic non-decreasing cost structure (need not be piecewise linear), but with $\underline{a}_j = 0$.

2.2 Related Knapsack Problems

There are two well studied knapsack problems, *tree knapsack problems* and *multiple choice knapsack problems*, which are similar and closely related to the PLKP. The piecewise linear nature of the cost function enables one to view the PLKP in two different ways. In both the ways, consider each *item* j of PLKP to be a *class* consisting of some knapsack items. Each linear segment can be considered as a knapsack item with weight δ_j^s and cost $n_j^s + \beta_j^s \delta_j^s$. However, these items have a precedence constraint (across each j) and therefore s can be selected only if $s-1$ has been already selected. This is a *precedence constrained knapsack* [33], more specifically a *tree knapsack problem* [19]. However, it is different from the above problems as in PLKP the weights δ_j^s are divisible except for the $\underline{a}_j$.

The second way of interpreting PLKP is that a class j consists of l_j mutually exclusive kanpsack items, out of which, at most one can be selected. In this case, each segment s has weight $\tilde{\delta}_j^{s-1}$ and cost $\tilde{n}_j^s$. This is similar to the *multiple choice knapsack problem* [25, 31]. Indeed PLKP is a generalization of this problem as it can accept partial allocation of these items in a given range. The tree knapsack problems and multiple choice knapsack problems are well studied. PLKP in some sense contains both these structures, but still it is different from the above two. These two structures are exploited extensively in this paper for mathematical formulation and algorithms design.

2.3 A Test Suite

A test suite was developed to generate problem instances of various size and features. The intention is to study in a comprehensive way the performance of the mathematical programming formulations and algorithms for different problem instances using computational experiments. The following parameters were considered in the generation of the problem instance: *continuity*, *marginally de-*

Q_j	**CR**	**UR**
n_j^s	$\{0.8, 0.9, 1\}$	$\{0.2, 0.3, \dots, 1\}$
δ_j^s	$\{30, 40, 50\}$	$\{20, 30, \dots, 100\}$
β_j^s	$\{0.8, 0.85, \dots, 1\}$	$\{0.4, 0.45, \dots, 1\}$

Table 2: The set of values for parameters of Q_j for CR and UR cost functions

creasing cost, *similarity of the cost functions*, $\underline{a}_j = 0$ or > 0, and $n_j^0 = 0$ or > 0. Each of the above parameters has two possible values. A continuous function has all the jump costs $n_j^s = 0$, whereas a discontinuous function has non-zero values. A marginally decreasing function will have $\beta_j^s > \beta_j^{s+1}$ and an arbitrary cost function need not have any order over β_j^s. For similar cost functions, the functions are closely related (CR) with values for parameters l_j, β_j^s , n_j^s, and δ_j^s chosen randomly in a close range. To make the cost functions unrelated across the items (UR), these parameters are chosen randomly from a wide range as shown in Table 2. The specific value for a parameter is chosen randomly (with uniform distribution) from the associated set of values.

The lower bound $\underline{a}_j$ can be zero or non-zero. Further for zero lower bound, the associated cost $\tilde{n}_j^0$ can be zero or non-zero. The above parameterization helps in generating problem instances related to different real world scenarios. For example, to study a procurement auction, a discontinuous, marginally decreasing, similar cost functions with $\underline{a}_j > 0$ can be used. A marginally decreasing function has $\beta_j^1 = 1$ and the rest of the β_j^s are chosen with decreasing values from the corresponding set of values. An arbitrary function can have any possible value for any β_j^s . The number of segments l_j is chosen randomly from the set $\{3, 4, 5\}$. The number of items N and the demand B (as a fraction of the total supply) are inputs to the test suite. The test suite can generate 16 types of problem instances based on the above parameterization (two possible values for each of the four parameters: continuity, marginally decreasing, similarity, and lower bound). With the option of $\tilde{n}_j^s$ being zero or non-zero for $\underline{a}_j = 0$, the test suite can generate 24 different problem instances. A problem instance is compactly represented by the truth values of the 5-tuple: (*similarity, continuity, marginally decreasing,* $\underline{a}_j$*,* n_j^0). For example, TFTFT refers to a problem instance with similar cost functions that are discontinuous and marginally decreasing with $\underline{a}_j = 0$ and $n_j^0 > 0$.

3 Mixed Integer Linear Programming Formulations for the PLKP

The cost function Q_j of Figure 1 is nonlinear but due to the piecewise linear nature, the nonlinear knapsack problem can be modeled as a mixed integer linear programming (MILP) problem. There are three standard textbook models for modeling piecewise linear cost functions, which were formally studied in [6, 7]. They are *incremental* (IM), *multiple choice* (MCM), and *convex combination* (CCM) models. In this paper, two more equivalent formulations are proposed, with two different knapsack structures: *precedence constrained knapsack model* (PCKM) and *multiple choice knapsack model* (MCKM). The decision variables and the constraints for the above formulations are given in

Model	**Variables**	**Constraints**	**Quantity** q_j	**Cost** $Q_j(q_j)$
IM	$d_j^s \in \{0,1\}$	$d_j^{s+1} \le d_j^s$	$\underline{a}_j d_j^0 + \sum_{s=1}^{l_j} q_j^s$	$n_j^0 d_j^0 +$
	$q_j^s \ge 0$	$\delta_j^s d_j^{s+1} \le q_j^s \le \delta_j^s d_j^s$		$\sum_{s=1}^{l_j} (n_j^s d_j^s + \beta_j^s q_j^s)$
MCM	$v_j^s \in \{0,1\}$	$\sum_{s=0}^{l_j} v_j^s \le 1$	$\underline{a}_j v_j^0 + \sum_{s=1}^{l_j} q_j^s$	$\tilde{n}_j^0 v_j^0 + \sum_{s=1}^{l_j} \Big(\tilde{n}_j^s v_j^s +$
	$q_j^s \ge 0$	$\tilde{\delta}_j^{s-1} v_j^s \le q_j^s \le \tilde{\delta}_j^s v_j^s$		$\beta_j^s (q_j^s - \tilde{\delta}_j^{s-1} v_j^s)\Big)$
CCM	$v_j^s \in \{0,1\}$	$\sum_{s=0}^{l_j} v_j^s \le 1$	$\underline{a}_j v_j^0 +$	$\tilde{n}_j^0 v_j^0 + \sum_{s=1}^{l_j} \Big(\tilde{n}_j^s \phi_j^s +$
	$\phi_j^s,\, \tau_j^s \ge 0$	$\phi_j^s + \tau_j^s = v_j^s$	$\sum_{s=1}^{l_j} (\tilde{\delta}_j^{s-1} \phi_j^s + \tilde{\delta}_j^s \tau_j^s)$	$(\tilde{n}_j^s + \beta_j^s \delta_j^s) \tau_j^s\Big)$
PCKM	$d_j^s \in \{0,1\}$	$d_j^{s+1} \le d_j^s$	$\underline{a}_j d_j^0 + \sum_{s=1}^{l_j} \delta_j^s x_j^s$	$n_j^0 d_j^0 +$
	$x_j^s \ge 0$	$d_j^{s+1} \le x_j^s \le d_j^s$		$\sum_{s=1}^{l_j} (n_j^s d_j^s + \beta_j^s \delta_j^s x_j^s)$
MCKM	$v_j^s \in \{0,1\}$	$\sum_{s=0}^{l_j} v_j^s \le 1$	$\underline{a}_j v_j^0 +$	$\tilde{n}_j^0 v_j^0 +$
	$y_j^s \ge 0$	$y_j^s \le v_j^s$	$\sum_{s=1}^{l_j} (\tilde{\delta}_j^{s-1} v_j^s + \delta_j^s y_j^s)$	$\sum_{s=1}^{l_j} (\tilde{n}_j^s v_j^s + \delta_j^s \beta_j^s y_j^s)$

Table 3: MILP formulations for PLKP

Table 3.

In all the formulations, a binary variable and a continuous variable are used for each of the linear segments in the cost function. In the IM, the continuous variable q_j^s $(1 \le s \le l_j)$ denotes the quantity chosen from segment s. Note that continuous variable is not used for the *indivisible* segment 0. Hence if $q_j^s > 0$ then $q_j^{s'} = \delta_j^{s'}$ for $s' < s$. The binary variables d_j^s $(0 \le s \le l_j)$ are used to handle the above logical implications. The quantity chosen for an item *incrementally* adds up along the linear segments and hence the name. In the MCM, the binary variable $v_j^s = 1$ indicates that the total quantity chosen lies in the segment s. Hence, if $q_j^s > 0$, then the total quantity $q_j = q_j^s$. At most one of the binary variables (and hence the continuous variables) can be non zero and hence the name multiple choice model. The CCM is similar to the MCM, where the quantity in segment s is chosen as the convex combination of the end points of the segment. The above three textbook models are the same with respect to the (1) set of feasible solutions, (2) Lagrangian relaxation (with respect to the demand constraint), and (3) linear programming relaxation [7].

The proposed new formulations PCKM and MCKM are similar to the IM and the MCM, respectively. The continuous variables in the proposed models are normalized to vary between 0 and 1. These two models, however, reveal the hidden knapsack structures discussed in Section 2.2. The proposed formulations are useful in developing different algorithms due their knapsack structures. The heuristic algorithm based on LP relaxation (Section 4) and the 2-approximation algorithm (Section 6) are developed using the PCKM formulation while the dynamic programming based algorithms (Section 5) and the fully polynomial time approximation scheme (Section 6) are developed using the MCKM formulation.

3.1 Computational Experiments

The MILP formulations were modeled and solved using CPLEX 10.0 for different problem instances generated by the test suite. The intention is to evaluate the formulations in terms of solution time using a commercial package. This will help the practitioners in choosing the appropriate formulation for their requirements. The experiments were carried out on a Windows XP based PC equipped with a 3GHz Intel P4 processor with 760MB RAM. The experiments and the test suite were coded in Java. The average solution time for different problem types with $N = 250$ is shown in Table 4. The time is averaged over 100 instances of each problem type. It is immediately obvious from the table that the solution time varies across the formulations for each problem type and across problem types for each formulation. With respect to the formulations, the MCKM formulation proposed by us took the least time across all problem types, except for few where it was slightly less than that of CCM. Though MCKM is similar to MCM, the solution time of MCKM was significantly less than that of MCM. However, no such significant difference was noted between IM and PCKM. The solution times of the formulations greatly depend on the algorithms used and hence it should be noted that these observations are with respect to CPLEX 10.0.

As expected, the problem type also influenced the solution time. For MCKM, the solution time varied from 75 milliseconds to as long as 7998 milliseconds, depending on the problem type. Henceforth in the analysis we will refer to only the MCKM formulation. In the 5-tuple truth value representation of problem types, let X denote either of the truth values T or F, but fixed for each parameter. For example, XTXFT refer to problem types with the X for each parameter have been fixed to either T or F independent of the X value of the other parameters. Thus XTXFT can represent 4 problem types. To know the influence of continuity on the solution time, one has to compare problem types XTXXX with XFXXX.

Each of the parameters that defines the problem type influenced the solution time. The following values individually showed an increase in solution time:

- $n_j^0 > 0$, $\underline{a}_j = 0$ (compare XXXFT and XXXFF)
- marginally decreasing, with more than 10 fold increase in time for many types (compare XXTXX with XXFXX)
- discontinuity, except for 14 and 20 (compare XFXXX with XTXXX)

No	**Problem**	**MILP formulations**				
	Type	**IM**	**MCM**	**CCM**	**PCKM**	**MCKM**
1	TTTTT	1732.21	1146.44	1280.37	2051.69	615.97
2	TTTFT	23076.69	11649.96	22503.14	23560.95	5089.44
3	TTTFF	4023.32	2115.67	2796.61	3344.4	864.87
4	TTFTT	117.35	88.74	67.44	107.91	74.53
5	TTFFT	3691.92	2026.5	3342.2	3030.62	512.59
6	TTFFF	90.86	104.37	91.21	77.18	77.57
7	TFTTT	3647.41	3845.58	5595.38	4572.51	1536.33
8	TFTFT	28786.77	16741.67	36642.2	26593.54	7998.1
9	TFTFF	5556.0	5522.8	5673.51	5011.49	2099.8
10	TFFTT	246.99	513.46	1092.86	236.8	204.18
11	TFFFT	4639.74	2784.79	4068.47	4210.77	906.04
12	TFFFF	340.23	860.74	822.66	368.36	308.85
13	FTTTT	10506.55	4445.04	12980.21	11377.45	2191.64
14	FTTFT	15674.23	6427.8	11291.93	18253.29	3679.93
15	FTTFF	10530.46	2831.02	6643.53	9815.13	1466.85
16	FTFTT	238.58	222.85	176.64	247.5	168.75
17	FTFFT	337.16	279.84	217.66	273.28	223.91
18	FTFFF	181.5	257.06	165.32	182.5	143.27
19	FFTTT	12934.52	5881.72	18635.35	13675.17	2970.93
20	FFTFT	13327.64	6388.28	10373.00	15246.01	3538.14
21	FFTFF	10378.57	4664.7	9797.2	10815.12	2899.79
22	FFFTT	516.26	386.39	720.96	508.11	285.87
23	FFFFT	715.94	339.5	567.87	630.07	284.53
24	FFFFF	340.94	325.88	546.89	333.17	190.43

Table 4: Average solution time in milliseconds for different MILP formulations

However, the similarity parameter did not reveal any coherent pattern (compare TXXXX with FXXXX). Six problem types (2, 5, 7, 8, 10, and 12) with CR property had solution times greater than that of with UR (14, 17, 19, 20, 22, and 24, respectively) and vice versa for six other problem types (1, 3, 4, 6, 9, and 11 took less time than that of 13, 15, 16, 21, and 23, respectively) . From the above individual influences, one is tempted to believe that either of TFTFT or FFTFT should have taken the longest time and indeed TFTFT (number 8) took the longest time.

4 A Heuristic Algorithm based on LP Relaxation and Convex Envelopes

Certain applications demand a *good quality* solution in a relatively short time. For example, in *iterative* or *multi-round* procurement auctions, the winner determination problem has to be solved in each round. With automated agents participating in the auctions, several hundreds of such rounds will not be uncommon. In such scenarios it may be advantageous to solve optimally the winner determination problem only in the later rounds, where less number of bidders will be left in the auctions. In the initial rounds with more number of bidders, solving the problem fast using a heuristic which provides a good feasible solution will help in earlier termination of the auction. In certain cases like spot purchase markets and exchanges, even the single round auctions need to be cleared in very less time. In this section, we develop a *fast* heuristic that finds a near-optimal solution to the PLKP. The heuristic is based on LP relaxation of PCKM using the convex envelope of the cost function.

In LP relaxation based heuristics, the integral variables are relaxed to take on continuous values and the resulting linear program is solved. If the optimal solution to the relaxed problem is integral, then it is also the optimal solution to the original problem. If not, the continuous solution obtained should be converted into integral solutions using some heuristic rounding techniques so that it is feasible to the original problem. Often, the integral solutions obtained this way are *close* to optimality, depending on the structure of the problem. We first propose a polynomial time algorithm for solving the LP relaxation of PCKM, followed by a rounding heuristic to construct a mixed integer feasible solution from the continuous solution.

4.1 Convex Envelopes and LP Relaxation

The LP relaxation of PCKM can be solved using the convex envelope of the cost function Q_j. It was shown in [6] that the LP relaxation of the incremental model approximates the cost function Q_j with its convex envelope. Since the LP relaxations of PCKM and the incremental model are equivalent ($q_j^s = \delta_j^s x_j^s$), the LP relaxation of PCKM approximates the function Q_j with its convex envelope. This means that solving the LP relaxation is equivalent to solving the PCKM with the convex envelope of Q_j as its cost function. The notion of convex envelope is introduced first and then an algorithm is proposed to construct the convex envelope for a nonconvex piecewise linear function.

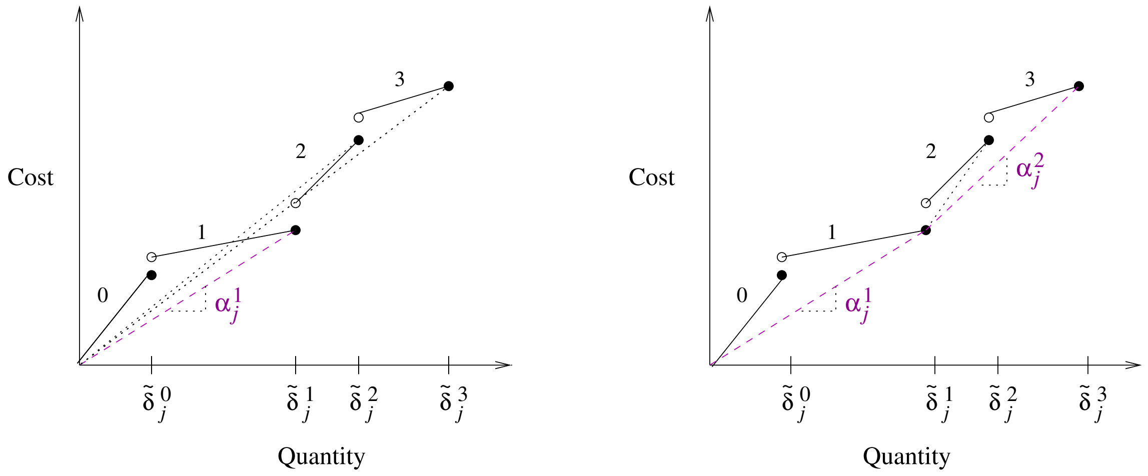


Figure 2: Convex envelope $\tilde{Q}_j$ of the piecewise linear cost function Q_j

Definition 1 (Convex Envelope [16]) *Let $M \subset \mathcal{R}^n$ be convex and compact, and let $f : M \to \mathcal{R}$ be lower semi-continuous on M. A function $g : M \to \mathcal{R}$ is called the convex envelope of f on M if it satisfies:*
1. $g(x)$ is convex on M,
2. $g(x) \leq f(x)$ for all $x \in M$,
3. there is no function $h : M \to \mathcal{R}$ satisfying (1), (2), and $g(\overline{x}) < h(\overline{x})$ for some point $\overline{x} \in M$.

In other words, the convex envelope is the *best* underestimating convex function that approximates the original function. Let $\tilde{Q}_j$ be the convex envelope of the piecewise linear cost function Q_j. Let LP:PCKM denote the LP relaxation of PCKM and let $\{q_j^*\}$ be the optimal quantity chosen in LP:PCKM. Then the result that relates LP relaxation and convex envelopes (see [6]) states that $Z(\text{LP} : \text{PCKM}) = \sum_j \tilde{Q}_j(q_j^*)$. Thus solving the LP relaxation is equivalent to solving the convex envelope problem. Based on this, LP:PCKM will be solved using the convex envelopes.

It is assumed here that the piecewise linear cost function is defined over $[0, \overline{a}_j]$ instead of $[\underline{a}_j, \overline{a}_j]$. This is for mathematical convenience as the convex envelope is used here for solving the LP relaxation and thus the binary variable d_j^0 can take continuous values and partial allocation of the indivisible segment is possible. The slope for the indivisible segment is defined as $\beta_j^0 = \frac{\tilde{n}_j^0}{\underline{a}_j}$ and hence for $q_j \in [0, \underline{a}_j]$ and $Q_j(q_j) = \frac{\tilde{n}_j^0}{\underline{a}_j} q_j$. The convex envelope $\tilde{Q}_j$ is also piecewise linear and hence the algorithm iteratively determines the break points and the slopes from that of Q_j. Let the slopes of $\tilde{Q}_j$ be denoted by $\{\alpha_j^r\}$. The mechanism is illustrated in Figure 2. The first break point is obviously 0, and from $(0,0)$ the slopes to Q_j at all other break of Q_j points are evaluated. The minimum of it is chosen to determine the next break point and slope α_j^1 of $\tilde{Q}_j$. The next slope is determined by proceeding in the same way: determine the slopes from the current break point $(\tilde{\delta}_j^1, Q_j(\tilde{\delta}_j^1))$ to Q_j at the remaining break points. The algorithm is presented first followed by the theorem of correctness.

Algorithm 1 `Convex_Env`*: Algorithm to find the convex envelope of a piecewise linear cost function Q_j.*

```
1. (Initialize)
   $c = 0;\ b = 0;\ u = 0;\ r = 1;$

2. while $(u \leq l_j)$ do:

   2.1. $\alpha_j^r = \min_{u \leq s \leq l_j} \dot{\beta}_j^s = \frac{\tilde{n}_j^s + \beta_j^s \delta_j^s - c}{\tilde{\delta}_j^s - b};\ \tilde{s} = \arg \min_{u \leq s \leq l_j} \dot{\beta}_j^s;$

   2.2. $S_j^r = \{u, \ldots, \tilde{s}\};\ \gamma_j^r = \sum_{s=u}^{\tilde{s}} \delta_j^s;$

   2.3. $c = \tilde{n}_j^{\tilde{s}} + \beta_j^{\tilde{s}} \delta_j^{\tilde{s}};\ b = \tilde{\delta}_j^{\tilde{s}};$

   2.4. $u = \tilde{s} + 1;\ r \leftarrow r + 1;$

   od;

3. $t_j = r;$
```

In Step 2.1, the segment with maximum index s is chosen in case of ties. The time complexity of the algorithm is $O(l_j^2)$. The following properties of the function $\tilde{Q}_j$ can be easily verified:

- $\tilde{Q}_j$ is continuous and piecewise linear, and $\tilde{Q}_j(q_j) \leq Q_j(q_j)$ over the its domain $[0, \overline{a}_j]$.
- If $s' = \max\{s \in S_j^r\}$ for some $1 \leq r \leq t_j$, then $\tilde{Q}_j(\tilde{\delta}_j^{s'}) = Q_j(\tilde{\delta}_j^{s'})$
- $\gamma_j^r = \sum_{s \in S_j^r} \delta_j^s$ for $1 \leq r \leq t_j$ and $\sum_{r=1}^{t_j} |S_j^r| = l_j + 1$
- $\alpha_j^r = \frac{\sum_{s \in S_j^r} \hat{\beta}_j^s \delta_j^s}{\sum_{s \in S_j^r} \delta_j^s}$ and $\alpha_j^r < \alpha_j^{r+1}$ for $1 \leq r < t_j$ and hence convex (as it is continuous and piecewise linear)

With reference to Figure 2, $l_j = 3$, $t_j = 2$, $S_1 = \{0, 1\}$, and $S_2 = \{2, 3\}$. The correctness of the algorithm is stated in the following theorem.

Theorem 1 *Algorithm 1 determines the convex envelope $\tilde{Q}_j$ of the piecewise linear function Q_j.*

Proof: See Appendix.

4.2 LP Relaxation based Heuristic for PCKM

The binary variables d_j^s, when relaxed for LP:PCKM, become redundant for $s > 0$. The constraints of PCKM imply that $d_j^s \geq x_j^s$, for all $j \in J$ and $0 < s \leq l_j$. The objective is of minimization type and all coefficients and variables in the objective are non-negative. Hence in the optimal solution, d_j^s would take the minimal possible value: $d_j^s = x_j^s$. Thus the d_j^s variables are redundant and can be removed from the formulation.

Proposition 1 *The* PCKM *with the convex envelope* $\tilde{Q}_j$ *as the cost function is a continuous knapsack problem.*

Proof: The formulation of the problem is:

$$\text{(CE : PCKM)} : \qquad \min \sum_{j \in J} \sum_{r=1}^{t_j} \left(\alpha_j^r \gamma_j^r z_j^r \right) \tag{2}$$

subject to

$$z_j^r \geq z_j^{r+1} \qquad \forall j \in J; 1 \leq s < t_j \tag{3}$$

$$\sum_{j \in J} \sum_{r=1}^{t_j} \gamma_j^r z_j^r \geq \underline{b} \tag{4}$$

$$z_j^r \in [0,1] \qquad \forall j \in J; 1 \leq r \leq t_j$$

Note that $\alpha_j^r < \alpha_j^{r+1}$ as it is a convex function. Since all $\gamma_j^r > 0$ and $z_j^r \in [0,1]$, segment r will be obviously chosen before $r+1$ in the optimal solution. Hence the constraints $z_j^r \geq z_j^{r+1}$ are redundant. The resulting problem is a continuous knapsack problem with each segment r of j as an item with weight γ_j^r and unit cost α_j^r. ■

The LP relaxation of PCKM is:

$$\text{(LP : PCKM)} : \qquad \min \sum_{j \in J} \sum_{s=0}^{l_j} \left(\hat{\beta}_j^s \delta_j^s \hat{x}_j^s \right) \tag{5}$$

subject to

$$\hat{x}_j^s \geq \hat{x}_j^{s+1} \qquad \forall j \in J; 0 \leq s < l_j \tag{6}$$

$$\sum_{j \in J} \sum_{s=0}^{l_j} \delta_j^s \hat{x}_j^s \geq \underline{b} \tag{7}$$

$$\hat{x}_j^s \in [0,1]$$

The optimal values for $\hat{\mathbf{x}}$ of LP:PCKM can be obtained from $\mathbf{z}$ of CE:PCKM.

Proposition 2 *Let* $\mathbf{z}$ *be the optimal solution to* CE:PCKM. *The optimal solution* $\hat{\mathbf{x}}$ *to* LP:PCKM *is* $\hat{x}_j^s = z_j^r$ *for* $s \in S_r$ *and* $1 \leq r \leq t_j$*, for all* j.

Proof: See Appendix.

Finally one has to construct a feasible solution to PCKM from the optimal continuous solution of LP:PCKM. Let q_j be the optimal quantity chosen in LP:PCKM for item j. Then the rounding heuristic is to allocate q_j units for item j across the segments without violating the precedence constraints. If $q_j \in (0, \underline{a}_j)$, then allocate $\underline{a}_j$ units. One can easily see that the solution constructed by the above heuristic is feasible with respect to the demand and the precedence constraints.

Algorithm 2 `LP_PCKM_Heu`: *Heuristic based on LP relaxation of* PCKM.

N	**Simplex**	**Convex Envelope**
100	17.65	0.01
200	42.79	0.16
300	60.72	0.32
400	89.70	0.78
500	120.92	0.96
600	162.78	1.11
700	201.59	1.43
800	235.78	1.76
900	269.8	2.30
1000	328.48	2.83

Table 5: Average computational time in milliseconds for LP based heuristic solved by simplex algorithm and the proposed convex envelope based algorithm

1. Construct the convex envelope $\tilde{Q}_j$ of Q_j, $\forall j \in J$.
2. Solve CE:PCKM using the algorithm for continuous knapsack problem.
3. Determine the optimal solution $\hat{\mathbf{x}}$ to LP:PCKM from the optimal solution $\mathbf{z}$ of CE:PCKM according to Proposition 2.
4. Construct feasible solution $(\mathbf{d}, \mathbf{x})$ to PCKM from $\hat{\mathbf{x}}$ using the rounding heuristic.

The time complexity if step 1 is $O(L^2)$, that of remaining steps is $O(L)$, and hence the total time complexity is $O(L^2)$.

4.3 Computational Experiments

Computational experiments were performed for the problem types generated using the test suite. The intentions were to study the reduction in computational time by using the proposed convex envelope based algorithm (as against solving by the simplex algorithm) and the optimality gap of the solution provided by the heuristic. As before, 100 instances of each of the problem types were generated using the test suite for various N values. The MCKM formulation was used to model the problem as an MILP. Each instance was solved as an MILP to optimality. Its corresponding LP relaxation based heuristic was solved by simplex algorithm and by the proposed convex envelope based algorithm. CPLEX was used to solve the MILP and the LP relaxation using simplex.

The structure of the problem had very negligible effect on the solution time of the LP based heuristics. The average time in milliseconds is shown in Table 5 for the LP based heuristic using CPLEX and the proposed convex envelope based algorithm. As these values did not change significantly across problem types, the time for problem type TFTFT is only shown in the table. The proposed algorithm is more than 100 folds faster than the simplex algorithm implemented

No	Problem Type	Number of Items N					
		100	200	400	600	800	1000
1	TTTTT	0.0878	0.0382	0.0174	0.0086	0.0066	0.0051
2	TTTFT	0.3627	0.2072	0.0973	0.0685	0.4962	0.0406
3	TTTFF	0.0728	0.0375	0.0171	0.0098	0.0053	0.0041
4	TTFTT	0.1265	0.0547	0.0229	0.0202	0.0110	0.0099
5	TTFFT	0.3352	0.1552	0.0867	0.0534	0.0392	0.0301
6	TTFFF	0.0360	0.0153	0.0052	0.0019	0.0009	0.0000
7	TFTTT	0.0942	0.0439	0.0183	0.0101	0.0074	0.0064
8	TFTFT	0.3997	0.1759	0.0988	0.0685	0.0493	0.0424
9	TFTFF	0.0842	0.0435	0.0166	0.0106	0.0082	0.0051
10	TFFTT	0.0815	0.0512	0.0252	0.0104	0.0089	0.0065
11	TFFFT	0.2818	0.1770	0.0913	0.0508	0.0444	0.0301
12	TFFFF	0.0348	0.0185	0.0073	0.0034	0.0012	0.0001
13	FTTTT	0.2252	0.1165	0.0633	0.0423	0.0224	0.0217
14	FTTFT	0.3977	0.2075	0.1106	0.0784	0.0547	0.0393
15	FTTFF	0.1351	0.0894	0.0439	0.0236	0.0197	0.0154
16	FTFTT	0.2938	0.1435	0.0666	0.0490	0.0334	0.0298
17	FTFFT	0.4062	0.2526	0.1087	0.0818	0.0466	0.0443
18	FTFFF	0.1285	0.0729	0.0323	0.0219	0.0189	0.0107
19	FFTTT	0.2034	0.1038	0.0635	0.0335	0.0236	0.0215
20	FFTFT	0.3853	0.2122	0.1208	0.0790	0.0583	0.0503
21	FFTFF	0.1487	0.0828	0.0394	0.0249	0.0187	0.0140
22	FFFTT	0.3154	0.1499	0.0875	0.0569	0.0298	0.0283
23	FFFFT	0.4059	0.2246	0.1190	0.0803	0.0544	0.0388
24	FFFFF	0.1732	0.0742	0.0393	0.0225	0.0125	0.0117

Table 6: Average optimality gap (%) of the LP based heuristic

by CPLEX. Thus, it is clearly preferable in time constrained applications than the commercial optimizers.

The quality of the solution given by the heuristic was measured using the optimality gap, calculated against the optimal solution provided by the MCKM formulation. It is obvious that the problem structure will influence the optimality gap, unlike the solution time. The results are shown in Table 6. One can easily verify from the Table 6 that problem types with $n_j^0 > 0$ have more optimality gap than with $n_j^0 = 0$ for $\underline{a}_j = 0$ (by comparing problems XXXFT and XXXFF). With respect to similarity parameter (TXXXX versus FXXXX), UR functions had more gap than that of the CR functions. However, there were no significant differences in gap with respect to continuity (XTXXX versus XFXXX) and marginally decreasing (XXTXX versus XXFXX).

5 Exact Algorithms based on Dynamic Programming

Dynamic programming (DP) is the earliest exact solution technique to the knapsack problem [1] and hence been successfully applied to its variants. With the knapsack problems having two parameters, *cost* and *demand*, DP formulations based on each of the parameters are commonly developed. The idea is to make on of these parameters as independent variable and determine the other recursively. In the DP based on demand, the minimum cost for a given demand is determined recursively. For the DP based on cost, the maximum demand that can be accommodated for a given cost is determined recursively.

5.1 Motivation

The DP algorithms for knapsack problems are pseudo polynomial time algorithms, which work relatively fast for small problem instances. However, with larger problem instances, they tend to be inefficient in time and memory. Our purpose of developing DP algorithms for PLKP is mainly twofold: the DP algorithm based on cost is used to develop fully polynomial time approximation scheme in Section 6 and the DP algorithm based on demand is useful for multi-attribute procurement with configurable bids. The DP based on demand does not only provide the optimal solution for demand B, but for any demand $b \in [0, B]$. The DP is basically a recursion and hence with a single run, one can get the optimal solutions for any demand b. This redundancy is useful in certain decision making scenarios and in particular for PLKP, it is useful auctions with configurable bids.

In multi-attribute auctions, there are multiple attributes to be negotiated, in addition to price. Configurable bids allows the suppliers to communicate all of their capabilities and gives the provision for the buyer to configure the bid based on the requirements. For example, consider a steel procurement scenario, with just two attributes: cost and delivery lead time. With large quantities of steel being traded, the suppliers submit configurable bids, as shown in Figure 3. In the figure, the attribute delivery lead time has three possible attribute values, due to the different transportation modes. Further the cost of delivery depends on the quantity. The buyer has to make two decisions: choose the optimal quantity q_j^* from each bid j and an optimal configuration of the delivery modes

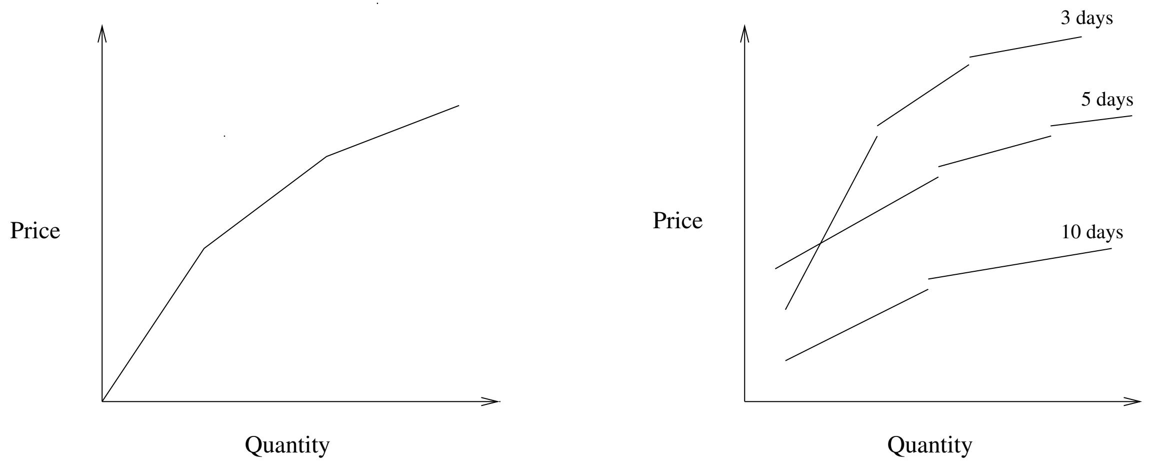


Figure 3: Configurable bids for attributes *cost* and *delivery lead time*

for that optimal quantity. The optimal configuration for q_j^* means splitting this quantity across the various transportation modes to minimize the transportation cost. One can see that this is clearly a PLKP, for a given q_j^*. The overall procurement decision is complex as the two decisions of choosing q_j^* and the optimal configuration are interdependent. On the other hand, if the optimal configuration for each possible q_j is known in advance, the problem of finding q_j^* is relatively easy. Thus if we use the DP based on demand for the configuration of transportation modes, we can get the optimal configuration for all possible q_j by just solving it once. Note that the dimension of this problem is generally very small (few alternate attribute values) that the curse of dimensionality of DP has negligible effect. First we present two naive DP formulations based on the formulation in [26].

5.2 Naive DP Algorithms

In [26], a nonlinear knapsack problem was considered (called as capacitated plant location problem) with nondecreasing cost functions. The problem is generic than the PLKP, as the cost need not be piecewise linear. A DP algorithm based on cost was proposed to solve the problem to optimality. Based on this algorithm, we propose here two naive algorithms, one based on cost and the other based on demand. Without loss of generality, it is assumed that B, $\{\delta_j^s\}$, and $\{Q_j b\}$ are integers.

5.2.1 DP based on Cost

Let $H_j(c)$ denote the maximum demand that can be satisfied with cost c from items $\{1, \ldots, j\}$. The PLKP can be reformulated as:

$$\min\{c : \; H_N(c) \geq B\} \tag{8}$$

Based on this new formulation, the DP algorithm iteratively finds the $H_j(c)$ for each j and for each possible cost $c = 0, \ldots, \overline{C}$, where $\overline{C}$ is an upper bound on the optimal cost. Let $Q_j^{-1}(c)$ denote the inverse price function over the domain $[Q_j(\underline{a}_j), Q_j(\overline{a}_j)]$:

$$Q_j^{-1}(c) = \arg\max_b \{Q_j(b) \leq c\},\ c \in [Q_j(\underline{a}_j), Q_j(\overline{a}_j)] \tag{9}$$

The boundary conditions for $H_j(c)$ are:

$$\begin{aligned}
H_1(c) &= \left\{ \begin{array}{ll} Q_1^{-1}(c) & c \in [Q_1(\underline{a}_1), Q_1(\overline{a}_1)] \\ 0 & c < Q_1(\underline{a}_1) \\ \overline{a}_1 & c > Q_1(\overline{a}_1) \end{array} \right\} && (10) \\
H_j(0) &= 0\ \forall j \in J && (11) \\
H_j(c) &= H_{j-1}(c)\ \forall j > 1,\ c < Q_j(\underline{a}_j) && (12) \\
H_j(c) &= H_j(\sum_{i \leq j} \overline{a}_i),\ c > \sum_{i \leq j} Q_i(\overline{a}_i) && (13)
\end{aligned}$$

The other values of $H_j(c)$ can be recursively found using the follow relation:

$$H_j(c) = \max \left\{ \max_{c' \in [Q_j(\underline{a}_j), Q_j(\overline{a}_j)] \cap [0,c]} \left\{ H_{j-1}(c - c') + Q_j^{-1}(c') \right\}, H_{j-1}(c) \right\} \tag{14}$$

The optimality of the above recursion can be verified as follows. The possible contribution from j in terms of cost is from the set $\{0 \cup [Q_j(\underline{a}_j), Q_j(\overline{a}_j)]\}$. The first term of 14 chooses the optimal positive contribution c' that maximizes the accumulated demand for cost c. The second term is for zero contribution from j ($H_j(c) = H_{j-1}(c)$). For each item j, the time complexity is $O(\overline{C}^2)$ and the optimal solution can be found in $O(N\overline{C}^2)$ steps. The size of the DP table, which stores all the values of H is $N\overline{C}$. The $\overline{C}$ can be any upper bound on the optimal cost. A simple heuristic to determine $\overline{C}$ is to select items in the order of the increasing unit costs $Q_j(\overline{a}_j)/\overline{a}_j$, till the demand B is satisfied. The heuristic runs in $O(N \log N)$ time.

5.2.2 DP based on Demand

In the similar way, the DP algorithm based on demand can be developed. Let $G_j(b)$ denote the minimum cost at which the demand b can be met using the items $1, \ldots, j$. The PLKP can restated as:

$$\min_{b \in [B, \overline{B}]} G_N(b) \tag{15}$$

The $\overline{B}$ is an upper bound on the accumulated demand of the optimal solution. The demand of the optimal solution can be greater than B, if the allocation from each of the items (in the optimal solution) consists only of the indivisible segment $\underline{a}_j$. Hence, the natural choice for the upper bound is $\overline{B} = B + \max_j\{\underline{a}_j\}$. If $\sum_j \underline{a}_j < B$, then $\overline{B} = B$. The boundary conditions are:

$$G_1(b) = \left\{ \begin{array}{ll} Q_1(b) & b \in [\underline{a}_1, \overline{a}_1] \\ 0 & b = 0 \\ \infty & \text{otherwise} \end{array} \right\} \tag{16}$$

$$G_j(0) = 0 \; \forall j \in J \tag{17}$$

$$G_j(b) = G_{j-1}(b) \; \forall j > 1, \; b < \underline{a}_j \tag{18}$$

$$G_j(b) = \infty \text{ if } b > \sum_{i \leq j} \overline{a}_i \tag{19}$$

Given the above conditions, the $G_j(b)$ can be determined from the following recursive equation:

$$G_j(b) = \min \left\{ \min_{b' \in [\underline{a}_j, \overline{a}_j] \cap [0,b]} \{G_{j-1}(b - b') + Q_j(b')\}, G_{j-1}(b) \right\} \tag{20}$$

The optimality of the recursion can be easily verified. The possible contribution from j in terms of demand is from the set $\{0 \cup [\underline{a}_j, \overline{a}_j]\}$ and the recursion searches for the minimum cost from this set. For each j, the time complexity is $O(\overline{B}^2)$ and hence the total time complexity is $O(N\overline{B}^2)$. The size of the DP table to store all the G values is $N\overline{B}$. Next, we present DP formulations with improved time complexity, which exploit the multiple choice knapsack structure of PLKP.

5.3 Improved DP Algorithms

The *multiple choice knapsack problem* (MKP) [25, 31] has N classes, where each class j has l_j knapsack items, each with a cost and a weight. The problem is to choose at most one item from each class such that the demand constraint is met and the overall cost is minimized. As observed earlier, the PLKP is a generalization of MKP, as it allows partial or fractional allocation of an item. Let item i be called as *fractionally allocated* if the allocation $q_i \notin \{0, \underline{a}_i, \tilde{\delta}_i^1, \ldots, \overline{a}_i\}$. Let $B^* = \sum_j q_j$ denote the total allocation accumulated from the optimal solution. Clearly, $B^* \geq B$ and the following properties are straightforward.

Proposition 3 *Let $\{q_j\}$ be an optimal allocation, with $B^* = \sum_j q_j$.*

1. *If $B^* > B$, then $q_j \in \{0, \underline{a}_j\}$, $\forall j \in J$.*

2. *If $B^* = B$, there exists an optimal solution with at most one fractionally allocated item.*

3. *If item i is fractionally allocated with quantity q_i, then the optimal allocation from items $J \setminus \{i\}$ is equivalent to the optimal allocation of MKP with classes $J \setminus \{i\}$ and demand $B - q_i$.*

Exploiting the above properties, we propose DP algorithms, which have better time complexity than the naive formulations. The outline of the algorithms is as follows. The fractional item i could be any item in the set J with $q_i \in [\underline{a}_i, \overline{a}_i]$. Hence, we first solve N MKP problems, each without an item i. For this we use the DP formulations of MKP. Then with each of the above MKP, we combine the corresponding fractional item i to determine the optimal allocation for PLKP.

For DP formulations of MKP, define $c_j^s = Q_j(\tilde{\delta}_j^s)$, for all $j \in J$ and $0 \leq s \leq l_j$. Note that each item of PLKP is a *class* in its corresponding MKP. The DP formulations for MKP were proposed in [9]. The idea is to consider one class of MKP at a time and the problem is solved sequentially. Therefore for a MKP with N classes, the solution is found in N stages.

5.3.1 DP based on Cost

First we present the DP algorithm for MKP. Let $V_j(c)$ denote the maximum demand that can be satisfied for a cost c with MKP classes $1,\dots,j$. The MKP can then be reformulated as:

$$Z(MKP) = \min\{c:\ V_N(c) \geq B\} \tag{21}$$

The $V_j(c)$ can be recursively defined as:

$$V_j(c) = \max\Big\{\max_s\{V_{j-1}(c-\tilde{c}_j^s)+\tilde{\delta}_j^s\},\ V_{j-1}(c)\Big\} \tag{22}$$

where $c = 0,\dots,\overline{C}$ for an upper bound $\overline{C}$ on optimal cost of MKP. It is easy to verify the optimality of the recursion. At stage j, there are two possibilities: either include or not to include one of the MKP items from class j. The boundary conditions to the recursion are $V_j(0) = 0$ and $V_j(c) = -\infty$ if one cannot accumulate any demand for cost c with classes $0,\dots,j$.

The heuristic to determine $\overline{C}$ for MKP is similar to that of the PLKP. The total time complexity to evaluate the DP table is $O(L\overline{C})$ and the size of the DP table is $N\overline{C}$. It is worth noting that the above optimal value is an upper bound to that of PLKP. Using the above formulation, we solve the PLKP as follows. Any item i could have fractional allocation in PLKP. So, we solve the MKP N additional times, each time without an item i. Let MKP^{-i} denote the MKP without i. With a slight abuse of notation, we use $V_j^{-i}(c)$ to denote the accumulated demand for the MKP^{-i}. It clearly follows that for all $i \in J$:

$$V_j^{-i}(c) = V_j(c)\ \forall j < i,\ \forall c \tag{23}$$

Hence MKP^{-i} requires additional $N-i$ evaluations of V. The total space to store all the V values of all the MKPs is $N^2\overline{C}$ and the time complexity is $O(NL\overline{C})$. If i has the fractional allocation $b \in [\underline{a}_i, \overline{a}_i]$ in the optimal solution, then the optimal cost of PLKP is $Q_i(b)+c$, where $V_N^{-i}(c) = B-b$. Let Z^i denote optimal cost such that i has fractional allocation.

$$Z^i = \min_{b\in[\underline{a}_i,\overline{a}_i]} \Big\{Q_i(b) + \min\{c:\ V_N^{-i}(c) = B-b\}\Big\} \tag{24}$$

The minimum is taken over all possible allocations b from i in $[\underline{a}_i, \overline{a}_i]$, with contribution $Q_i(b)$. The demand $B-b$ is exactly satisfied from MKP^{-i} (if possible). The V determines the maximum demand for a given cost and hence many different costs can be achieved for the same demand $B-b$. The minimum of such costs is chosen as the contribution from MKP^{-i}. For the ease of establishing the time complexity, we use the following technique to determine the Z^i.

$$Z^i = \min_{c\in[0,\overline{C}]} \Big\{c + Q_i(b):\ b = \Big(B - V_N^{-i}(c)\Big) \in [\underline{a}_i, \overline{a}_i]\Big\} \tag{25}$$

It is same as that of above, but the minimum is taken over all possible contributions from MKP^{-i}. Clearly, it takes $O(\overline{C})$ steps to determine Z^i using the MKP^{-i}. The optimal solution to PLKP can be easily determined by investigating all Z^i and MKP (solutions with $B^* \geq B$).

$$Z(\text{PLKP}) = \min\left\{\min_{i\in J} Z^i,\ \min_{c\in[0,\overline{C}]}\ \{c:\ V_N(c) \geq B\}\right\} \tag{26}$$

The first term takes $O(N\overline{C})$ steps and the second term takes $O(L\overline{C})$ steps. Hence, the overall time complexity including the MKP^{-i}, is $O(NL\overline{C})$.

5.3.2 DP based on Demand

The DP based on demand exploiting the MKP can be developed in the similar way as above. Let $U_j(b)$ be the minimum cost at which the demand b can be met with MKP from classes $1,\ldots,j$. Then the optimal cost of the MKP is:

$$Z(\text{MKP}) = \min\{U_N(b):\ b \geq B\} \tag{27}$$

The $U_j(b)$ can be recursively defined as:

$$U_j(b) = \min\left\{\min_s\{U_{j-1}(b-\tilde{\delta}_j^s) + c_j^s\},\ U_{j-1}(b)\right\} \tag{28}$$

where $b = 0,\ldots,\overline{B}$ and $\overline{B}$ is an upper bound on the accumulated demand from the optimal solution to MKP. The $\overline{B} = B + \max_j\{\underline{a}_j\}$ is an upper bound on the accumulated demand. The boundary conditions are $U_j(0) = 0$ and $U_j(b) = \infty$ if b units cannot be exactly met from classes $1,\ldots,j$. The space required to maintain all the values is $N\overline{B}$ and time complexity is $O(L\overline{B})$.

Let $U_j^{-i}(b)$ denote the corresponding recursion of MKP^{-i}. Similar to that of the previous algorithm, one can easily see that the total time complexity for evaluating all MKP^{-i} is $O(NL\overline{B})$ and the total space required is in $O(N^2\overline{B})$. If item i has fractional allocation in the optimal solution of PLKP, then the optimal cost is given by

$$Z^i = \min_{b\in[\underline{a}_i,\overline{a}_i]}\left\{Q_i(b) + U_N^{-i}(B-b)\right\} \tag{29}$$

The optimal solution to PLKP is then the minimum of all Z^i and MKP with no items with fractional allocation.

$$Z(\text{PLKP}) = \min\left\{\min_{i\in J} Z^i, \min_{b\geq B}\{U_N(b)\}\right\} \tag{30}$$

The time complexity of first term is $O(N\overline{B})$ and that of second is $O(L\overline{B})$, amounting to the total time complexity of $O(NL\overline{B})$. This is better than the naive formulation with time complexity $O(N\overline{B}^2)$, as L will be usually less than the demand $\overline{B}$.

5.4 Computational Experiments

Computational experiments were performed to study the improvement in computational time of the proposed MKP based DP algorithms. Extensive experimentation using the test suite is not required as the structure of the problem instance is irrelevant to the DP algorithms (which depend

Problem Type	N	$\overline{C}$	**DP-Cost**		$\overline{B}$	**DP-Demand**	
			Naive	**Improved**		**Naive**	**Improved**
DP1	5	603.5	6.91	0.79	286.94	2.0	0.77
	10	1105.5	47.8	6.42	517.85	5.91	3.78
	15	1606.4	113.95	22.35	753.68	16.68	8.59
	20	2168.0	219.35	53.58	995.93	31.43	20.96
	25	2723.5	354.96	105.93	1228.31	51.74	41.09
DP2	5	3694.0	323.24	7.08	2482.6	80.77	3.12
	10	6909.3	2038.31	46.93	4588.1	507.95	24.03
	15	10407.4	5440.15	159.69	6656.0	1275.11	84.87
	20	13191.5	9583.26	359.89	8653.4	2321.41	209.51
	25	16333.5	15298.69	651.19	10577.9	3651.48	389.56

Table 7: Average computational time in milliseconds for DP algorithms

only on N, L, $\overline{C}$, and $\overline{B}$). Two types of problems were considered with the following characteristics: (DP1) $\delta_j^s \in \{20, 30, 40, 50\}$ and (DP2) $\delta_j^s \in \{100, 200, \ldots, 500\}$. The values of other parameters for both problem types were: $l_j \in \{2, 3, 4, 5\}$, $\beta_j^s \in \{1, 2, \ldots, 5\}$, and $n_j^s \in \{20, 30, 40, 50\}$. The two types differ only by the values of $\{\delta_j^s\}$, which results in varying values for $\overline{B}$ and $\overline{C}$. The experiments were conducted for small values of N and results are shown in Table 7. The computational time is averaged over 100 instances of each problem type. The upper bounds $\overline{B}$ and $\overline{C}$ shown in the table are average values. The approximate average values of L for $N = 5, 10, 15, 20$, and 25 were $17, 35, 52, 70$, and 87, respectively. There is significant reduction in the computational time of improved DP algorithms based on cost. For the DP algorithms based on demand, the reduction in time is more for problems with larger $\overline{B}$.

6 Approximation Algorithms

The PLKP is $\mathcal{NP}$-hard and therefore it is of immediate interest to investigate the possibility of approximation algorithms. An approximation algorithm is necessarily polynomial, and is evaluated by the worst or average case possible relative error over all possible instances of the problem. In this section, a 2-approximation algorithm is proposed for PLKP, which will be subsequently used to design a fully polynomial time approximation scheme.

6.1 A 2-Approximation Algorithm for PLKP

For a minimization problem, an ϵ-*approximation* ($\epsilon \geq 1$) algorithm is a polynomial time algorithm that yields a solution with a value that is at most ϵ times the optimum solution value. A 2-approximation algorithm is proposed here for the PLKP. The approximation algorithms for maximization version of the knapsack problems are well studied [32, 18, 27, 28]. However, these algorithms do not give the same approximation ratio for the minimization version and in some cases it is not even possible to characterize the ratio (though the exact algorithms for maximization problems can solve the minimization problems). On the contrary, very few papers [12, 26, 13] consider the minimization problem. The general idea for the algorithm is outlined in [12]. It is a multi-run greedy algorithm that identifies a *critical item* at each run and the final solution guarantees a 2-approximation ratio. In the same lines, we design our algorithm by using the LP relaxation to identify a segment of an item, which we call as the *break segment.* The break segment can be either in or not in the optimal solution and if it is in the optimal solution it can either be partially or fully allocated. Considering all these possibilities, we construct a 2-approximation solution. If it cannot be guaranteed, the segment is removed and the LP relaxation is again applied. The algorithm stops when a 2-approximation is guaranteed or when it is impossible to remove any more segments. In the following, we use the decision variables $\mathbf{d}, \mathbf{x}, \hat{\mathbf{x}}$ consistent with the notation of LP based heuristic of Section4. The $\hat{\mathbf{x}}$ denote the solution of LP:PCKM and $\mathbf{d}, \mathbf{x}$ denote the constructed feasible solution of PCKM.

Proposition 4 *The feasible solution to* PCKM, *constructed using the LP rounding heuristic, has at most one x_j^s such that $0 < x_j^s < 1$. If all x_j^s are either 1 or 0, then either the solution is optimal or there is exactly one $0 \le \hat{x}_j^0 < 1$ in the LP solution that was rounded to obtain $d_j^0 = 1$.*

Proof: See appendix.

The above proposition identifies an unique segment of a feasible PCKM solution that was constructed from the LP solution. The variable corresponding to this segment is an $x_j^s \in (0, 1)$ or d_j^0, whose corresponding LP variable $\hat{x}_j^0$ had a fractional value. This segment will hereafter be referred to as the *break segment.* Let I denote a set of segments $\{(j, s)\}$ that preserve the precedence constraints and `lp_pckm`(I, A, F, $\hat{j}$, $\hat{s}$, b) denote the LP relaxation procedure on the set I. The other arguments in the procedure are outputs: A is the set of segments with LP solution $\hat{x}_j^s = 1$, F is the set of segments with fractional values $0 < \hat{x}_j^s < 1$, $\hat{j}$ and $\hat{s}$ define the break segment $(\hat{j}, \hat{s})$, and $b = x_{\hat{j}}^{\hat{s}} \delta_{\hat{j}}^{\hat{s}}$ if $\hat{s} \neq 0$, else $b = \underline{a}_{\hat{j}}$. Let the quantity b accepted from the break segment be called as the *break quantity.* Note that this quantity is the accepted quantity in the PCKM solution (and not in the LP solution). Let $co(Set)$ be the cost of segments in set Set evaluated using the values of PCKM variables $\{\mathbf{x}\}$ and $wt(Set)$ denote the demand accumulated from the segments in Set.

$$co(Set) \quad = \quad \sum_{(j,s) \in Set} \beta_j^s \delta_j^s x_j^s \tag{31}$$

$$wt(Set) \quad = \quad \sum_{(j,s) \in Set} \delta_j^s \tag{32}$$

Following is our 2-approximation algorithm for PLKP using the PCKM formulation.

Algorithm 3 `2-APPROX`: *A 2-approximation algorithm for* PLKP.

1. (Initialize) $I = \{(j, s) : \ 0 \le s \le l_j, \ \forall j\}$; $R = \emptyset$; $z = \infty$; $I' = \emptyset$;

```
2. while $wt(I) \geq B$ do:
   2.1. lp_pckm($I$, $A$, $F$, $\hat{j}$, $\hat{s}$, $b$);
   2.2. $z \leftarrow \min(z, co(A) + co(F))$;
   2.3. if $F = \emptyset$ then stop fi;
   2.4. if $co(F) \leq co(A)$ then stop fi;
   2.5. $I \leftarrow I \setminus \{\cup_{s \geq \hat{s}} (\hat{j}, s)\}$; $R \leftarrow R \cup \{\cup_{s \geq \hat{s}} (\hat{j}, s)\}$;
   2.6. if $\hat{s} \neq 0$:
      2.6.1. $\delta_{\hat{j}}^{\hat{s}} \leftarrow b - 1$; $I' \leftarrow I \cup (\hat{j}, \hat{s}) \setminus \{\cup_{s < \hat{s}} (\hat{j}, s)\}$;
      2.6.2. $c'' = \sum_{s<\hat{s}} \delta_{\hat{j}}^{s} \beta_{\hat{j}}^{s}$; $b'' = \sum_{s<\hat{s}} \delta_{\hat{j}}^{s}$;
      2.6.3. while $wt(I') \geq (B - b'')$ do:
         (a) lp_pckm($I'$, $A'$, $F'$, $j'$, $s'$, $b'$);
         (b) $z \leftarrow \min(z, c'' + co(A') + co(F'))$;
         (c) if $F' = \emptyset$ then break fi;
         (d) if $j' = \hat{j}$ then break fi;
         (e) if $co(F') \leq (c'' + co(A'))$ then break fi;
         (f) $I' \leftarrow I' \setminus \{\cup_{s \geq s'} (j', s)\}$;
         od;
   fi;
od;
```

The 2-approximation solution value is stored in z. The set F in lp_pckm($\cdot$) is the set of segments with fractional LP solutions $\hat{x}_j^s$. All the segments in F would belong to the same item j and they will all be contiguous in s. This is because there will be only one *segment* of the convex envelope with fractional value in CE:PCKM (as it is equivalent to solving continuous knapsack problem) and this fractional value results in PLKP segments (of that convex segment) with fractional values. Note that though the F is determined using LP variables $\hat{\mathbf{x}}$, in Steps 2.2 and 2.4, the $co(F)$ is evaluated using the PCKM variables $\mathbf{x}$. This gives the contribution (in terms of cost) of the segments from F to the original problem. To prove the correctness of 2-approximation of the algorithm, the following observations are made first:

- If $F = \emptyset$, then the break segment $(\hat{j}, \hat{s})$ is *null* and A has the optimal segments to I.
- If $F \neq \emptyset$, then $wt(A) < B$ and A is the set of optimal segments to I for demand $wt(A)$. Then it follows that $co(A) < Z(\text{PCKM})$.

Theorem 2 *Algorithm* 2-APPROX *is a 2-approximation algorithm to* PLKP.

Proof: It is required to prove that $Z(\text{PCKM}) \leq z \leq 2 \times Z(\text{PCKM})$. The first inequality can be validated easily as z always stores the value of a feasible solution. Consider the first iteration. If $F = \emptyset$, then the solution is optimal and thus the theorem is true. If $F \neq \emptyset$, and if $co(F) \leq co(A)$ then,

$$z \leq (co(A) + co(F)) \leq 2 \times co(A) \leq 2 \times Z(\text{PCKM}) \tag{33}$$

Thus if the algorithm terminates at Steps 2.3 or 2.4, the theorem is proved. If $co(F) > co(A)$, then there are four possibilities:

1. The break segment $(\hat{j}, \hat{s})$, with quantity greater than or equal to b, is in the optimal solution.
2. The break segment $(\hat{j}, \hat{s})$, with quantity greater than or equal to b, is not in the optimal solution.
3. The break segment $(\hat{j}, \hat{s})$, with quantity less than b, is in the optimal solution.
4. The break segment $(\hat{j}, \hat{s})$, with quantity less than b, is not in the optimal solution.

The above four exhaust all the possibilities and at least one of them is true. If (1) is true then $co(F) \leq Z(\text{PCKM})$ and hence

$$z \leq (co(A) + co(F)) \leq 2 \times Z(\text{PCKM}) \tag{34}$$

If (2) or (4) is true then the break segment can be removed from I as it will not alter the optimal solution. The algorithm can be stopped if (1) is known to be true, but since it is not known which of the possibilities is true the algorithm is continued further. Thus if (1) is true, the 2-approximation is guaranteed. If not, remove the break item from I (Step 2.5). Due to the precedence constraints, removal of break segment leads to the removal of the subsequent segments in $\hat{j}$ from I. Ignoring the Steps 2.6 to 2.8, z is a 2-approximation value if either of the possibilities (1), (2), and (4) is true. This is because, at every run of `lp_pckm(·)` at least one of the segments is removed from I and the set R has the removed items. The algorithm terminates at 2.3 or 2.4 guaranteeing a 2-approximation solution, otherwise it terminates when $wt(I) < B$. At this stage, at least one of the segments from R should be in the optimal solution, otherwise it leads to infeasibility. Since the possibility of the segments in the optimal solution is taken care of, the 2-approximation is guaranteed.

The possibility (3) is true only if the break segment is not the indivisible segment. This is taken care of in Step 2.6. If break segment is in the optimal solution with quantity less than b, then all segments preceding $\hat{s}$ in j belong to the optimal solution. A new PLKP is created with segments I' and demand b''. The I', however, has the break segment with quantity $b - 1$, as the optimal quantity is less than b. The Step 2.6.3 is similar to Step 2, except that the problem considered at 2.6.3 assumes that the break item is in the optimal solution with quantity less than b. Thus the solution obtained in 2.6.3 guarantees a 2-approximation for I'. The algorithm, hence, yields a 2-approximation solution for all the above four possibilities.

The Step 2.1 takes $O(L^2)$ time and Step 2.6.3 takes $O(L^3)$ time. The Step 2 has to repeated $O(L)$ times and hence the total time complexity is $O(L^4)$. The algorithm could be implemented in a more efficient way by implementing Step 2.6 independently from Step 2. The efficiency is achieved by avoiding the calling the LP relaxation routine repeatedly. Note that in every run of Step 2, the segments in set A are preserved, that is A is a non-decreasing set, only including new segments in the consecutive runs. Thus, in every run, only the new segments to be added to A are to be identified. This can be implemented in the following way. First evaluate the convex segments of all the cost functions in $O(L^2)$ time. Next, create a min-binary heap of the convex segments in $O(L)$ time [11]. The CE:PCKM can be solved by choosing the convex segments in increasing slopes till the demand is satisfied. This is equivalent to removing the root (minimum element) of the binary heap, till the demand is satisfied. The break segment and the break quantity can be found from the last convex segment removed. Note that F contains the linear segments corresponding to this last convex segment and A contains all the segments removed, but the last segment. According to Step 2.5, segments that follow the break segment should be removed from I. This is equivalent to removing the remaining convex segments of $\hat{j}$ from the heap. However, linear segments preceding $\hat{s}$ in F, should be added in I. Find the convex envelopes just for these segments and add to the heap. The I is now the segments in A, plus the segments in the binary heap. Now the new demand is $B - wt(A)$. Remove the root elements from the heap till this new demand is satisfied. Thus the Step 2 (without 2.6) can be implemented by maintaining a single binary heap and a set A. The time complexity of deletion of the root in the heap is $O(\log L)$. At worst L segments need to be removed and hence the complexity is $O(L \log L)$. At every run, convex envelope needs to be evaluated for the segments $s < \hat{s}$, which are in F. In the worst case, it has to be done for all the segments and hence the time complexity is $O(L^2)$. The new convex segments then need to be inserted into the heap. Each insertion has $O(\log L)$ complexity and hence the total complexity is $O(L \log L)$. Thus the total time complexity of Step 2, excluding 2.6, is $O(L^2)$. The Step 2.6 is similar to the above, except that the demand and certain segments are changed. Hence, Step 2.6 also takes $O(L^2)$ time. However, it may be required to do this for every segment and hence the total complexity is $O(L^3)$. Thus we have the total time complexity of the 2-approximation algorithm as $O(L^3)$. ■

6.2 A Fully Polynomial Time Approximation Scheme

The 2-approximation algorithm developed above is a constant ratio approximation. The *approximation schemes* are superior to the constant ratio approximations as their performance does not limit the ratio on the error. A *fully polynomial time approximation scheme* (FPTAS) is proposed here for the PLKP.

Definition 2 (FPTAS) *An algorithm is called an FPTAS if for a given error parameter ϵ, it provides a solution with value z such that $z \leq (1+\epsilon)z^*$ for a problem instance with optimal objective value z^*, in a running time that is polynomial in the size of the problem and $\frac{1}{\epsilon}$.*

The ϵ in 2-approximation is 1 but in FPTAS it can be made as close to 0 as desired with a running time as a function of $\frac{1}{\epsilon}$, N, and L. FPTAS generally exist for problems with knapsack structure due to the pseudo-polynomial time dynamic programming algorithm. The DP based on cost is the essential building block for the FPTAS [15]. Recall that the running time of DP based on cost is $O(NL\overline{C})$ where $\overline{C}$ is some upper bound on the objective value. The idea of the FPTAS is to exploit this DP algorithm and the 2-approximation algorithm. The cost $c_j^s = \delta_j^s \beta_j^s$ are scaled, thus reducing the running time of the algorithm to depend on the new scaled value. However, the optimal solution for the scaled problem need not be optimal to the original. With a judicious selection of the scaling factor, the optimal solution of the scaled problem can be made close to that of the original problem. Following is the FPTAS for the PLKP. Let C_2 be the objective value of the 2-approximation algorithm and ϵ the required approximation ratio.

Algorithm 4 `DP-FPTAS`*: FPTAS for* PLKP

```
1. $k = \frac{\epsilon C_2}{2L}$

2. if $k > 1$
   then $\hat{c}_j^s = \left\lfloor \frac{c_j^s}{k} \right\rfloor$; $\overline{C} = \left\lfloor \frac{\tilde{C}}{k} \right\rfloor$;
   else $\hat{c}_j^s = c_j^s$; $\overline{C} = C_2$;
   fi;

3. $C_\epsilon \leftarrow$ DP_Cost($\hat{\mathbf{c}}$, $\overline{C}$)
```

The algorithm first determines the scaling factor k based on the desired error parameter ϵ (if $k \leq 1$, scaling is not required). The `DP_Cost`($\hat{\mathbf{c}}$, $\overline{C}$) is the DP algorithm developed in Section 5.3.1, applied to the scaled problem with cost $\hat{\mathbf{c}}$ and upper bound $\overline{C}$. Let $Cost(Set, \mathbf{ct})$ denote the total cost of items in set Set evaluated with cost $\mathbf{ct}$.

Proposition 5 *The value $\overline{C}$ is the upper bound to the scaled problem.*

Proof: See appendix.

The above proposition is necessary for the `DP-FPTAS` algorithm, since without that, one cannot guarantee an optimal solution to the scaled problem.

Theorem 3 *Algorithm* `DP-FPTAS` *is an FPTAS for the* PLKP.

Proof: Let ϵ be the given error parameter and C_ϵ the optimal objective value of the scaled problem obtained by `DP-FPTAS`. Then according to the definition of FPTAS, one has to prove $C_\epsilon \leq (1+\epsilon)C^*$ and running time of `DP-FPTAS` is polynomial in N, L, and $\frac{1}{\epsilon}$. First consider the case where $k > 1$. Let A^* be the set of items with optimal quantities to the PLKP and A_ϵ the set of items with optimal quantities to the scaled problem. Then, $co(A^*, \mathbf{c}) = C^*$ and $co(A_\epsilon, \hat{\mathbf{c}}) = C_\epsilon$. As $\hat{c}_j^s = \left\lfloor \frac{c_j^s}{k} \right\rfloor$,

$$c_j^s \geq k\hat{c}_j^s \geq c_j^s - k \tag{35}$$

The first inequality is obvious from the definition and the second inequality follows from the inequality $x - 1 \leq \lfloor x \rfloor$. By the definition of A^*, A_ϵ, and $\hat{\mathbf{c}}$,

$$co(A_\epsilon, \mathbf{c}) \geq co(A^*, \mathbf{c}) \geq k \times co(A^*, \hat{\mathbf{c}}) \geq k \times co(A_\epsilon, \hat{\mathbf{c}}) \geq co(A_\epsilon, \mathbf{c} - k) \tag{36}$$

The first inequality follows from the fact that items of A^* are optimal to the unscaled problem, second follows from $c_j^s \geq k\hat{c}_j^s$, third is due to the optimality of items of A_ϵ to the scaled problem, and the last one is because $k\hat{c}_j^s \geq c_j^s - k$. Thus,

$$\begin{aligned} co(A^*, \mathbf{c}) &\geq co(A_\epsilon, \mathbf{c} - k) \geq co(A_\epsilon, \mathbf{c}) - Lk \\ \Rightarrow co(A_\epsilon, \mathbf{c}) &\leq C^* + Lk \\ \Rightarrow co(A_\epsilon, \hat{\mathbf{c}}) &\leq C^* + Lk \\ &\leq C^* + \frac{\epsilon C_2}{2} \\ &\leq C^* + \epsilon C^* \\ \Rightarrow C_\epsilon &\leq (1+\epsilon) C^* \end{aligned}$$

The fourth inequality follows from the 2-approximation bound C_2. The time taken by the algorithm is the time taken to determine the 2-approximate solution C_2 ($O(L^3)$) plus the time taken by the DP ($O(NL\overline{C})$).

$$NL\overline{C} = NL \left\lfloor \frac{C_2}{k} \right\rfloor \leq NL\frac{C_2}{k} = \frac{2NL^2}{\epsilon} \tag{37}$$

The total time complexity is $O(\frac{L^3}{\epsilon})$ (as $N \leq L$) and hence the algorithm `DP-FPTAS` is an FPTAS for PLKP. ■

7 Conclusions

This paper considered an important class of nonlinear knapsack problems in which the cost of inclusion of the quantity of an item is a nonconvex piecewise linear function. The cost function considered was generic enough to include various special cases like continuous, linear, concave, convex, marginally decreasing, discontinuous fixed costs, etc. This abstract function can thus model cost structures arising in logistics and supply chains. The problem was particularly motivated by its application to winner determination in volume discount procurement auctions. The problem is NP-hard and was studied from two perspectives: (1) the practical purpose of solving the problem in real-world applications such as e-commerce by developing heuristic and exact algorithms to solve the problem with less computational effort and (2) theoretical viewpoint of studying the approximability of the hard problem.

Two mixed integer programming formulations were proposed which are generalizations of precedence constrained knapsack problems and multiple choice knapsack problems. These formulations were compared with the standard textbook formulations in terms of solution time by a commercial optimization package. The computational experiments conducted over a set 24 representative problem types show that the MCKM formulation proposed by us took less computational time than that of the other formulations for all the problem types.

A fast polynomial time heuristic was proposed to solve the linear relaxation of the problem using convex envelopes. Computational experiments showed that the proposed heuristic was 100 folds faster than that of solving using traditional simplex algorithm. Two pseudo polynomial time exact algorithms based on dynamic programming were devised using the multiple choice formulation. These formulations are faster than the existing naive formulations. A 2-approximation algorithm and a fully polynomial time approximation scheme were also developed.

There are some interesting research problems with respect to solution techniques. An intelligent branch and bound technique that exploits the fast convex envelopes based linear relaxation can possibly solve the problem at a much faster rate than other enumerative search techniques. There is also scope for improving the computational and memory requirements of the dynamic programming algorithms. The information from the LP relaxation can possibly leveraged to fix some of the variables in the dynamic programming to their optimal values. Another research direction is to develop the concept of *core problems* for PLKP, similar to that of the knapsack problem [31], where only a subset of items are considered having a high probability of being in the optimal solution.

References

[1] R. E. Bellman. *Dynamic Programming.* Princeton University Press, Princeton, NJ, 1957.

[2] G. R. Bitran and A. C. Hax. Disaggregation and resource allocation using convex knapsack problems with bounded variables. *Management Science*, 27:431–441, 1981.

[3] K. M. Bretthauer and B. Shetty. The nonlinear resource allocation problem. *Operations Research*, 43:670–683, 1995.

[4] K. M. Bretthauer and B. Shetty. A pegging algorithm for the nonlinear resource allocation problem. *Computers and Operations Research*, 29(5):505–527, 2001.

[5] K. M. Bretthauer and B. Shetty. The nonlinear knapsack problem - algorithms and applications. *European Journal of Operations Research*, 138:459–472, 2002.

[6] K. L. Croxton. *Modeling and solving network flow problems with piecewise linear costs, with applications in supply chain management.* PhD thesis, Operations Research Center, MIT, Cambridge, MA, 1999.

[7] K. L. Croxton, B. Gendron, and T. L. Magnanti. A comparison of mixed-integer programming models for nonconvex piecewise linear cost minimization problems. *Management Science*, 49(9):1268–1273, 2003.

[8] A. Davenport and J. Kalagnanam. Price negotiations for procurement of direct inputs. Research Report RC 22078, IBM Research, Yorktown Heights, NJ, USA, 2001.

[9] K. Dudziński and S. Walukiewicz. Exact methods for the knapsack problem and its generalizations. *European Journal of Operational Research*, 28:3–21, 1987.

[10] M. Eso, S. Ghosh, J. Kalagnanam, and L. Ladanyi. Bid evaluation in procurement auctions with piece-wise linear supply curves. Research Report RC 22219, IBM Research, Yorktown Heights, NJ, USA, 2001.

[11] R. Floyd. Algorithm 245: Treesort3. *Communications of the ACM*, 7(12):701, 1964.

[12] G.V. Gens and E.V. Levner. Computational complexity of approximation algorithms for combinatorial problems. In *Mathematical Foundations of Computer Science*, volume 74 of *Lecture Notes in Computer Science*, pages 292–300. Springer, 1979.

[13] M.M. Güntzer and D. Jungnickel. Approximate minimization algorithms for the 0/1 knapsack and subset-sum problem. *Operations Research Letters*, 26:55–66, 2000.

[14] D. S. Hochbaum. A nonlinear knapsack problem. *Operations Research Letters*, 17:103–110, 1995.

[15] D. S. Hochbaum. Various notions of approximations: good, better, best, and more. In D. S. Hochbaum, editor, *Approximation algorithms for NP-hard problems*, pages 346–398. PWS Publishing Company, 1995.

[16] R. Horst and H. Tuy. *Global Optimization: Deterministic Approaches.* Springer, Berlin, 1990.

[17] T. Ibaraki and N. Katoh. *Resource Allocation Problems: Algorithmic Approaches.* MIT Press, Cambridge, Massachusetts, 1988.

[18] O. H. Ibarra and C. E. Kim. Fast approximation algorithms for the knapsack and sum of subset problems. *Journal of the ACM*, 22:463–468, 1975.

[19] D. S. Johnson and K. A. Niemi. On knapsacks, partitions, and a new dynamic programming technique for trees. *Mathematics of Operations Research*, 8:1–14, 1983.

[20] S. Kameshwaran. *Algorithms for piecewise linear knapsack problems with applications in electronic commerce.* PhD thesis, Department of Computer Science and Automation, Indian Institute of Science, Bangalore, 2004.

[21] S. Kameshwaran and Y. Narahari. Trade determination in multiattribute exchanges. In *Proceedings of IEEE Conference on E-Commerce CEC' 03*, pages 173–180. IEEE Computer Society, 2003.

[22] H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack Problems.* Springer, 2004.

[23] M. S. Kodialam and H. Luss. Algorithms for separable nonlinear resource allocation problems. *Operational Research*, 46:272–284, 1998.

[24] A. Kothari, D. Parkes, and S. Suri. Approximately strategy proof and tractable multi-unit auctions. In *Proceedings of ACM Conference on Electronic Commerce (EC-03)*, 2003.

[25] D. S. Kung. *The multiple choice knapsack problem: Algorithms and applications.* PhD thesis, University of Texas, Austin, 1982.

[26] M. Labbé, E. F. Schmeichel, and S. L. Hakimi. Approximation algorithms for the capacitated plant allocation problem. *Operations Research Letters*, 15:115–126, 1994.

[27] E. L. Lawler. Fast approximation algorithms for knapsack problems. *Mathematics of Operations Research*, 4:339–356, 1979.

[28] S. Martello and P. Toth. *Knapsack Problems: Algorithms and Computer Implementation.* John Wiley and Sons, 1990.

[29] K. Mathur, H. M. Salkin, and B. B. Mohanty. A note on a general non-linear knapsack problems. *Operations Research Letters*, 5:79–81, 1986.

[30] J. J. Moré and S. A. Vavasis. On the solution of concave knapsack problems. *Mathematical Programming*, 49:397–411, 1991.

[31] D. Pisinger and P. Toth. Knapsack problems. In Ding-Zhu Du and P. M. Pardalos, editors, *Handbook of Combinatorial Optimization*, pages 299–428. Kluwer Academic Publications, 1998.

[32] S. Sahni. Approximate algorithms for the 0-1 knapsack problem. *Journal of the ACM*, 22:115–124, 1975.

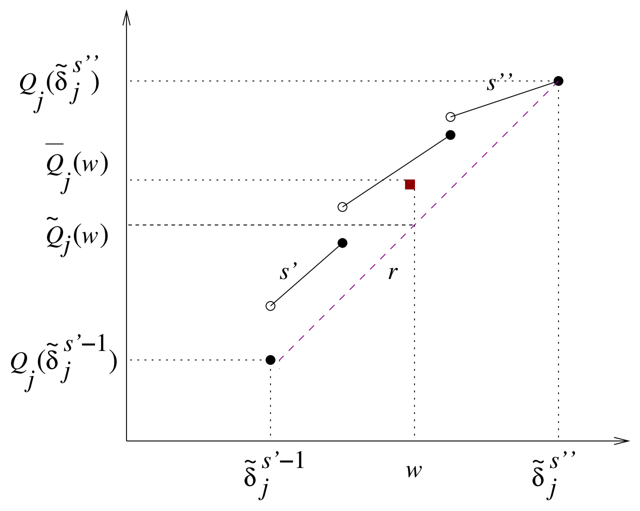
[33] N. Samphaiboon and T. Yamada. Heuristic and exact algorithms for the precedence constrained knapsack problem. *Journal of Optimization Theory and Applications*, 105:659–676, 2000.

[34] P. H. Zipkin. Simple ranking methods for allocation of one resource. *Management Science*, 26:34–43, 1980.

Appendix

Proof of Theorem 1

Proof: It can be easily verified from the construction of the algorithm that $\tilde{Q}_j$ is convex over $[0, \overline{a}_j]$ (it is continuous and piecewise linear with increasing slopes). It is also underestimating, $\tilde{Q}_j(q_j) \leq Q_j(q_j)$. Thus the conditions 1 and 2 of Definition 1 are satisfied. For proving condition 3, let there exist a function $\overline{Q}_j$ which satisfies conditions 1 and 2, and $\overline{Q}_j(w) > \tilde{Q}_j(w)$ for some $w \in [0, \overline{a}_j]$. Let r be the segment of $\tilde{Q}_j$ which includes w. If $s' = \min\{s \in S_r\}$ and $s'' = \max\{s \in S_r\}$, then



$Q_j(\tilde{\delta}_j^{s''})$
$\overline{Q}_j(w)$
$\tilde{Q}_j(w)$
$Q_j(\tilde{\delta}_j^{s'-1})$
s''
s'
r
$\tilde{\delta}_j^{s'-1}$
w
$\tilde{\delta}_j^{s''}$

Figure 4: Proof of $\tilde{Q}_j$ as the convex envelope

$w \in [\tilde{\delta}_j^{s'-1}, \tilde{\delta}_j^{s''}]$ (refer Figure 4). By construction of the algorithm, at the end points of the segment r, $\tilde{Q}_j(\cdot) = Q_j(\cdot)$. Since $\overline{Q}_j$ is an underestimating convex function,

$$\begin{array}{rcl}
& \overline{Q}_j(q_j) \leq Q_j(q_j) = \tilde{Q}_j(q_j) & \text{for } q_j = \tilde{\delta}_j^{s'-1},\ \tilde{\delta}_j^{s''} \\
\Rightarrow & k\overline{Q}_j(\tilde{\delta}_j^{s'-1}) + (1-k)\overline{Q}_j(\tilde{\delta}_j^{s''}) \leq k\tilde{Q}_j(\tilde{\delta}_j^{s'-1}) + (1-k)\tilde{Q}_j(\tilde{\delta}_j^{s''}) & \forall k \in [0,1] \\
\Rightarrow & \overline{Q}_j\left(k\tilde{\delta}_j^{s'-1} + (1-k)\tilde{\delta}_j^{s''}\right) \leq \tilde{Q}_j\left(k\tilde{\delta}_j^{s'-1} + (1-k)\tilde{\delta}_j^{s''}\right) & \forall k \in [0,1] \\
\Rightarrow & \overline{Q}_j(w) \leq \tilde{Q}_j(w) &
\end{array}$$

which contradicts the assumption of $\overline{Q}_j$. Thus condition 3 is also satisfied and $\tilde{Q}_j$ is the convex envelope of Q_j. (The above implications can be proved as follows: first implication is a direct inequality due to the inequality at the end points, the second implication is due to the fact that $\overline{Q}_j$ is a convex function and $\tilde{Q}_j$ is linear over the segment r, and the third implication is straightforward by the definition of w.) ■

Proof of Proposition 2

Proof: Assigning $\hat{x}_j^s = z_j^r$ for $s \in S_r$ and $1 \leq r \leq t_j$, for all j, the precedence constraints are clearly satisfied. For the demand constraint,

$$\begin{aligned}
\sum_j \sum_{s=1}^{l_j} \delta_j^s \hat{x}_j^s &= \sum_j \sum_{r=1}^{t_j} \sum_{s \in S_r} \delta_j^s \hat{x}_j^s \\
&= \sum_j \sum_{r=1}^{t_j} z_j^r \sum_{s \in S_r} \delta_j^s \\
&= \sum_j \sum_{r=1}^{t_j} z_j^r \gamma_j^r
\end{aligned}$$

which is a feasible allocation as $\mathbf{z}$ is feasible. To show that the allocation is optimal,

$$\begin{aligned}
\textstyle\sum_j \sum_{s=0}^{l_j} \hat{\beta}_j^s \delta_j^s \hat{x}_j^s &= \sum_j \sum_{r=1}^{t_j} \sum_{s \in S_r} \hat{\beta}_j^s \delta_j^s \hat{x}_j^s \\
&= \sum_j \sum_{r=1}^{t_j} z_j^r \sum_{s \in S_r} \hat{\beta}_j^s \delta_j^s \\
&= \sum_j \sum_{r=0}^{t_j} \alpha_j^r \gamma_j^r z_j^r \\
\Rightarrow \quad Z(\mathrm{LP} : \mathrm{PCKM}) &= Z(\mathrm{CE} : \mathrm{PCKM})
\end{aligned}$$

■

Proof of Proposition 4

Proof: The CE:PCKM is a continuous knapsack problem (refer Proposition 1) and therefore the optimal solution will have at most one fractional variable. The convex segment of this fractional variable gives rise to fractional variables in the LP solution. If the variable of the convex segment is not fractional, so are the corresponding variables of the LP solution (by Proposition 1). Hence the fractional variables in the optimal LP solution are contiguous and belong to the same item j. When converted into a feasible PCKM solution, there will be at most one fractional x_j^s. If there are no fractional variables, then either the solution is optimal or the fractional convex segment was the first segment which made the corresponding $\hat{x}_j^0$ in the LP solution fractional, and finally rounded the $d_j^0 = 1$. ■

Proof of Proposition 5

Proof: If $k \leq 1$ then the proposition is obvious. For $k > 1$, let the proposition be false. Then, $\left\lfloor \frac{C_2}{k} \right\rfloor < C_\epsilon$, where C_ϵ is the optimal objective value of the scaled problem. Let A_2 be the set of items with quantities for the 2-approximate solution for the unscaled problem. Then, $co(A_2, \mathbf{c}) = C_2$.

$$co(A_2, \hat{\mathbf{c}}) \leq \left\lfloor \frac{C_2}{k} \right\rfloor < C_\epsilon \tag{38}$$

The first inequality follows from the fact that $\lfloor x_1 \rfloor + \ldots + \lfloor x_n \rfloor \leq \lfloor (x_1 + \ldots + x_n) \rfloor$ for any real numbers $x_1, \ldots, x_n$. Note that the set of items in A_2 are feasible to PLKP (irrespective of the cost structure) and therefore $co(A_2, \hat{\mathbf{c}}) \geq C_\epsilon$. This contradicts the above relation and hence $C_\epsilon \leq \overline{C}$. ■