Cooperating Memetic and Branch-and-Cut Algorithms for Solving the Multidimensional Knapsack Problem

Jakob Puchinger* Günther R. Raidl* Martin Gruber*

*Institute of Computer Graphics and Algorithms, Vienna University of Technology
Favoritenstrae 9-11/1861, 1040 Vienna, Austria
{puchinger,raidl,gruber}@ads.tuwien.ac.at

1 Introduction

Recent work in combinatorial optimization indicates the high potential of combining metaheuristics with integer linear programming (ILP) techniques. We study here a hybrid system in which a memetic algorithm (MA) and a general purpose ILP solver based on branch-and-cut (B&C) are executed in parallel and continuously exchange information in a bidirectional, asynchronous way. As target problem, we consider the multidimensional knapsack problem (MKP). The memetic algorithm uses a direct binary encoding of candidate solutions and repair and local improvement strategies that are steered by pseudo-utility ratios. As B&C framework we use the general purpose commercial ILP-solver CPLEX. The information exchanged between the two heterogenous algorithms are so-far best primal solutions and promising dual variable values of solutions to certain linear programming (LP) relaxations. These dual variable values are used in the MA to update the pseudo-utility ratios of local improvement and repair.

We will see that this combination of a metaheuristic and an exact optimization method is able to benefit from synergy: Experimental results document that within the same limited total time, the cooperative system yields better heuristic solutions than each algorithm alone. In particular, the cooperative system also competes well with today's best algorithms for the MKP, needing substantially shorter total running times.

The next section introduces the MKP formally and gives some references to state-of-the-art algorithms for solving it. Section 3 explains the MA-part. The used ILP formulation and B&C solver are described in Sect. 4. Details about the parallel execution and asynchronous communication are given in Sect. 5. Finally, Sect. 6 presents exemplary results, and conclusions are drawn in Sect. 7.

This work is supported by the RTN ADONET under grant 504438 and the Austrian Science Fund (FWF) under grant P16263-N04.

2 The Multidimensional Knapsack Problem

The MKP is a well-studied, strongly NP-hard combinatorial optimization problem occurring in many different applications. It can be defined by the following ILP:

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

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

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

Given are n items with profits $p_j > 0$ and m resources with capacities $c_i > 0$. Each item j consumes an amount $w_{ij} \geq 0$ from each resource i. The goal is to select a subset of the items with maximum total profit, see (1); chosen items must, however, not exceed resource capacities, see (2). The 0–1 decision variables x_j indicate which items are selected.

A general overview on practical and theoretical results for the MKP can be found in the monograph by Kellerer et al. [4]. Besides exact techniques for solving small to moderately sized instances, many kinds of metaheuristics have already been applied to the MKP.

To our knowledge, the method currently yielding the best results, at least for commonly used benchmark instances, has been described by Vasquez and Hao [7] and was recently refined by Vasquez and Vimont [8]. The search space is reduced and partitioned via additional constraints, 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 [8] lies mainly in an additional variable fixing heuristic.

Besides this tabu search approach, various other metaheuristics have been described for the MKP, including several variants of hybrid evolutionary algorithms (EAs); see [6] for a recent survey and comparison of EAs for the MKP. The basics of today's most effective EAs go back to Chu and Beasley [1]: Candidate solutions are directly represented by their 0–1 vectors x; standard crossover and mutation operators and – most importantly – clever repair and local improvement strategies are applied. Such combinations of EAs with problem-specific local improvement strategies are in general also called memetic algorithms, and we adopt this nomenclature in the following.

3 The Memetic Algorithm Part

The MA, which we consider here for parallel execution with B&C, is based on the principles from Chu and Beasley and includes some improvements suggested in [5, 3, 6]. The MA framework is steady-state. The creation of initial solutions is guided by the LP-relaxation of the MKP, as described in [3]. 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 to 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, where 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 capacity 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. Various heuristic measures can be found in the literature for calculating utility values for estimating the likelihood with which each item appears in an optimal solution. For the unidimensional knapsack problem ($m = 1$), for example, ordering items according to increasing ratios p_j/w_{1j} is straight-forward and works generally well. In case of the MKP, an appropriate choice is much more difficult.

As in [1], we determine Π by ordering the items according to pseudo-utility ratios

$$u_j = \frac{p_j}{\sum_{i=1}^{m} a_i w_{ij}}, \tag{4}$$

where we set the surrogate multipliers a_i to the dual variable values (i.e. shadow prices of the i-th constraints) of the solution to the LP-relaxation of the MKP.

While this ordering Π remains fixed throughout the whole optimization if the MA runs independently, we will continuously adapt the surrogate multipliers according to more promising dual variable values when B&C is performed in parallel, see Sect. 5.

4 The Branch-and-Cut Part

The heuristics in [7, 8] exploit the property that good solutions to the MKP often lie in the neighborhood of the solution x^{LP} to the corresponding LP relaxation. We performed an empirical in-depth examination on smaller instances of Chu and Beasley's benchmark library [1] for which we were able to compute optimal solutions x^* and observed that the Hamming distance between x^* and the (possibly infeasible) rounded LP solution x^{RLP} with

$$x_j^{\mathrm{RLP}} = \lceil x_j^{\mathrm{LP}} - 0.5 \rceil, \quad j = 1, \ldots, n, \tag{5}$$

is almost always smaller than 10% of the total number of variables. Focusing the optimization to such a neighborhood seems therefore to be highly promising. This can be done by adding a single constraint to the MKP similar to the local branching constraints presented by Fischetti and Lodi [2]. The following inequality restricts the search space to a neighborhood of Hamming distance k around the rounded LP solution x^{RLP}:

$$\Delta(x, x^{\mathrm{RLP}}) = \sum_{j \in S^{\mathrm{LP}}} (1 - x_j) + \sum_{j \notin S^{\mathrm{LP}}} x_j \leq k, \tag{6}$$

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

In our implementation we use CPLEX as B&C system and initially partition the search space by constraint (6) into the more promising part and by the inverse constraint $\Delta(x, x^{\mathrm{RLP}}) \geq k+1$ into a second, remaining part. CPLEX is forced to first completely solve the neighborhood of x^{RLP} before considering the remaining search space.

5 Parallel Execution and Communication

The intention is to run the MA and the B&C approach in parallel on two individual machines. In our tests, however, we executed the algorithms in a pseudo-parallel way as individual processes on a single machine. 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 the population by replacing the worst solution, as in the case of any other newly created solution candidate. In B&C, a received solution is set as new incumbent solution, providing a new global lower bound.

When B&C finds a new incumbent solution, it also sends the current dual variable values associated to the MKP-constraints, which are devised from the LP relaxation of the currently processed node in the B&C tree. When the MA receives these dual variable values, it recalculates the pseudo-utility ratios and the item ordering Π for repair and local improvement as described in Sec. 3.

6 Computational Experiments

We considered four variants of the described algorithms: the independent MA, independent B&C, the cooperative approach exchanging best solutions only (CO-b), and the cooperative approach exchanging best solutions and dual variable values (CO-bd). The MA runs for 1 000 000 iterations and then restarts, keeping only the so-far best solution. The neighborhood size k is set to 10% of the number of variables n.

The approaches were tested on the three largest standard benchmark sets of Beasley's OR-Library (`http://people.brunel.ac.uk/~mastjjb/jeb/info.html`) consisting of a total of 90 instances with $n = 500$ items and $m \in \{5, 10, 30\}$ constraints. Each of these three instance sets is divided into three subsets of 10 instances with tightness ratios $\alpha = c_i / \sum_{j=1}^{n} w_{ij}$, $\alpha \in \{0.25, 0.5, 0.75\}$. The tested algorithms were implemented using GNU C++ 3.3.1 and CPLEX 9.0. All experiments were performed on a 2.8GHz Pentium 4 machine.

We consider here two different computational experiments, one with shorter total CPU-times of 850s, see Table 1, and one with longer total CPU-times of 1 800s, see Table 2. Preliminary tests with the cooperative approaches indicated that due to its relatively quick convergence, the MA mainly contributes to the finding of improved solutions during the early stages of the optimization process. We therefore started the MA and B&C at the same time but terminated the MA earlier. The MA was given 250s (400s in the long runs), while B&C was performed with a time limit of 600s (1 400s).

Table 1: Average performance over the 90 largest OR-Library instances; CPU-times: 850s.

		MA	B&C	CO-b	CO-bd
m	α	$\overline{z}$	$\overline{z}$	$\overline{z}$	$\overline{z}$
5	0.25	120 624	120 626	120 626	**120 627**
	0.5	219 511	219 511	**219 512**	219 510
	0.75	302 360	302 361	**302 362**	**302 362**
10	0.25	118 586	118 591	**118 600**	118 595
	0.5	217 295	217 306	**217 308**	**217 308**
	0.75	302 573	**302 582**	**302 582**	**302 582**
30	0.25	115 500	115 493	115 521	**115 523**
	0.5	216 192	216 171	216 192	**216 204**
	0.75	302 378	**302 390**	302 388	**302 390**

Table 2: Average performance over the 90 largest OR-Library instances; long runs.

		[7]		[8]		MA		B&C		CO-bd	
m	α	$\overline{z}$	t[h]	$\overline{z}$	t[h]	$\overline{z}$	t[h]	$\overline{z}$	t[h]	$\overline{z}$	t[h]
5	0.25	120 623	5	120 628	8.5	**120 629**	0.5	**120 629**	0.5	**120 629**	0.5
	0.5	219 507	5	**219 512**	8.5	219 509	0.5	**219 512**	0.5	219 511	0.5
	0.75	302 360	5	**302 363**	8.5	302 362	0.5	302 362	0.5	**302 363**	0.5
10	0.25	118 600	9	**118 629**	7.6	118 589	0.5	118 603	0.5	118 605	0.5
	0.5	217 298	9	**217 326**	7.6	217 296	0.5	217 310	0.5	217 317	0.5
	0.75	302 575	9	**302 603**	7.6	302 575	0.5	302 583	0.5	302 584	0.5
30	0.25	115 547	12	**115 624**	33	115 509	0.5	115 520	0.5	115 550	0.5
	0.5	216 211	12	**216 275**	33	216 196	0.5	216 213	0.5	216 222	0.5
	0.75	302 404	12	**302 447**	33	302 379	0.5	302 400	0.5	302 408	0.5

The results of the short runs (Table 1) show an advantage for the hybrid strategies. They always yield solutions with better or equal average objective values than the independent algorithms. Obviously, exchanging dual variable information is fruitful for the optimization process since the CO-bd approach yields the best average objective values for 7 out of the 9 instance sets of the OR-Library, whereas CO-b achieves best average objective values in only 5 out of the 9 instance sets. Especially for the instances with $m = 30$, the benefits of CO-bd become apparent.

The cooperative approach also outperforms the individual algorithms in the case of the longer runs (Table 2). CO-bd yields better or equal average objective values than the the independent MA and independent B&C for 8 out of the 9 instance sets. Additionally the results of the longer runs are compared to the results presented in [7] and [8] obtained on different computers (a 2GHz Pentium 4 machine was used in [8]). CO-bd yields almost equally good or better results than the algorithms presented in [7] and [8] for the instances with $m = 5$. For $m = 10$ and $m = 30$, better or equally good results than in [7] are achieved, whereas the solution quality of [8] could in general not be reached.

The approaches from [7] and [8] still achieve most of the best known solutions for the tested instances. However, the main drawbacks of these approaches are their huge running times of up to 33 hours for the largest OR-Library instances. Running CO-bd for up to 20 hours on one instance of each type indicated that the results of [8] can be reached by our approach in 6 of 9 cases.

7 Conclusions

We presented a cooperative combination of a memetic algorithm and a branch-and-cut approach for solving the MKP. The two heterogeneous algorithms run in parallel and asynchronously exchange information about the ongoing optimization process. Besides primal solutions, also dual variable values are communicated from B&C to the MA for updating its repair and local improvement strategies. The computational experiments we performed showed the benefits of the cooperative approach, which yields better results than the individually executed algorithms. The obtained results further indicate that this research direction is promising, since many of the best known results from the literature were achieved in substantially shorter times. Future research will be directed towards the exchange of more information about the ongoing search, such as statistical or negative information, and also towards other combinatorial optimization problems.

References

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

[2] M. Fischetti and A. Lodi. Local Branching. *Mathematical Programming Series B*, 98:23–47, 2003.

[3] J. Gottlieb. On the effectivity of evolutionary algorithms for multidimensional knapsack problems. In C. Fonlupt et al., editors, *Proceedings of Artificial Evolution: Fourth European Conference*, volume 1829 of *LNCS*, pages 22–37. Springer, 1999.

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

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

[6] G. R. 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 Journal*, 13(4), to appear 2005.

[7] M. Vasquez and J.-K. Hao. A hybrid approach for the 0–1 multidimensional knapsack problem. In *Proceedings of the International Joint Conference on Artificial Intelligence 2001*, pages 328–333, 2001.

[8] M. Vasquez and Y. Vimont. Improved results on the 0-1 multidimensional knapsack problem. *European Journal of Operational Research*, 165:70–81, 2005.