2008

Exact and heuristic solution approaches for the mixed integer setup knapsack problem

Nezih Altay, *University of Richmond*
Powell E Robinson, *Texas A & M University - College Station*
Kurt M Bretthauer, *Indiana University - Bloomington*

Available online at www.sciencedirect.com
ScienceDirect
European Journal of Operational Research 190 (2008) 598–609
EUROPEAN JOURNAL OF OPERATIONAL RESEARCH
www.elsevier.com/locate/ejor

Discrete Optimization

Exact and heuristic solution approaches for the mixed integer setup knapsack problem

Nezih Altay [a,*], Powell E. Robinson Jr. [b], Kurt M. Bretthauer [c]

[a] *Management Department, Robins School of Business, University of Richmond, Richmond, VA 23173, United States*
[b] *Department of Information and Operations Management, Mays Business School, Texas A&M University, College Station, TX 77843-4217, United States*
[c] *Operations and Decision Technologies Department, Kelley School of Business, Indiana University, Bloomington, IN 47405, United States*

Received 13 September 2006; accepted 3 July 2007
Available online 15 August 2007

Abstract

We consider a class of knapsack problems that include setup costs for families of items. An individual item can be loaded into the knapsack only if a setup cost is incurred for the family to which it belongs. A mixed integer programming formulation for the problem is provided along with exact and heuristic solution methods. The exact algorithm uses cross decomposition. The proposed heuristic gives fast and tight bounds. In addition, a Benders decomposition algorithm is presented to solve the continuous relaxation of the problem. This method for solving the continuous relaxation can be used to improve the performance of a branch and bound algorithm for solving the integer problem. Computational performance of the algorithms are reported and compared to CPLEX.

Keywords: Integer programming; Knapsack; Cross decomposition; Benders decomposition

1. Introduction

Knapsack problems are widely studied due to their ability to closely represent real world problems and their frequent appearance as subproblems in more complex models. Martello and Toth (1990) and Kellerer et al. (2004) provide extensive reviews of the major classes of knapsack problems, theoretical results, and solution algorithms. Lin (1998) surveyed well-known non-standard knapsack problems and identified the setup knapsack problem (SKP) as worthy of further investigation. The objective of the SKP is to select specific items, which belong to mutually exclusive product family sets, for placement in a capacitated knapsack while either maximizing its value or minimizing its cost. However, an item can only be selected if a setup charge for placing the family of items in the knapsack is incurred.

The SKP has been identified as a significant subproblem for the solution of capacitated scheduling problems. Guignard (1993) proposed a Lagrangean Decomposition scheme for finding a lower bound on the optimal makespan of a scheduling problem

* Corresponding author. Tel.: +1 804 289 8259; fax: +1 804 289 8878.
E-mail address: naltay@richmond.edu (N. Altay).

0377-2217/$ - see front matter
doi:10.1016/j.ejor.2007.07.003

with parallel unrelated machines with setups. A series of 0–1 programming SKPs, one for each capacitated machine, is solved each iteration of a subgradient algorithm. Chajakis and Guignard (1994) presented a dynamic programming algorithm and two versions of a two-phase enumerative scheme, which solve the SKP in pseudo-polynomial time. However, due to the relatively large storage requirements of dynamic programming, they emphasized the importance of designing good upper bounding schemes and heuristics to be embedded in efficient branch and bound algorithms.

In addition to appearing as a subproblem in scheduling capacitated machines, SKP by itself can model a variety of resource allocation problems. Consider a freight consolidation problem in which a capacitated transport vehicle can carry several different product families as defined by commodity type, shipper, or destination. The decision problem objective is to select the product families and quantities of items to ship in order to maximize the value of the line haul. However, each family requires a setup cost (e.g. a pickup or delivery charge, or specialized freight services such as dunning, refrigeration, or government inspection) before any items in its family can be included in the shipment.

Product category management, where a supplier is allocated limited retail shelf capacity and must determine the mix of product families and items to stock, provides a second example. Each product family included in the lineup has a fixed setup cost for administration, inventory maintenance, and billing, while associated with each item and stocking quantity is an expected profit contribution. The decision is to select the mix of families and items that maximizes the expected value of the product category. Other potential applications are in portfolio management where each investment opportunity is associated with an account setup, subscription or membership cost and expected return. The objective is to select the mix of investments and quantity of each to maximize the expected return of the portfolio.

Recently, Akinc (2006) developed a variety of algorithmic components to improve the efficiency of branch and bound to solve the SKP. The first component solves the LP relaxation of the problem by a non-iterative procedure similar to the one described in Section 4.1. Akinc reported that the LP relaxation of SKP is very tight, with at most a 1.7% gap. The second component he developed is geared towards finding good feasible solutions to accelerate fathoming branch and bound nodes. One approach mentioned simply involved rounding the fractional setup variables up or down to obtain two feasible solutions. Then the better of the two solutions was used for fathoming. The second approach Akinc presented reallocated total capacity by solving a traditional knapsack problem for all the individual items for which a setup is incurred in the LP solution. The last component Akinc developed to improve the efficiency of branch and bound consists of a set of rules to peg the setup variables to zero or one. The three approaches described above reportedly improved the performance of the branch and bound, although no computational times were reported.

In this paper, we focus on a variant of the SKP where fractions of individual items are allowed to load into the knapsack. This mixed-integer setup knapsack problem (MISKP) is equivalent to a continuous relaxation of the binary constraints on items in the SKP of Chajakis and Guignard (1994). However, MISKP allows multiple units of each item to be included in the knapsack and they are continuously divisible (e.g. dollars in an investment fund, weight/volume in assignment to a transportation vehicle, units in a production schedule). Another difference is that in MISKP, family setups do not consume any of the limited resource. The problem appears as a subproblem of the capacitated coordinated replenishment problem (Robinson and Lawrence, 2004). MISKP plays a critical role in developing efficient solution approaches to the capacitated coordinated replenishment problem but has not received much attention in the literature. With this paper we provide the first comprehensive study of this problem.

The contribution of the paper is threefold. First, we present a mixed-integer programming model and investigate the potential of using an exact solution algorithm based on Van Roy's (1983) cross decomposition procedure. Secondly, we propose an efficient upper bounding heuristic and develop a Benders decomposition based approach to obtain the lower bound as an alternative to the method described in Akinc (2006). The development of efficient exact solution approaches to MISKP makes solving large problems possible. Good heuristics to generate tight bounds and efficient methods to solve the LP-relaxation of the problem promotes the development of algorithms for more complex decision problems that utilize MISKP as subproblems. Lastly, the contributions of the paper draw from

better understanding of the problem. We designed an experiment to investigate what makes the problem difficult to solve, and how effective is the state-of-the-art commercially available software.

In the next section, we define the notation and formulate the problem. In Section 3 we describe the cross decomposition algorithm. Section 4 focuses on generating upper and lower bounds. Section 5 discusses experimental design and presents the results of the computational experiments. Section 6 concludes the manuscript.

2. Problem formulation

We formulate MISKP as a minimization problem. For notational simplicity in presentation of the solution algorithms, and without compromising the generality of the model, it is assumed that each family has exactly T unique items. Modifications to allow different numbers of items in each family are straightforward. For each $k \in \{1,2,\ldots,K\}$ and $t \in \{1,2,\ldots,T\}$ define:

- C_{kt} value of including item t in family k in the knapsack. C_{kt} takes a negative value to conform to the objective of the minimization problem. Thus, any $C_{kt} \geqslant 0$ can a priori be dropped from further consideration.
- D_{kt} resources consumed if item t in family k is included in the knapsack; $D_{kt} \geqslant 0$.
- S_k setup cost to include family k in the knapsack; $S_k \geqslant 0$.
- P capacity of the knapsack.
- Y_k binary decision variable to setup family k in the knapsack.
- X_{kt} the fraction of item t that is included in the knapsack.

The MISKP formulation is

(MISKP)

$$\text{Min} \quad \sum_{k=1}^{K} S_k Y_k + \sum_{k=1}^{K} \sum_{t=1}^{T} C_{kt} X_{kt}, \tag{1}$$

Subject to:

$$\sum_{k=1}^{K} \sum_{t=1}^{T} D_{kt} X_{kt} \leqslant P, \tag{2}$$

$$X_{kt} \leqslant Y_k \quad \forall k,t, \tag{3}$$

$$X_{kt} \geqslant 0 \quad \forall k,t, \tag{4}$$

$$Y_k \in \{0,1\} \quad \forall k. \tag{5}$$

We formulate MISKP using the "tight" fractional representation of the continuous decision variables (see Denizel et al. (1996) for convex envelop and strong formulation results on the general class of mixed integer programs). The first term in the objective function represents family setup costs. The second term collects the contribution value of including specific items in the knapsack where a negative value of C_{kt} indicates positive value in the minimization objective function. Constraint (2) is the resource capacity constraint. Constraints (3) are variable upper bound (VUB) constraints that prevent an item from entering the knapsack unless the fixed setup cost is paid. Constraints (4) and (5) are the nonnegativity and binary variable conditions.

3. Cross decomposition algorithm

Cross decomposition, developed by Van Roy (1983) and successfully applied to capacitated facility location (Van Roy, 1986), unifies Benders decomposition and Lagrangean relaxation into a single framework for solving mixed-integer programming problems. It exploits MISKP's special mathematical structure. In MISKP, the Benders decomposition procedure iteratively fixes the values of the complicating Y_k decision variables thereby providing an easily solved continuous knapsack problem. The Lagrangean relaxation side of the algorithm dualizes the VUB constraints also yielding a continuous knapsack problem.

While cross decomposition breaks down complex problems into easier to solve subproblems, its major drawback relates to solving an unstructured master problem to guarantee convergence to optimality. Van Roy (1983) showed that when the Lagrangean relaxation has a nonzero duality gap, cycling may occur with a solution repeating itself every four iterations. Hence, the master problem should be solved every fifth iteration to ensure convergence.

The cross decomposition algorithm proposed here utilizes only a primal master problem to guarantee convergence. Holmberg (1990) suggested this strategy for mixed-integer programming problems since the procedure converges to optimality in a finite number of steps, as is the case in Benders Decomposition. If instead, only the dual master problem (generated using Dantzig-Wolfe's column generation) is solved, the gap between lower and upper bounds would need to be closed using an exact procedure such as a branch and bound algorithm. Following these guidelines, the cross

decomposition algorithm developed for MISKP contains primal and dual subproblems, a primal master problem, and a primal convergence test that is applied after solving the dual subproblem. Details of the algorithm follow.

3.1. Primal subproblem SP

The primal subproblem SP is obtained by fixing the values of the setup variables, $\overline{Y}_k$ to either 0 or 1. This is the same subproblem that would be obtained if Benders decomposition were applied to MISKP. Fixing the setup variable values converts the second component of the objective function below into a constant and VUB constraints (8) into simple upper bounds. The result is a continuous knapsack problem given by:

(SP)

$$\text{Min} \quad \sum_{k=1}^{K}\sum_{t=1}^{T} C_{kt}X_{kt} + \sum_{k=1}^{K} S_k\overline{Y}_k, \tag{6}$$

Subject to:

$$\sum_{k=1B}^{K}\sum_{t=1}^{T} D_{kt}X_{kt} \leqslant P \tag{7}$$

$$X_{kt} \leqslant \overline{Y}_k \quad \forall k,t, \tag{8}$$

$$X_{kt} \geqslant 0 \quad \forall k,t. \tag{9}$$

SP is solved using the procedure proposed by Dantzig (1957). The procedure starts by sorting the X_{kt} variables in non-decreasing order of the contribution value per unit of resource consumed, C_j/D_j, according to index j as indicted by X_j, where $j = 1,2,\ldots,KT$. Since the order of the X_j variables is constant, this sort is performed only once. Permissible items are loaded into the knapsack following this ordering until a critical item r is found for which its resource consumption exceeds the available resources. The optimal solution X_j^* is:

$$X_j^* = 1 \quad \text{for } j = 1,\ldots,r-1, \tag{10}$$

$$X_j^* = \frac{P - \sum_{j=1}^{r-1} D_j}{D_r} \quad \text{for } j = r \tag{11}$$

$$X_j^* = 0 \quad \text{for } j = r+1,\ldots,KT. \tag{12}$$

The time complexity is O(n), plus O($n \log n$) for the initial sorting (Martello and Toth, 1990). Next, we generate the values of the dual variables α and β_{kt}, which are required input for the dual subproblem SD. The dual variable associated with the capacity constraint, $\alpha = -C_r/D_r$ associated with the critical item r. The values of the dual variables, β_{kt}, for constraints (8) are given by $\beta_{kt} = \text{Max}\{0, -C_{kt} - D_{kt}\alpha\}$. This is derived from Eq. (20) of DSP in Section 3.3, and the complementary slackness condition (13). For any feasible solution of α, β_{kt} can be set at its lowest feasible value and leave the objective function of DSP (19) unchanged. The values of β_{kt} are then used in solving the dual sub problem explained in the following section.

$$(\overline{Y}_k - X_{kt})(\text{Max}\{0, -C_{kt} - D_{kt}\alpha\}) = 0 \tag{13}$$

3.2. Dual subproblem SD

The dual subproblem is the Lagrangean relaxation of MISKP with respect to constraints (3). For specified dual variable values, the solution of SD provides a lower bound on MISKP. The formulation of SD is given below, where $\bar{\beta}_{kt}$ is the value of β_{kt} provided by the most recent solution of SP and constraint set (16) is used to guarantee feasibility.

(SD)

$$\text{Min} \quad \sum_{k=1}^{K}\left(S_k - \sum_{t=1}^{T}\bar{\beta}_{kt}\right)Y_k + \sum_{k=1}^{K}\sum_{t=1}^{T}\left(C_{kt} + \bar{\beta}_{kt}\right)X_{kt} \tag{14}$$

Subject to:

$$\sum_{k=1}^{K}\sum_{t=1}^{T} D_{kt}X_{kt} \leqslant P \tag{15}$$

$$X_{kt} \leqslant 1 \quad \forall k,t \tag{16}$$

$$X_{kt} \geqslant 0 \quad \forall k,t \tag{17}$$

$$Y_k \in \{0,1\} \quad \forall k \tag{18}$$

The solution procedure for the dual subproblem sets $Y_k = 1$ for all $k = 1,2,\ldots,K$ with $S_k - \sum_{t=1}^{T}\bar{\beta}_{kt} \leqslant 0$, and $Y_k = 0$ otherwise; and then applies Dantzig's (1957) continuous knapsack algorithm to find the optimal values of the X_{kt}. In order to account for potential updates in the values of the $\bar{\beta}_{kt}$, the X_{kt} variables are sorted each time before the dual subproblem is solved. The Y_k values found as the solution will be input into the primal subproblem.

3.3. Primal master problem MP

MISKP is a relatively easy to solve continuous knapsack problem when the primal variables Y_k are fixed. The solutions to the primal subproblem are used to generate cuts for a Benders master

problem. When the dual variables β_{kt} are fixed, SD decomposes into two subproblems since the integer setup variables and the continuous loading variables become independent from each other. The solution approach for the former was described above. The latter is a continuous knapsack problem. Consequently, one could construct a feasible primal solution and iterate between the two subproblems using the solution of one as input for the other to potentially solve the problem to optimality. These steps iteratively provide upper and lower bounds on the optimal solution, but cannot guarantee a monotonic improvement in bounds or convergence to optimality. Consequently, cycling may occur with a solution repeating itself every four iterations (Van Roy, 1983). Hence, every fifth iteration or whenever an improvement in the bounds ceases to improve, a Benders master problem is solved to obtain a new set of Y_k variables and restart the iterative process. Van Roy (1986) describes using the dual of the primal subproblem, DSP given below, to construct the primal master problem.

(DSP)

$$\text{Max} \quad \sum_{k=1}^{K}\left(S_k - \sum_{t=1}^{T}\beta_{kt}\right)Y_k - \alpha P, \tag{19}$$

Subject to:

$$-\beta_{kt} - D_{kt}\alpha \leqslant C_{kt} \quad \forall k,t, \tag{20}$$

$$\beta_{kt} \geqslant 0 \quad \forall k,t, \tag{21}$$

$$\alpha \geqslant 0. \tag{22}$$

Since the optimum of DSP is bounded, reformulating it provides the primal or Benders master problem MP given below. MP is a mixed-integer programming problem with a single continuous variable, ρ, and given by:

(MP)

$$\text{Min} \quad \rho, \tag{23}$$

Subject to:

$$\sum_{k=1}^{K}\left(S_k - \sum_{t=1}^{T}\beta_{kt}^{l}\right)Y_k - \alpha^{l}P \leqslant \rho \quad \forall l \in L_p, \tag{24}$$

$$Y_k \in \{0,1\} \quad \forall k, \tag{25}$$

$$\rho \text{ unrestricted.} \tag{26}$$

where $l \in L_{\text{p}}$ are extreme points of the feasible region of DSP. If all the extreme point constraints are included in (24), the problem is equivalent to the original MISKP. However, the objective is to find the optimal solution by generating only a subset $L_{\text{Sp}} \subset L_{\text{p}}$ of the extreme points. One extreme point, called a primal cut, is generated every time the dual subproblem, SD, is solved. Benders (1962) showed the algorithm is finite and converges to an optimal solution.

Since the Benders problem lacks favorable mathematical structure, it is solved only when the solutions of the subproblems stop improving. This is identified by the primal convergence tests which check if the Y_k variables obtained from SD can improve upon the incumbent upper bound. The primal convergence test PCT is:

$$\sum_{k=1}^{K}\left(S_k - \sum_{t=1}^{T}\beta_{kt}^{l}\right)\overline{Y}_k - \alpha^{l}P < UB \quad \forall l \in L_{\text{Sp}} \tag{27}$$

where L_{Sp} is the set of primal cuts generated up to this stage of the solution process and UB indicates the incumbent upper bound. If the convergence test fails, the algorithm solves the Benders master problem using L_{Sp} to obtain a new set of setup variables and a new lower bound on MISKP.

To make more use of the subproblem phase, the first time cross decomposition calls for MP its continuous relaxation is solved to optimality rather than stepping into branch and bound. This approach originally suggested by McDaniel and Devine (1977) is used to obtain a quick lower bound. This provides a weaker lower bound, but cycles the procedure back to the primal subproblem faster. After this first call for the master problem, a rudimentary, depth-first, last-in-first-out branch and bound algorithm is employed to solve MP each time it is called. Nodes are fathomed if the node solution exceeds the incumbent upper bound, or contains all integer Y_k values. Branching is performed lexicographically.

To highlight the effect of the cross decomposition algorithm we did not attempt to accelerate the branch and bound by applying any cut generation heuristics. Thus, the master problem is expected to take most of the solution time in cross decomposition. The primal and dual subproblems are simple continuous knapsack problems that are very easy to solve. However, if the lower bound produced by SD is not strong enough the primal subproblem will not be able to make use of it and the convergence test would fail. This would force the algorithm to enter the master problem phase to obtain a better lower bound and go through branch and bound.

Hence, if our computational study indicates that cross decomposition is a competitive alternative for

solving the MISKP, faster solutions can be generated by accelerating the Benders decomposition or by increasing the efficiency of the branch and bound. Magnanti and Wong (1981) suggest accelerating Benders decomposition by generating strong or Pareto-optimal cuts from the alternate optima of the subproblem. Van Roy (1986) implemented a similar approach solving the capacitated location problem but did not report on the resulted improvement. Geoffrion and Graves (1974) emphasized that the exact solution of the master problem contains little information and therefore it is not necessary to solve the master problem optimally. Balas and Martin (1980) suggested solving the master problem only heuristically. Wentges (1996) implemented these suggestions along with Van Roy's cut generation approach on the capacitated facility location problem and found solving the master problems at the beginning only heuristically advantageous.

3.4. Cross decomposition algorithm statement

The cross decomposition algorithm for MISKP is summarized below. As proven by Holmberg (1990) the cross decomposition algorithm using a Benders master problem is guaranteed to converge in a finite number of iterations.

Step 1: Set $LB = -\infty$. Get $\overline{Y}_k$ and UB from the initialization heuristic (see Section 4.1).
Step 2: Solve SP with $\overline{Y}_k$ for α and $\bar{\beta}_{kt}$.
$UB = \text{Min}\{\upsilon(\text{SP}),\ UB\}$
Step 3: If $UB = LB$ then: Stop
Else: Solve SD with $\bar{\beta}_{kt}$ for $\overline{Y}_k$.
$LB = \text{Max}\{\upsilon(\text{SD}), LB\}$.
Step 4: Check PCT.
If TRUE then: go to Step 2
Else: Solve MP for $\overline{Y}_k$.
$LB = \text{Max}\{\upsilon(\text{MP}), LB\}$.
Step 5: If $UB = LB$ then: Stop
Else: go to Step 2.

The advantage of this algorithm is in the use of the subproblems. For MISKP both subproblems are trivial and can be solved very efficiently. However, similar to the case of Benders decomposition the master problem is the handicap of cross decomposition. Strong lower and upper bounds become critical in generating strong cuts for the master problem. Section 5 provides a computational experiment testing the ability of cross decomposition and compares it to off-the-shelf optimization software.

4. Generating upper and lower bounds

4.1. Upper bounding heuristic

Next, we present a two-phase heuristic procedure for finding a good initial upper bound solution. Phase-one solves an integer knapsack problem assuming that each family of items must be loaded into the knapsack as a whole, or not at all. The total value contribution, TC_k, and total resource consumption, TD_k, for each family k, are calculated using the following formulas.

$$TC_k = S_k - \sum_{t=1}^{T} C_{kt} \tag{28}$$

$$TD_k = \sum_{t=1}^{T} D_{kt} \tag{29}$$

Phase-one follows Dantzig's approach where the families are sorted in non-decreasing order by TC_k/TD_k and indexed by j where Y_k^j for $j = 1,2,\ldots,K$. The families are sequentially loaded into the knapsack beginning with $j = 1$ until a family $j = f$ is found which cannot completely fit in the knapsack. All families with $j < f$ are loaded into the knapsack by setting $Y_k^j = 1$, and $X_{kt} = 1$ for all t. All other decision variables are set equal to zero. Capacity remaining to be allocated at the conclusion of phase-one, P', is:

$$P' = P - \sum_{k=1}^{f-1} TD_k \tag{30}$$

Phase-two allocates the remaining capacity following a three-step process.

Step 1: Solve a continuous knapsack problem with capacity P' considering only the items in family f and calculate the total value contribution of the added items.
Step 2: If (S_f – value contribution of the added items) $\geqslant 0$, go to step 3. Otherwise, set $Y_k^f = 1$ and X_{ft} values following Eqs. (10)–(12) and Stop.
Step 3: If $f = |K|$, stop, otherwise set $f = f + 1$ and go to step 1.

The combined solution of these two-phases provides the heuristic solution. This set of Y_k values will be used in the primal subproblem to start cross decomposition. The quality of this upper bound is critical since the branch and bound algorithm in

the master problem will use the upper bound for fathoming.

4.2. Benders decomposition to solve the LP relaxation of MISKP

As mentioned earlier, the first pass through the master problem phase in the cross decomposition algorithm contains no branching and bounding (the integrality requirement is relaxed) to get a quick lower bound. If the algorithm is forced to iterate between the primal subproblem and this integrality relaxed master problem the resulting lower bound would be equal to the LP-relaxation of MISKP. Since Akinc (2006) already showed that LP-MISKP provides a tight lower bound developing an efficient algorithm for this variant of the problem provides an advantage in developing new solution algorithms for MISKP.

The Benders decomposition based procedure to solve LP-MISKP is summarized below:

Step 1: Set $LB = -\infty$. Get $\overline{Y}_k$ and UB from the initialization heuristic.

Step 2: Solve SP with $\overline{Y}_k$ for α^l and β^l_{kt}. Add primal cut PCT

$$UB = \text{Min}\{\upsilon(\text{SP}),\ UB\}$$

Step 3: If $UB = LB$ then: Stop

Else: Solve MP for $\overline{Y}_k$.

$$LB = \text{Max}\{\upsilon(\text{MP}),\ LB\}.$$

Step 4: If $UB = LB$ then: Stop

Else go to Step 2.

We expect this procedure to be very efficient since the primal subproblem is very easy to solve and the master problem should only use a handful primal cuts (Benders, 1962). That means the size of the master problem would be relatively small and could be efficiently solved using the simplex method.

5. Computational experiments

We conducted two experiments in this study. The first experiment compared the performance of the cross decomposition algorithm for solving MISKP with CPLEX Linear Optimizer Version 6.6, a general-purpose software package. The second experiment studied the application of Benders Decomposition to solve the LP-relaxation of MISKP. In addition, performance of the upper bounding heuristic was also analyzed.

5.1. Experimental factors

Balas and Zemel (1980) suggested that a larger gap between the optimal solution and the LP-relaxation would make knapsack problems difficult to solve. Martello and Toth (1990) found that knapsack problems with correlated value and resource consumption parameters are more difficult to solve for both exact and heuristic algorithms. Furthermore, we anticipate that problems with higher capacity utilization will be more difficult to solve. Capacity utilization is defined as the ratio of the total potential resources to the knapsack capacity. If capacity utilization is less than or equal to one then all product families with justifiable setup costs are loaded into the knapsack.

$$\text{Capacity Utilization} = \frac{\sum_{k=1}^{K}\sum_{t=1}^{T} D_{kt}}{P} \tag{31}$$

Large numbers of binary variables increases the number of different combinations of possible family loadings in the knapsack. We expect efficiency to decrease with increasing number of binary variables. In the experiments, we tested the number of binary variables at three levels with the ratio of binary to continuous variables held constant at 0.1 in all test problems.

Setup cost level can also impact algorithm efficiency. A higher setup cost makes it more difficult to justify family inclusion, while lower setup costs will cause more families to be eligible for loading. We expect problems with lower setup costs to be more difficult to solve. Based on these expectations, we utilized a full factorial experimental design of four problem characteristics: capacity utilization (1.5/2.0/2.5), number of product families (50/100/200), setup cost (low/high), and the correlation of item contribution value and resource consumption (low/medium/high), resulting in 54 different combinations of environmental parameters. For each combination, 10 problem instances were randomly generated using the data in Table 1. The demand and setup cost data were drawn from uniform probability distributions.

5.2. Performance of the cross decomposition procedure

The first experiment tested the performances of CPLEX and cross decomposition on MISKP. We made two separate runs with CPLEX. CPLEX utilizes multiple cut generation techniques as well as

Table 1
Experimental data

Factor	Levels		
	Low	Medium	High
Capacity utilization	1.5	2.0	2.5
Binary variables	50	100	200
Continuous variables	500	1000	2000
Data correlation	$D_{kt} = [1,10]$ $C_{kt} = -[1,10]$	$D_{kt} = [1,10]$ $C_{kt} = -[D_{kt} - 2,\ D_{kt} + 2]$	$D_{kt} = [1,10]$ $C_{kt} = -[D_{kt} + 2]$
Fixed cost	$S_k = [40,60]$		$S_k = [60,80]$

Table 2
CPLEX system parameter settings

Preprocessing presolve no	Mip strategy order 1
Simplex pgradient 1	Mip strategy bbinterval 2
Mip strategy covers 2	Mip strategy nodeselect 2
Mip strategy rootheuristic −1	Mip tolerances uppercutoff 0.0
Mip strategy variableselect 3	

heuristics to improve the efficiency of branch and bound. The software allows the user to design custom solution strategies by modifying the settings of these generators and heuristics to enhance computation efficiency. For the first set of runs, we "fine-tuned" several system settings for solving the MISKP to make CPLEX run faster than its default settings. This fine-tuning reduced standard solution times by 50–67%. All settings that were altered from their defaults are listed in Table 2.

With the second set of runs we aimed to compare cross decomposition to a standard branch and bound and "stripped" CPLEX from all of its cut generators and pruning heuristics. For the sake of simplicity we will refer to CPLEX in the first set of experiments as the "fine-tuned" version, and in the second set as the "stripped" version. The default setting of the software then falls in between the "fine-tuned" and the "stripped" version. Interested readers should consult the CPLEX Manual (1999) for additional information. In the experiments, CPLEX was called from AMPL, a front-end problem generator. The Pre-solver option of AMPL was turned off so as not to disturb the experimental results.

Performance metrics included solution time, solution gaps, and the number of nodes solved in the branch and bound tree. Table 3 summarizes the results of the first experiment. Since cross decomposition may visit the master problem several times, the number of nodes reported is the total number of nodes solved. Columns labeled "UB Gap" and "LB Gap" present average gaps between the optimal and initial upper and lower bounds, respectively. Here the initial lower bound is the LB produced by the first run of the master problem rather than the first time the dual subproblem is solved. "MP time" is the average CPU time spent in the master problem phase. "Time" indicates the average CPU time to find and verify an optimal solution. "Root time" is the average CPU time CPLEX uses at Node 0 to find an initial lower bound.

Table 3 indicates that the cross decomposition procedure (CD) is faster than the "stripped" version of CPLEX in all cases except one (utilization = 2.5 with low correlation). In addition, problems with 50 product families were solved faster than both versions of CPLEX in all scenarios. When capacity utilization is low (i.e. 1.5) CD solves problems with low to medium parameter correlation and high setup costs solved faster.

The heuristic procedure developed in this paper provides tight upper bounds. The initial upper bound - optimal solution gap found through the heuristic is on average 50.6% tighter than the initial upper bound the "fine-tuned" CPLEX utilizes and about the same (2.01% tighter to be exact) with the initial upper bound of the "stripped" version. The heuristic upper bound was optimal for problems with capacity utilization equal to 1.5 and low to medium parameter correlation as well as when setup costs were high. In these cases cross decomposition seems to verify optimality by the time CPLEX obtains a node 0 solution.

As problem size increases the upper bound gap shrinks. This result also holds for the lower bound used by CPLEX (LP relaxation of the problem). Hence, this becomes a major advantage for generic optimization approaches. One explanation for the decreasing optimality gaps is that with more candidate families, it is relatively easier or more likely to

Table 3
Results of the cross decomposition experiment

Utilization	1.5	Cross decomposition					CPLEX with cuts & heuristics					CPLEX without cuts & heuristics				
		Initial		MP-time[c]	Time[c]	Nodes	Node 0 Performance			Time	Nodes	Node 0 Performance			Time	Nodes
		UB Gap[a]	LB Gap[b]				UB Gap	LB Gap	Root time[c]			UB Gap	LB Gap	Root time		
Families	50	0.09	0.28	0.41	0.42	217.20	0.29	0.12	0.11	0.69	62.47	0.11	0.14	0.09	2.16	330.93
	100	0.06	0.18	5.01	5.02	988.10	0.17	0.05	0.24	4.01	178.88	0.06	0.06	0.25	39.12	2933.37
	200	0.02	0.14	79.21	79.23	4046.53	0.10	0.02	0.80	32.60	613.12	0.02	0.03	0.83	240.65	6516.72
Correlation	Low	0.00	0.07	0.20	0.21	27.67	0.01	0.03	0.31	1.05	17.77	0.01	0.03	0.29	1.20	34.97
	Med	0.00	0.17	4.08	4.09	245.87	0.06	0.06	0.40	5.59	127.10	0.02	0.07	0.43	26.58	1024.23
	High	0.17	0.35	80.35	80.37	4978.30	0.50	0.10	0.45	30.66	709.60	0.16	0.14	0.45	254.15	8721.82
Setup cost	Low	0.12	0.33	54.31	54.32	3362.91	0.14	0.09	0.41	21.17	470.07	0.11	0.10	0.42	157.78	5427.27
	High	0.00	0.07	2.12	2.12	138.31	0.24	0.04	0.36	3.69	99.58	0.02	0.06	0.36	30.18	1093.41
Utilization	2															
Families	50	0.18	1.00	0.79	0.80	545.30	0.56	0.28	0.13	1.18	142.98	0.21	0.26	0.12	2.10	361.75
	100	0.12	0.96	13.91	13.93	3572.37	0.27	0.09	0.32	7.70	483.77	0.17	0.08	0.32	21.68	1692.57
	200	0.07	0.79	99.81	99.84	6284.77	0.11	0.03	1.18	35.50	914.97	0.09	0.03	1.17	547.29	12784.40
Correlation	Low	0.03	1.04	1.33	1.35	130.53	0.01	0.03	0.40	0.89	12.13	0.09	0.06	0.41	1.52	53.22
	Med	0.05	0.86	8.73	8.74	1118.00	0.14	0.12	0.56	5.87	228.03	0.12	0.08	0.55	31.54	1270.67
	High	0.29	0.84	104.46	104.48	9153.90	0.79	0.24	0.67	37.61	1301.55	0.27	0.24	0.65	538.00	13514.83
Setup cost	Low	0.18	1.62	54.55	54.58	4965.64	0.24	0.15	0.56	15.72	561.07	0.26	0.13	0.55	240.14	6970.92
	High	0.07	0.21	21.79	21.80	1969.31	0.38	0.12	0.53	13.87	466.74	0.06	0.12	0.52	140.56	2921.56
Utilization	2.5															
Families	50	0.44	1.69	0.99	1.00	898.30	0.74	0.29	0.15	1.30	182.13	0.39	0.39	0.14	1.86	335.75
	100	0.21	1.44	7.59	7.61	2413.60	0.27	0.10	0.39	4.79	308.75	0.21	0.11	0.40	22.66	1820.05
	200	0.13	1.48	94.97	95.00	8671.17	0.17	0.03	1.12	20.96	813.10	0.10	0.04	1.08	153.89	4889.63
Correlation	Low	0.16	1.81	6.82	6.84	653.23	0.01	0.03	0.44	0.63	3.87	0.16	0.06	0.43	0.97	34.38
	Med	0.11	1.56	10.98	10.99	1735.14	0.17	0.14	0.44	2.98	144.53	0.15	0.20	0.44	19.03	909.58
	High	0.50	1.24	85.76	85.78	9594.70	1.01	0.25	0.78	23.44	1155.58	0.38	0.28	0.75	158.42	6101.47
Setup cost	Low	0.41	2.73	47.07	47.09	5158.40	0.34	0.15	0.56	9.93	572.87	0.36	0.21	0.55	72.18	3293.43
	High	0.10	0.34	21.97	21.98	2830.31	0.45	0.13	0.55	8.10	296.46	0.10	0.15	0.53	46.77	1403.52

[a] [(UB-Optimal)/Optimal] × 100.
[b] [(Optimal-LB)/Optimal] × 100.
[c] In CPU seconds.

find a family of which the setup cost can be justified by its individual items. On the other hand, the upper bound gap increases with increasing capacity utilization and problem parameter correlation.

During our experiment whenever the initial upper bound was optimal cross decomposition solved the problem faster than both versions of CPLEX. This is due to the quality of the primal cut generated by the initial solution. Once the algorithm goes through the subproblems and enters the master problem phase this strong cut allows the master problem to quickly confirm optimality. This is a notable advantage of cross decomposition.

The disadvantage of cross decomposition lies, not surprisingly in the primal master problem. On average the master problem is responsible for 99.96% of the total solution time. This becomes a handicap with larger problems. Additionally, the initial lower bound provided by the dual subproblem when it is first called is quite weak for cross decomposition. This forces the algorithm to enter the master problem phase very early to get a better lower bound. We solved this problem by solving the LP-relaxation of the master problem and returning to the subproblem phases as soon as possible.

5.3. Performance of the Benders method to solve the LP-relaxation of MISKP

Computational tests with CPLEX revealed, as confirmed by Akinc (2006), that the LP-relaxation of the MISKP provides a very tight lower bound. Thus, finding an efficient method to solve the LP-relaxation is very important. We conducted a second set of experiments to test a Benders decomposition based approach to solve LP–MISKP. Table 4 presents the results of these experiments. Since the LP–MISKP behaves similarly to MISKP under the experimental conditions considered here, for the sake of brevity we only present the results for the hardest problem scenarios, namely, problems with high capacity utilization, high parameter correlation, and low setup costs. We run experiments for 200, 500, and 1000 families with 10 randomly generated test problems for each combination of experimental factors. Each test problem contains 10 items in each family, thus resulting in 10,000 continuous variables in the largest cases.

The Benders decomposition approach solves LP–MISKP to optimality on average in 0.517 CPU seconds suggesting a potential computational improvement is associated with using this algorithm instead of the simplex method when obtaining a LB solution within a branch and bound procedure. The Benders decomposition approach solves LP–MISKP on average eight times faster than CPLEX.

Table 4 also displays the performance of the solution approach Akinc (2006) suggests for solving the LP–MISKP. His algorithm, a non-iterative procedure, takes advantage of a property of the optimal solution of the LP relaxation. The complexity of his algorithm is dominated by the sorting of the setups in order of their cost/resource ratio similar to the procedure described in Section 3.1. The result is a very efficient procedure as indicated by the solution times displayed in Table 4. Akinc's approach solves the LP-relaxation of MISKP in 0.013 CPU seconds.

5.4. Factors affecting problem difficulty

Balas and Zemel (1980) argue that as the problem size increases the gap between the integer optimal solution and the LP-relaxation solution of knapsack problems tend to decrease. Results of our experiments confirm this theory. However, this tightening LB gap does not necessarily mean that the problem becomes easier to solve. Table 3 shows that problem difficulty increases with increasing problem size. When capacity utilization is of concern, problem difficulty peaks with capacity utilization 2.0 and decreases when utilization reaches 2.5. Contrary to most capacitated mixed-integer programming problems, this behavior signals that the

Table 4
Results of the Benders decomposition experiment

Families	UB Gap	UB time	BD iter	BD time	CPLEX time	Akinc time
200	0.383	0.011	3.3	0.067	0.502	0.000
500	0.118	0.032	3.4	0.306	2.980	0.000
1000	0.054	0.083	3.4	1.178	9.050	0.038
Average	0.185	0.042	3.37	0.517	4.177	0.013

* Columns four and five show results for Benders decomposition.

Table 5
Effects of parameter correlation and cost ratio on problem difficulty

Correlation	Setup	Ratio	CPLEX with cuts & heuristics			CPLEX without cuts & heuristics			CD time
			UB Gap	LB Gap	Time	UB Gap	LB Gap	Time	
Correlation effect									
1	1	1.1	0.01	0.06	1.33	0.17	0.09	2.08	5.59
2	1	1.1	0.24	0.21	9.14	0.19	0.24	50.94	15.88
1	2	0.78	0.00	0.00	0.39	0.00	0.00	0.38	0.00
2	2	0.78	0.00	0.00	0.48	0.00	0.00	0.49	0.00
Cost ratio effect									
2	2	0.78	0.00	0.00	0.48	0.00	0.00	0.49	0.00
2	1	1.1	0.24	0.21	9.14	0.19	0.24	50.94	15.88
3	2	1.14	1.07	0.29	24.79	0.17	0.33	216.64	45.90
3	1	1.6	0.46	0.11	36.35	0.37	0.10	417.08	134.52

problem may be easier to solve with higher levels of capacity utilization.

Our results also confirm the argument that the higher the correlation between value and resource consumption parameters the harder the problem. To isolate the correlation effect the expected "total cost to setup cost ratio" for each family was calculated. For example, the expected value of low setup cost is 50 (based on a Uniform distribution between 40 and 60) and the expected value of low correlation is 5.5 (based on a Uniform distribution between 1 and 10). With 10 items in each family, the total cost to setup cost ratio for low correlation and low setup cost would be $10(5.5)/50 = 1.10$. Table 5 lists these ratios and displays performances of CPLEX and cross decomposition (as indicated by "CD time") for various ratios.

The first four rows of Table 5 demonstrate the effect of correlation on problem difficulty. When the total cost to setup cost ratio is kept constant, increasing parameter correlation increases the optimality gap as well as solution times. On the other hand, when correlation is kept constant solution times still increase with increasing cost ratio as indicated in the last four rows.

6. Conclusion

We presented two variants of the Setup Knapsack Problem, namely the Mixed Integer Setup Knapsack Problem and its LP-relaxation, which have important implications for solving capacitated scheduling problems, as well as potential applications in freight consolidation and portfolio management. A cross decomposition based exact algorithm was developed and its performance compared against CPLEX linear optimization software. Computational experiments indicate that the proposed algorithm has great potential in solving the MISKP. A major factor limiting the algorithm's computational efficiency is solving the primal master problem. Improvements on the solution algorithm for the master problem could improve overall solution times significantly.

Our computational results also showed that the proposed upper bounding heuristic is effective and provides a higher quality upper bound than the one associated with CPLEX for every experimental factor setting. We also developed a Benders Decomposition based procedure to solve the LP-relaxation of MISKP. This procedure was, on average, eight times faster than CPLEX, offering an improved lower bounding procedure for LP relaxation based branch and bound methods.

Finally, the experimental design and analysis provides a comprehensive set of test problems that identify the factors driving problem difficulty from both a heuristic and optimization perspective. Both optimization approaches experience significant challenge solving problems with higher numbers of families, demand/cost correlation, and lower setup costs. When applying cross decomposition and CPLEX as heuristics, higher levels of capacity utilization and demand/cost correlation and lower levels of product families and setup cost result in lower quality lower and upper bounds. These results suggest the need for additional heuristic and optimization-based research on this important problem class.

Acknowledgement

We would like to thank Professor Umit Akinc for the feedback he has provided concerning his algorithm during the preparation of the experiments.

References

Akinc, U., 2006. Approximate and exact algorithms for the fixed-charge knapsack problem. European Journal of Operational Research 170, 363–375.

Balas, E., Martin, C.H., 1980. Pivot and complement – a heuristic for 0–1 programming. Management Science 26, 86–96.

Balas, E., Zemel, E., 1980. An algorithm for large zero-one knapsack problems. Operations Research 28, 1130–1154.

Benders, J.F., 1962. Partitioning procedures for solving mixed variables programming problems. Numerische Mathematik 4, 238–252.

Chajakis, E.D., Guignard, M., 1994. Exact Algorithms for the Setup Knapsack Problem. INFOR 32, 124–142.

Dantzig, G.B., 1957. Discrete variable extremum problems. Operations Research 5, 266–277.

Denizel, M., Erenguc, S., Sherali, H.D., 1996. Convex envelope results and strong formulations for a class of mixed-integer programs. Naval Research Logistics 43, 503–518.

Geoffrion, A.M., Graves, G.W., 1974. Multicommodity distribution system design by Benders decomposition. Management Science 20, 822–844.

Guignard, M., 1993. Solving makespan minimization problems with Lagrangean Decomposition. Discrete Applied Mathematics 42, 17–29.

Holmberg, K., 1990. On the convergence of cross decomposition. Mathematical Programming 47, 269–296.

ILOG CPLEX User's Manual, 1999. Version 6.5., ILOG, Incline Village, Nevada.

Kellerer, H., Pferschy, U., Pisinger, D., 2004. Knapsack problems. Springer, Berlin.

Lin, E.Y., 1998. A bibliographical survey on some well-known non-standard knapsack problems. INFOR 36, 274–317.

Magnanti, T.L., Wong, R.T., 1981. Accelerating Benders decomposition: algorithmic enhancement and model selection criteria. Operations Research 29, 464–484.

Martello, S., Toth, P., 1990. Knapsack problems: Algorithms and computer implementations. John Wiley and Sons, Chichester, England.

McDaniel, D., Devine, M., 1977. A modified Benders' partitioning algorithm for mixed integer programming. Management Science 24, 312–319.

Robinson, E.P., Lawrence, F.B., 2004. Coordinated capacitated lot-sizing problem with dynamic demand: A Lagrangean heuristic. Decision Sciences 35, 25–53.

Van Roy, T.J., 1983. Cross decomposition for mixed-integer programming. Mathematical Programming 25, 46–63.

Van Roy, T.J., 1986. A cross decomposition algorithm for capacitated facility location. Operations Research 34, 145–163.

Wentges, P., 1996. Accelerating Benders decomposition for the capacitated facility location problem. Mathematical Methods of Operations Research 44, 267–290.