Analysis of Greedy Heuristics and Weight-Coded EAs for Multidimensional Knapsack Problems and Multi-Unit Combinatorial Auctions

Jella Pfeiffer and Franz Rothlauf

Working paper 1/2007
March 2007

Working Papers in Information Systems and Business Administration

Johannes Gutenberg-University Mainz
Department of Information Systems and Business Administration
D-55128 Mainz/Germany
Phone +49 6131 39 22734, Fax +49 6131 39 22185
E-Mail: sekretariat[at]wi.bwl.uni-mainz.de
Internet: `http://wi.bwl.uni-mainz.de`

Analysis of Greedy Heuristics and Weight-Coded EAs for Multidimensional Knapsack Problems and Multi-Unit Combinatorial Auctions

Jella Pfeiffer
Dept. of Information Systems and Business Administration
University of Mainz
jella.pfeiffer@uni-mainz.de

Franz Rothlauf
Dept. of Information Systems and Business Administration
University of Mainz
rothlauf@uni-mainz.de

ABSTRACT

Until recently, multidimensional knapsack problems (MDKP) and winner determination problems (WDP) have been studied independently of each other, although WDPs can be modelled as MDKPs. State-of-the-art optimization methods for WDPs are exact algorithms whereas MDKPs are mainly solved using heuristics or metaheuristics such as evolutionary algorithms (EAs). This paper compares and studies the performance of these different approaches for MDKP and WDP test instances. It shows that all currently used WDP test instances can be solved optimally and in a short length of time by exact optimization methods. Thus, these test instances are only of limited usefulness for estimating the performance of optimization methods for the WDP. For MDKPs, simple greedy heuristics are very fast and provide high-quality solutions since the gaps towards optimal solutions are usually only a few percent. Furthermore, the gaps decrease with an increasing tightness ratio of the underlying MDKP. Weight-coded EAs significantly improve solution quality of greedy heuristics at the expense of much higher computational effort. However, as the main quality improvement occurs in the very early stages of EA runs, running times observed in the literature can be greatly reduced with only minor effects on the resulting solution quality.

1. INTRODUCTION

The multidimensional knapsack problem (MDKP) is a well-known combinatorial optimization problem that finds a subset of given items in such a way that the total profit of the selected items is maximized while satisfying a set of knapsack constraints. Recently [17], it was noticed that the winner determination problem (WDP), which is a relevant problem in the context of combinatorial auctions (CA), can be modelled as MDKP. The current state-of-the-art solution approaches for WDPs are exact optimization methods, whereas MDKPs are mainly solved using heuristics and metaheuristics such as evolutionary algorithms (EA).

The goal of this paper is to directly compare and study the solution approaches used in the two different research communities. We focus on the comparison of different heuristic optimization methods for standard MDKP and WDP test instances. Consequently, we present results for exact problem solvers such as CPLEX, compare the performance of different types of greedy heuristics and Raidl's weight-coded EA [28], and examine the EA's trade-off between solution quality and number of generations. Thus, this study also extends previous work [28], which examined the influence of the underlying greedy heuristic on the performance of weight-coded EAs.

The following section defines the MDKP, presents current solution approaches and shows its close relationship to the WDP, and reviews various WDP and MDKP test instances. Section 3 discusses four different construction heuristics and describes the weight-coded EA from Raidl [28]. Experimental results are presented in Section 4. We start by studying the performance of CPLEX, then continue by comparing the performance of different heuristics and the weight-coded EA, and finally examine the anytime performance of the EA.

2. MDKP

2.1 Problem Definition

The *knapsack problem* (KP) is a common problem in combinatorial optimization, which is $\mathcal{NP}$-complete [19]. The standard variant assumes one knapsack being packed with a number of items x_j $(j = 1, \ldots, n)$. Given the profit p_j and weight r_j of each item x_j, the goal is to pack items into the knapsack such that the total profit is maximized while not exceeding the maximal capacity c of the knapsack.

In the MDKP, each item has m different properties such as the weight or volume consuming capacity of the knapsack. The MDKP is also known as m- or d-dimensional knapsack problem, multiconstraint knapsack problem, or multiple knapsack problem. It can be formulated as follows:

$$\begin{aligned} \textit{maximize} \quad & \sum_{j=1}^{n} p_j x_j \\ \textit{subject to} \quad & \sum_{j=1}^{n} r_{ij} x_j \leq c_i, \quad i = 1, ..., m \\ \textit{with} \quad & x_j \in \{0, 1\}, \quad j = 1, ..., n \\ & p_j \in \mathbb{N},\ r_{ij} \in \mathbb{N}_0,\ c_i \in \mathbb{N}, \end{aligned} \tag{1}$$

where n is the number of items and m the number of dimensions. A number of relevant real-world problems can be modelled as MDKPs such as allocation problems, logistic problems, or cutting stock problems [20].

2.2 Solution Approaches

Due to the $\mathcal{NP}$-completeness of the MDKP, optimal algorithms face run-time problems when applied to problem instances with a large number of items. Although branch

and bound algorithms are exact optimization methods and thus need exponential effort, they are the method of choice for small problems instances. The reader is referred to Osorio et al. [26] for a detailed overview of exact approaches for MDKP.

Caprara et al. published a polynomial-time approximation scheme for the MDKP which provides an $\frac{1}{m+1}$ approximation [2]. However, the running time of the proposed algorithm is $O(n^{\lceil \frac{m}{\epsilon} \rceil - m})$, which grows polynomially with n but exponentially with m.

There are a large number of heuristic optimization methods (such tabu search [16, 10] or ant colony optimization [6]) which have been applied to the MDKP. The most successful approaches are based on EAs. The two current state-of-the-art algorithms are EAs with repair steps using a direct encoding proposed by Chu and Beasley [3], and a weight-coded approach proposed by Raidl [28]. Other promising approaches are hybrids such as the approach presented by Thiel and Voss [36], who combined an EA with tabu search, or an approach from Gallardo et al. [9], who combined branch and bound algorithms with EAs. The latter approach runs both algorithms in parallel and performs mutual exchanges of high-quality solutions. A comprehensive review of heuristic optimization methods for the MDKP can be found in [3, 13, 20].

2.3 Multi-Unit Combinatorial Auctions

Different problems can be modelled as the MDKP. A prominent example is the allocation problem of *multi-unit combinatorial auctions* (MUCA) in the field of electronic market design. In CAs bidding is allowed on bundles of goods allowing bidding agents to express synergies between goods they want to obtain. Synergies exist when the agent's utility function is either *sub-additive* or *super-additive*. Sub-additivity expresses substitution effects between goods, which means that the value of a bundle of goods is lower than the sum of the values of the single goods. Super-additivity expresses complementarities between goods and occurs when the value of a bundle is higher than the sum of the values of the single goods. The benefit of CAs is that bids on bundles of goods are allowed reducing an agents' risk of receiving unfavorable combinations of goods. Therefore, agents are encouraged to bid higher prices for bundles of goods, leading to a higher revenue for the auctioneer. Hence, the use of CAs result in a win-win situation for both agents and auctioneers.

CAs consist of several steps. In the first step the agents submit their bids and in the second step the auctioneer allocates the goods to the agents such that his revenue is maximized. The revenue is the sum of all submitted bids which are accepted by the auctioneer. This allocation problem (which bids should be accepted by the auctioneer to maximize his profit) is called the *winner determination problem* (WDP). The WDP in CAs is more demanding than in ordinary auctions (such as Vickrey or English auction), since a subset of bids which maximize the auctioneer's revenue must be found.

Research often focuses on single-unit CAs where only one unit per good is available, $c_i = 1, \forall i = 1, ..., m$. In this paper, we focus on the more general MUCA problem which allows several copies of the same type of good i. Formulating the WDP of MUCAs leads to the MDKP (see equation 1). The profit p_j used in the MDKP corresponds to the price of bid j, while the resource consumption r_{ij} corresponds to the number of units of good i requested in bid j. The decision variable x_j denotes whether bid j wins (is accepted by the auctioneer) or looses (is not accepted by the auctioneer).

As the literature had ignored the close relationship between MDKP and WDP for a long time, two different lines of research emerged independently of each other. The close relationship was first mentioned by Holte in 2001 [17]. Only recently, Kelly [21] presented the first direct comparison between the WDP for different kinds of auctions and their equivalent KPs. Kelly [21] states that research on WDP algorithms might profit from algorithms developed for the MDKP.

The development of optimal algorithms for the WDP has mainly focused on single-unit CAs. Commonly used solution techniques are branch and bound approaches [8, 34], IDA* search, or dynamic programming [30, 33]. Yet, in the last few years, the commercial solver ILOG CPLEX has become the state-of-the-art solution tool for solving CA problems. For MUCAs, only two algorithms have been developed, both are based on branch and bound methods. The first algorithm is CAMUS [25], a generalization and extension of CASS [8]. The algorithm is tested on instances with 10 and 14 goods with 40 units in total and up to 2500 bids. The largest problems are solved optimally in two minutes with a good any-time performance. The second exact algorithm for MUCA uses bounds by solving several relaxed linear problems [12] and shows similar performance.

Also non-optimal algorithms have been proposed for the single-unit case. Zurel and Nisan [38] presented a greedy heuristic with some local improvements which is up to 1000 times faster than CPLEX while obtaining solutions with a maximum gap of 4% to the optimum. Furthermore, some metaheuristics have been developed. Hoos and Boutilier [18] applied a stochastic local search technique to problems with up to 500 goods and 5000 bids. The quality of the solutions was reported to be similar to using CASS as the time limit for the running time (60 seconds) was so low that CASS was not able to find optimal solutions. Sakurai et al. [32] proposed a limited discrepancy search which resulted in a maximum gap between 6% and 15% to the optimal solution for test problems of up to 100 goods and 5000 bids. Guo et al. [15] proposed a simulated annealing approach which outperforms simple greedy methods and achieves better results than the anytime performance of CPLEX 8.0 with strict time limits. To our knowledge, the only GA for the WDP was presented by Bai et al. [1]. The presented results are only of limited relevance as the method was compared to the exact method from Sandholm [33] where it showed worse performance but lower running time on larger test instances. The only non-optimal approach for the MUCA is a simple hill-climber presented by Holte [17]. Since this approach showed good performance on some MDKP instances, we later examine in this paper whether the use of simple heuristics alone already leads to good results for the MDKP.

CAs are relevant for many real-world scenarios. For example, at the University of Chicago the students can enroll in courses by placing bundled bids, and a CA mechanism allocates the available capacity of the courses to the students [14]. In other applications, transport capacities [22] or school meals for Chilean students [5] are auctioned. Overall, the application of CAs in real-world scenarios is of special interest for logistics and transportation, procurement, e-finance, the allocation of public goods, and supply chain manage-

ment with scheduling problems.

2.4 Test Instances

We choose three different types of test instances for the experiments, denoted as *LB*, *MULTIPATH*, and *MKNAPCB*. LB and MULTIPATH are from the MUCA literature and MKNAPCB is from the MDKP literature. The LB test instances are generated using the data generation method presented by Leyton-Brown et al. [25]. The MULTIPATH test instances are extended variants of the single-unit PATH test instances from the well-known CATS[1] test suite. The CATS test suite was created by Leyton-Brown et al. [24] as an attempt to generate realistic, economically motivated test data for single-unit CAs. The MKNAPCB test instances are taken from Beasley's OR-library[2]. We have chosen the same test instances as used by Chu and Beasley [3] for evaluating the performance of methods for the MDKP.

In the MDKP literature, the structure of problem instances is described by the *tightness ratio* α. This ratio expresses the scarcity of capacities defined as:

$$\alpha_i = \sum_{j=1}^{n} \frac{c_i}{r_{ij}}. \quad (2)$$

For example, a tightness ratio of 0.75 expresses that only about 75% of the bids can be satisfied (WDP), or about 75% of all available goods can be packed into the knapsack (MDKP). The lower the tightness ratio, the more restricted is the problem instance.

MKNAPCB test instances.

The MKNAPCB test data consists of 30 instances for each of the nine combinations of $m = \{5, 10, 30\}$ and $n = \{100, 250, 500\}$. The resource consumptions r_{ij} are uniformly distributed in $[0, 1000]$, resulting in a dense constraint matrix. For each of the nine combinations, 10 random problem instances are created with $\alpha_i = 0.25$, $\alpha_i = 0.5$, and $\alpha_i = 0.75$ ($i \in \{1, \ldots, m\}$). The price of an item is determined as:

$$p_j = \sum_{i=1}^{m} \frac{r_{ij}}{m} + 500\ \mathcal{R}[0, 1], \quad (3)$$

where $\mathcal{R}$ is a uniform distribution.

LB test instances.

The LB data sets are generated according to Leyton-Brown et al. [25]. We use problem instances with $n \in \{1500, 2500\}$ and $m \in \{10, 14\}$. The maximum number of units of each good is set to $g = 5$. Other variables are relevant for the calculation of the prices of the bids and are set to $p^{avg}_{base} = 50$, $p^{avg}_{var} = 25$, and $p_{var} = 0.5$. The problem instances are constructed as follows:

For each good $i \in [1, m]$, c_i is chosen randomly from $[1, g]$. This process is repeated until $\sum_i c_i = m \sum_{j=1}^{g} j/g$ to ensure that each trial involves the same total number of units [25]. The average price p^{avg}_i of each good i is chosen uniformly from $[p^{avg}_{base} - p^{avg}_{var}, p^{avg}_{base} + p^{avg}_{var}]$. The number of goods in a bid as well as the number r_{ij} of units demanded of each good follow a decay distribution: we iteratively add a random good/unit to a bid without replacement. The adding process is stopped with probability P, or if no more goods/units remain. We use $P = 0.2$ for increasing the number of goods and $P = 0.35$ for increasing r_{ij}. Finally, the price of each bid is calculated as $p_j = \mathcal{R}[1 - p_{var}, 1 + p_{var}] \sum_{i=1}^{m} (p^{avg}_i r_{ij})$, where $\mathcal{R}$ is a uniformly drawn number.

MULTIPATH test instances.

The MULTIPATH test instances are an extension of the PATH instances from the CATS test suite [24]. The PATH instances make use of paths in graphs where a good is equivalent to an edge. Let $G = (V, E)$ be a connected, undirected graph where the nodes are randomly placed on a 2-dimensional grid and edges are added according to an algorithm which ensures the creation of a nearly planar graph [24]. The goal of an agent is to obtain all edges of a path between i and j since incomplete paths have no value for an agent. The price of a path $path_{ij}$ is calculated as $v_{ij} - \sum_{e_{lm} \in p_{ij}} d_{lm}$, where d_{lm} is proportional to the distance between node l and m and v_{ij} is the agent's benefit from obtaining the path $path_{ij}$. In this standard CATS test suite, each edge can be assigned to only one agent. For more details on the CATS test suite we refer to [24].

We extended the CATS test problems to multi-unit test problems by assigning a uniformly distributed capacity $c_i \in [1, 10]$ to each edge. The capacity specifies the number of units which are available of each edge (good). The resource consumption r_{ij} in bid_i is determined uniformly in the range of $[1, c_i]$ for each edge which is desired. For the test problems used in this study, we chose standard parameter settings as proposed in [24] and created 10 test instances with $n = 1500$. For the algorithm, we used $m = 50$. Since the problem construction algorithm introduces "dummy goods" to make bids of the same agent mutually exclusive, the average number of goods for the 10 problem increases to $m_{avg} = 595$.

In most MUCA test instances from the literature, each agent only bids for a small ratio of the available goods. Therefore, in contrast to the MKNAPCB test instances with dense constraint matrices, MUCA test instances have only sparse constraint matrices. However, it is unclear whether most real-world WDPs are structured like existing MUCA test instances with sparse constraint matrices, or like MDKP test instances with dense constraint matrices. WDPs with dense constraint matrices can be, for example, the result of agents that are interested in all available goods.

3. HEURISTICS FOR THE MDKP

This section describes different heuristics that can be used for solving MDKPs. We focus on construction heuristics and EAs using weighted encodings.

3.1 Construction Heuristics

MDKP and MUCA literature distinguish between two basic construction heuristics for building a solution. *Primal greedy heuristics* use a pre-processing step where all items (bids) are sorted according to some criteria (usually r_{ij}, p_j, or a combination of both) by decreasing value [37]. When constructing a solution, the heuristics add the items one after another to the knapsack as long as no restrictions are violated. *Dual greedy heuristics* start the other way around [35]. They first set all decision variables x_j to one, then sort the items by increasing value according to some sorting-heuristic, and finally remove items one after the other as

[1] `http://cats.stanford.edu`

[2] `http://people.brunel.ac.uk/~mastjjb/jeb/orlib/mknapinfo.html`

long as constraints are violated. In a final step, all removed items must be considered again for inclusion since by removing items new capacity might become available. This final (post-processing) step is performed in descending order of the items, as the most valuable items should be included first.

The following paragraphs present four heuristics for sorting items. All four heuristics are easy to implement and fast. Please note that an item is equivalent to a bid, and the profit of an item is equivalent to the price of a bid.

3.1.1 Normalized Bid Price (NBP)

A common sorting heuristic is based on the ratio [4]

$$\frac{p_j}{\sum_{i=1}^{m} r_{ij}}. \tag{4}$$

The profit of an item j is weighted by the number of units that are consumed by the item. This yields the average profit per unit. Apparently, all goods are treated equally irrespectively of the scarcity of the capacities c_i.

In the CA community, a more general version of this ratio was proposed for the single-unit case ($r_{ij} \in \{0, 1\}$) [23]:

$$NBP_j := \frac{p_j}{(\sum_{i=1}^{m} r_{ij})^l}, \quad l \geq 0. \tag{5}$$

Lehmann et al. [23] showed that for $l = 0.5$ the resulting primal greedy heuristic ensures the best possible worst-case bound since it approximates the optimal allocation within a factor of $\frac{1}{\sqrt{m}}$. In a later paper [11], the same bound was proved to hold for the multi-unit case.

3.1.2 Relaxed LP Solution (RLPS)

This approach calculates the optimal solution of the relaxed MDKP problem using a linear problem solver such as CPLEX. Then, the RLPS sorting heuristic sorts the continuous decision variables $x_j \in [0, 1]$ in a decreasing order. This approach is motivated by the observation that "the more of the bid is accepted in the relaxed LP, the more likely it is to be competitive" [34].

3.1.3 Scaled NBP (SNBP)

Fox and Scudder [7] proposed a *relevance* value μ_i which measures the scarcity of capacities (see also equation (2)) and sorts the items according to $p_j / \sum_{i=1}^{m} \mu_i r_{ij}$. The underlying idea is to choose a high μ_i for dimensions with low c_i to penalize the consumption of that resource. $\mu_i = 1/c_i$ [20] results in

$$SNBP_j := \frac{p_j}{\sum_{i=1}^{m} \frac{r_{ij}}{c_i}}. \tag{6}$$

The SNBP improves the NBP as it does not treat all dimensions equally, but considers different capacities c_i.

3.1.4 Shadow Surplus (SS)

Pirkul [27] proposed using *surrogate multipliers* a_i as relevance values μ_i. The a_i aggregate all capacity constraints to a single constraint by using weighted sums (see [20]). Pirkul proposed several possibilities to derive surrogate multipliers. A straightforward approach is to use the dual variables of the relaxed LP. They serve as an approximation of how valuable a unit of a good is and are often called shadow prices. We define the shadow surplus of a bid j as

$$SS_j := \frac{p_j}{\sum_{i=1}^{m} a_i r_{ij}}, \tag{7}$$

where a_i ($i \in \{1, \ldots, m\}$) are the solutions of the dual LP.

3.2 Weighted Encodings

As discussed in Sect.2.2, the current state-of-the-art approaches for MDKPs are the EA from Chu and Beasley [3] and the weight-coded approach from Raidl [28]. Both show an approximately similar performance. For the study performed in this paper, we focus on the EA from Raidl which can easily be combined with the construction heuristics from section 3.1.

In Raidl's approach, an indirect representation is used. The genotypes are vectors of real-valued weights $(w_1, \ldots, w_n)$, where the weight at position j indicates the importance of the jth item. The decoder constructs a solution from the genotype in two steps. In the first step, the profits p_j of the items are biased according to the corresponding w_j either as $p'_j = p_j w_j$ or $p'_j = p_j + w_j$, where p'_j are the resulting biased profits. Subsequently, the items are ordered in descending order according to p'_j. In the second step, a primal greedy heuristic (see Sect. 3.1) is applied to the sorted list to obtain a complete problem solution.

Raidl studied the performance of the SS heuristic and one other using Lagrangian multipliers. Furthermore, he examined different approaches for the initialization and mutation operator. The best results were reported for calculating the biased weights as $p'_j = p_j w_j$, using the SS heuristic as the primal greedy heuristic, and using the LOGNORM biasing

$$w_j = (1 + \gamma)^{\mathcal{N}(0,1)} \tag{8}$$

for initializing and mutating the weights. This initialization and mutation results in a bias towards small weights due to the use of a normal distribution $\mathcal{N}$. Raidl recommended setting γ to 0.05.

Weighted encodings allow us to apply standard search operators. Raidl uses a standard uniform crossover with the probability 1 and the mutation operator (8) with probability $3/n$. Experiments are performed using a steady-state GA with a population of 100 and phenotypic duplicate elimination. GA runs are stopped after evaluating 100,000 individuals without finding a new best solution.

The indirect representation used by Raidl results in a strong bias towards solutions which are constructed using the underlying greedy heuristic. The strong bias might mislead evolutionary search if the optimal solution of the problem is not very similar to the solution generated by applying the greedy heuristic alone [31]. Additional problems can occur due to the redundancy of the encoding as many mutations of the w_j do not result in a new phenotype. This problem can be addressed by using duplicate elimination (as proposed by Raidl) but results in additional search effort.

4. EXPERIMENTS

All experiments of this study were run on a 64-bit Linux machine with an AMD Opteron CPU with 2.2 GHz and 2 GB RAM.

4.1 CPLEX

CPLEX is a commercial solver for linear problems. Due to its rapid development in the last few years, it has become the state-of-the-art method for solving WDPs. However, MDKP problems like the MKNAPCB test instances are more difficult to solve for CPLEX. In 1998, Chu and Beasley [3] used

Table 1: Performance of CPLEX

	m	n	opt. found	time (sec.)
MKNAPCB				
1	5	100	yes	224
2	5	250	yes	3084
3	5	500	yes	3840
4	10	100	yes	378
5	10	250	no	322
6	10	250	no	820
7	30	100	yes	1152
8	30	250	no	608
9	30	500	no	266
LB				
1-10	10	1500	yes	1.2
1-10	14	2500	yes	1.6
MULTIPATH				
1-10	595	1500	yes	9.1

CPLEX 4.0 for solving the MKNAPCB test instances and found that only test instances 1,2, and 4 could be solved to optimality.

Table 1 extends those experiments and presents the performance of CPLEX 9.0 for the test instances from Sect. 2.4. We list the number n of items and the number m of dimensions, indicate whether the optimal solution was found, and report the necessary CPU time. For each of the nine MKNAPCB parameter settings, we averaged over all 30 test instances. For the LB and MULTIPATH problems, we averaged over 10 randomly generated problem instances per parameter setting. For problems where CPLEX runs out of memory (optimum not found), we list the time when CPLEX aborted.

The results show that CPLEX solves all MUCA test instances (LB as well as MULTIPATH) in a short length of time. In contrast, only small MKNAPCB test instances can be solved optimally. Therefore, we are able to confirm the results from the CA literature that CPLEX is a good choice for solving MUCA test instances. In contrast, MDKP test instances are more difficult to solve for CPLEX as only small problem instances can be solved optimally. Therefore, heuristic optimization methods are a valid choice for solving MDKPs. Heuristic optimization methods might also be useful for MUCAs if it is important to obtain optimal or near-optimal solutions in a short time or when MUCA instances are structurally similar to MDKPs. Short solution times are relevant for online MUCAs where agents (users) are not willing to wait several minutes until being informed about the auction's outcome.

4.2 Greedy Heuristics versus Weight Coding

The following experiments compare the solution quality of Raidl's weight-coded approach to the primal greedy heuristics from Section 3.1. Furthermore, we extend Raidl's study [28] and examine how the solution quality of weight-coded EAs depend on the underlying primal greedy heuristic. As the dual greedy heuristic is supposed to perform similarly to the primal greedy heuristic, we consider only the latter.

The performance of different methods are measured by $gapB$ and $gapR$ (in percent) which are the gaps between the best found solution and the optimal solution ($gapB$) and relaxed optimal solution ($gapR$), respectively. $gapR$ ($gapR \geq gapB$) is relevant for problems where the optimal solution is not known. Table 3 shows the experimental results. We only show running times for the Raidl EA, since all construction heuristics need only a fraction of a second. Furthermore, we only provide results for MKNAPCB test instances since all MUCA test instances can be solved optimally by CPLEX in a few seconds.

Table 2: Performance of primal greedy heuristics on MUCA instances in %.

m	n	α	NBP gapB	SNBP gapB	RLPS gapB	SS gapB
LB						
10	1500	.0023	2.844	18.287	4.722	9.617
14	2500	.0018	7.350	12.209	6.482	6.722
MULTIPATH						
595	1500	0.05	6.328	16.131	4.110	4.565

The results reveal that $gapR$ ranges from about 0.03% up to 11.5%, whereas $gapB$ is constantly below 10%. Comparing the different primal greedy heuristics, RLPS (avg. $gapB = 0.93$ %) and SS (avg. $gapB = 1.56$ %) show the highest average performance. However, the situation is slightly different for small LB instances (see Table 2) since then NBP performs the best. Therefore, the results indicate that the performance of greedy heuristics depends on the specific type of problems and there is no unique optimal heuristic. As expected [28], the gaps are lower for the weight-coded EA than for the greedy heuristics. However, the EAs need a running time of up to 364 seconds, while the heuristics only take some ms.

Table 3 indicates that the gaps decrease with larger tightness ratio α. This influence was also partially studied by Raidl and Gottlieb [29] who only considered the relaxed gap $gapR$, and concluded that there is no general evidence that the performance of greedy heuristics increases with larger α. Instead, the trend is due to the tighter bound of the relaxed LP for higher α. The presented results extend previous work and are contradictory to the theory of Raidl and Gottlieb as $gapB$ also decreases with larger α. We are able to confirm this hypothesis by an analysis of variances (ANOVA) of $gabB$ and α for a level of significance of 5%. As a dependent value we used $gapB$ averaged over all five algorithms. The ANOVA shows that with increasing α, the gap $gapB$ decreases significantly ($M = 3.74$ vs. 1.86 vs. 1.07, $F = 8.749$, $p < 0.001$).

Raidl [28] recommended studying how EA performance depends on the type of the underlying greedy heuristic. Consequently, Table 4 compares the $gapR$ of the EA for different greedy heuristics. We have chosen $gapR$ as $gapB$ is not available for larger problem instances. Furthermore, we only present results for NBP and SNBP because RLPS creates relaxed optimal solutions where only up to m variables are fractional and all others are either 0 or 1. Therefore, in the initial population the diversification is very low and duplicate elimination continuously discards new initial solutions (prohibiting the creation of a complete initial population). Again, the presented results are averaged over ten problem instances. The parameter γ was chosen with respect to the

Table 3: Performance of primal greedy heuristics and Raidl EA for MKNAPCB test instances.

			NBP[%]		SNBP[%]		RLPS[%]		SS[%]		Raidl-SS*[%]		
m	n	α	gapR	gapB	gapR	gapB	gapR	gapB	gapR	gapB	gapR	gapB	time[sec.]
MKNAPCB													
5	100	.25	9.611	8.706	7.453	6.525	2.452	1.478	2.530	1.557	1.011	0.023	33.26
		.50	4.595	4.163	3.203	2.764	1.283	0.835	1.621	1.175	0.465	0.014	36.71
		.75	2.696	2.386	1.695	1.381	0.877	0.561	0.981	0.665	0.325	0.007	35.98
5	250	.25	7.515	7.310	4.657	4.446	0.836	0.617	1.006	0.787	0.289	0.069	129.58
		.50	4.126	4.022	2.113	2.006	0.509	0.401	0.491	0.383	0.135	0.026	115.08
		.75	2.192	2.117	1.009	0.933	0.301	0.235	0.324	0.248	0.083	0.007	89.16
5	500	.25	7.012	6.945	3.170	3.101	0.455	0.383	0.398	0.326	0.136	0.065	307.89
		.50	3.626	3.587	2.148	2.108	0.252	0.212	0.219	0.179	0.059	0.021	268.65
		.75	2.247	2.221	1.115	1.089	0.155	0.129	0.141	0.116	0.033	0.009	222.85
10	100	.25	11.099	9.687	8.959	7.513	3.612	2.082	4.695	3.182	1.730	0.171	39.64
		.50	5.255	4.500	4.029	3.263	2.000	1.218	2.320	1.541	0.846	0.055	39.03
		.75	2.821	2.350	3.086	2.616	1.140	0.661	1.618	1.141	0.502	0.019	40.79
10	250	.25	7.912	-	6.005	-	1.429	-	1.998	-	0.653	-	132.87
		.50	3.952	-	3.435	-	0.737	-	0.924	-	0.294	-	125.33
		.75	1.852	-	1.575	-	0.383	-	0.516	-	0.169	-	102.55
10	500	.25	6.367	-	3.524	-	0.665		1.006	-	0.354	-	282.42
		.50	3.270	-	2.347	-	0.338	-	0.408	-	0.156	-	291.38
		.75	1.636	-	1.248	-	0.205	-	0.290	-	0.089	-	241.28
30	100	.25	11.394	8.758	11.526	8.896	6.098	3.306	9.893	7.215	3.164	0.286	51.31
		.50	6.431	5.179	5.975	4.717	2.181	0.873	4.453	3.175	1.411	0.093	53.34
		.75	3.480	2.678	3.318	2.514	1.788	0.972	2.530	1.720	0.859	0.035	61.84
30	250	.25	7.899	-	7.186	-	2.634	-	4.393	-	1.473	-	154.3
		.50	5.001	-	4.206	-	1.077	-	1.719	-	0.617	-	157.31
		.75	2.267	-	2.214	-	0.693	-	1.269	-	0.345	-	140.09
30	500	.25	6.972	-	5.515	-	1.259	-	2.155	-	0.884	-	356.4
		.50	3.199	-	2.829	-	0.659	-	0.979	-	0.356	-	294.13
		.75	1.624	-	1.542	-	0.315	-	0.603	-	0.206	-	364.74
average			5.039	4.974	3.897	3.592	1.272	0.931	1.864	1.560	0.617	0.06	154.37

used heuristic.

Table 4: EA performance for different heuristics.

m	n	α	Raidl-SS* $\gamma = 0.05$	Raidl-SNBP* $\gamma = 0.08$	Raidl-NBP* $\gamma = 0.5$
MKNAPCB					
5	100	.25	1.010	1.004	1.021
		.50	0.465	0.455	0.453
		.75	0.325	0.318	0.318
10	250	.25	0.653	0.614	0.785
		.50	0.294	0.298	0.354
		.75	0.169	0.185	0.209
30	500	.25	0.884	0.946	1.300
		.50	0.356	0.402	0.516
		.75	0.208	0.255	0.279
average			0.485	0.497	0.582

We tested the hypothesis H_0 that the three averaged gaps do not differ significantly (significance level 5%) using an ANOVA. The hypotheses could not be rejected ($p = 0.79$). However, H_0 can only be accepted with caution since a Post-Hoc analysis showed that for an average effect size of 0.06 the power was too low. In summary, the results show that although greedy performance is significantly different (Table 3), the resulting EA performance seems to be independent of the used greedy heuristic.

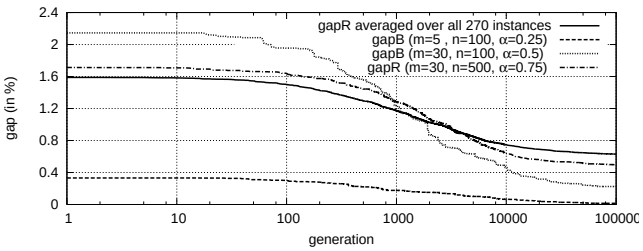

Figure 1: Anytime performance of weight-coded EA

4.3 Anytime Performance of Weight-coded EA

Finally, we study the anytime performance of the weight-coded approach. The objective is to analyze the trade-off between solution quality and running-time. We postulate that excellent solutions are already created in the initialization phase, since the initial population is located near the solutions of the primal greedy heuristic. Furthermore, we suppose that the increase in performance in later generations is low.

Figure 1 shows the average gaps (either *gapR* or *gapB*) towards optimal solutions over the number of generations averaged over all 270 instances and, separately, for three

different MKNAPCB problem instances (*gapB* for $m = 5$, $n = 100$, $\alpha = 0.25$; *gapB* for $m = 30$, $n = 100$, $\alpha = 0.5$; *gapR* for $m = 30$, $n = 500$, $\alpha = 0.75$). We only plot the first 100,000 generations. The results indicate that using a population size $N = 100$ already results in very small gaps (1.59% in average) in the initial population. Initial solutions have high fitness as they are only slightly different from the solutions generated by the underlying greedy heuristic. In the first few generations, the gap decreases and stays approximately constant after a few hundred or thousand generations (please note that one generation is equivalent to one fitness evaluation). In conclusion, the high number of fitness evaluations for the EA proposed by Raidl [28] are not necessary and can be greatly reduced to a few hundred or thousand evaluation steps with only a minor effect on the resulting solution performance.

5. SUMMARY AND CONCLUSIONS

This paper provides an integrated view of the MDKP and the WDP for which optimization methods as well as test instances have been developed independently of each other. The paper carefully studies the performance of different algorithms like CPLEX, simple greedy heuristics, and a weight-coded EA proposed by Raidl [28] for existing MDKP and WDP test problems. In particular, we examine the performance of current CPLEX implementations, compare the performance of different heuristics, study how the performance of heuristics depends on characteristics of the underlying test problem, and examine the anytime performance of weight-coded EAs.

The results show that all currently used WDP test instances as well as some small MDKP test instances can be easily and optimally solved by CPLEX in a short time. Therefore, the currently used WDP test instances are not difficult and thus of only limited usefulness for estimating the performance of new optimization approaches for the WDP. The study of heuristics for larger MDKP test instances confirms that simple greedy heuristics are able to find high-quality solutions in a few milliseconds. For all test instances, the gaps between solutions found by the greedy heuristics and the optimal, or relaxed optimal solution are only a few percent. Furthermore, the gap between the solution found by the heuristics and the optimal solution decreases with an increasing tightness ratio of the WDKP. When taking into account MDKP as well as WDP test problems, there is no heuristic that consistently outperforms the others.

The good results of the heuristics can significantly be improved by Raidl's weight-coded EA [28] at the expense of a much higher computational effort. Studying the anytime performance of the weight-coded EA shows that the main quality improvement occurs in the initial population and in the very early stages of an EA run. Therefore, the high running time proposed by Raidl [28] can be greatly reduced with only minor effects on the resulting solution quality. To design fast solution approaches is, for example, necessary for Internet-based online combinatorial auctions where solution time is a critical factor.

6. REFERENCES

[1] J. Bai, H. Chang, and Y. Yi. An immune partheno-genetic algorithm for winner determination in combinatorial auctions. In L. Wang, K. Chen, and Y.-S. Ong, editors, *Proceedings of Advances in Natural Computation, First International Conference (ICNC 2005)*, volume 3612 of *LNCS*, pages 74–85. Springer, 2005.

[2] A. Caprara, H. Kellerer, U. Pferschy, and D. Pisinger. Approximation algorithms for knapsack problems with cardinality constraints. *European Journal of Operation Research*, 123:333–345, 2000.

[3] P. C. Chu and J. E. Beasley. A genetic algorithm for the multidimensional knapsack problem. *Journal of Heuristics*, 4(1):63–86, 1998.

[4] G. Dobson. Worst-case analysis of greedy heuristics for integer programming with nonnegative data. *Mathematics of Operations Research*, 7:515–531, 1982.

[5] R. Epstein, L. Henríques, J. Catalán, G. Y.Weintraub, and C. Martínez. A combinatorial auction improves school meals in chile. *Interfaces*, 32, Issue 6:1–14, 2002.

[6] S. Fidanova. ACO algorithm for MKP using various heuristic information. In I. Dimov, I. Lirkov, S. Margenov, and Z. Zlatev, editors, *Numerical Methods and Application*, volume 2542 of *LNCS*, pages 438–444. Springer, 2002.

[7] G. E. Fox and G. D. Scudder. A heuristic with tie breaking for certain 0-1 integer programming models. *Naval Research Logistics Quarterly*, 32:613–623, 1985.

[8] Y. Fujishima, K. Leyton-Brown, and Y. Shoham. Taming the computational complexity of combinatorial auctions: Optimal and approximate approaches. In D. Thomas, editor, *Proceedings of the 16th International Joint Conference on Artificial Intelligence 1999 (IJCAI-99)*. Morgan Kaufmann Publishers, 1999.

[9] J. E. Gallardo, C. Cotta, and A. J. Fernández. Solving the multidimensional knapsack problem using an evolutionary algorithm hybridized with branch and bound. In J. Mira and J. R. Álvarez, editors, *Artificial Intelligence and Knowledge Engineering Applications: A Bioinspired Approach: First International Work-Conference on the Interplay Between Natural and Artificial Computation (IWINAC-2005)*, volume 3562 of *LNCS*, pages 21–30, 2005.

[10] F. Glover and G. Kochenberger. Critical event tabu search for multidimensional knapsack problems. In I. Osman and J. Kelly, editors, *Meta-Heuristics: The theory and applications*, pages 407–427. Kluwer Academic Publishers, 1996.

[11] R. Gonen and D. J. Lehmann. Optimal solutions for multi-unit combinatorial auctions: Branch and bound heuristics. In *Proceedings of the 2nd ACM Conference on Electronic Commerce (EC-00)*, pages 13–20. ACM Press, 2000.

[12] R. Gonen and D. J. Lehmann. Linear programming helps solving large multi-unit combinatorial auctions. Technical Report 2001-8, Hebrew University, Leibniz Center for Reaseach in Computer Science, Jerusalem, Israel, 2002.

[13] J. Gottlieb. *Evolutionary algorithms for constrained optimization problems*. PhD thesis, Technische Universität Clausthal, 1999.

[14] R. Graves, L. Schrage, and J. Sankaran. An auction method of course registration. *Interfaces*, 23:81–92,

1993.

[15] Y. Guo, A. Lim, B. Rodrigues, and Y. Zhu. A non-exact approach and experiment studies on the combinatorial auction problem. In *Thirty-Eight Hawaii International Conference on System Sciences (HICSS-38)*, pages 82–89, 2005.

[16] S. Hanafi and A. Freville. An efficient tabu search approach for the 0-1 multidimensional knapsack problem. *European Journal of Operational Research*, 106:659–675, 1997.

[17] R. C. Holte. Combinatorial auctions, knapsack problems, and hill-climbing search. In E. Stroulia and S. Matwin, editors, *Canadian Conference on AI*, pages 57–66. Springer, 2001.

[18] H. H. Hoos and C. Boutilier. Solving combinatorial auctions using stochastic local search. In *Proceedings of the 7th Conference on Artificial Intelligence (AAAI-00) and of the 12th Conference on Innovative Applications of Artificial Intelligence (IAAI-00)*, pages 22–29, Menlo Park, CA, 2000. AAAI Press.

[19] R. M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*. Plenum Press, 1972.

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

[21] T. Kelly. Generalized knapsack solvers for multi-unit combinatorial auctions. Technical Report HPL-2004-21, HP Laboratories Palo Alto, 2004.

[22] J. Ledyard, M. Olson, D. Porter, J. Swanson, and D. Torma. The first use of a combined-value auction for transportation servies. *Interfaces*, 32:4–12, 2002.

[23] D. Lehmann, L. I. O'Callaghan, and Y. Shoham. Truth revelation in rapid, approximately efficient combinatorial auctions. In *Proceedings of the ACM Conference on Electronic Commerce (EC-99)*, pages 96–102. ACM Press, 1999.

[24] K. Leyton-Brown, M. Pearson, and Y. Shoham. Towards a universal test suite for combinatorial auction algorithms. In *Proceedings of the 2nd ACM Conference on Electronic Commerce (EC-00)*, pages 66–76, N.Y., 2000. ACM.

[25] K. Leyton-Brown, Y. Shoham, and M. Tennenholtz. An algorithm for multi-unit combinatorial auctions. In *Proceedings of the 7th Conference on Artificial Intelligence (AAAI-00) and of the 12th Conference on Innovative Applications of Artificial Intelligence (IAAI-00)*, pages 56–61, Menlo Park, CA, 2000. AAAI Press.

[26] M. A. Osorio, F. Glover, and P. Hammer. Cutting and surrogate constraint analysis for improved multidimensional knapsack solutions. *Annals of Operations Research*, 117:71–93, 2002.

[27] H. Pirkul. A heuristic solution procedure for the multiconstraint zero-one knapsack problem. *Naval Research Logistics*, 43:161–172, 1987.

[28] G. Raidl. Weight-codings in a genetic algorithm for the multiconstraint knapsack problem. In P. J. Angeline, Z. Michalewicz, M. Schoenauer, X. Yao, and A. Zalzala, editors, *Proceedings of the Congress on Evolutionary Computation*, volume 1, pages 596–603. IEEE Press, 1999.

[29] G. Raidl and J. Gottlieb. Empirical analysis of locality, heritability and heuristic bias in evolutionary algorithms: A case study for the multidimensional knapsack problem. *Evolutionary Computation*, 13, issue 4:441–475, 2005.

[30] M. Rothkopf, A. Pekec, and R. M. Harstad. Computationally manageable combinatorial auctions. *Management Science*, 44(8):1131–1147, 1998.

[31] F. Rothlauf and D. E. Goldberg. Redundant representations in evolutionary computation. *Evolutionary Computation*, 11(4):381–415, 2003.

[32] Y. Sakurai, M. Yokoo, and K. Kamei. An efficient approximate algorithm for winner determination in combinatorial auctions. In *Proceedings of ACM Conference on Electronic Commerce (EC-00)*, pages 30–37, 2000.

[33] T. Sandholm. Algorithm for optimal winner determination in combinatorial auctions. *Artificial Intelligence*, 135:1–54, 2002.

[34] T. Sandholm, S. Suri, A. Gilpin, and D. Levine. CABOB: A fast optimal algorithm for winner determination in combinatorial auctions. *Management Science*, 51(3):374–390, March 2005.

[35] S. Senju and Y. Toyoda. An approach to linear programming with 0-1 variables. *Management Science*, 15:196–207, 1968.

[36] J. Thiel and S. Voss. Some experiences on solving multiconstraint zero-one knapsack problems with genetic algorithms. *INFOR*, 32(4):226–242, 1994.

[37] Y. Toyoda. A simplified algorithm for obtaining approximate solutions to zero-one programming problems. *Management Science*, 21:1417–1427, 1975.

[38] E. Zurel and N. Nisan. An efficient approximate allocation algorithm for combinatorial auctions. In *Proceedings of the ACM Conference on Electronic Commerce*, pages 125–136, Tampa, Floridam, USA, 2001. ACM Press.