TU

TECHNISCHE UNIVERSITÄT WIEN
Institut für Computergraphik und Algorithmen

The Multidimensional Knapsack Problem: Structure and Algorithms

Jakob Puchinger, Günther R. Raidl,
and Ulrich Pferschy

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

Institute 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.

---

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).

The MKP first appeared in the context of capital budgeting Lorie and Savage (1955); Manne and Markowitz (1957). 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). A recent review of the MKP was given by Fréville (2004). 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.

To our knowledge, the method currently yielding the best results, at least for commonly used benchmark instances, was described by Vasquez and Hao (2001) and has recently 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 by more detailed analyses and refined algorithms.

1.0.1. 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 all experiments 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

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

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.

---

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

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 the 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) 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

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

4

the integral part of x^{LP}

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

and the fractional part of x^{LP}

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

We further display the Hamming distance between x^* and the (possibly infeasible) arithmetically rounded LP solution x^{RLP}

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

and 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 around the rounded LP solution x^{RLP}:

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

where $S^{\text{RLP}} = \{j = 1, \ldots, n \mid x_j^{\text{RLP}} = 1\}$ 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 $\Delta(x, x^{\text{RLP}}) \geq k + 1$ into a second, remaining part. CPLEX is forced to first completely solve the neighborhood of x^{RLP} before investigating the remaining search space.

6

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

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

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. 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	$0.025_{0.004}$	7	5.24E5	$0.025_{0.004}$	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. 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 in Gomes da Silva et al. (2005).

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 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

$$(\text{MKPC}_e) \qquad \text{maximize} \quad z = \sum_{j \in C_e} p_j x_j + \tilde{p} \tag{19}$$

$$\text{subject to} \quad \sum_{j \in C_e} w_{ij} x_j \le c_i - \tilde{w}_i, \quad i = 1, \ldots, m \tag{20}$$

$$x_j \in \{0, 1\}, \quad j \in C_e, \tag{21}$$

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:*

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

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

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

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

$$u_i, v_j \geq 0, \quad i = 1, \ldots, m, \; j = 1, \ldots, n, \qquad (25)$$

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)):

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

$$v_j (x_j - 1) = 0 \qquad (27)$$

Recall that $e_j^{\text{duals}} = \frac{p_j}{\sum_{i=1}^{m} u_i w_{ij}}$. 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 any 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).

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	$|S_e|$	$|C_e|$	ScC	CcS	C_{dist}	$|S_e|$	$|C_e|$	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	$|S_e|$	$|C_e|$	ScC	CcS	C_{dist}	$|S_e|$	$|C_e|$	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

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 $e(\text{duals})$, on which we will concentrate our further investigations.

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 runtime, we propose a fixed core size algorithm, where we solve approximate cores using the general purpose ILP-solver 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 e_j^{duals} was used. The different values of δ where 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.

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

14

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.		$\delta=0.1n$			$\delta=0.15n$			$\delta=0.2n$			$\delta=2m+0.1n$			$\delta=2m+0.2n$		
m	z	$t[s]$	$\%_{opt}$	#	$\%t$	$\%_{opt}$	#	$\%t$	$\%_{opt}$	#	$\%t$	$\%_{opt}$	#	$\%t$	$\%_{opt}$	#	$\%t$
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

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, 50, 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 ($\%_{LP}$), standard deviations as subscripts, numbers of times this core size led to best solutions of these experiments (#), and numbers of explored nodes of the branch-and-bound tree.

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 except $t = 50$ an $m = 5$, for which the error probability is 2.4%. The number of explored nodes increases with decreasing problem/core size. 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.

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, 50, 100, and 500 seconds).

m	α	orig. prob. $\%_{LP}$	#	$Nnodes$	$\delta=0.1n$ $\%_{LP}$	#	$Nnodes$	$\delta=0.15n$ $\%_{LP}$	#	$Nnodes$	$\delta=0.2n$ $\%_{LP}$	#	$Nnodes$
$t[s]=5$													
5	0.25	$0.146_{0.034}$	2	4.99E+3	$\mathbf{0.112}_{0.024}$	9	1.61E+4	$0.122_{0.024}$	4	1.26E+4	$0.120_{0.023}$	3	1.04E+4
	0.5	$0.063_{0.012}$	3	4.52E+3	$\mathbf{0.053}_{0.011}$	8	1.58E+4	$0.058_{0.010}$	6	1.27E+4	$0.060_{0.014}$	6	9.98E+3
	0.75	$0.032_{0.008}$	6	5.08E+3	$0.030_{0.006}$	7	1.60E+4	$\mathbf{0.030}_{0.006}$	7	1.28E+4	$0.031_{0.008}$	7	1.06E+4
10	0.25	$0.309_{0.056}$	2	2.72E+3	$0.275_{0.030}$	4	1.20E+4	$0.280_{0.025}$	4	8.47E+3	$\mathbf{0.273}_{0.031}$	5	6.64E+3
	0.5	$0.131_{0.017}$	4	2.54E+3	$\mathbf{0.120}_{0.018}$	6	1.22E+4	$0.126_{0.016}$	3	8.29E+3	$0.128_{0.015}$	2	6.53E+3
	0.75	$0.090_{0.011}$	3	2.62E+3	$\mathbf{0.081}_{0.009}$	5	1.16E+4	$\mathbf{0.081}_{0.008}$	5	8.26E+3	$0.082_{0.006}$	3	6.32E+3
30	0.25	$0.728_{0.078}$	3	7.07E+2	$0.710_{0.014}$	2	3.99E+3	$0.690_{0.058}$	3	2.75E+3	$\mathbf{0.680}_{0.052}$	3	1.90E+3
	0.5	$0.316_{0.036}$	3	7.59E+2	$0.302_{0.017}$	3	4.22E+3	$\mathbf{0.297}_{0.023}$	4	3.01E+3	$0.308_{0.021}$	3	2.03E+3
	0.75	$0.194_{0.018}$	2	7.19E+2	$\mathbf{0.183}_{0.016}$	4	4.22E+3	$0.187_{0.016}$	3	2.92E+3	$0.185_{0.016}$	4	2.01E+3
Average		$0.223_{0.030}$	3.1	2.74E+3	$0.207_{0.016}$	5.3	1.07E+4	$0.208_{0.021}$	4.3	7.98E+3	$0.208_{0.021}$	4.0	6.27E+3
$t[s]=10$													
5	0.25	$0.118_{0.020}$	3	1.06E+4	$\mathbf{0.106}_{0.019}$	6	3.15E+4	$0.111_{0.018}$	7	2.54E+4	$0.113_{0.016}$	4	2.06E+4
	0.5	$0.061_{0.013}$	3	9.65E+3	$\mathbf{0.045}_{0.007}$	9	3.10E+4	$0.049_{0.008}$	6	2.49E+4	$0.048_{0.007}$	7	2.00E+4
	0.75	$0.032_{0.008}$	5	1.08E+4	$0.029_{0.006}$	7	3.17E+4	$0.029_{0.006}$	6	2.57E+4	$\mathbf{0.028}_{0.005}$	7	2.12E+4
10	0.25	$0.295_{0.048}$	2	5.90E+3	$\mathbf{0.257}_{0.037}$	5	2.43E+4	$0.266_{0.027}$	3	1.69E+4	$0.262_{0.033}$	6	1.34E+4
	0.5	$0.126_{0.013}$	2	5.61E+3	$\mathbf{0.108}_{0.010}$	7	2.43E+4	$0.117_{0.011}$	5	1.67E+4	$0.118_{0.014}$	4	1.35E+4
	0.75	$0.088_{0.010}$	2	5.76E+3	$\mathbf{0.077}_{0.006}$	6	2.35E+4	$\mathbf{0.077}_{0.007}$	6	1.65E+4	$0.079_{0.007}$	5	1.29E+4
30	0.25	$0.715_{0.073}$	2	1.76E+3	$0.691_{0.041}$	3	8.30E+3	$0.686_{0.055}$	3	5.96E+3	$\mathbf{0.644}_{0.097}$	2	4.16E+3
	0.5	$0.308_{0.027}$	3	1.88E+3	$0.295_{0.021}$	3	8.88E+3	$\mathbf{0.294}_{0.024}$	4	6.44E+3	$0.302_{0.017}$	3	4.52E+3
	0.75	$0.181_{0.027}$	3	1.75E+3	$\mathbf{0.178}_{0.010}$	2	8.77E+3	$0.180_{0.019}$	4	6.31E+3	$\mathbf{0.178}_{0.016}$	4	4.45E+3
Average		$0.214_{0.026}$	2.8	5.97E+3	$0.198_{0.018}$	5.3	2.14E+4	$0.201_{0.019}$	4.9	1.61E+4	$0.197_{0.024}$	4.7	1.28E+4
$t[s]=50$													
5	0.25	$0.102_{0.014}$	1	5.83E+4	$\mathbf{0.085}_{0.016}$	8	1.54E+5	$0.090_{0.015}$	6	1.25E+5	$0.094_{0.016}$	3	1.03E+5
	0.5	$0.046_{0.007}$	3	5.26E+4	$\mathbf{0.042}_{0.003}$	9	1.55E+5	$0.043_{0.005}$	8	1.24E+5	$0.044_{0.005}$	5	1.03E+5
	0.75	$0.027_{0.006}$	8	5.66E+4	$\mathbf{0.026}_{0.004}$	10	1.58E+5	$0.027_{0.004}$	9	1.27E+5	$0.027_{0.004}$	9	1.09E+5
10	0.25	$0.229_{0.037}$	5	3.18E+4	$0.218_{0.024}$	8	1.24E+5	$\mathbf{0.216}_{0.027}$	7	8.40E+4	$0.218_{0.027}$	7	6.70E+4
	0.5	$0.105_{0.011}$	3	3.05E+4	$\mathbf{0.099}_{0.009}$	9	1.24E+5	$0.100_{0.011}$	8	8.43E+4	$0.107_{0.013}$	4	6.72E+4
	0.75	$0.073_{0.010}$	4	3.09E+4	$0.071_{0.006}$	2	1.20E+5	$\mathbf{0.069}_{0.008}$	7	8.31E+4	$0.071_{0.009}$	4	6.56E+4
30	0.25	$0.634_{0.054}$	2	1.07E+4	$0.666_{0.039}$	0	4.29E+4	$0.635_{0.043}$	1	3.13E+4	$\mathbf{0.592}_{0.082}$	7	2.29E+4
	0.5	$0.278_{0.015}$	4	1.13E+4	$0.281_{0.015}$	4	4.54E+4	$0.280_{0.016}$	3	3.37E+4	$\mathbf{0.270}_{0.021}$	6	2.42E+4
	0.75	$0.169_{0.018}$	3	1.09E+4	$0.170_{0.012}$	1	4.56E+4	$0.172_{0.014}$	2	3.34E+4	$\mathbf{0.166}_{0.012}$	6	2.40E+4
Average		$0.185_{0.019}$	3.7	3.26E+4	$0.184_{0.014}$	5.7	1.08E+5	$0.181_{0.016}$	5.7	8.06E+4	$0.177_{0.021}$	3.8	6.51E+4
$t[s]=100$													
5	0.25	$0.094_{0.019}$	3	1.19E+5	$\mathbf{0.082}_{0.012}$	8	3.07E+5	$0.086_{0.013}$	7	2.51E+5	$0.089_{0.015}$	5	2.11E+5
	0.5	$0.044_{0.005}$	4	1.07E+5	$\mathbf{0.040}_{0.005}$	7	3.05E+5	$0.041_{0.004}$	9	2.51E+5	$0.042_{0.005}$	8	2.10E+5
	0.75	$0.027_{0.006}$	7	1.14E+5	$\mathbf{0.026}_{0.004}$	9	3.16E+5	$0.026_{0.004}$	9	2.59E+5	$\mathbf{0.026}_{0.004}$	8	2.23E+5
10	0.25	$0.221_{0.030}$	4	6.45E+4	$0.213_{0.020}$	8	2.51E+5	$\mathbf{0.208}_{0.022}$	6	1.70E+5	$0.214_{0.026}$	5	1.34E+5
	0.5	$0.101_{0.013}$	4	6.19E+4	$\mathbf{0.095}_{0.011}$	8	2.50E+5	$0.099_{0.011}$	6	1.71E+5	$0.099_{0.009}$	6	1.36E+5
	0.75	$0.072_{0.008}$	3	6.29E+4	$\mathbf{0.068}_{0.008}$	4	2.44E+5	$0.069_{0.008}$	4	1.68E+5	$0.069_{0.008}$	5	1.32E+5
30	0.25	$0.630_{0.051}$	1	2.21E+4	$0.646_{0.048}$	0	8.61E+4	$0.609_{0.047}$	2	6.35E+4	$\mathbf{0.586}_{0.085}$	7	4.69E+4
	0.5	$0.271_{0.015}$	4	2.31E+4	$0.270_{0.017}$	2	9.15E+4	$0.273_{0.012}$	2	6.81E+4	$\mathbf{0.265}_{0.019}$	5	4.89E+4
	0.75	$0.167_{0.016}$	1	2.25E+4	$0.165_{0.013}$	4	9.18E+4	$0.170_{0.016}$	3	6.72E+4	$\mathbf{0.163}_{0.016}$	4	4.88E+4
Average		$0.181_{0.018}$	3.4	6.63E+4	$0.178_{0.013}$	5.4	2.16E+5	$0.176_{0.015}$	5.3	1.63E+5	$0.173_{0.021}$	5.9	1.32E+5
$t[s]=500$													
5	0.25	$0.080_{0.010}$	5	5.50E+5	$\mathbf{0.075}_{0.008}$	9	1.00E+6	$0.076_{0.008}$	9	9.85E+5	$0.076_{0.010}$	8	8.34E+5
	0.50	$0.040_{0.005}$	6	5.06E+5	$\mathbf{0.039}_{0.005}$	7	1.05E+6	$\mathbf{0.039}_{0.005}$	9	1.00E+6	$\mathbf{0.039}_{0.006}$	9	8.38E+5
	0.75	$0.025_{0.004}$	6	5.36E+5	$\mathbf{0.024}_{0.003}$	10	1.05E+6	$0.025_{0.004}$	8	1.02E+6	$0.025_{0.004}$	8	9.04E+5
10	0.25	$0.206_{0.022}$	1	3.15E+5	$0.198_{0.021}$	5	1.10E+6	$\mathbf{0.195}_{0.023}$	6	6.99E+5	$0.198_{0.023}$	4	5.68E+5
	0.50	$0.094_{0.013}$	4	3.01E+5	$\mathbf{0.088}_{0.009}$	8	1.11E+6	$0.090_{0.009}$	6	6.95E+5	$0.092_{0.012}$	5	5.73E+5
	0.75	$0.066_{0.009}$	4	3.05E+5	$0.065_{0.009}$	5	1.07E+6	$\mathbf{0.064}_{0.007}$	7	6.83E+5	$0.065_{0.008}$	7	5.59E+5
30	0.25	$0.598_{0.038}$	2	1.11E+5	$0.621_{0.034}$	0	4.22E+5	$0.566_{0.049}$	4	3.06E+5	$\mathbf{0.537}_{0.061}$	6	2.28E+5
	0.50	$0.258_{0.009}$	2	1.15E+5	$0.246_{0.021}$	4	4.50E+5	$\mathbf{0.243}_{0.027}$	3	3.28E+5	$0.250_{0.024}$	2	2.38E+5
	0.75	$0.158_{0.013}$	2	1.12E+5	$\mathbf{0.151}_{0.013}$	6	4.48E+5	$0.160_{0.011}$	1	3.14E+5	$\mathbf{0.151}_{0.013}$	5	2.36E+5
Average		$0.169_{0.013}$	3.6	3.17E+5	$0.167_{0.014}$	5.9	8.55E+5	$0.162_{0.016}$	6.0	6.70E+5	$0.159_{0.018}$	6.0	5.53E+5

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 meta-heuristic 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 performing 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).

As in case of truncated CPLEX, CPU-time limits of 5, 10, 50, 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. As before, the hardest benchmark instances with $n = 500$ were used. The population size was 100.

Table 6 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

Table 6: 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, 50, 100, and 500 seconds).

m	α	orig. prob.			$\delta = 0.1n$			$\delta = 0.15n$			$\delta = 0.2n$		
		$\%_{\mathrm{LP}}$	#	$Niter$	$\%_{\mathrm{LP}}$	#	$Niter$	$\%_{\mathrm{LP}}$	#	$Niter$	$\%_{\mathrm{LP}}$	#	$Niter$
$t[s]=5$													
5	0.25	$0.099_{0.015}$	2	1.41E+5	$\mathbf{0.091_{0.006}}$	6	5.21E+5	$0.092_{0.008}$	5	4.16E+5	$0.097_{0.014}$	4	3.42E+5
	0.5	$0.050_{0.009}$	1	1.36E+5	$\mathbf{0.043_{0.006}}$	6	5.21E+5	$0.044_{0.006}$	5	4.16E+5	$\mathbf{0.043_{0.004}}$	5	3.41E+5
	0.75	$0.031_{0.004}$	2	1.47E+5	$\mathbf{0.026_{0.002}}$	7	5.20E+5	$\mathbf{0.026_{0.003}}$	7	4.17E+5	$0.029_{0.005}$	5	3.43E+5
10	0.25	$0.269_{0.034}$	1	1.28E+5	$0.245_{0.025}$	4	4.73E+5	$\mathbf{0.242_{0.015}}$	4	3.76E+5	$0.243_{0.027}$	5	3.06E+5
	0.5	$0.120_{0.015}$	3	1.23E+5	$\mathbf{0.112_{0.008}}$	5	4.70E+5	$0.113_{0.011}$	7	3.72E+5	$0.115_{0.012}$	5	3.04E+5
	0.75	$0.078_{0.008}$	3	1.33E+5	$\mathbf{0.073_{0.007}}$	8	4.72E+5	$0.074_{0.006}$	5	3.73E+5	$0.075_{0.007}$	4	3.05E+5
30	0.25	$0.696_{0.053}$	1	9.13E+4	$0.656_{0.028}$	4	3.25E+5	$0.652_{0.051}$	2	2.52E+5	$\mathbf{0.648_{0.055}}$	5	2.03E+5
	0.5	$0.290_{0.020}$	1	8.33E+4	$0.285_{0.019}$	2	3.26E+5	$\mathbf{0.273_{0.027}}$	8	2.52E+5	$0.284_{0.025}$	2	2.02E+5
	0.75	$0.177_{0.018}$	3	8.35E+4	$\mathbf{0.170_{0.014}}$	7	3.33E+5	$0.172_{0.018}$	5	2.57E+5	$0.176_{0.017}$	4	2.07E+5
Average		$0.201_{0.019}$	1.9	1.18E+5	$0.189_{0.013}$	5.4	4.40E+5	$0.188_{0.016}$	5.3	3.48E+5	$0.190_{0.018}$	4.3	2.84E+5
$t[s]=10$													
5	0.25	$0.103_{0.030}$	0	2.84E+5	$\mathbf{0.081_{0.014}}$	6	1.04E+6	$0.088_{0.014}$	3	8.31E+5	$0.088_{0.015}$	5	6.86E+5
	0.5	$0.047_{0.011}$	1	2.74E+5	$\mathbf{0.041_{0.004}}$	8	1.04E+6	$0.043_{0.006}$	2	8.30E+5	$0.044_{0.006}$	3	6.84E+5
	0.75	$0.030_{0.005}$	2	2.96E+5	$\mathbf{0.026_{0.003}}$	8	1.04E+6	$0.027_{0.003}$	6	8.35E+5	$\mathbf{0.026_{0.003}}$	7	6.87E+5
10	0.25	$0.264_{0.038}$	1	2.58E+5	$\mathbf{0.226_{0.020}}$	6	9.47E+5	$0.235_{0.025}$	3	7.50E+5	$0.241_{0.018}$	3	6.12E+5
	0.5	$0.116_{0.011}$	2	2.47E+5	$\mathbf{0.107_{0.011}}$	8	9.40E+5	$\mathbf{0.107_{0.009}}$	6	7.45E+5	$0.110_{0.013}$	4	6.09E+5
	0.75	$0.074_{0.007}$	3	2.69E+5	$0.072_{0.005}$	6	9.43E+5	$\mathbf{0.070_{0.006}}$	9	7.47E+5	$0.075_{0.006}$	3	6.11E+5
30	0.25	$0.655_{0.045}$	2	1.87E+5	$0.627_{0.049}$	4	6.53E+5	$\mathbf{0.609_{0.079}}$	7	5.05E+5	$0.631_{0.058}$	4	4.09E+5
	0.5	$0.290_{0.025}$	1	1.67E+5	$0.274_{0.019}$	4	6.49E+5	$\mathbf{0.264_{0.025}}$	5	5.03E+5	$0.267_{0.031}$	5	4.05E+5
	0.75	$0.171_{0.015}$	3	1.69E+5	$\mathbf{0.166_{0.013}}$	4	6.64E+5	$0.171_{0.015}$	3	5.15E+5	$0.167_{0.015}$	6	4.15E+5
Average		$0.195_{0.018}$	1.7	2.39E+5	$0.180_{0.015}$	6.0	8.80E+5	$0.179_{0.020}$	4.9	6.96E+5	$0.183_{0.019}$	4.4	5.69E+5
$t[s]=50$													
5	0.25	$0.090_{0.009}$	1	1.42E+6	$\mathbf{0.080_{0.009}}$	7	5.22E+6	$0.082_{0.010}$	4	4.16E+6	$\mathbf{0.080_{0.012}}$	7	3.43E+6
	0.5	$0.043_{0.006}$	3	1.37E+6	$\mathbf{0.040_{0.004}}$	9	5.21E+6	$\mathbf{0.040_{0.003}}$	7	4.16E+6	$\mathbf{0.040_{0.004}}$	5	3.42E+6
	0.75	$0.027_{0.003}$	6	1.48E+6	$0.026_{0.003}$	7	5.20E+6	$\mathbf{0.025_{0.004}}$	9	4.16E+6	$0.026_{0.003}$	6	3.43E+6
10	0.25	$0.238_{0.016}$	1	1.29E+6	$\mathbf{0.217_{0.012}}$	4	4.73E+6	$0.221_{0.019}$	3	3.75E+6	$0.221_{0.018}$	4	3.06E+6
	0.5	$0.108_{0.008}$	2	1.24E+6	$\mathbf{0.100_{0.009}}$	4	4.70E+6	$0.104_{0.012}$	3	3.72E+6	$\mathbf{0.100_{0.008}}$	5	3.04E+6
	0.75	$0.071_{0.006}$	3	1.35E+6	$\mathbf{0.067_{0.006}}$	7	4.72E+6	$0.069_{0.005}$	5	3.74E+6	$0.069_{0.006}$	4	3.05E+6
30	0.25	$0.642_{0.044}$	0	9.36E+5	$0.596_{0.075}$	5	3.26E+6	$\mathbf{0.594_{0.074}}$	5	2.52E+6	$0.612_{0.056}$	3	2.04E+6
	0.5	$0.264_{0.021}$	2	8.39E+5	$0.265_{0.016}$	4	3.26E+6	$\mathbf{0.262_{0.015}}$	3	2.52E+6	$0.267_{0.012}$	4	2.02E+6
	0.75	$0.172_{0.016}$	2	8.48E+5	$\mathbf{0.160_{0.011}}$	6	3.32E+6	$0.161_{0.012}$	7	2.57E+6	$0.162_{0.010}$	6	2.07E+6
Average		$0.184_{0.015}$	2.2	1.20E+6	$0.172_{0.016}$	5.9	4.40E+6	$0.173_{0.017}$	5.1	3.48E+6	$0.175_{0.015}$	4.9	2.84E+6
$t[s]=100$													
5	0.25	$0.085_{0.011}$	3	2.84E+6	$\mathbf{0.075_{0.009}}$	9	1.04E+7	$0.077_{0.008}$	8	8.32E+6	$0.079_{0.009}$	5	6.85E+6
	0.5	$0.042_{0.006}$	3	2.75E+6	$\mathbf{0.039_{0.004}}$	8	1.04E+7	$0.040_{0.004}$	8	8.31E+6	$0.040_{0.004}$	5	6.83E+6
	0.75	$0.026_{0.002}$	3	2.97E+6	$\mathbf{0.024_{0.003}}$	9	1.04E+7	$\mathbf{0.024_{0.003}}$	9	8.32E+6	$0.025_{0.004}$	6	6.85E+6
10	0.25	$0.230_{0.016}$	0	2.58E+6	$\mathbf{0.209_{0.023}}$	7	9.46E+6	$0.216_{0.011}$	5	7.51E+6	$0.216_{0.015}$	5	6.12E+6
	0.5	$0.101_{0.011}$	3	2.47E+6	$\mathbf{0.099_{0.007}}$	5	9.40E+6	$0.097_{0.011}$	6	7.45E+6	$0.100_{0.009}$	4	6.08E+6
	0.75	$0.069_{0.006}$	5	2.69E+6	$\mathbf{0.067_{0.006}}$	7	9.44E+6	$0.068_{0.006}$	5	7.47E+6	$0.069_{0.006}$	5	6.10E+6
30	0.25	$0.624_{0.042}$	3	1.87E+6	$0.595_{0.073}$	5	6.51E+6	$\mathbf{0.593_{0.072}}$	4	5.05E+6	$0.602_{0.076}$	3	4.07E+6
	0.5	$0.272_{0.019}$	0	1.68E+6	$0.257_{0.014}$	7	6.51E+6	$0.262_{0.022}$	3	5.04E+6	$\mathbf{0.256_{0.019}}$	6	4.04E+6
	0.75	$0.166_{0.013}$	3	1.69E+6	$\mathbf{0.158_{0.012}}$	7	6.63E+6	$0.161_{0.011}$	5	5.14E+6	$\mathbf{0.158_{0.010}}$	7	4.13E+6
Average		$0.180_{0.014}$	2.6	2.39E+6	$0.169_{0.017}$	7.1	8.80E+6	$0.171_{0.016}$	5.9	6.96E+6	$0.172_{0.017}$	5.1	5.67E+6
$t[s]=500$													
5	0.25	$0.078_{0.011}$	6	1.40E+7	$\mathbf{0.073_{0.008}}$	10	5.08E+7	$0.074_{0.008}$	9	4.07E+7	$0.074_{0.008}$	9	3.33E+7
	0.50	$0.040_{0.004}$	6	1.35E+7	$\mathbf{0.039_{0.004}}$	9	5.07E+7	$\mathbf{0.039_{0.004}}$	9	4.07E+7	$0.040_{0.004}$	7	3.33E+7
	0.75	$0.025_{0.003}$	7	1.46E+7	$\mathbf{0.024_{0.003}}$	9	5.07E+7	$\mathbf{0.024_{0.003}}$	10	4.08E+7	$\mathbf{0.024_{0.003}}$	9	3.34E+7
10	0.25	$0.208_{0.019}$	5	1.26E+7	$\mathbf{0.202_{0.017}}$	5	4.54E+7	$\mathbf{0.202_{0.013}}$	6	3.62E+7	$0.208_{0.018}$	4	2.90E+7
	0.50	$0.099_{0.007}$	2	1.21E+7	$0.093_{0.007}$	6	4.51E+7	$\mathbf{0.091_{0.010}}$	8	3.59E+7	$0.093_{0.009}$	5	2.89E+7
	0.75	$0.066_{0.005}$	6	1.31E+7	$\mathbf{0.065_{0.005}}$	8	4.53E+7	$0.067_{0.005}$	4	3.59E+7	$0.068_{0.006}$	4	2.87E+7
30	0.25	$0.604_{0.046}$	1	9.10E+6	$0.573_{0.078}$	5	3.08E+7	$0.575_{0.063}$	5	2.39E+7	$\mathbf{0.569_{0.068}}$	6	1.92E+7
	0.50	$0.254_{0.021}$	3	8.10E+6	$0.257_{0.015}$	1	3.08E+7	$\mathbf{0.246_{0.019}}$	7	2.37E+7	$0.253_{0.021}$	3	1.90E+7
	0.75	$0.159_{0.012}$	4	8.12E+6	$\mathbf{0.156_{0.013}}$	5	3.14E+7	$0.157_{0.012}$	3	2.35E+7	$0.157_{0.011}$	5	1.96E+7
Average		$0.170_{0.014}$	4.4	1.17E+7	$0.165_{0.017}$	6.4	4.23E+7	$0.164_{0.015}$	6.8	3.35E+7	$0.165_{0.017}$	5.8	2.72E+7

18

percentage gaps ($\%_{\mathrm{LP}}$), standard deviations as subscripts, numbers of times each core size yielded the best solutions of these experiments (#), and average numbers of MA iterations (*Niter*).

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 perform within the allowed time. The larger number of candidate solutions the MA can examine when it is applied to a restricted 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.

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 ex-

perimentally 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 7 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.

The results indicate a small 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 CPLEX_MA_D/skewed achieves better average results than CPLEX with an error probability of 1.5% and better than the MA with an error probability of 5.3%.

Table 7: 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	$\%_{LP}$	#	$\%_{LP}$	#	$\%_{LP}$	#	$\%_{LP}$	#	$\%_{LP}$	#	$\%_{LP}$	#
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 8 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.

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.

Table 8: 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	$\%_{LP}$	#	$\%_{LP}$	#	$\%_{LP}$	#	$\%_{LP}$	#	$\%_{LP}$	#	$\%_{LP}$	#
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	$0.025_{0.004}$	7	$0.025_{0.004}$	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

5.3. Applying Collaborative Approaches to MKP Cores

Table 9 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 9: 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	$\%_{LP}$	#	$\%_{LP}$	#	$\%_{LP}$	#	$\%_{LP}$	#	$\%_{LP}$	#	$\%_{LP}$	#
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 10 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 ($\%_{\mathrm{LP}}$), 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.

Table 10: 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	$\%_{\mathrm{LP}}$	$\#$	$t[s]$	$\%_{\mathrm{LP}}$	$\#$	$\%_{\mathrm{LP}}$	$\#$	$\%_{\mathrm{LP}}$	$\#$	$\%_{\mathrm{LP}}$	$\#$
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

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 outperform the tabu search from Vasquez and Vimont (2005) for the $m = 5$ class, since we achieve slightly better results in substantially shorter running times. For the classes with $m \in \{10, 30\}$ the

results provided in Vasquez and Vimont (2005) are usually better than those of our approach in terms of solution quality, but not in terms of running time.

The results achieved by our parallel approach with 14400 seconds of total CPU-time are again slightly better than those of Vasquez and Vimont (2005) for $m = 5$. For $m = 10$ our results are comparable to 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. We then applied CPLEX, as well as a memetic algorithm, to the core problems of larger 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

References

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

Bertsimas, Dimitris, John N. Tsitsiklis. 1997. *Introduction to Linear Optimization*. Athena Scientific.

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

Danna, Emilie, Edward Rothberg, Claude 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. *Mathematics of Operations Research* **7** 515–531.

Dyer, M.E., A.M. Frieze. 1989. Probabilistic analysis of the multidimensional knapsack problem. *Mathematics of Operations Research* **14** 162–176.

Fischetti, Matteo, Andrea Lodi. 2003. Local Branching. *Mathematical Programming Series B* **98** 23–47.

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

Fréville, Arnaud. 2004. The multidimensional 0–1 knapsack problem: An overview. *European Journal of Operational Research* **155** 1–21.

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

Gilmore, P.C., R. Gomory. 1966. The theory and computation of knapsack functions. *Operations Research* **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, Carlos, João Clímaco, José Figueira. 2005. Core problems in bi-criteria {0,1}-knapsack: new developments. Tech. Rep. 12/2005, INESC-Coimbra.

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, Hans, Ulrich Pferschy, David Pisinger. 2004. *Knapsack Problems*. Springer.

Lorie, J., L.J. Savage. 1955. Three problems in capital rationing. *The Journal of Business* **28** 229–239.

Manne, A.S., H.M. Markowitz. 1957. On the solution of discrete programming problems. *Econometrica* **25** 84–110.

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

Pisinger, D. 1995. An expanding-core algorithm for the exact 0–1 knapsack problem. *European Journal of Operational Research* **87** 175–187.

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

Pisinger, D. 1999. Core problems in knapsack algorithms. *Operations Research* **47** 570–575.

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

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

Raidl, Günther R., Jens Gottlieb. 2005. Empirical analysis of locality, heritability and heuristic bias in evolutionary algorithms: A case study for the multidimensional knapsack problem. *Evolutionary Computation Journal* **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 Applied Mathematics* **42** 279–290.

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

Shih, W. 1979. A branch and bound method for the multiconstraint zero-one knapsack problem. *Journal of the Operational Research Society* **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, Michel, Yannick Vimont. 2005. Improved results on the 0–1 multidimensional knapsack problem. *European Journal of Operational Research* **165** 70–81.

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