The Multidimensional Knapsack Problem: Structure and Algorithms

Jakob Puchinger
NICTA Victoria Laboratory
Department of Computer Science & Software Engineering
University of Melbourne, Australia
jakobp@csse.unimelb.edu.au

Günther R. Raidl
Institute of Computer Graphics and Algorithms
Vienna University of Technology, Austria
raidl@ads.tuwien.ac.at

Ulrich Pferschy
Department of Statistics and Operations Research
University of Graz, Austria
pferschy@uni-graz.at

We study the multidimensional knapsack problem, present some theoretical and empirical results about its structure, and evaluate different Integer Linear Programming (ILP) based, metaheuristic, and collaborative approaches for it. We start by considering the distances between optimal solutions to the LP-relaxation and the original problem and then introduce a new core concept for the MKP, which we study extensively. The empirical analysis is then used to develop new concepts for solving the MKP using ILP-based and memetic algorithms. Different collaborative combinations of the presented methods are discussed and evaluated. Further computational experiments with longer run-times are also performed in order to compare the solutions of our approaches to the best known solutions of another so far leading approach for common MKP benchmark instances. The extensive computational experiments show the effectiveness of the proposed methods, which yield highly competitive results in significantly shorter run-times than previously described approaches.

Key words: multidimensional knapsack problem; integer linear programming; heuristics;
History: Submitted March 2007.

---

1. Introduction

The Multidimensional Knapsack Problem (MKP) is a well-studied, strongly NP-hard combinatorial optimization problem occurring in many different applications. In this paper, we

present some theoretical and empirical results about the MKP's structure and evaluate different Integer Linear Programming (ILP) based, metaheuristic, and collaborative approaches for it. We will first give a short introduction to the problem, followed by an empirical analysis based on widely used benchmark instances. Firstly the distances between optimal solutions to the LP-relaxation and the original problem are considered. Secondly we introduce a new core concept for the MKP, which we study extensively. The results of this empirical analysis are then used to develop new concepts for solving the MKP using ILP-based and memetic algorithms. Different collaborative combinations of the presented algorithms are discussed and evaluated. More extensive computational experiments involving longer run-times are also performed in order to compare the solutions of our approaches to the best solutions of a so far leading parallel tabu search for the MKP. Obtained results indicate the competitiveness of the new methods. Finally, we conclude with a summary of the developed methods and an outlook for future work.

The MKP can be defined by the following ILP:

$$\text{(MKP)} \qquad \text{maximize} \quad z = \sum_{j=1}^{n} p_j x_j \tag{1}$$

$$\text{subject to} \quad \sum_{j=1}^{n} w_{ij} x_j \leq c_i, \quad i = 1, \ldots, m, \tag{2}$$

$$x_j \in \{0, 1\}, \quad j = 1, \ldots, n. \tag{3}$$

A set of n items with profits $p_j > 0$ and m resources with capacities $c_i > 0$ are given. Each item j consumes an amount $w_{ij} \geq 0$ from each resource i. The 0–1 decision variables x_j indicate which items are selected. According to (1), the goal is to choose a subset of items with maximum total profit. Selected items must, however, not exceed resource capacities; this is expressed by the *knapsack constraints* (2).

A comprehensive overview of practical and theoretical results for the MKP can be found in the monograph on knapsack problems by Kellerer et al. (2004). Recent reviews of the MKP with different focuses were given by Fréville (2004) and Fréville and Hanafi (2005). Besides exact techniques for solving small to moderately sized instances, based on dynamic programming (Gilmore and Gomory, 1966; Weingartner and Ness, 1967) and branch-and-bound (Shih, 1979; Gavish and Pirkul, 1985) many kinds of metaheuristics have already been applied to the MKP (Glover and Kochenberger, 1996; Chu and Beasley, 1998), including also several variants of hybrid evolutionary algorithms (EAs); see Raidl and Gottlieb (2005) for

a recent survey and comparison of EAs for the MKP. A notable preprocessing scheme was recently proposed by Balev et al. (2008).

To our knowledge, the method currently yielding the best results for commonly used benchmark instances was described by Vasquez and Hao (2001) and has been refined by Vasquez and Vimont (2005). It is a hybrid approach based on tabu search. The search space is reduced and partitioned via additional cardinality constraints, thereby fixing the total number of items to be packed. Bounds for these constraints are calculated by solving a modified LP-relaxation. For each remaining part of the search space, tabu search is independently applied, starting with a solution derived from the LP-relaxation of the partial problem. The improvement described in Vasquez and Vimont (2005) lies mainly in an additional variable fixing heuristic.

The current authors originally suggested a core concept for the MKP in Puchinger et al. (2006). Preliminary results with a metaheuristic/ILP collaborative approach have been presented in Puchinger et al. (2005). More details can also be found in the first author's PhD thesis (Puchinger, 2006). The current article summarizes this previous work and extends it to a large degree. In particular, it contains the completely new collaborative approach combining heuristic and exact algorithms for the solution of the core problem. Moreover, many additional computational analyses have been performed showing more instructive details and new comparisons. Larger instances of a different type with up to 2500 items are now also considered.

Benchmark Instances

Chu and Beasley's benchmark library[1] for the MKP (Chu and Beasley, 1998) is widely used in the literature and will also be the basis of most of the experiments presented in this paper. The library contains classes of randomly created instances for each combination of $n \in \{100, 250, 500\}$ items, $m \in \{5, 10, 30\}$ constraints, and tightness ratios

$$\alpha = c_i / \sum_{j=1}^{n} w_{ij} \in \{0.25, 0.5, 0.75\}.$$

Resource consumption values w_{ij} are integers uniformly chosen from $(0, 1000)$. Profits are correlated to the weights and generated as

$$p_j = \sum_{i=1}^{m} w_{ij}/m + \lfloor 500\, r_j \rfloor,$$

[1] `http://people.brunel.ac.uk/~mastjjb/jeb/info.html`

where r_j is a randomly uniformly chosen real number from $(0, 1]$. For each class, i.e., for each combination of n, m, and α, 10 different instances are available. It can be argued that these instances have uncorrelated weight coefficients for every item which may not always be the case in real-world scenarios. However, this class of instances is the only widely used benchmark library where the number of instances allows a reasonable and statistically sound analysis. Therefore, most papers dealing with the MKP stick to these instances.

To test the core concept also for larger instances we performed additional tests on a set of 11 benchmark instances proposed by Glover and Kochenberger[2]. The instance sizes range from 100 items and 15 constraints to 2500 items and 100 constraints. A major difference of these instances in comparison to those of Chu and Beasley is the fact that the resource consumption values are correlated, which is also the case in many practical applications of the MKP. We also performed experiments with metaheuristics using those instances. Obtained results are ambiguous and do not allow any serious conclusions, we therefore omit the details in Section 5. In fact, the small number of only 11 instances hardly permits a statistical analysis but encourages specific parameter tuning. Moreover, no comparative values for the method by (Vasquez and Vimont, 2005) are available.

2. The MKP and its LP-relaxation

In the LP-relaxation of the MKP, the integrality constraints (3) are replaced by

$$0 \leq x_j \leq 1, \quad j = 1, \ldots, n. \tag{4}$$

Basic LP-theory implies the following important property characterizing the structure of an optimal solution x^{LP} to the linear programming (LP) relaxation of the MKP (Kellerer et al., 2004):

Proposition 1 *There exists an optimal solution x^{LP} with at most* $\min\{m, n\}$ *fractional values.*

An interesting question, which arises for almost any integer linear program, concerns the difference between the ILP and the corresponding LP-relaxation with respect to optimal solutions' values and their structures. Concerning the latter, a probabilistic result was given for the classical 0/1-knapsack problem (KP): Goldberg and Marchetti-Spaccamela (1984)

[2] `http://hces.bus.olemiss.edu/tools.html`

showed that the number of items which have to be changed when moving from an optimal solution of the KP's LP-relaxation to one of the integer problem grows logarithmically in expectation with increasing problem size for uniformly distributed profits and weights.

For MKP, Dyer and Frieze (1989) showed for an analogous probabilistic model that the afore mentioned number of changes grows stronger than logarithmically in expectation with increasing problem size.

2.1 Empirical Analysis

Since there is so far only this negative result by Dyer and Frieze (1989) on the distance of the LP-optimum and the optimum of the MKP, we performed an empirical in-depth examination on smaller instances of Chu and Beasley's benchmark library for which we were able to compute optimal solutions x^* (with $n = 100$ items, $m \in \{5, 10\}$ constraints, and $n = 250$ items, $m = 5$ constraints).

Table 1 displays the average distances between optimal solutions x^* of the MKP and optimal solutions x^{LP} of the LP-relaxation

$$\Delta LP = \sum_{j=1}^{n} |x_j^* - x_j^{\mathrm{LP}}|, \tag{5}$$

the integral part of x^{LP}

$$\Delta LP_{\mathrm{int}} = \sum_{j \in J_{\mathrm{int}}} |x_j^* - x_j^{\mathrm{LP}}|, \quad \text{with } J_{\mathrm{int}} = \{j = 1, \ldots, n \mid x_j^{\mathrm{LP}} \text{ is integral}\}, \tag{6}$$

and the fractional part of x^{LP}

$$\Delta LP_{\mathrm{frac}} = \sum_{j \in J_{\mathrm{frac}}} |x_j^* - x_j^{\mathrm{LP}}|, \quad \text{with } J_{\mathrm{frac}} = \{j = 1, \ldots, n \mid x_j^{\mathrm{LP}} \text{ is fractional}\}. \tag{7}$$

We further display the Hamming distance between x^* and the arithmetically rounded LP solution x^{RLP}. This solution can be infeasible, since arithmetic rounding (rounding down if the decimal part is less than or equal to 0.5 and rouding up else) can violate the capacity constraints.

$$\Delta LP_{\mathrm{rounded}} = \sum_{j=1}^{n} |x_j^* - x_j^{\mathrm{RLP}}| \quad \text{with} \quad x_j^{\mathrm{RLP}} = \lceil x_j^{\mathrm{LP}} - 0.5 \rceil, \; j = 1, \ldots, n. \tag{8}$$

Also shown is the Hamming distance between x^* and a feasible solution x' created by sorting the items according to decreasing LP-relaxation solution values and applying a greedy-fill

procedure

$$\Delta LP_{\text{feasible}} = \sum_{j=1}^{n} |x_j^* - x_j'|. \tag{9}$$

All distances are displayed as percentages of the total number of items (%n), except ΔLP_{frac} which is displayed as a percentage of the number of knapsack constraints (%m).

Table 1: Distances between LP and integer optimal solutions (average values over 10 instances per problem class and total averages).

n	m	α	ΔLP %n	ΔLP_{int} %n	ΔLP_{frac} %m	$\Delta LP_{\text{rounded}}$ %n	$\Delta LP_{\text{feasible}}$ %n
100	5	0.25	5.88	3.60	45.68	5.60	7.70
		0.50	6.72	4.40	46.32	6.60	9.30
		0.75	6.56	4.30	45.17	6.50	11.60
250	5	0.25	3.12	2.20	46.25	3.12	3.80
		0.50	3.42	2.56	42.81	3.36	5.52
		0.75	3.15	2.28	43.25	3.20	7.04
100	10	0.25	9.01	4.50	45.12	8.40	11.50
		0.50	6.88	3.40	34.75	5.70	14.60
		0.75	6.75	2.60	41.51	6.50	17.40
Average			5.72	3.32	43.43	5.44	9.83

The distance $\Delta LP_{\text{feasible}}$ between heuristically obtained feasible solutions and the optimal ones is quite important and can grow up to an average of 17.4% of the number of items for the instance class with 100 items, 10 constraints, and $\alpha = 0.75$.

We further observe that $\Delta LP_{\text{rounded}}$ is almost always smaller than 10% of the total number of variables and is 5.44% on average. When the available time for optimization is restricted, it therefore is reasonable for these instances to reduce the search space to a reasonably sized neighborhood of the solution to the LP-relaxation, or to explore this more promising part of the search space first. The most successful algorithms for the MKP exploit this fact (Raidl and Gottlieb, 2005; Vasquez and Hao, 2001; Vasquez and Vimont, 2005).

The distance between the integral parts ΔLP_{int} increases slower than the number of variables. The distance between the fractional part and an optimal MKP solution ΔLP_{frac} seems to depend on the number of constraints (about 45% of the number of constraints). This can partly be explained with the result from Proposition 1. If we assume that our LP solution is the one with, at most, $\min\{m, n\}$ fractional values, the distance to the optimum of the fractional values can never be larger than $\min\{m, n\}$. The total distance ΔLP does therefore depend more on the number of constraints than on the number of variables which can also be observed in the shown results.

2.2 Exploiting the LP-Relaxation in Exact Solution Procedures

Based on the empirical results of Section 2.1 it seems to be worth to guide a classical Branch and Bound method to explore the neighborhood of the LP-relaxation first before exploring other regions of the solution space. This approach has similarities with the concepts of *local branching* (Fischetti and Lodi, 2003) and *relaxation induced neighborhood search* (Danna et al., 2003).

In more detail, we focus the optimization to the neighborhood of the arithmetically rounded LP solutions. This is achieved by adding a single constraint to the MKP similar to the local branching constraints from Fischetti and Lodi (2003). The following inequality restricts the search space to a neighborhood of Hamming distance k (the value of k is defined by the user) around the rounded LP solution x^{RLP}:

$$\Delta(x, x^{\mathrm{RLP}}) = \sum_{j \in S^{\mathrm{RLP}}} (1 - x_j) + \sum_{j \notin S^{\mathrm{RLP}}} x_j \leq k, \tag{10}$$

where $S^{\mathrm{RLP}} = \{j = 1, \ldots, n \mid x_j^{\mathrm{RLP}} = 1\}$ is the binary support of x^{RLP}.

In our implementation we use CPLEX as branch-and-cut system and initially partition the search space by constraint (10) into the more promising part and by the inverse constraint $\Delta(x, x^{\mathrm{RLP}}) \geq k + 1$ into a second, remaining part. CPLEX is forced to first completely solve the neighborhood of x^{RLP} before investigating the remaining search space.

Alternatively, we can consider a variant of this constraint which bounds only the deviation from the integral values of the LP solution and does not restrict variables with fractional LP values. In this case we replace (10) by

$$\sum_{j | x_j^{\mathrm{LP}} = 1} (1 - x_j) + \sum_{j | x_j^{\mathrm{LP}} = 0} x_j \leq k. \tag{11}$$

2.2.1 Computational Results

We performed an experimental investigation on the hardest instances of Chu and Beasley's benchmark library with $n = 500$ items and $m \in \{5, 10, 30\}$ constraints. The general purpose ILP-solver CPLEX 9.0 was used on a 2.4 GHz Intel Pentium 4 PC.

Table 2 shows results when adding constraint (10) with different values of k and limiting the CPU-time to 500 seconds. Listed are average percentage gaps of obtained solution values z to the optimal objective value z^{LP} of the LP-relaxation ($\%_{\mathrm{LP}} = 100 \cdot (z^{\mathrm{LP}} - z)/z^{\mathrm{LP}}$). We display standard deviations as subscripts, the numbers of times this neighborhood size

yields the best solutions of this experiment (#), and average numbers of explored nodes of the branch-and-bound tree (*Nnodes*).

Table 2: Results on large MKP instances when including constraint (10) to only search the neighborhood of x^{RLP} (average values over 10 instances per problem class and total averages, $n = 500$).

m	α	no constraint			$k = 10$			$k = 25$			$k = 50$		
		$\%_{\mathrm{LP}}$	#	*Nnodes*	$\%_{\mathrm{LP}}$	#	*Nnodes*	$\%_{\mathrm{LP}}$	#	*Nnodes*	$\%_{\mathrm{LP}}$	#	*Nnodes*
5	0.25	$0.080_{0.010}$	8	5.50E5	$\mathbf{0.079}_{0.009}$	9	5.58E5	$0.080_{0.009}$	8	5.38E5	$\mathbf{0.079}_{0.008}$	8	5.38E5
	0.50	$0.040_{0.005}$	7	5.06E5	$0.040_{0.006}$	7	5.09E5	$\mathbf{0.039}_{0.005}$	10	4.88E5	$\mathbf{0.039}_{0.005}$	10	4.92E5
	0.75	$\mathbf{0.025}_{0.004}$	8	5.36E5	$\mathbf{0.025}_{0.003}$	9	5.49E5	$\mathbf{0.025}_{0.004}$	7	5.24E5	$\mathbf{0.025}_{0.004}$	7	5.28E5
10	0.25	$\mathbf{0.206}_{0.022}$	9	3.15E5	$0.221_{0.024}$	4	3.00E5	$\mathbf{0.206}_{0.022}$	9	3.03E5	$\mathbf{0.206}_{0.022}$	9	3.06E5
	0.50	$\mathbf{0.094}_{0.013}$	8	3.01E5	$0.102_{0.012}$	5	2.87E5	$0.095_{0.014}$	7	2.91E5	$\mathbf{0.094}_{0.014}$	8	2.93E5
	0.75	$\mathbf{0.066}_{0.009}$	8	3.05E5	$0.068_{0.007}$	5	2.98E5	$\mathbf{0.066}_{0.008}$	8	2.95E5	$\mathbf{0.066}_{0.008}$	9	2.98E5
30	0.25	$0.598_{0.038}$	5	1.11E5	$0.601_{0.036}$	1	1.02E5	$\mathbf{0.555}_{0.067}$	9	1.08E5	$0.605_{0.042}$	4	1.09E5
	0.50	$0.258_{0.009}$	2	1.15E5	$0.258_{0.024}$	5	1.07E5	$\mathbf{0.257}_{0.012}$	4	1.12E5	$\mathbf{0.257}_{0.010}$	4	1.12E5
	0.75	$0.158_{0.013}$	5	1.12E5	$0.162_{0.014}$	4	1.07E5	$\mathbf{0.155}_{0.011}$	8	1.07E5	$0.159_{0.012}$	4	1.07E5
Average		$0.169_{0.013}$	6.7	3.17E5	$0.173_{0.015}$	5.4	3.13E5	$0.164_{0.017}$	7.8	3.07E5	$0.170_{0.014}$	7.0	3.09E5

Obtained results indicate that forcing CPLEX to first explore the more promising part of the search space can be advantageous, since this could yield better primal bounds earlier on during the branch-and-bound search and thus allow a faster pruning of the search tree. Especially for $k = 25$, which corresponds to 5% of the total number of variables, we almost always obtain slightly better solutions than with the standard approach. A one-sided Wilcoxon signed rank test over all the instances showed that the $k = 25$ version provides better results than standard CPLEX with an error probability of 1.4%. For $k = 10$ results were worse than those of CPLEX without additional constraint, for $k = 50$ results are not improved on average, whereas the mean number of best solutions reached is higher.

Further experiments with constraint (11) showed that the performance is worse than for the case with (10); in particular, no significant improvements upon the solution method without additional constraint could be observed. This may be due to the fact that the search space is not as strongly reduced as it is the case with (10). Detailed results can be found in Puchinger (2006).

3. The Core Concept

The core concept was first presented for the classical 0/1-knapsack problem (Balas and Zemel, 1980) and led to very successful KP algorithms (Martello and Toth, 1988; Pisinger, 1995, 1997). The main idea is to reduce the original problem to a core of items for which it is hard to decide whether or not they will occur in an optimal solution, whereas all variables

corresponding to items outside the core are initially fixed to their presumably optimal values. The core concept was also studied for bicriteria KP by Gomes da Silva et al. (2008).

3.1 The Core Concept for KP

The (one-dimensional) 0/1-knapsack problem is the special case of MKP arising for $m = 1$. Every item j has associated a profit p_j and a single weight w_j. A subset of these items with maximal total profit has to be packed into a knapsack of capacity c. The classical greedy heuristic for KP packs the items into the knapsack in decreasing order of their *efficiencies* $e_j := \frac{p_j}{w_j}$ as long as the knapsack constraint is not violated. It is well known that the same ordering also defines the solution structure of the LP-relaxation, which consists of three parts: The first part contains all variables set to one, the second part consists of at most one *split item* s, whose corresponding LP-value is fractional, and finally the remaining variables, which are always set to zero, form the third part.

For most instances of KP (except those with a very special structure of profits and weights) the integer optimal solution closely corresponds to this partitioning in the sense that it contains most of the highly efficient items of the first part, some items with medium efficiencies near the split item, and almost no items with low efficiencies from the third part. Items of medium efficiency constitute the so-called core.

The precise definition of the core of KP introduced by Balas and Zemel (1980) requires the knowledge of an optimal integer solution x^*. Assume that the items are sorted according to decreasing efficiencies and let

$$a := \min\{j \mid x_j^* = 0\}, \qquad b := \max\{j \mid x_j^* = 1\}. \tag{12}$$

The core is given by the items in the interval $C = \{a, \ldots, b\}$. It is obvious that the split item is always part of the core.

The KP Core (KPC) problem is derived from KP by setting all variables x_j with $j < a$ to 1 and those with $j > b$ to 0. Thus the optimization is restricted to the items in the core with appropriately updated capacity and objective. Obviously, the solution of KPC would suffice to compute the optimal solution of KP, which, however, has to be already partially known to determine C. Pisinger (1997) reported experimental investigations of the exact core size. He also studied the hardness of core problems and gave a model for their expected hardness in Pisinger (1999).

In an algorithmic application of the core concept, only an approximate core including the actual unknown core with high probability can be used. A first class of core algorithms is based on an approximate core of fixed size $c = \{s-\delta, \ldots, s+\delta\}$ with various choices of δ, e.g. δ being a predefined constant or $\delta = \sqrt{n}$. An example is the MT2 algorithm by Martello and Toth (1988). First the core is solved, then an upper bound is derived in order to eventually prove optimality. If this is not possible, a variable reduction is performed, which tries to fix as many variables as possible to their optimal values. Finally the remaining problem is solved to optimality.

Since the estimation of the core size remains a weak point of fixed core algorithms, Pisinger proposed two expanding core algorithms. Expknap (Pisinger, 1995) uses branch-and-bound for enumeration, whereas Minknap (Pisinger, 1997) applies dynamic programming and enumerates at most the smallest symmetrical core. For more details we also refer to Kellerer et al. (2004).

3.2 Efficiency Measures for MKP

Trying to apply a core concept to MKP the sorting of items raises an important question since in contrast to KP there is no obvious definition of efficiency anymore. Consider the most obvious form of efficiency for the MKP, which is a direct generalization of the one-dimensional case (Dobson, 1982):

$$e_j^{\text{simple}} = \frac{p_j}{\sum_{i=1}^{m} w_{ij}}. \tag{13}$$

Different orders of magnitude of the constraints' coefficients are not considered and a single constraint may easily dominate all others. This drawback can be avoided by scaling:

$$e_j^{\text{scaled}} = \frac{p_j}{\sum_{i=1}^{m} \frac{w_{ij}}{c_i}}. \tag{14}$$

Taking into account the relative contribution of the constraints, Senju and Toyoda (1968) propose:

$$e_j^{\text{st}} = \frac{p_j}{\sum_{i=1}^{m} w_{ij}(\sum_{l=1}^{n} w_{il} - c_i)}. \tag{15}$$

For more details on efficiency measures we refer to Kellerer et al. (2004) where a general form of efficiency is defined by introducing relevance values $r_i \geq 0$ for every constraint:

$$e_j^{\text{general}} = \frac{p_j}{\sum_{i=1}^{m} r_i w_{ij}}. \tag{16}$$

These relevance values r_i can also be interpreted as a kind of surrogate multipliers. In the optimal solution of the LP-relaxation the dual variable u_i for every constraint (2), $i = 1, \ldots, m$, signifies the opportunity cost of the constraint. Moreover, it is well-known that for the LP-relaxed problem, the dual variables are the optimal surrogate multipliers and lead to the optimal LP-solution. Recalling the results of Section 2.1 it is therefore an obvious choice to set $r_i = u_i$ yielding the efficiency measure e_j^{duals}; compare Chu and Beasley (1998).

Finally, applying the relative scarcity of every constraint as a relevance value, Fréville and Plateau (1994) suggested setting

$$r_i = \frac{\sum_{j=1}^{n} w_{ij} - c_i}{\sum_{j=1}^{n} w_{ij}}, \tag{17}$$

yielding the efficiency measure e_j^{fp}.

Rinnooy Kan et al. (1993) study the quality of greedy heuristic solutions as a function of the relevance values. They emphasize the importance of using an optimal dual solution for deriving the relevance values, since those values yield for the greedy heuristic the upper bound $z^* + m \cdot \max\{p_j\}$, where z^* is the optimal solution value, and this bound cannot be improved.

3.3 The Core Concept for MKP

The basic concept can be expanded from KP to MKP without major difficulties. The main problem, however, lies in the fact that the core and the core problem crucially depend on the used efficiency measure e. Let x^* be an optimal solution and assume that the items are sorted according to decreasing efficiency e, then define

$$a_e := \min\{j \mid x_j^* = 0\} \quad \text{and} \quad b_e := \max\{j \mid x_j^* = 1\}. \tag{18}$$

The core is given by the items in the interval $C_e := \{a_e, \ldots, b_e\}$, and the core problem is defined as

$$\begin{aligned}
(\text{MKPC}_e) \qquad & \text{maximize} \quad z = \sum_{j \in C_e} p_j x_j + \tilde{p} && (19)\\
& \text{subject to} \quad \sum_{j \in C_e} w_{ij} x_j \le c_i - \tilde{w}_i, \quad i = 1, \ldots, m && (20)\\
& \qquad\qquad x_j \in \{0, 1\}, \quad j \in C_e, && (21)
\end{aligned}$$

with $\tilde{p} = \sum_{j=1}^{a_e - 1} p_j$ and $\tilde{w}_i = \sum_{j=1}^{a_e - 1} w_{ij},\ i = 1, \ldots, m$.

In contrast to KP, the solution of the LP-relaxation of MKP does not consist of a single fractional split item, but its up to m fractional values give rise to a whole *split interval* $S_e := \{s_e, \ldots, t_e\}$, where s_e and t_e are the first and the last index of variables with fractional values after sorting by efficiency e. Note that depending on the choice of the efficiency measure, the split interval can also contain variables with integer values. Moreover, sets S_e and C_e can in principle have almost any relation to each other, from inclusion to disjointness. However, for a "reasonable" choice of e they are expected to overlap to a large extent.

If the dual optimal solution values of the LP-relaxation are taken as relevance values, the split interval S_e can be precisely characterized. Let x^{LP} be the optimal solution of the LP-relaxation of MKP.

Theorem 1 *For efficiency values e_j^{duals} there is:*

$$x_j^{\mathrm{LP}} = \begin{cases} 1 & \text{if } e_j > 1\,, \\ \in [0,1] & \text{if } e_j = 1\,, \\ 0 & \text{if } e_j < 1\,. \end{cases} \tag{22}$$

Proof The dual LP associated with the LP-relaxation of MKP is given by

$$\text{(D(MKP))} \qquad \text{minimize} \quad \sum_{i=1}^{m} c_i u_i + \sum_{j=1}^{n} v_j \tag{23}$$

$$\text{subject to} \quad \sum_{i=1}^{m} w_{ij} u_i + v_j \geq p_j, \quad j = 1, \ldots, n \tag{24}$$

$$u_i, v_j \geq 0, \quad i = 1, \ldots, m,\ j = 1, \ldots, n, \tag{25}$$

where u_i are the dual variables corresponding to the knapsack constraints (2) and v_j correspond to the inequalities $x_j \leq 1$. For the optimal primal and dual solutions the following complementary slackness conditions hold for $j = 1, \ldots, n$ (see any textbook on linear programming, e.g. Bertsimas and Tsitsiklis (1997)):

$$x_j \left(\sum_{i=1}^{m} w_{ij} u_i + v_j - p_j \right) = 0 \tag{26}$$

$$v_j (x_j - 1) = 0 \tag{27}$$

Recall that $e_j^{\mathrm{duals}} = \frac{p_j}{\sum_{i=1}^{m} u_i w_{ij}}$. Hence, $e_j > 1$ implies $p_j > \sum_{i=1}^{m} w_{ij} u_i$, which means that (24) can only be fulfilled by $v_j > 0$. Now, (27) immediately yields $x_j = 1$, which proves the first part of the theorem.

If $e_j < 1$, there is $p_j < \sum_{i=1}^{m} w_{ij} u_i$ which together with $v_j \geq 0$ makes the second factor of (26) strictly positive and requires $x_j = 0$. This proves the remainder of the theorem since nothing has to be shown for $e_j = 1$. $\square$

It follows from Theorem 1 that $S_e \subseteq \{j \mid e_j = 1,\ j = 1, \ldots, n\}$. Together with Proposition 1, this means that there exists an optimal solution x^{LP} yielding a split interval with size at most $\min\{m, n\}$.

It should be noted that the theorem gives only a structural result which does not yield an immediate algorithmic advantage in computing the primal solution x^{LP}, since knowing the dual optimal solution is required.

3.4 Experimental Study of MKP Cores and Core Sizes

In order to analyze the core sizes in dependence of different efficiency values, we performed an empirical in-depth examination on smaller instances of Chu and Beasley's benchmark library for which we were able to compute optimal solutions x^* (with $n = 100$ items, $m \in \{5, 10\}$ constraints, and $n = 250$ items, $m = 5$ constraints).

In Table 3 we examine cores generated by using the scaled efficiency e_j^{scaled} as defined in equation (14), the efficiency e_j^{st} as defined in equation (15), the efficiency e_j^{fp} as defined in equations (16) and (17), and finally the efficiency e_j^{duals} setting the relevance values r_i of equation (16) to the optimal dual variable values of the MKP's LP-relaxation. Listed are average values of the sizes of the split interval ($|S_e|$) and of the exact core ($|C_e|$) as a percentage of the number of items n, the percentage of how much the split interval covers the core (ScC) and how much the core covers the split interval (CcS), and the distance between the center of the split interval and the center of the core (C_{dist}) as a percentage of n.

As expected from Theorem 1, the smallest split intervals, consisting of the fractional variables only, are derived with e_j^{duals}. They further yield the smallest cores. Using any of the other efficiency measures results in substantially larger split intervals and observed sizes; coverages, and distances are roughly comparable for them. The smallest distances between the centers of the split intervals and the cores are also obtained with e_j^{duals} for almost all instance classes. The most promising information for devising approximate cores is therefore available from the split intervals generated with e_j^{duals}, on which we will concentrate our further investigations.

Table 3: Relative sizes of split intervals and cores and their mutual coverages and distances for efficiencies e_j^{scaled}, e_j^{st}, e_j^{fp}, and e_j^{duals} (average values over 10 instances per problem class and total averages).

			e_j^{scaled}					e_j^{st}				
n	m	α	$\lvert S_e\rvert$	$\lvert C_e\rvert$	ScC	CcS	C_{dist}	$\lvert S_e\rvert$	$\lvert C_e\rvert$	ScC	CcS	C_{dist}
100	5	0.25	23.40	30.50	72.69	94.71	4.05	27.20	30.20	78.85	88.11	4.80
		0.50	29.50	37.60	71.93	88.45	5.95	27.00	35.60	69.88	89.01	5.90
		0.75	24.30	27.00	72.61	83.13	5.05	22.80	25.20	77.72	84.08	4.30
250	5	0.25	17.44	22.40	77.20	97.38	1.88	17.12	22.20	76.91	94.62	2.46
		0.50	22.88	29.44	71.71	94.25	3.44	23.76	30.88	74.95	94.69	4.04
		0.75	11.44	17.84	56.14	88.45	4.60	11.96	16.64	63.82	85.86	3.62
100	10	0.25	42.60	38.30	92.62	84.39	4.35	43.30	38.20	88.78	79.36	5.55
		0.50	39.40	45.20	80.80	91.20	5.30	44.40	46.50	85.43	88.49	5.65
		0.75	37.50	34.80	94.29	86.42	2.55	38.60	36.20	93.04	87.16	2.10
Average			27.61	31.45	76.67	89.82	4.13	28.46	31.29	78.82	87.93	4.27
			e_j^{fp}					e_j^{duals}				
n	m	α	$\lvert S_e\rvert$	$\lvert C_e\rvert$	ScC	CcS	C_{dist}	$\lvert S_e\rvert$	$\lvert C_e\rvert$	ScC	CcS	C_{dist}
100	5	0.25	24.70	30.10	75.50	91.94	4.20	5.00	20.20	28.12	100.00	3.30
		0.50	27.10	35.80	70.36	89.74	6.35	5.00	22.10	27.49	100.00	3.45
		0.75	23.20	26.10	74.47	84.22	4.55	5.00	19.60	26.95	100.00	3.20
250	5	0.25	16.92	21.72	76.87	95.63	2.24	2.00	12.68	18.16	100.00	2.46
		0.50	22.96	29.68	74.79	95.02	3.56	2.00	12.20	18.45	100.00	1.38
		0.75	11.40	17.12	59.00	87.27	4.06	2.00	10.40	20.18	100.00	1.56
100	10	0.25	42.10	38.20	90.41	83.74	4.75	10.00	23.20	46.57	100.00	2.90
		0.50	41.90	45.60	84.52	90.85	5.15	9.80	25.70	48.17	95.00	3.15
		0.75	37.90	35.30	94.55	86.96	2.40	9.70	18.80	55.74	99.00	2.75
Average			27.58	31.07	77.83	89.49	4.14	5.61	18.32	32.20	99.33	2.68

4. Core-Based Algorithms

After establishing the definition of a core for the MKP and investigating different approximate core sizes, we now concentrate on methods for solving approximate core problems and exploiting them for computing near optimal MKP solutions.

4.1 Exact Solution of Core Problems

In order to evaluate the influence of approximate core sizes on solution quality and run-time, we propose a fixed core size algorithm, where we solve approximate cores using CPLEX 9.0. We performed the experiments on a 2.4 GHz Intel Pentium 4 computer.

In analogy to KP, the approximate core is generated by adding δ items on each side of the center of the split interval which coincides fairly well with the center of the (unknown)

exact core. We created the approximate cores by setting δ to $0.1n$, $0.15n$, $0.2n$, $2m + 0.1n$, and $2m + 0.2n$, respectively. As efficiency measure for the sorting of items from which we determine the approximate cores we used e_j^{duals} (see Section 3.2 for the definition of efficiency measures). The different values of δ were chosen in accordance to the results of the previous section, where an average core size of about $0.2n$ has been observed. Eventual outliers and the distances between the centers of the core and the split interval were the motivation for also considering the larger approximate core sizes. We further used linear combinations of m and n, since the core sizes in general do not depend on the number of items only, but also on the number of constraints. Table 4 lists average solution values and CPU-times for completely solving the original problem, and percentage gaps to these optimal values ($\%_{\text{opt}} = 100 \cdot (z^* - z)/z^*$), numbers of times the optimum was reached (#), as well as average CPU-times as a percentage of the times required for solving the original problem ($\%t$) for the approximate cores of different sizes.

Table 4: Solving approximate cores of different sizes to optimality (average values over 10 instances per problem class and total averages).

n	α	orig. prob.		$\delta = 0.1n$			$\delta = 0.15n$			$\delta = 0.2n$			δ=$2m$+$0.1n$			δ=$2m$+$0.2n$		
m		z	$t[s]$	$\%_{\text{opt}}$	#	$\%t$	$\%_{\text{opt}}$	#	$\%t$	$\%_{\text{opt}}$	#	$\%t$	$\%_{\text{opt}}$	#	$\%t$	$\%_{\text{opt}}$	#	$\%t$
100	0.25	24197	21	0.097	5	1	0.034	7	9	0.015	9	32	0.015	9	32	0.000	10	62
5	0.50	43253	27	0.053	4	1	0.018	6	6	0.002	9	24	0.002	9	24	0.002	9	64
	0.75	60471	6	0.038	5	4	0.021	7	17	0.001	9	39	0.001	9	39	0.000	10	61
250	0.25	60414	1474	0.008	7	36	0.003	9	81	0.000	10	82	0.003	9	69	0.000	10	91
5	0.50	109293	1767	0.002	8	21	0.000	10	63	0.000	10	67	0.000	10	59	0.000	10	73
	0.75	151560	817	0.000	10	17	0.000	10	47	0.000	10	72	0.000	10	40	0.000	10	61
100	0.25	22602	189	0.473	1	0	0.152	4	1	0.002	9	10	0.000	10	46	0.000	10	66
10	0.50	42661	97	0.234	3	0	0.084	5	1	0.030	8	13	0.022	8	60	0.000	10	75
	0.75	59556	29	0.036	6	0	0.015	8	3	0.011	9	22	0.000	10	54	0.000	10	70
Average		63778	492	0.105	5.4	9	0.036	7.3	25	0.007	9.2	40	0.005	9.3	47	0.000	9.9	69

The results of CPLEX applied to cores of different sizes clearly indicate that smaller cores can be solved substantially faster and the obtained solution values are only slightly worse than the optimal ones given by the *orig. prob.* column. The best results concerning run-times were achieved with $\delta = 0.1n$, with which the time could be reduced by factors ranging from 3 to 1000. Despite this strong speedup, the obtained solution values are very close to the respective optima ($\approx 0.1\%$ on average). Solving the larger cores requires more time, but almost all of the optimal solutions can be reached with substantial time savings.

For large MKP instances the exact solution of an approximate core often still consumes

too much time. Therefore, we also consider truncated runs of CPLEX as approximate solution strategies. In our experiments, we used the hardest instances of Chu and Beasley's benchmark library with $n = 500$ items and $m \in \{5, 10, 30\}$ constraints and imposed CPU-time limits of 5, 10, 100, and 500 seconds on the runs. Table 5 lists the following average results over ten instances per problem class: percentage gaps to the optimal solution values of the LP-relaxations ($\%_{\mathrm{LP}}$), standard deviations as subscripts, and numbers of times this core size led to best solutions of these experiments (#).

It can be observed that for all considered time limits, CPLEX applied to approximate cores of any tested size consistently yields better average results than when applied to the original MKP. This has been confirmed by one-sided Wilcoxon signed rank tests yielding error probabilities less than 1% for all considered time limits and approximate core sizes. The best average results for a time limit of 500 seconds are obtained with core sizes of $\delta = 0.2n$. For instances with $m \in \{5, 10\}$ better results are achieved with smaller approximate cores, whereas for $m = 30$ larger approximate cores are usually better. Of course, the number of nodes explored in the branch-and-bound tree increases with decreasing problem/core size.

In order to test the scalability of our approach, we performed further tests on the set of 11 instances proposed by Glover and Kochenberger. These instances range from $n = 100$, $m = 15$ up to $n = 2500$ and $m = 100$. We tested with approximate core sizes $\delta \in \{0.1n, 0.15n, 0.2n, 0.3n\}$ and run-times of 5, 10, 100, and 500 seconds. The results of these experiments are displayed in Table 6. Listed are percentage gaps ($\%_{\mathrm{LP}}$), and the number of investigated branch-and-bound nodes.

For each instance overall best obtained objective values are printed bold. The cases where solving an approximate core led to equally good or better solutions in comparison to the application of CPLEX to the original problem are marked by light- and dark-gray backgrounds, respectively.

These results confirm the trend that optimizing approximate cores in general yields better solutions than optimizing the original problem in the same time. The longer the run-times become, the better the results on the original problems are. The instances of Glover and Kochenberger are occasionally argued to be more complicated to solve due to their correlation between the resouce consumption values. In fact, our results indicate that slightly higher values for the approximate core size tend to be here beneficial for small to medium size instances. In particular, it turns out that a core size of $\delta = 0.1n$ yields rather inferior

Table 5: Solving approximate cores of different sizes with truncated CPLEX (average values over 10 instances per problem class and total averages, $n = 500$, time limits of 5, 10, 100, and 500 seconds).

m	α	orig. prob. $\%_{\text{LP}}$	#	$\delta = 0.1n$ $\%_{\text{LP}}$	#	$\delta = 0.15n$ $\%_{\text{LP}}$	#	$\delta = 0.2n$ $\%_{\text{LP}}$	#
$t[s] = 5$									
5	0.25	$0.146_{0.034}$	2	$\mathbf{0.112}_{0.024}$	9	$0.122_{0.024}$	4	$0.120_{0.023}$	3
	0.5	$0.063_{0.012}$	3	$\mathbf{0.053}_{0.011}$	8	$0.058_{0.010}$	6	$0.060_{0.014}$	6
	0.75	$0.032_{0.008}$	6	$\mathbf{0.030}_{0.006}$	7	$\mathbf{0.030}_{0.006}$	7	$0.031_{0.008}$	7
10	0.25	$0.309_{0.056}$	2	$0.275_{0.030}$	4	$0.280_{0.025}$	4	$\mathbf{0.273}_{0.031}$	5
	0.5	$0.131_{0.017}$	4	$\mathbf{0.120}_{0.018}$	6	$0.126_{0.016}$	3	$0.128_{0.015}$	2
	0.75	$0.090_{0.011}$	3	$\mathbf{0.081}_{0.009}$	5	$\mathbf{0.081}_{0.008}$	5	$0.082_{0.006}$	3
30	0.25	$0.728_{0.078}$	3	$0.710_{0.014}$	2	$0.690_{0.058}$	3	$\mathbf{0.680}_{0.052}$	3
	0.5	$0.316_{0.036}$	3	$0.302_{0.017}$	3	$\mathbf{0.297}_{0.023}$	4	$0.308_{0.021}$	3
	0.75	$0.194_{0.018}$	2	$\mathbf{0.183}_{0.016}$	4	$0.187_{0.016}$	3	$0.185_{0.016}$	4
Average		$0.223_{0.030}$	3.1	$0.207_{0.016}$	5.3	$0.208_{0.021}$	4.3	$0.208_{0.021}$	4.0
$t[s] = 10$									
5	0.25	$0.118_{0.020}$	3	$\mathbf{0.106}_{0.019}$	6	$0.111_{0.018}$	7	$0.113_{0.016}$	4
	0.5	$0.061_{0.013}$	3	$\mathbf{0.045}_{0.007}$	9	$0.049_{0.008}$	6	$0.048_{0.007}$	7
	0.75	$0.032_{0.008}$	5	$0.029_{0.006}$	7	$0.029_{0.006}$	6	$\mathbf{0.028}_{0.005}$	7
10	0.25	$0.295_{0.048}$	2	$\mathbf{0.257}_{0.037}$	5	$0.266_{0.027}$	3	$0.262_{0.033}$	6
	0.5	$0.126_{0.013}$	2	$\mathbf{0.108}_{0.010}$	7	$0.117_{0.011}$	5	$0.118_{0.014}$	4
	0.75	$0.088_{0.010}$	2	$\mathbf{0.077}_{0.006}$	6	$\mathbf{0.077}_{0.007}$	6	$0.079_{0.007}$	5
30	0.25	$0.715_{0.073}$	2	$0.691_{0.041}$	3	$0.686_{0.055}$	3	$\mathbf{0.644}_{0.097}$	2
	0.5	$0.308_{0.027}$	3	$0.295_{0.021}$	3	$\mathbf{0.294}_{0.024}$	4	$0.302_{0.017}$	3
	0.75	$0.181_{0.027}$	3	$\mathbf{0.178}_{0.010}$	2	$0.180_{0.019}$	4	$\mathbf{0.178}_{0.016}$	4
Average		$0.214_{0.026}$	2.8	$0.198_{0.018}$	5.3	$0.201_{0.019}$	4.9	$0.197_{0.024}$	4.7
$t[s] = 100$									
5	0.25	$0.094_{0.019}$	3	$\mathbf{0.082}_{0.012}$	8	$0.086_{0.013}$	7	$0.089_{0.015}$	5
	0.5	$0.044_{0.005}$	4	$\mathbf{0.040}_{0.005}$	7	$0.041_{0.004}$	9	$0.042_{0.005}$	8
	0.75	$0.027_{0.006}$	7	$\mathbf{0.026}_{0.004}$	9	$0.026_{0.004}$	9	$\mathbf{0.026}_{0.004}$	8
10	0.25	$0.221_{0.030}$	4	$0.213_{0.020}$	7	$\mathbf{0.208}_{0.022}$	6	$0.214_{0.026}$	5
	0.5	$0.101_{0.013}$	4	$\mathbf{0.095}_{0.011}$	8	$0.099_{0.011}$	6	$0.099_{0.009}$	6
	0.75	$0.072_{0.008}$	3	$\mathbf{0.068}_{0.008}$	4	$0.069_{0.008}$	4	$0.069_{0.008}$	5
30	0.25	$0.630_{0.051}$	1	$0.646_{0.048}$	0	$0.609_{0.047}$	2	$\mathbf{0.586}_{0.085}$	7
	0.5	$0.271_{0.015}$	4	$0.270_{0.017}$	2	$0.273_{0.012}$	2	$\mathbf{0.265}_{0.019}$	5
	0.75	$0.167_{0.016}$	1	$0.165_{0.013}$	4	$0.170_{0.016}$	3	$\mathbf{0.163}_{0.016}$	4
Average		$0.181_{0.018}$	3.4	$0.178_{0.013}$	5.4	$0.176_{0.015}$	5.3	$0.173_{0.021}$	5.9
$t[s] = 500$									
5	0.25	$0.080_{0.010}$	5	$\mathbf{0.075}_{0.008}$	9	$0.076_{0.008}$	9	$0.076_{0.010}$	8
	0.50	$0.040_{0.005}$	6	$\mathbf{0.039}_{0.005}$	7	$\mathbf{0.039}_{0.005}$	9	$\mathbf{0.039}_{0.006}$	9
	0.75	$0.025_{0.004}$	6	$\mathbf{0.024}_{0.003}$	10	$0.025_{0.004}$	8	$0.025_{0.004}$	8
10	0.25	$0.206_{0.022}$	1	$0.198_{0.021}$	5	$\mathbf{0.195}_{0.023}$	6	$0.198_{0.023}$	4
	0.50	$0.094_{0.013}$	4	$\mathbf{0.088}_{0.009}$	8	$0.090_{0.009}$	6	$0.092_{0.012}$	5
	0.75	$0.066_{0.009}$	4	$0.065_{0.009}$	5	$\mathbf{0.064}_{0.007}$	7	$0.065_{0.008}$	7
30	0.25	$0.598_{0.038}$	2	$0.621_{0.034}$	0	$0.566_{0.049}$	4	$\mathbf{0.537}_{0.061}$	6
	0.50	$0.258_{0.009}$	2	$0.246_{0.021}$	3	$\mathbf{0.243}_{0.027}$	4	$0.250_{0.024}$	2
	0.75	$0.158_{0.013}$	2	$\mathbf{0.151}_{0.013}$	6	$0.160_{0.011}$	1	$\mathbf{0.151}_{0.013}$	5
Average		$0.169_{0.013}$	3.6	$0.167_{0.014}$	5.9	$0.162_{0.016}$	6.0	$0.159_{0}.018$	6.0

results for smaller instances (≤ 200 items). In contrast, however, for larger instances and in particular longer run-times, $\delta = 0.1n$ seems to be very well suited again.

Considering for $\delta \in \{0.15n, 0.2n, 0.3n\}$ and each run-time the number of instances (out of 11) where the original problem yielded better solutions than the core problem, it turns out that this number is usually only 1, sometimes 2, but at most 3. The same holds for the smaller core size $\delta = 0.1n$ applied to larger instances.

4.2 Heuristic Solution of Core Problems by a Memetic Algorithm

As an alternative to truncated runs of CPLEX, we now consider the application of a metaheuristic for heuristically solving core problems of large MKP instances within reasonable time. From another perspective, this approach can also be seen as a study of how the reduction to MKP cores influences the performance of metaheuristics. The hope is that the core concept enables us also in this case to find better solutions within given time limits.

We consider a state-of-the-art memetic algorithm (MA) for solving the MKP and again apply it to differently sized approximate cores. The MA is based on Chu and Beasley's principles and includes some improvements suggested in Raidl (1998); Gottlieb (1999); Raidl and Gottlieb (2005). The framework is steady-state and the creation of initial solutions is guided by the solution to the LP-relaxation of the MKP, as described in Gottlieb (1999). Each new candidate solution is derived by selecting two parents via binary tournaments, performing uniform crossover on their characteristic vectors x, flipping each bit with probability $1/n$, performing repair if a capacity constraint is violated, and always applying local improvement. If such a new candidate solution is different from all solutions in the current population, it replaces the worst of them.

Both, repair and local improvement are based on greedy first-fit strategies and guarantee that any resulting candidate solution lies at the boundary of the feasible region, on which optimal solutions are always located. The repair procedure considers all items in a specific order Π and removes selected items ($x_j = 1 \;\rightarrow\; x_j = 0$) as long as any knapsack constraint is violated. Local improvement works vice-versa: It considers all items in the reverse order $\overline{\Pi}$ and selects items not yet appearing in the solution as long as no capacity limit is exceeded.

Crucial for these strategies to work well is the choice of the ordering Π. Items that are likely to be selected in an optimal solution must appear near the end of Π. Following the results of Section 3.4 it is most promising to determine Π by ordering the items according to efficiency measure e_j^{duals}, as it has already been suggested in Chu and Beasley (1998).

Table 6: Solving approximate cores of different sizes with truncated CPLEX, Glover Kochenberger benchmark set (time limits of 5, 10, 100, and 500 seconds).

n	m	orig. prob.		$\delta = 0.1n$		$\delta = 0.15n$		$\delta = 0.2n$		$\delta = 0.3n$	
		Obj	*Nnodes*	Obj	*Nnodes*	Obj	*Nnodes*	Obj	*Nnodes*	Obj	*Nnodes*
$t[s] = 5$											
100	15	**0.263**	29557	0.554	1007	0.501	9268	**0.263**	2725	**0.263**	24016
100	25	0.583	13276	0.810	747	**0.533**	1719	**0.533**	15322	**0.533**	19042
150	25	**0.398**	7963	0.486	10104	**0.398**	26554	0.415	17623	**0.398**	13166
150	50	0.668	2878	1.117	3049	0.582	13892	0.599	8078	**0.565**	4673
200	25	0.295	5731	0.308	28759	**0.255**	16696	**0.255**	13155	0.295	10072
200	50	0.560	1985	0.521	17882	0.430	8256	**0.404**	5026	0.508	3318
500	25	0.106	1655	**0.101**	11271	0.106	7961	0.106	5502	**0.101**	3391
500	50	0.201	485	0.201	3248	0.243	2407	**0.185**	1619	0.206	966
1500	25	0.037	289	0.032	3191	0.039	2048	0.037	1366	**0.030**	701
1500	50	0.077	2	0.065	859	**0.060**	507	0.067	319	0.081	133
2500	100	0.112	0	0.086	0	**0.066**	0	0.075	0	0.084	0
$t[s] = 10$											
100	15	**0.263**	29557	0.554	1007	0.501	9268	**0.263**	2725	**0.263**	24016
100	25	0.583	27393	0.810	747	0.533	1719	0.533	15322	**0.508**	40401
150	25	**0.380**	16732	0.486	10104	0.398	56984	0.398	36078	0.398	26561
150	50	0.668	6339	1.117	3049	**0.565**	29870	0.582	17641	**0.565**	10040
200	25	0.295	12001	0.295	72445	0.255	34233	0.255	26744	**0.242**	20595
200	50	0.469	4334	0.521	38558	0.430	17412	**0.404**	10756	0.508	7126
500	25	0.101	4178	0.101	23031	0.106	16733	0.096	11542	**0.080**	7625
500	50	0.201	1321	0.201	7216	0.243	5330	**0.185**	3803	0.206	2424
1500	25	0.037	865	0.032	6653	0.039	4857	0.037	3351	**0.030**	1879
1500	50	0.077	199	0.065	2001	**0.060**	1358	0.067	954	0.065	551
2500	100	0.082	0	0.086	41	**0.066**	0	0.068	0	0.076	0
$t[s] = 100$											
100	15	**0.263**	29557	0.554	1007	0.501	9268	**0.263**	2725	**0.263**	24016
100	25	0.533	309993	0.810	747	0.533	1719	0.533	15322	**0.458**	195889
150	25	0.380	180789	0.486	10104	0.380	190920	0.345	399922	**0.327**	272864
150	50	0.565	68090	1.117	3049	0.513	191459	**0.461**	211827	0.565	109941
200	25	**0.242**	121883	0.281	79609	**0.242**	378119	0.255	277355	**0.242**	206943
200	50	0.456	49095	0.508	132496	0.430	205375	**0.404**	119730	0.430	78021
500	25	0.096	48584	0.090	229000	0.096	170792	0.090	121344	**0.080**	82100
500	50	0.201	18763	0.195	84821	0.190	57975	0.185	44511	**0.174**	30438
1500	25	0.034	15362	**0.025**	81001	0.034	54664	0.032	40921	0.030	27673
1500	50	0.067	5409	0.065	28373	**0.060**	19029	**0.060**	14990	0.063	10210
2500	100	**0.063**	201	0.069	4554	0.066	2299	0.066	1041	0.069	950
$t[s] = 500$											
100	15	**0.263**	29557	0.554	1007	0.501	9268	**0.263**	2725	**0.263**	24016
100	25	**0.458**	944476	0.810	747	0.533	1719	0.533	15322	**0.458**	195889
150	25	**0.257**	929892	0.486	10104	0.380	190920	0.345	2094931	0.327	1333451
150	50	0.530	353671	1.117	3049	0.513	191459	**0.461**	1576936	0.496	581966
200	25	**0.242**	612130	0.281	79609	**0.242**	2148333	0.255	1347741	**0.242**	1017448
200	50	0.417	245609	0.508	132496	**0.404**	1097270	**0.404**	646145	**0.404**	390232
500	25	0.096	243026	0.090	1126323	0.075	848853	0.080	603189	**0.070**	400001
500	50	**0.169**	96813	**0.169**	456310	0.190	291099	**0.169**	224554	0.174	159790
1500	25	0.032	83251	0.025	393755	0.027	264968	**0.023**	195736	0.025	142998
1500	50	0.067	31623	**0.049**	145821	0.056	98285	0.060	80401	0.056	57045
2500	100	0.062	4683	**0.057**	26596	0.058	17219	0.060	10283	0.059	8833

As in case of truncated CPLEX, CPU-time limits of 5, 10, 100, and 500 seconds were imposed on the MA. Since the MA usually converges much earlier, it has been restarted every 1 000 000 iterations, always keeping the so-far best solution in the population. The hardest benchmark instances of Chu and Beasley's library with $n = 500$ were used. The population size was 100. Table 7 shows average results of the MA applied to the original problem and to approximate cores of different sizes. Similarly as for the truncated CPLEX experiments, listed are percentage gaps ($\%_{\mathrm{LP}}$), standard deviations as subscripts, and numbers of times each core size yielded the best solutions of these experiments (#).

As observed with truncated CPLEX, the use of approximate cores also leads in case of the MA consistently to better average solution qualities for all tested time limits. This was confirmed by one-sided Wilcoxon signed rank tests yielding error probabilities of less than 0.001% for all tested time limits, except for $t = 500$, where the error probabilities are less than 1%. Obviously, the core size has a substantial influence on the number of iterations the MA can carry out within the allowed time: With the core sizes of $0.1n$, $0.15n$, and $0.2n$, about 3.6, 2.9, and 2.3 times more iterations were performed than when applying the MA to the original problem.

The larger number of candidate solutions the MA can examine when it is restricted to a core problem seems to be one reason for the usually better final solutions. Most of the best results for all considered run-time limits were obtained with $\delta = 0.1n$ and $\delta = 0.15n$, thus, with relatively small approximate cores.

In summary, we applied CPLEX and a MA to approximate cores of hard to solve benchmark instances and observed that using approximate cores of fixed size instead of the original problem clearly and consistently improves the average solution quality when using different time limits between 5 and 500 seconds.

5. Collaborative Approaches

So far, we have individually looked at ILP-based and metaheuristic methods for solving the MKP and corresponding approximate core problems. Now, we consider a hybrid architecture in which the ILP-based approach and the MA are performed (quasi-)parallel and continuously exchange information in a bidirectional asynchronous way. In general, such hybrid systems have drawn much attention over the recent years since they often significantly outperform the individual "pure" approaches; see e.g. Puchinger and Raidl (2005) for an overview.

Table 7: Solving approximate cores of different sizes with the MA (average values over 10 instances per problem class and total average values, $n = 500$, time limits of 5, 10, 100, and 500 seconds).

m	α	orig. prob. $\%_{\text{LP}}$	#	$\delta = 0.1n$ $\%_{\text{LP}}$	#	$\delta = 0.15n$ $\%_{\text{LP}}$	#	$\delta = 0.2n$ $\%_{\text{LP}}$	#
$t[s] = 5$									
5	0.25	$0.099_{0.015}$	2	$\mathbf{0.091}_{0.006}$	6	$0.092_{0.008}$	5	$0.097_{0.014}$	4
	0.5	$0.050_{0.009}$	1	$\mathbf{0.043}_{0.006}$	6	$0.044_{0.006}$	5	$\mathbf{0.043}_{0.004}$	5
	0.75	$0.031_{0.004}$	2	$\mathbf{0.026}_{0.002}$	7	$\mathbf{0.026}_{0.003}$	7	$0.029_{0.005}$	5
10	0.25	$0.269_{0.034}$	1	$0.245_{0.025}$	4	$\mathbf{0.242}_{0.015}$	4	$0.243_{0.027}$	5
	0.5	$0.120_{0.015}$	3	$\mathbf{0.112}_{0.008}$	5	$0.113_{0.011}$	7	$0.115_{0.012}$	5
	0.75	$0.078_{0.008}$	3	$\mathbf{0.073}_{0.007}$	8	$0.074_{0.006}$	5	$0.075_{0.007}$	4
30	0.25	$0.696_{0.053}$	1	$0.656_{0.028}$	4	$0.652_{0.051}$	2	$\mathbf{0.648}_{0.055}$	5
	0.5	$0.290_{0.020}$	1	$0.285_{0.019}$	2	$\mathbf{0.273}_{0.027}$	8	$0.284_{0.025}$	2
	0.75	$0.177_{0.018}$	3	$\mathbf{0.170}_{0.014}$	7	$0.172_{0.018}$	5	$0.176_{0.017}$	4
Average		$0.201_{0.019}$	1.9	$0.189_{0.013}$	5.4	$0.188_{0.016}$	5.3	$0.190_{0.018}$	4.3
$t[s] = 10$									
5	0.25	$0.103_{0.030}$	0	$\mathbf{0.081}_{0.014}$	6	$0.088_{0.014}$	3	$0.088_{0.015}$	5
	0.5	$0.047_{0.011}$	1	$\mathbf{0.041}_{0.004}$	8	$0.043_{0.006}$	2	$0.044_{0.006}$	3
	0.75	$0.030_{0.005}$	2	$\mathbf{0.026}_{0.003}$	8	$0.027_{0.003}$	6	$\mathbf{0.026}_{0.003}$	7
10	0.25	$0.264_{0.038}$	1	$\mathbf{0.226}_{0.020}$	6	$0.235_{0.025}$	3	$0.241_{0.018}$	3
	0.5	$0.116_{0.011}$	2	$\mathbf{0.107}_{0.011}$	8	$\mathbf{0.107}_{0.009}$	6	$0.110_{0.013}$	4
	0.75	$0.074_{0.007}$	3	$0.072_{0.005}$	6	$\mathbf{0.070}_{0.006}$	9	$0.075_{0.006}$	3
30	0.25	$0.655_{0.045}$	2	$0.627_{0.049}$	4	$\mathbf{0.609}_{0.079}$	7	$0.631_{0.058}$	4
	0.5	$0.290_{0.025}$	1	$0.274_{0.019}$	4	$\mathbf{0.264}_{0.025}$	5	$0.267_{0.031}$	5
	0.75	$0.171_{0.015}$	3	$\mathbf{0.166}_{0.013}$	4	$0.171_{0.015}$	3	$0.167_{0.015}$	6
Average		$0.195_{0.018}$	1.7	$0.180_{0.015}$	6.0	$0.179_{0.020}$	4.9	$0.183_{0.019}$	4.4
$t[s] = 100$									
5	0.25	$0.085_{0.011}$	3	$\mathbf{0.075}_{0.009}$	9	$0.077_{0.008}$	8	$0.079_{0.009}$	5
	0.5	$0.042_{0.006}$	3	$\mathbf{0.039}_{0.004}$	8	$0.040_{0.004}$	8	$0.040_{0.004}$	5
	0.75	$0.026_{0.002}$	3	$\mathbf{0.024}_{0.003}$	9	$\mathbf{0.024}_{0.003}$	9	$0.025_{0.004}$	6
10	0.25	$0.230_{0.016}$	0	$\mathbf{0.209}_{0.023}$	7	$0.216_{0.011}$	5	$0.216_{0.015}$	5
	0.5	$0.101_{0.011}$	3	$\mathbf{0.099}_{0.007}$	5	$0.097_{0.011}$	6	$0.100_{0.009}$	4
	0.75	$0.069_{0.006}$	5	$\mathbf{0.067}_{0.006}$	7	$0.068_{0.006}$	5	$0.069_{0.006}$	5
30	0.25	$0.624_{0.042}$	3	$0.595_{0.073}$	5	$\mathbf{0.593}_{0.072}$	4	$0.602_{0.076}$	3
	0.5	$0.272_{0.019}$	0	$0.257_{0.014}$	7	$0.262_{0.022}$	3	$\mathbf{0.256}_{0.019}$	6
	0.75	$0.166_{0.013}$	3	$\mathbf{0.158}_{0.012}$	7	$0.161_{0.011}$	5	$\mathbf{0.158}_{0.010}$	7
Average		$0.180_{0.014}$	2.6	$0.169_{0.017}$	7.1	$0.171_{0.016}$	5.9	$0.172_{0.017}$	5.1
$t[s] = 500$									
5	0.25	$0.078_{0.011}$	6	$\mathbf{0.073}_{0.008}$	10	$0.074_{0.008}$	9	$0.074_{0.008}$	9
	0.50	$0.040_{0.004}$	6	$\mathbf{0.039}_{0.004}$	9	$\mathbf{0.039}_{0.004}$	9	$0.040_{0.004}$	7
	0.75	$0.025_{0.003}$	7	$\mathbf{0.024}_{0.003}$	9	$\mathbf{0.024}_{0.003}$	10	$\mathbf{0.024}_{0.003}$	9
10	0.25	$0.208_{0.019}$	5	$\mathbf{0.202}_{0.017}$	5	$\mathbf{0.202}_{0.013}$	6	$0.208_{0.018}$	4
	0.50	$0.099_{0.007}$	2	$0.093_{0.007}$	6	$\mathbf{0.091}_{0.010}$	8	$0.093_{0.009}$	5
	0.75	$0.066_{0.005}$	6	$\mathbf{0.065}_{0.005}$	8	$0.067_{0.005}$	4	$0.068_{0.006}$	4
30	0.25	$0.604_{0.046}$	1	$0.573_{0.078}$	5	$0.575_{0.063}$	5	$\mathbf{0.569}_{0.068}$	6
	0.50	$0.254_{0.021}$	3	$0.257_{0.015}$	1	$\mathbf{0.246}_{0.019}$	7	$0.253_{0.021}$	3
	0.75	$0.159_{0.012}$	4	$\mathbf{0.156}_{0.013}$	5	$0.157_{0.012}$	3	$0.157_{0.011}$	5
Average		$0.170_{0.014}$	4.4	$0.165_{0.017}$	6.4	$0.164_{0.015}$	6.8	$0.165_{0.017}$	5.8

The basic concept is to run CPLEX and the MA from Section 4.2 in parallel on two individual machines/CPUs or in a pseudo-parallel way as individual processes on a single machine. Inter-process communication takes place over standard TCP/IP-socket connections. In the experiments documented in the following, we only used a single-CPU 2.4 GHz Intel Pentium 4 PC. The two processes are started at the same time and their pseudo-parallel execution is handled by the operating system.

We consider primal and dual information to be exchanged between the algorithms, as explained in the next section. Different variants of the collaborative combination are experimentally compared for the complete MKP in Section 5.2. In Section 5.3, we put our attention back to core problems of different sizes, also approaching them by collaboration.

5.1 Information Exchange

If a new so-far best solution is encountered by one of the algorithms, it is immediately sent to the partner. If the MA receives such a solution, it is included into its population by replacing the worst solution, as in the case of any other newly created solution candidate. In CPLEX, a received solution is set as new incumbent solution, providing a new global lower bound, possibly enabling the fathoming of further B&B tree nodes.

Additionally, when CPLEX finds a new incumbent solution, it also sends the current dual variable values associated to the knapsack constraints, which are devised from the LP-relaxation of the node in the B&B tree currently being processed, to the MA. When receiving these dual variable values, the MA recalculates the efficiencies and the item ordering Π for repair and local improvement as described in Section 4.2.

5.2 Applying Collaborative Approaches to the MKP

The computational experiments were performed, as before, with a total CPU-time limit of 500 seconds. The MA and CPLEX were started at the same time and were each given 250 seconds (*equal* case), or running time was assigned by a 2:1 ratio, terminating the MA after 167 seconds and performing CPLEX with a time-limit of 333 seconds (*skewed* case). We studied these two variants, since preliminary tests with the cooperative approach suggested that the MA often was the main contributor in finding improved solutions during the early stages of the optimization process.

Table 8 compares the results of the independent application of CPLEX and the MA to those of equal and skewed cooperation. Regarding information exchange, we further

differentiate between the case where only so far best solutions are exchanged (CPLEX_MA) and the case where additionally also dual variable values are sent from CPLEX to the MA (CPLEX_MA_D). Subscripts again display standard deviations.

Results indicate a small but significant advantage for the cooperative strategies. Use of the skewed collaboration scheme and the additional exchange of dual variable values improved the solution quality obtained on average for almost all instance classes. Interestingly, when the MA was executed independently, it achieved the highest number of best solutions obtained, whereas it yielded on average the worst solution quality. The best average solution quality and the second highest number of obtained best solutions is achieved with CPLEX_MA_D using the skewed cooperation strategy. A one-sided Wilcoxon signed rank test showed that in general CPLEX_MA_D/skewed achieves better results than CPLEX with an error probability of 1.5% and better results than the MA with an error probability of 5.3%.

Table 8: Results of collaborative strategies (average values over 10 instances per problem class and total averages, $n = 500$).

		No Cooperation				Equal Cooperation				Skewed Cooperation			
		CPLEX		MA		CPLEX_MA		CPLEX_MA_D		CPLEX_MA		CPLEX_MA_D	
m	α	$\%_{\text{LP}}$	#	$\%_{\text{LP}}$	#	$\%_{\text{LP}}$	#	$\%_{\text{LP}}$	#	$\%_{\text{LP}}$	#	$\%_{\text{LP}}$	#
5	0.25	$0.080_{0.010}$	6	$0.078_{0.011}$	7	$0.079_{0.010}$	6	$\mathbf{0.077}_{0.010}$	7	$0.078_{0.010}$	6	$0.078_{0.008}$	8
	0.50	$0.040_{0.005}$	7	$0.040_{0.004}$	9	$0.041_{0.004}$	5	$0.041_{0.003}$	5	$\mathbf{0.039}_{0.005}$	8	$\mathbf{0.039}_{0.005}$	10
	0.75	$\mathbf{0.025}_{0.004}$	7	$\mathbf{0.025}_{0.003}$	8	$\mathbf{0.025}_{0.004}$	8	$\mathbf{0.025}_{0.004}$	7	$\mathbf{0.025}_{0.004}$	7	$\mathbf{0.025}_{0.004}$	7
10	0.25	$0.206_{0.022}$	4	$0.208_{0.019}$	5	$0.207_{0.016}$	2	$0.203_{0.022}$	5	$0.205_{0.015}$	3	$\mathbf{0.199}_{0.015}$	5
	0.50	$0.094_{0.013}$	5	$0.099_{0.007}$	3	$\mathbf{0.093}_{0.012}$	5	$0.098_{0.009}$	2	$0.095_{0.012}$	3	$0.096_{0.013}$	4
	0.75	$\mathbf{0.066}_{0.009}$	4	$\mathbf{0.066}_{0.005}$	4	$0.067_{0.008}$	3	$0.067_{0.007}$	4	$\mathbf{0.066}_{0.008}$	5	$\mathbf{0.066}_{0.008}$	2
30	0.25	$0.598_{0.038}$	3	$0.604_{0.046}$	3	$0.594_{0.035}$	3	$0.596_{0.034}$	3	$0.592_{0.039}$	3	$\mathbf{0.574}_{0.065}$	5
	0.50	$0.258_{0.009}$	2	$\mathbf{0.254}_{0.021}$	5	$0.257_{0.012}$	4	$0.255_{0.009}$	4	$\mathbf{0.254}_{0.019}$	3	$0.257_{0.011}$	4
	0.75	$0.158_{0.013}$	3	$0.159_{0.012}$	6	$0.158_{0.011}$	3	$\mathbf{0.156}_{0.011}$	7	$0.158_{0.010}$	2	$\mathbf{0.156}_{0.011}$	4
Average		$0.169_{0.013}$	4.6	$0.170_{0.014}$	5.6	$0.169_{0.012}$	4.3	$0.169_{0.012}$	4.9	$0.168_{0.013}$	4.4	$0.166_{0.015}$	5.4

In our next experiments, we focused the optimization of CPLEX to the more promising part of the search space in the neighborhood of the solution to the LP-relaxation, as we did already in Section 2.2. The local branching constraint (10) is added to the ILP formulation (1)–(3) of the MKP, and only if this restricted problem could be solved to optimality within the time-limit, CPLEX continues with the remainder of the problem. Table 9 shows the results of the collaboration between the MA and this locally constrained CPLEX (LC). The neighborhood size parameter k was set to 25, which yielded on average the best results in Section 2.2. Again, we considered the variants with exchange of so far best solutions only (LC_MA) and with the additional exchange of dual variable values (LC_MA_D). Both cases were tested with equal and skewed cooperation strategies. For comparison purposes we also list results of LC and the MA performed alone. Subscripts have the same meanings as before.

Table 9: Results of collaborative strategies with locally constraint CPLEX (average values over 10 instances per problem class and total averages, $n = 500$).

		No Cooperation				Equal Cooperation				Skewed Cooperation			
		LC		MA		LC_MA		LC_MA_D		LC_MA		LC_MA_D	
m	α	$\%_{\text{LP}}$	#	$\%_{\text{LP}}$	#	$\%_{\text{LP}}$	#	$\%_{\text{LP}}$	#	$\%_{\text{LP}}$	#	$\%_{\text{LP}}$	#
5	0.25	$0.080_{0.009}$	6	$0.078_{0.011}$	7	$0.077_{0.010}$	7	$0.080_{0.011}$	7	$\mathbf{0.075}_{0.007}$	10	$0.078_{0.010}$	7
	0.50	$\mathbf{0.039}_{0.005}$	9	$0.040_{0.004}$	8	$0.040_{0.004}$	7	$\mathbf{0.039}_{0.005}$	7	$\mathbf{0.039}_{0.004}$	8	$\mathbf{0.039}_{0.005}$	8
	0.75	$\mathbf{0.025}_{0.004}$	7	$\mathbf{0.025}_{0.003}$	7	$\mathbf{0.025}_{0.003}$	9	$\mathbf{0.025}_{0.003}$	6	$\mathbf{0.025}_{0.004}$	7	$\mathbf{0.025}_{0.004}$	9
10	0.25	$0.206_{0.022}$	3	$0.208_{0.019}$	5	$0.206_{0.017}$	2	$\mathbf{0.200}_{0.019}$	5	$0.202_{0.012}$	3	$0.202_{0.021}$	4
	0.50	$0.095_{0.014}$	3	$0.099_{0.007}$	2	$0.095_{0.012}$	4	$\mathbf{0.092}_{0.013}$	6	$\mathbf{0.092}_{0.013}$	5	$0.094_{0.011}$	4
	0.75	$0.066_{0.008}$	4	$0.066_{0.005}$	5	$0.067_{0.006}$	5	$0.066_{0.008}$	5	$0.066_{0.007}$	5	$\mathbf{0.065}_{0.008}$	6
30	0.25	$\mathbf{0.555}_{0.067}$	7	$0.604_{0.046}$	3	$0.594_{0.041}$	4	$0.607_{0.039}$	2	$0.591_{0.036}$	2	$0.571_{0.067}$	5
	0.50	$0.257_{0.012}$	2	$0.254_{0.021}$	3	$\mathbf{0.251}_{0.019}$	6	$0.257_{0.012}$	3	$\mathbf{0.251}_{0.010}$	5	$0.260_{0.012}$	2
	0.75	$0.155_{0.011}$	6	$0.159_{0.012}$	4	$0.156_{0.011}$	5	$\mathbf{0.154}_{0.010}$	6	$0.156_{0.09}$	5	$0.155_{0.011}$	8
Average		$0.164_{0.017}$	5.2	$0.170_{0.014}$	4.9	$0.168_{0.014}$	5.4	$0.169_{0.013}$	5.2	$0.166_{0.011}$	5.6	$0.165_{0.016}$	5.9

These results do not allow clear conclusions. The best average solution quality over all instance classes is observed when using LC alone; however, a one-sided Wilcoxon signed rank test showed that this difference is statistically insignificant. This result primarily comes from the extraordinary good solutions obtained for $m = 30$, $\alpha = 0.25$. For the remaining instance classes, LC_MA_D provides better or equal results, which can be seen by the highest average number of times it obtained the best solutions. Again the skewed collaboration strategy provides slightly better results than the equal strategy.

5.3 Applying Collaborative Approaches to MKP Cores

Table 10 shows results of the collaboration between the MA and CPLEX applied to approximate core problems of different sizes ($\delta = 0.15n$ and $\delta = 0.2n$); efficiency measure e_j^{duals} was used. We list the results for the variant where so far best solutions and dual variable values are exchanged with the skewed cooperation strategy (CPLEX_MA_D). For comparison the table also contains the results of the independently performed CPLEX and MA.

When solving approximate cores of different sizes, the cooperative approach cannot always improve average results of the individual algorithms. Considering the core size $\delta = 0.15n$, the collaborative approach dominates the individual algorithms, as confirmed by a one-sided Wilcoxon signed rank test yielding an error probability of less than 5%. In case of $\delta = 0.2n$ results are not as clear anymore, since restricting the search space to cores enables the individual algorithms to find very high quality solutions.

Table 10: Results of collaborative strategies applied to MKP cores of different sizes (average values over 10 instances per problem class and total averages, $n = 500$).

		CPLEX				MA				CPLEX_MA_D			
		$\delta = 0.15n$		$\delta = 0.2n$		$\delta = 0.15n$		$\delta = 0.2n$		$\delta = 0.15n$		$\delta = 0.2n$	
m	α	$\%_{\text{LP}}$	#	$\%_{\text{LP}}$	#	$\%_{\text{LP}}$	#	$\%_{\text{LP}}$	#	$\%_{\text{LP}}$	#	$\%_{\text{LP}}$	#
5	0.25	$0.076_{0.008}$	6	$0.076_{0.010}$	6	$\mathbf{0.074}_{0.008}$	8	$\mathbf{0.074}_{0.008}$	8	$0.076_{0.011}$	6	$0.075_{0.007}$	7
	0.50	$\mathbf{0.039}_{0.005}$	8	$\mathbf{0.039}_{0.006}$	7	$\mathbf{0.039}_{0.004}$	8	$0.040_{0.004}$	7	$\mathbf{0.039}_{0.005}$	8	$\mathbf{0.039}_{0.005}$	7
	0.75	$0.025_{0.004}$	8	$0.025_{0.004}$	8	$\mathbf{0.024}_{0.003}$	9	$\mathbf{0.024}_{0.003}$	8	$\mathbf{0.024}_{0.003}$	8	$0.025_{0.004}$	8
10	0.25	$\mathbf{0.195}_{0.023}$	6	$0.198_{0.023}$	5	$0.202_{0.013}$	3	$0.208_{0.018}$	1	$0.197_{0.022}$	5	$0.202_{0.019}$	3
	0.50	$0.090_{0.009}$	5	$0.092_{0.012}$	4	$0.091_{0.010}$	4	$0.093_{0.009}$	3	$\mathbf{0.088}_{0.009}$	7	$0.089_{0.009}$	6
	0.75	$\mathbf{0.064}_{0.007}$	7	$0.065_{0.008}$	8	$0.067_{0.005}$	2	$0.068_{0.006}$	1	$0.065_{0.007}$	5	$0.065_{0.008}$	6
30	0.25	$0.566_{0.049}$	3	$\mathbf{0.537}_{0.061}$	5	$0.575_{0.063}$	2	$0.569_{0.068}$	3	$0.549_{0.070}$	5	$0.548_{0.064}$	3
	0.50	$0.243_{0.027}$	4	$0.250_{0.024}$	2	$0.246_{0.019}$	2	$0.253_{0.021}$	1	$\mathbf{0.241}_{0.029}$	3	$0.246_{0.025}$	3
	0.75	$0.160_{0.011}$	0	$\mathbf{0.151}_{0.013}$	6	$0.157_{0.012}$	2	$0.157_{0.011}$	1	$0.154_{0.009}$	2	$0.154_{0.010}$	4
Average		$0.162_{0.016}$	5.2	$0.159_{0.018}$	5.7	$0.164_{0.015}$	4.4	$0.165_{0.017}$	3.7	$0.159_{0.018}$	5.4	$0.160_{0.017}$	5.2

6. Comparison to Currently Best Solutions

In order to compare the approaches we developed to the tabu search based approach from Vasquez and Vimont (2005), which yielded the best known results for the benchmark instances used, we tested some of our methods on a dual AMD Opteron 250 machine with 2.4 GHz, with total CPU-times of 1800 seconds.

Table 11 lists the results of Vasquez and Vimont (2005), CPLEX without additional constraints (CPLEX), CPLEX applied to approximate cores generated with e_j^{duals} and $\delta = 0.25n$ (CPLEX C), CPLEX_MA_D applied to the same cores with the skewed cooperation strategy (CPLEX_MA_D C s), and finally a version with the equal cooperation strategy and 7200 seconds per algorithm (CPLEX_MA_D C el). Since we used a dual-processor machine, the parallel approaches were actually executed in parallel, and wall-clock times of about 900, 1200, and 7200 seconds, respectively, were needed per instance. Shown are average percentage gaps to the optimal objective values of the LP-relaxations ($\%_{\text{LP}}$), corresponding standard deviations as subscripts, the number of times this algorithm yielded the best solution for these experiments (#), and for Vasquez and Vimont (2005) we further display the average running times in seconds on an Intel Pentium 4 computer with 2 GHz. The detailed results of this experiment can be found in the on-line supplement accompanying this paper.

With respect to our approaches, the obtained results provide a similar overall picture as in the previous sections. The parallel approach with the skewed cooperation strategy can slightly improve the results of the individual algorithms. Our approaches yield mostly similar and sometimes even better solutions than the tabu search from Vasquez and Vimont (2005) for the $m = 5$ class, with substantially shorter running times. For the classes with

Table 11: Solving the MKP with different variants and total CPU times of 1800 seconds per instance, for the last column of 14400 seconds, compared to best known approach (average values over 10 instances per problem class and total averages, $n = 500$).

		Vasquez and Vimont (2005)			CPLEX		CPLEX C		CPLEX_MA_D C s		CPLEX_MA_D C el	
m	α	$\%_{\text{LP}}$	#	$t[s]$	$\%_{\text{LP}}$	#	$\%_{\text{LP}}$	#	$\%_{\text{LP}}$	#	$\%_{\text{LP}}$	#
5	0.25	$0.074_{0.010}$	8	47469	$0.073_{0.008}$	9	$0.073_{0.008}$	9	$0.073_{0.008}$	9	$\mathbf{0.072}_{0.009}$	10
	0.50	$\mathbf{0.038}_{0.005}$	7	20486	$\mathbf{0.038}_{0.005}$	10	$\mathbf{0.038}_{0.005}$	10	$\mathbf{0.038}_{0.005}$	9	$\mathbf{0.038}_{0.005}$	10
	0.75	$\mathbf{0.024}_{0.003}$	10	24883	$\mathbf{0.024}_{0.003}$	8	$\mathbf{0.024}_{0.003}$	9	$\mathbf{0.024}_{0.003}$	9	$\mathbf{0.024}_{0.003}$	10
10	0.25	$\mathbf{0.174}_{0.016}$	9	34964	$0.190_{0.020}$	2	$0.189_{0.019}$	2	$0.185_{0.019}$	3	$0.179_{0.017}$	4
	0.50	$0.082_{0.004}$	8	26333	$0.087_{0.012}$	4	$0.083_{0.011}$	6	$0.082_{0.009}$	6	$\mathbf{0.080}_{0.008}$	8
	0.75	$\mathbf{0.057}_{0.009}$	10	21156	$0.063_{0.007}$	2	$0.061_{0.008}$	3	$0.061_{0.007}$	3	$0.058_{0.009}$	7
30	0.25	$\mathbf{0.482}_{0.045}$	10	97234	$0.546_{0.052}$	1	$0.544_{0.040}$	1	$0.534_{0.057}$	3	$0.502_{0.055}$	3
	0.50	$\mathbf{0.210}_{0.015}$	10	113418	$0.237_{0.022}$	0	$0.234_{0.017}$	0	$0.235_{0.016}$	0	$0.229_{0.016}$	0
	0.75	$\mathbf{0.135}_{0.008}$	10	148378	$0.150_{0.011}$	0	$0.147_{0.013}$	2	$0.148_{0.012}$	1	$0.143_{0.010}$	3
Average		$0.142_{0.013}$	9.1	59369	$0.156_{0.016}$	4.0	$0.155_{0.014}$	4.7	$0.153_{0.015}$	4.8	$0.147_{0.015}$	6.1

$m \in \{10, 30\}$ the results provided in Vasquez and Vimont (2005) are mostly better than those of our approach in terms of solution quality, but at much higher computational costs.

The results achieved by our parallel approach with 14400 seconds of total CPU-time are slightly better than those of Vasquez and Vimont (2005) for $m = 5$. For $m = 10$ our results are sometimes slightly worse than the state-of-the-art, whereas for the instances with $m = 30$ we were not able to obtain the best known solutions. Most of them are achieved by the approach proposed in Vasquez and Vimont (2005). However, the main drawback of this approach is its huge running time of more than 80 hours for the largest OR-Library instances. On the other hand, our approach reaches only very minor additional improvements if its running time is increased extensively.

7. Conclusions

We started by studying the distance between LP-relaxed and optimal solutions of the MKP. For the benchmark instances used we empirically observed that theses distances are small, i.e. $< 10\%$ of the problem size on average, and depended on the number of variables as well as on the number of constraints. This fact was explored for solving hard to solve benchmark instances, where we restricted our search to explore this more promising neighborhood of the LP-relaxations first, which improved the performance of CPLEX applied to those instances.

We presented the new core concept for the multidimensional knapsack problem, extending the core concept for the classical one-dimensional 0/1-knapsack problem. An empirical study of the exact core sizes of widely used benchmark instances with different efficiency measures

was performed. The efficiency value using dual-variable values as relevance factors yield the smallest possible split-intervals and the smallest cores. We further studied the influence of restricting problem solving to approximate cores of different sizes, and observed significant differences in terms of run-time when applying the general-purpose ILP-solver CPLEX to approximate cores or to the original problem, whereas the objective values remained very close to the respective optima. The approach was also applied to the largest currently available test instances.

We then applied CPLEX, as well as a memetic algorithm, to the core problems of larger benchmark instances and they provided clearly and consistently better results than solving the original problems within the given fixed run-time.

Finally we studied several collaborative combinations of the presented MA and the ILP-based approaches, where these algorithms are executed in parallel and exchanged information in an asynchronous way. These collaborative approaches were given the same total CPU-times as the individual algorithms, and they were able to obtain superior solutions in some of the tested variants. In general, we were able to achieve competitive results compared to best-known solutions needing significantly lower running times.

The structural analysis of LP-relaxed and optimal solutions of combinatorial optimization problems can lead to interesting results, such as the core concept, which in turn can be used in different ways for improving the solution quality of already available algorithms. In the future we want to investigate whether the core concept can be usefully extended to other combinatorial optimization problems. Finally, the cooperation of metaheuristics and ILP-based techniques has once again proven to be highly promising. Collaborative approaches often manage to achieve better or equally good results as the individual algorithms within the same total CPU-time. Using these approaches in a parallel computing environment (e.g. multiprocessor machines or clusters) could lead to strongly improved solution quality using the same wall clock times as the individual algorithms.

Acknowledgements

NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program.

This work is partly supported by the European RTN ADONET under grant 504438 and the "Hochschuljubiläumsstiftung" of Vienna, Austria, under contract number H-759/2005.

We would like to thank the anonymous referees for valuable comments and suggestions which helped a lot to improve the paper.

References

Balas, E., E. Zemel. 1980. An algorithm for large zero-one knapsack problems. *Oper. Res.* **28** 1130–1154.

Balev, S., N. Yanev, A. Fréville, R. Andonov. 2008. A dynamic programming based reduction procedure for the multidimensional 0-1 knapsack problem. *Eur. J. Oper. Res.* **186** 63–76.

Bertsimas, D., J.N. Tsitsiklis. 1997. *Introduction to Linear Optimization*. Athena Scientific.

Chu, P.C., J.E. Beasley. 1998. A genetic algorithm for the multiconstrained knapsack problem. *J. Heuristics* **4** 63–86.

Danna, E., E. Rothberg, C. Le Pape. 2003. Integrating mixed integer programming and local search: A case study on job-shop scheduling problems. *Fifth International Workshop on Integration of AI and OR techniques in Constraint Programming for Combinatorial Optimisation Problems (CP-AI-OR'2003)*. 65–79.

Dobson, G. 1982. Worst-case analysis of greedy heuristics for integer programming with nonnegative data. *Math. of Oper. Res.* **7** 515–531.

Dyer, M.E., A.M. Frieze. 1989. Probabilistic analysis of the multidimensional knapsack problem. *Math. Oper. Res.* **14** 162–176.

Fischetti, M., A. Lodi. 2003. Local Branching. *Math. Program. Series B* **98** 23–47.

Fréville, A. 2004. The multidimensional 0–1 knapsack problem: An overview. *Eur. J. Oper. Res.* **155** 1–21.

Fréville, A., S. Hanafi. 2005. The multidimensional 0-1 knapsack problem - bounds and computational aspects. *Ann. of Oper. Res.* **139** 195–227.

Fréville, A., G. Plateau. 1994. An efficient preprocessing procedure for the multidimensional 0–1 knapsack problem. *Discrete Appl. Math.* **49** 189–212.

Gavish, B., H. Pirkul. 1985. Efficient algorithms for solving the multiconstraint zero-one knapsack problem to optimality. *Math. Program.* **31** 78–105.

Gilmore, P.C., R. Gomory. 1966. The theory and computation of knapsack functions. *Oper. Res.* **14** 1045–1075.

Glover, F., G.A. Kochenberger. 1996. Critical event tabu search for multidimensional knapsack problems. I.H. Osman, J.P. Kelly, eds., *Metaheuristics: Theory and Applications*. Kluwer Academic Publishers, 407–427.

Goldberg, A.V., A. Marchetti-Spaccamela. 1984. On finding the exact solution of a zero-one knapsack problem. *STOC '84: Proceedings of the sixteenth annual ACM symposium on Theory of computing*. ACM Press, New York, NY, 359–368.

Gomes da Silva, C., J. Clímaco, J. Figueira. 2008. Core problems in bi-criteria {0,1}-knapsack. *Comput. Oper. Res.* **35** 2292–2306.

Gottlieb, J. 1999. On the effectivity of evolutionary algorithms for multidimensional knapsack problems. Cyril Fonlupt, et al., eds., *Proceedings of Artificial Evolution: Fourth European Conference*, *LNCS*, vol. 1829. Springer, 22–37.

Kellerer, H., U. Pferschy, D. Pisinger. 2004. *Knapsack Problems*. Springer.

Martello, S., P. Toth. 1988. A new algorithm for the 0–1 knapsack problem. *Manage. Sci.* **34** 633–644.

Pisinger, D. 1995. An expanding-core algorithm for the exact 0–1 knapsack problem. *Eur. J. Oper. Res.* **87** 175–187.

Pisinger, D. 1997. A minimal algorithm for the 0–1 knapsack problem. *Oper. Res.* **45** 758–767.

Pisinger, D. 1999. Core problems in knapsack algorithms. *Oper. Res.* **47** 570–575.

Puchinger, J. 2006. Combining metaheuristics and integer programming for solving cutting and packing problems. Ph.D. thesis, Vienna University of Technology, Institute of Computer Graphics and Algorithms.

Puchinger, J., G.R. Raidl. 2005. Combining metaheuristics and exact algorithms in combinatorial optimization: A survey and classification. *Proceedings of the First International Work-Conference on the Interplay Between Natural and Artificial Computation*, *LNCS*, vol. 3562. Springer, 41–53.

Puchinger, J., G.R. Raidl, M. Gruber. 2005. Cooperating memetic and branch-and-cut algorithms for solving the multidimensional knapsack problem. *Proceedings of MIC 2005, the 6th Metaheuristics International Conference*. Vienna, Austria, 775–780.

Puchinger, J., G.R. Raidl, U. Pferschy. 2006. The core concept for the multidimensional knapsack problem. *Evolutionary Computation in Combinatorial Optimization - EvoCOP 2006*, *LNCS*, vol. 3906. Springer, 195–208.

Raidl, G.R. 1998. An improved genetic algorithm for the multiconstrained 0–1 knapsack problem. D. Fogel et al., ed., *Proceedings of the 5th IEEE International Conference on Evolutionary Computation*. IEEE Press, 207–211.

Raidl, G.R., J. Gottlieb. 2005. Empirical analysis of locality, heritability and heuristic bias in evolutionary algorithms: A case study for the multidimensional knapsack problem. *Evolutionary Comput. J.* **13** 441–475.

Rinnooy Kan, A.H.G., L. Stougie, C. Vercellis. 1993. A class of generalized greedy algorithms for the multi-knapsack problem. *Discrete Appl. Math.* **42** 279–290.

Senju, S., Y. Toyoda. 1968. An approach to linear programming with 0–1 variables. *Manage. Sci.* **15** 196–207.

Shih, W. 1979. A branch and bound method for the multiconstraint zero-one knapsack problem. *J. Oper. Res. Soc.* **30** 369–378.

Vasquez, M., J.-K. Hao. 2001. A hybrid approach for the 0–1 multidimensional knapsack problem. *Proceedings of the Int. Joint Conference on Artificial Intelligence 2001*. Seattle, Washington, 328–333.

Vasquez, M., Y. Vimont. 2005. Improved results on the 0–1 multidimensional knapsack problem. *Eur. J. Oper. Res.* **165** 70–81.

Weingartner, H.M., D.N. Ness. 1967. Methods for the solution of the multidimensional 0/1 knapsack problem. *Oper. Res.* **15** 83–103.

On-Line Supplement for: The Multidimensional Knapsack Problem: Structure and Algorithms

Jakob Puchinger
NICTA Victoria Laboratory
Department of Computer Science & Software Engineering
University of Melbourne, Australia
jakobp@csse.unimelb.edu.au

Günther R. Raidl
Institute of Computer Graphics and Algorithms
Vienna University of Technology, Austria
raidl@ads.tuwien.ac.at

Ulrich Pferschy
Department of Statistics and Operations Research
University of Graz, Austria
pferschy@uni-graz.at

Table 1: Objective values of the MKP obtained with different algorithm variants and total CPU times of 1800 seconds per instance, for the last column of 14400 seconds, compared to best known approach (VV 2005); $n = 500$ $m = 5$.

VV 2005	CPLEX	CPLEX C	CPLEX_MA_D C s	CPLEX_MA_D C el
120148	120148	120148	120148	120148
117879	117879	117879	117879	117879
121131	121131	121131	121131	121131
120804	120804	120804	120804	120804
122319	122319	122319	122319	122319
122011	122024	122024	122024	122024
119127	119127	119127	119127	119127
120568	120568	120568	120568	120568
121575	121575	121575	121575	121586
120717	120717	120717	120717	120717
218426	218428	218428	218428	218428
221202	221202	221202	221202	221202
217542	217542	217542	217534	217542
223560	223560	223560	223560	223560
218965	218966	218966	218966	218966
220527	220530	220530	220530	220530
219989	219989	219989	219989	219989
218215	218215	218215	218215	218215
216976	216976	216976	216976	216976
219719	219719	219719	219719	219719
295828	295828	295828	295828	295828
308086	308086	308086	308086	308086
299796	299796	299796	299796	299796
306480	306480	306480	306480	306480
300342	300342	300342	300342	300342
302571	302571	302571	302571	302571
301339	301339	301339	301339	301339
306454	306454	306454	306454	306454
302828	302822	302828	302828	302828
299910	299904	299906	299906	299910

Table 2: Objective values of the MKP obtained with different algorithm variants and total CPU times of 1800 seconds per instance, for the last column of 14400 seconds, compared to best known approach (VV 2005); $n = 500$ $m = 10$.

VV 2005	CPLEX	CPLEX C	CPLEX_MA_D C s	CPLEX_MA_D C el
117811	117800	117809	117809	117809
119232	119175	119181	119181	119217
119215	119211	119211	119211	119211
118813	118813	118813	118813	118813
116509	116502	116514	116514	116514
119504	119504	119481	119504	119504
119827	119777	119790	119794	119800
118329	118323	118323	118323	118323
117815	117779	117781	117781	117815
119231	119211	119203	119228	119228
217377	217377	217377	217377	217377
219077	219066	219066	219077	219077
217806	217847	217847	217847	217847
216868	216868	216868	216868	216868
213850	213846	213846	213859	213859
215086	215073	215086	215075	215086
217940	217887	217931	217931	217931
219984	219984	219984	219984	219984
214375	214352	214375	214352	214375
220899	220852	220846	220886	220886
304387	304387	304387	304353	304387
302379	302348	302356	302371	302379
302416	302408	302416	302416	302416
300757	300747	300743	300747	300757
304374	304357	304357	304374	304366
301836	301767	301796	301796	301836
304952	304949	304949	304949	304952
296478	296456	296459	296456	296459
301359	301353	301353	301353	301357
307089	307089	307089	307089	307089

Table 3: Objective values of the MKP obtained with different algorithm variants and total CPU times of 1800 seconds per instance, for the last column of 14400 seconds, compared to best known approach (VV 2005); $n = 500$ $m = 30$.

VV 2005	CPLEX	CPLEX C	CPLEX_MA_D C s	CPLEX_MA_D C el
116056	115957	116014	116014	116014
114810	114701	114810	114810	114810
116712	116661	116661	116661	116699
115329	115268	115241	115241	115253
116525	116396	116413	116420	116516
115741	115726	115726	115741	115726
114181	114072	114058	114024	114181
114348	114270	114290	114290	114290
115419	115419	115287	115419	115419
117116	117023	117023	117023	117104
218104	218033	218073	218041	218073
214648	214645	214621	214626	214626
215978	215898	215918	215903	215942
217910	217863	217827	217836	217841
215689	215625	215601	215601	215635
215890	215848	215847	215847	215860
215907	215831	215883	215883	215883
216542	216425	216419	216448	216466
217340	217287	217333	217312	217312
214739	214701	214701	214701	214701
301675	301656	301656	301656	301656
300055	300004	300055	299992	300016
305087	305069	305051	305051	305062
302032	302001	302008	302008	302008
304462	304406	304423	304423	304423
297012	296959	296959	296959	296959
303364	303329	303322	303322	303335
307007	306920	306940	306961	307007
303199	303158	303199	303199	303199
300572	300524	300499	300509	300572