The Core Concept for 0/1 Integer Programming

Sam Huston[2] **Jakob Puchinger**[1,2] **Peter Stuckey**[1,2]

[1] NICTA Victoria Laboratory
[2] Department of Computer Science and Software Engineering,
University of Melbourne, Victoria 3010, Australia,
Email: {shuston,jakobp,pjs}@csse.unimelb.edu.au

Abstract

In this paper we examine an extension of the core concept for the 0/1 Multidimensional Knapsack Problem (MKP) towards general 0/1 Integer Programming (IP) by allowing negative profits, weights and capacities. The core concept provides opportunities for heuristically solving the MKP, achieving higher quality solutions and shorter runtimes than general IP methods. We provide the theoretical foundations of the extended core concept and further provide computational experiments showing that we can achieve similar computational behavior for extended MKP instances with negative weights, profits and capacities.

1 Introduction

The core concept for the 0/1 Multidimensional Knapsack Problem (MKP) (Puchinger et al. 2006, 2007) has been shown to be very effective in providing opportunities for heuristically solving the MKP, achieving higher quality solutions and shorter run-times than general IP methods. In this paper we will examine the possibilities of extending the core concept towards general 0/1 Integer Programming (IP).

The Multidimensional Knapsack Problem (MKP) is defined as:

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

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

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

where the profits p_j, the weights w_{ij}, and the capacities c_i are all positive. Allowing negative values for those parameters results in general 0/1 Integer Problems. This is because it is possible to transform any 0/1 IP into this format. (Bertsimas & Tsitsiklis 1997)

The aim of the core concept 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. All variables corresponding to items outside the core are fixed to their optimal values.

The *core* of a given MKP is defined with respect to some ordering of the variables in the problem. The ordering results in variables that are expected to be in the knapsack (set to one) occur before those which are not expected to be in the knapsack (set to zero). Given the optimal solution of the MKP the exact core is defined as the set of variables from the first variable that takes the value zero to the last variable that takes the value one. In order to devise an exact core for a given MKP, the optimal solution has to be known. However, being able to heuristically obtain good approximations to cores and solve those smaller problems, may lead to high quality solutions in short computational times.

The underlying concept of such a heuristic is to order the items of the MKP according to a specific efficiency measure. This ordering will allow to partition the items into three sections. The first section which contains items which are included in the knapsack (variables set to one). The second section, named the *approximate core*, containing the items which may or may not be included in the knapsack (variables set to either one or zero). Finally the third section contains the items which are not included in the knapsack (variables set to zero). The aim is to have the approximate core closely mimic the exact core.

The following example illustrates the core concept for a small 2-dimensional knapsack problem. The variables are ordered by an efficiency measure described later. The first line shows the optimal integer solution, while the second line shows the optimal solution to the LP-relaxation of the problem. The exact core is shown in bold in the first line, while an approximate core (adding 2 variables around the non 0-1 LP solution values) is shown in bold in the second line.

IP	1	1	**0**	**0**	**1**	**0**	**1**	0	0	0
LP	1	1	1	**1**	**1**	**0.96**	**0.23**	**0**	**0**	0

Copyright ©2008, Australian Computer Society, Inc. This paper appeared at Computing: The Australian Theory Symposium (CATS 2008), Wollongong, Australia. Conferences in Research and Practice in Information Technology, Vol. 77. James Harland and Prabhu Manyem, Ed. Reproduction for academic, not-for profit purposes permitted provided this text is included.

In the remainder of the paper, we first introduce the core concept in the context of its previous uses. We then extend the efficiency measure used for MKPs to general 0/1 Integer Programs, and prove that the efficiency measure is tightly related to the optimal solution. In Section 4 we show the result of experiments illustrating the effectiveness of the approximate core computations, the loss of precision that arises from restricting the problem to an approximate core, and the improvement in best solutions found if we use approximate cores.

2 Background

2.1 The Multidimensional Knapsack Problem

A comprehensive overview of practical and theoretical results for the MKP can be found in the monograph on knapsack problems by Kellerer et al. (Kellerer et al. 2004). Solving the MKP with heuristic methods seems to be the method of choice for the bigger instances described in the literature, since no exact method is known for solving these instances to optimality. Besides exact techniques for solving small to moderately sized instances, based on dynamic programming (Gilmore & Gomory 1966, Weingartner & Ness 1967) and branch-and-bound (Shih 1979, Gavish & Pirkul 1985), many kinds of meta-heuristics have been applied to the MKP (Glover & Kochenberger 1996, Chu & Beasley 1998). See (Raidl & Gottlieb 2005) for a recent survey and comparison of evolutionary algorithms for the MKP. The hybrid tabu-search methods presented in (Vasquez & Hao 2001, Vasquez & Vimont 2005) are currently yielding the best known results for the commonly used benchmark instances.

2.2 The core concept for KP and MKP

The core concept was first presented for the classical 0/1-Knapsack Problem (KP) (Balas & Zemel 1980) and led to very successful knapsack algorithms (Martello & Toth 1988, Pisinger 1995, 1997). These ideas were also studied in the context of bi-criteria knapsack problems in (Gomes da Silva et al. 2005). The core concept was successfully extended to the MKP (Puchinger et al. 2006, 2007), leading to highly competitive heuristic algorithms.

It should be noted here that the KP core concept is not effective for producing good heuristic solutions for strongly correlated problem instances. Pisinger (Pisinger 1995) discusses why these problems are difficult to solve using the core concept. We would expect similar results for strongly correlated general 0/1 IP problems.

The classical greedy heuristic for KP packs the items into the knapsack in decreasing order of their *efficiencies* $\frac{p_j}{w_j}$ as long as the knapsack constraint is not violated. 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.

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

$$a := \min\{j \mid x_j^* = 0\}, \qquad b := \max\{j \mid x_j^* = 1\}. \tag{4}$$

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. If the split item would not be part of the core, the core would either start after the split item or end before it. The first case is impossible, since this would break the capacity constraint. The second case would contradict the optimality of the solution, because we would still be able to add more items to the knapsack without violating the capacity constraint.

These ideas have been expanded to MKP without major difficulties (Puchinger et al. 2006, 2007). The main difference is that the choice of the efficiency measure is not obvious for the MKP any more. The efficiency measure:

$$e_j = \frac{p_j}{s_j},$$

where $s_j = \sum_{i=1}^{m} u_i w_{ij}$, and u_i are the dual variable values of the LP-relaxation of the MKP, provided the best theoretical and practical results.

Let x^* be an optimal solution and assume that the items are sorted according to the decreasing efficiency measure e, then define

$$a := \min\{j \mid x_j^* = 0\} \quad \text{and} \quad b := \max\{j \mid x_j^* = 1\}. \tag{5}$$

The core is given by the items in the interval $C := \{a, \ldots, b\}$, and the core problem (MKPC) is defined as

$$\text{maximize} \quad z = \sum_{j \in C} p_j x_j + \tilde{p} \tag{6}$$

$$\text{subject to} \quad \sum_{j \in C} w_{ij} x_j \leq c_i - \tilde{w}_i, \; i = 1, \ldots, m \tag{7}$$

$$x_j \in \{0, 1\}, \quad j \in C, \tag{8}$$

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

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* $S := \{s, \ldots, t\}$, where s and t are the first and the last index of variables with fractional values after sorting by efficiency e.

The split interval S_e has been precisely characterized. Let x^{LP} be the optimal solution of the LP-relaxation of MKP.

Theorem 1 ((Puchinger et al. 2006, 2007)) *For efficiency values e_j we have:*

$$x_j^{\mathrm{LP}} = \begin{cases} 1 & \textit{if} \;\; e_j > 1\,, \\ \in [0,1] & \textit{if} \;\; e_j = 1\,, \\ 0 & \textit{if} \;\; e_j < 1\,. \end{cases} \tag{9}$$

3 Extending the core concept

As mentioned above, the main goal of this paper is to extend the core concept to general 0/1 Integer Programs. We show how the efficiency measure for the classical MKP problem can be adapted in such a way that the ordering of the variables according to this measure remains valuable for devising good approximate cores. We further provide a characterization of the structure of the LP relaxation of the 0/1 Integer Program.

We take all 0/1 IP problems to be transformable into the same structure as MKP, see equation 1.

Such a transformation is possible for any 0/1 IP, see any linear programming text book, (e.g. (Bertsimas & Tsitsiklis 1997)). This means that the only difference between MKP and this formulation of 0/1 IP problems is that the coefficients are permitted to take either positive or negative values.

We extend the previously defined efficiency measure e_j to create a tuple based measure. We introduce ordering variables, o_j which take values representing a section of the ordering. Variables x_j are sorted in decreasing (lexicographic) order of efficiency (o_j, e_j), in other words they are first sorted by section variable, o_j, and then efficiency value e_j.

These extensions are implemented as indicated in equation 10:

$$(o_j, e_j) = \begin{cases} (7, \frac{p_j}{s_j}) & \text{if } p_j > 0 \wedge s_j < 0 \\ (6, p_j) & \text{if } p_j > 0 \wedge s_j = 0 \\ (5, \frac{1}{s_j}) & \text{if } p_j = 0 \wedge s_j < 0 \\ (4, \frac{p_j}{s_j}) & \text{if } p_j > 0 \wedge s_j > 0 \\ (4, \frac{s_j}{p_j}) & \text{if } p_j < 0 \wedge s_j < 0 \\ (4, 1) & \text{if } p_j = 0 \wedge s_j = 0 \\ (3, \frac{1}{s_j}) & \text{if } p_j = 0 \wedge s_j > 0 \\ (2, p_j) & \text{if } p_j < 0 \wedge s_j = 0 \\ (1, \frac{p_j}{s_j}) & \text{if } p_j < 0 \wedge s_j > 0 \end{cases} \tag{10}$$

The ordering is designed to minimize the size of the split interval, which is completely contained in section $o_j = 4$. Our experiments in Section 4 show that the center of the core and the center of the split interval are close for our benchmark instances.

The sections $o_j \in \{7, 6, 5\}$ contain items that are purely beneficial: they either increase profit or "on average" remove weight from the optimal solution without decreasing profit. They are ordered to maximize profit and removal of weight. Similarly the sections $o_j \in \{1, 2, 3\}$ contain items that are purely detrimental to the problem: they either decrease profit or add weight. Again they are ordered to maximize profit and removal of weight.

Using this efficiency measure the nature of the split interval can be characterized as follows. Let x^{LP} be the optimal solution of the LP-relaxation of general 0/1 IP.

Theorem 2

$$x_j^{LP} = \begin{cases} 1 & \textit{if } e_j > 1 \textit{ or } o_j > 4, \\ \in [0,1] & \textit{if } \; e_j = 1 \textit{ and } o_j = 4, \\ 0 & \textit{if } 0 \le e_j < 1 \textit{ or } o_j < 4. \end{cases} \tag{11}$$

Proof The dual LP (DLP) associated with the LP-relaxation of the general 0/1 IP formulation is given by:

$$\text{minimise} \sum_{i=1}^{m} c_i u_i + \sum_{j=1}^{n} v_j \tag{12}$$

$$\text{subject to} \sum_{i=1}^{m} w_{ij} u_i + v_j \ge p_j, j = 1, ..., n \tag{13}$$

$$u_i, v_j \ge 0, i = 1, ..., m, j = 1, ..., n, \tag{14}$$

where u_i are the dual variables corresponding to the problem's constraints and each v_j corresponds to the inequality $x_j \le 1$. For the optimal primal and dual solutions the following complementary slackness conditions hold for $j = 1, ..., n$. .

$$x_j \left(\sum_{i=1}^{m} w_{ij} u_i + v_j - p_j \right) = 0 \tag{15}$$

$$v_j \left(x_j - 1 \right) = 0 \tag{16}$$

(For more details on linear programming duality and complementary slackness conditions, refer to any textbook on linear programming, e.g.(Bertsimas & Tsitsiklis 1997).)

We illustrate the result for each section in the definitions of the ordered tuple (o_j, e_j).

Consider the top 3 sections ($o_j \in \{7, 6, 5\}$). Clearly in each case we have that $p_j > s_j$. Hence satisfying equation (13) requires that $v_j > 0$. Therefore equation (16) implies that $x_j = 1$.

Consider the bottom 3 sections ($o_j \in \{1, 2, 3\}$). Clearly in each case $p_j < s_j$. Hence the expression $\sum_{i=1}^{m} w_{ij} u_i + v_j - p_j$ or equivalently $s_j + v_j - p_j$ is greater than 0, since $v_j \ge 0$. In order to satisfy equation 15 $x_j = 0$.

For section $(4, \frac{p_j}{s_j})$ we consider two cases. If $e_j = \frac{p_j}{s_j} > 1$ then $p_j > s_j$ since $s_j > 0$ and the same reasoning as for cases $o_j \in \{7, 6, 5\}$ applies, while if $e_j < 1$ then $p_j < s_j$ and the same reasoning as cases $o_j \in \{1, 2, 3\}$ applies.

For section $(4, \frac{s_j}{p_j})$ we consider two cases. If $e_j = \frac{s_j}{p_j} > 1$ then $p_j > s_j$ since $p_j < 0$ and the same reasoning as for cases $o_j \in \{7, 6, 5\}$ applies,

while if $e_j < 1$ then $p_j < s_j$ and the same reasoning as cases $o_j \in \{1, 2, 3\}$ applies.

For the remaining case, $e_j = 1$ and $o_j = 4$, there is nothing to prove.

□

The ordering within the top and bottom groups $o_j \in \{1, 2, 3\}$, and $o_j \in \{5, 6, 7\}$ is not set by the proof, these orderings are based upon maximizing profit. Other possible orderings could be considered for these groups without changing our characterization of the split interval.

An illustration of the ordering (o_j, e_j) is given in Table 1. This example shows some of the sections described above. As predicted by the theorem, the split interval exists entirely within section $o_j = 4$.

4 Computational experiments

4.1 Benchmark Problems

All of the following computational experiments were performed on a 3GHz Intel Pentium D with 4 Gb RAM, using the programming language Mercury and the commercial mixed integer programming solver CPLEX 10.0.

In order to study the core concept on 0/1 IP we generated example problems using the Chu and Beasley (Chu & Beasley 1998) benchmark instances for the MKP, as the starting point.

These benchmark problems consist of classes of randomly created instances for each combination of $n \in \{100, 250, 500\}$ items, $m \in \{5, 10, 30\}$ constraints and tightness ratios $0.25, 0.50.75$. The tightness ratio refers to the ratio between the constraint value and the sum of the corresponding weights.

$$\alpha = \frac{c_i}{\sum_{j=1}^{n} w_{ij}} \in \{0.25, 0.5, 0.75\} \qquad (17)$$

Weights are integers between 0 and 1000. Profits are generated by the equation:

$$p_j = \sum_{i=1}^{m} \frac{w_{ij}}{m} + [500 r_i] \in \{0.25, 0.5, 0.75\}$$

where r_i is a random number generated from $(0, 1]$. For each permutation of n, m and α, 10 instances are provided.

In order to generate 0/1 Integer Programs with negative values we multiply a random percentage of the weights and profits of a given problem by -1. The percentages used are $5\%, 10\%, 20\%$. Larger percentages of negative values were tried, however the problems quickly became optimally solvable in short run-times.

The set of profits and the set of all weights are operated on separately. This ensures that there is a fixed percentage of negative profits and a fixed percentage of negative weights. Combinations of different percentages of negative weights and profits were tried, however almost invariably this made the problems easier and thus faster to solve.

This process will change the tightness ratio. In order to maintain the tightness ratio the capacities have to be adjusted:

$$\hat{c}_i = \frac{c_i * \sum_{j=1}^{n} \hat{w}_{i,j}}{\sum_{j=1}^{n} w_{i,j}},$$

$w_{i,j}$ represents the original weights, $\hat{w}_{i,j}$ represents the adjusted weights, and $\hat{c}_i$ represents the new constraint value. Since the sum of the adjusted weights may become negative, it is possible that the new capacity $\hat{c}_i$ will also be negative.

It can be seen that the generated problems are general 0/1 IP problems. There are nine classes of generated problems for each combination of n and m, corresponding to three different tightness ratios, and the three percentages of negative coefficients.

4.2 0/1 IP Core Analysis

We provide empirical results supporting our adaptation of the efficiency measure e, (see Table 2). This table shows information about actual cores when the above efficiency function is utilized. The problems shown in these tables are based upon the smaller instances in Chu and Beasley's benchmark library (Chu & Beasley 1998). Specifically these problems use $n = 100, m \in \{5, 10\}$, and $n = 250, m = 5$. These problems were chosen because they are solvable in reasonable runtime. This means that the optimal solutions can be found, and the size of the core can be determined.

The tables show the averaged values over 10 problem instances. Average values listed include size of the split interval ($|S_e|$), size of the exact core ($|C_e|$), percentage that the split interval covers the exact core (ScC), percentage that the exact core covers the split interval (ScC), and the distance between the center of the split interval and the center of the exact core ($|C_{dist}|$) as a percentage of the number of items in the problem.

The table entries for 0% negative coefficients shows that the newly defined efficiency value provides equivalent results for standard MKP problems as those reported in (Puchinger et al. 2006, 2007). As expected from Theorem 2, negative values do not increase the size of the split interval. The size of the split interval and the core actually decreases as the number of negative weights increases. The center of the core remains close to the center of the split interval. These results show that the chosen ordering, based on the optimal dual variable values of the LP-relaxation, is a good indicator of the actual location of the core.

4.3 Approximate core algorithm

In order to evaluate the influence of negative values on solution quality and run-times an approximate core algorithm was implemented. This algorithm is similar to the algorithm implemented by (Puchinger et al. 2006, 2007). The approximate core is generated by adding δ items to either side of the center of the split interval. The values of

$Weight_1$	$Weight_2$	Profit	s_j	(o_j, e_j)	LP	IP
-1	-1	1	-0.89	(7,0.89)	1.00	1
-9	12	7	-5.17	(7,0.74)	1.00	1
0	0	8	0.00	(6,8.00)	1.00	1
-2	13	12	0.24	(4,50)	1.00	1
5	0	19	3.77	(4,5.04)	1.00	1
-5	-4	-3	-4.31	(4,1.44)	1.00	1
16	21	18	14.88	(4,1.21)	1.00	1
13	15	14	11.81	(4,1.19)	1.00	**0**
-6	-10	-5	-5.87	(4,1.17)	1.00	**1**
20	-8	14	14.00	(4,1.00)	**0.71**	**1**
-6	-11	-6	-6.00	(4,1.00)	**0.85**	0
20	16	15	17.22	(4,0.87)	0.00	0
10	8	6	8.61	(4,0.70)	0.00	0
-1	-14	-4	-2.63	(4,0.66)	0.00	0
14	9	5	11.76	(4,0.43)	0.00	0
16	-3	3	11.66	(4,0.26)	0.00	0
-7	23	-9	-2.19	(4,0.24)	0.00	0
-1	10	0	0.59	(3,1.69)	0.00	0
7	0	-5	5.28	(1,0.95)	0.00	0
24	10	-9	19.43	(1,0.46)	0.00	0

Table 1: Example 2-dimensional 0/1 IP problem. The 3 sections are separated based upon the IP solution. This example shows an exact core using the efficiency measure defined in Theorem 2.

			e - 0% negative weights 0% negative profits					e - 5% negative weights 5% negative profits				
n	m	α	$\lvert S_e \rvert$	$\lvert C_e \rvert$	ScC	CcS	C_{dist}	$\lvert S_e \rvert$	$\lvert C_e \rvert$	ScC	CcS	C_{dist}
100	5	0.25	5.00	20.20	28.12	100.00	3.30	5.00	20.00	31.58	100.00	2.90
		0.5	5.00	22.10	27.49	100.00	3.45	5.00	15.90	28.33	86.00	2.65
		0.75	5.00	20.00	26.32	100.00	3.40	5.00	14.80	35.91	98.00	3.50
250	5	0.25	2.00	12.68	18.16	100.00	2.46	2.00	13.36	17.35	100.00	3.12
		0.5	2.00	12.20	18.45	100.00	1.38	2.00	9.60	21.47	100.00	1.20
		0.75	2.00	10.40	20.18	100.00	1.56	2.00	10.96	21.04	100.00	1.92
100	10	0.25	10.00	23.20	46.57	100.00	2.90	9.90	25.80	42.74	96.67	3.45
		0.5	9.80	25.80	48.17	96.00	3.10	9.70	23.70	44.06	100.00	3.00
		0.75	9.70	18.30	54.36	94.00	3.00	9.20	16.90	60.09	93.19	2.45
Average			5.61	18.32	31.98	98.89	2.73	5.53	16.75	33.62	97.10	2.69

			e - 10% negative weights 10% negative profits					e - 20% negative weights 20% negative profits				
n	m	α	$\lvert S_e \rvert$	$\lvert C_e \rvert$	ScC	CcS	C_{dist}	$\lvert S_e \rvert$	$\lvert C_e \rvert$	ScC	CcS	C_{dist}
100	5	0.25	5.00	23.60	22.31	100.00	4.30	5.00	18.60	29.55	100.00	2.70
		0.5	5.00	19.40	27.11	100.00	3.00	4.60	9.80	57.68	95.50	1.20
		0.75	4.80	12.40	47.28	98.00	1.60	2.50	16.90	30.34	86.67	7.00
250	5	0.25	2.00	10.36	20.27	100.00	1.22	2.00	10.52	20.22	98.00	2.02
		0.5	2.00	11.72	18.79	100.00	2.22	2.00	8.84	24.31	100.00	1.94
		0.75	2.00	7.28	30.13	98.00	1.24	1.56	6.08	40.80	100.00	1.74
100	10	0.25	9.70	24.20	42.19	97.89	4.05	9.70	28.00	37.91	100.00	4.15
		0.5	9.50	20.10	49.20	97.00	3.20	8.40	20.10	47.86	98.89	3.35
		0.75	8.80	14.30	65.99	92.70	2.05	4.30	13.80	35.51	92.50	4.05
Average			5.42	15.92	35.92	98.18	2.54	4.45	14.74	36.02	96.84	3.13

Table 2: Split intervals, core sizes, mutual coverage of the split interval and cores, distances of the centers for various percentages of negative values. (Values are averaged over 10 instances of each problem.)

δ were chosen to approximately reflect the size of the actual cores detected in the previous section: $\delta \in 0.1n, 0.15n, 0.2n, 0.1n + 2m, 0.2n + 2m$.

The problems shown in these tables are based upon the smaller instances in Chu and Beasley's benchmark library (Chu & Beasley 1998). They are the same set of problems used to investigate the actual core sizes in the previous section.

The results of this experiment are shown in Table 3. It shows the average values over 10 problems with the same tightness ratio. Values shown for the original problem include the average optimal IP solution for the problem ($\bar{z}$), then the average amount of CPU-time taken to produce the optimal IP solution in seconds (t[s]). Values shown for each core include the average percentage difference between the optimal IP solution(z^*) and the IP solution produced by the core problem(z), ($\%_{opt} = 100*(z^* - z)/z^*$), the number of times the optimal solution was reached (#), and the average CPU-time taken to solve the core IP, as a percentage of the CPU-time taken to solve the original IP problem, $\%t = 100 * (t_{core}/t_{original})$.

The solution to the approximate cores are (on average over 10 problem instances) always within 0.7 % of the optimal solution. The results shown in Table 3 show that smaller approximate core sizes produce a significant increase in speed. However they are less likely to produce the optimal solution, and on average produce solutions of lesser quality than the larger cores. As the percentage of negative values increases the problems become faster to solve. Larger negative percentages were examined, however run-times were too small to see any benefit from the core concept.

4.4 Larger 0/1 IP with Fixed Time Runs

We now investigate fixed-time runs over larger problem instances. These tests are performed over instances which are currently very hard or not at all solvable to optimality. The instances used are based on the hardest benchmarks provided by Chu and Beasley (Chu & Beasley 1998), $n = 500,\ m \in 5, 10, 30$

Again these problems were adjusted to contain negative values in a manner similar to the problems above. All of the results shown here are performed over problems with 10% negative weights, and 10% negative profits. The constraints are also adjusted accordingly.

Table 4 shows the best feasible solution for the original problem and the core problems as a percentage of the LP solution, ($\%_{LP} = 100 * (LP - IP)/LP$). These values are averaged over 10 instances of similar problems. Standard deviations are provided as subscripts. The smallest values for each row are highlighted in bold. This table also shows the number of times a particular core size has lead to the best solution for a problem, (#). The final column for each core size is the average number of nodes explored in the branch and cut tree used by CPLEX.

The experiments show that for the considered time limits the results obtained on the core problems are, on average, better that the results obtained from the original problem. There is also a inverse relationship between the size of the core and the number of nodes explored. As the size of the core decreases the number of nodes explored increases. The best average results for a time limit of 500 seconds is $\delta = 0.2$. It can be seen that smaller time limits provide best results with smaller approximate core sizes.

5 Related Work

The most closely related work to this paper is the application of the core concept to the MKP (Puchinger et al. 2006, 2007). We extend the results therein to general 0/1 Integer Programs, and show that the core concept continues to be valuable in the more general case.

Recently, very interesting results have been achieved with heuristics for 0/1 Mixed Integer Programming Problems with the goal of devising better feasible solutions earlier in the optimization process. Local Branching (Fischetti & Lodi 2003) combines local search and general branch-and-bound by introducing local branching constraints forcing the search to explore the neighborhoods of current feasible solutions first.

In Relaxation Induced Neighborhood Search (RINS) (Danna et al. 2005) subproblems for finding better feasible solutions are solved at some nodes of the branch-and-bound tree. The subproblems are obtained by fixing the variables having identical values in the current best feasible solution and in the current solution of the LP-relaxation, leaving the remaining variables free.

RINS and local branching are local-search based ideas, reducing the subproblems to certain neighborhoods around a currently feasible solution. Our approach requires an LP solution only, and does not make use of feasible solutions at all.

6 Conclusions

We have extended the core concept, previously successfully used for finding better solutions to Multiple Knapsack Problems to general 0/1 Integer Programs. We provided an ordering of the variables using dual information, which results in a compact split interval just as for the standard MKP. This ordering is used to reduce the size of the tackled instances and obtain near-optimal solutions in shorter run-times. Our computational experiments show, that for challenging 0/1 Integer Programs with a large number of variables compared to the number of constraints, the core concept provides better solutions than directly solving the original problem using a commercial solver. In the future we plan to test our approach on other widely used large 0/1 IP benchmarks.

Acknowledgements

National ICT Australia is funded by the Australian Government's Backing Australia's Ability initiative, in part through the Australian Research Council.

5 % negative weights, 5 % negative profits																			
			no core		$\delta = 0.1n$			$\delta = 0.15n$			$\delta = 0.2n$			$\delta = 0.1n + 2m$			$\delta = 0.2n + 2m$		
n	m	α	$\bar{z}$	t[s]	$\bar{\%}_{opt}$	#	$\%t$	$\bar{\%}_{opt}$	#	$\%t$	$\bar{\%}_{opt}$	#	$\%t$	$\bar{\%}_{opt}$	#	$\%t$	$\bar{\%}_{opt}$	#	$\%t$
100	5	0.25	26603	1.97	0.113	3	5	0.014	8	30	0.004	9	51	0.004	9	51	0.000	10	82
		0.50	44666	1.71	0.072	5	6	0.001	9	21	0.000	10	46	0.000	10	46	0.000	10	73
		0.75	60387	0.51	0.025	7	14	0.013	8	43	0.011	9	60	0.011	9	60	0.000	10	66
250	5	0.25	68598	56.11	0.007	7	39	0.004	8	70	0.003	9	119	0.004	8	70	0.003	9	130
		0.50	113794	93.78	0.000	10	22	0.000	10	47	0.000	10	68	0.000	10	42	0.000	10	66
		0.75	152330	42.46	0.002	7	39	0.000	10	62	0.000	10	65	0.000	10	45	0.000	10	70
100	10	0.25	24211	16.94	0.614	1	0	0.133	6	3	0.026	7	16	0.000	10	57	0.000	10	74
		0.50	43587	21.50	0.287	1	0	0.068	6	3	0.011	9	18	0.000	10	69	0.000	10	94
		0.75	59130	4.49	0.081	6	2	0.018	7	12	0.000	10	35	0.000	10	62	0.000	10	72
Average			65923	26.61	0.133	5.2	14	0.028	8.0	32	0.006	9.2	53	0.002	9.6	56	0.000	9.9	81

10 % negative weights, 10 % negative profits																			
			no core		$\delta = 0.1n$			$\delta = 0.15n$			$\delta = 0.2n$			$\delta = 0.1n + 2m$			$\delta = 0.2n + 2m$		
n	m	α	$\bar{z}$	t[s]	$\bar{\%}_{opt}$	#	$\%t$	$\bar{\%}_{opt}$	#	$\%t$	$\bar{\%}_{opt}$	#	$\%t$	$\bar{\%}_{opt}$	#	$\%t$	$\bar{\%}_{opt}$	#	$\%t$
100	5	0.25	28582	2.04	0.136	2	6	0.067	6	38	0.000	10	57	0.000	10	57	0.000	10	75
		0.50	45222	1.92	0.051	4	5	0.004	8	30	0.000	10	47	0.000	10	47	0.000	10	74
		0.75	59180	0.16	0.003	8	23	0.000	10	47	0.000	10	66	0.000	10	66	0.000	10	81
250	5	0.25	74521	44.80	0.000	10	30	0.000	10	46	0.000	10	68	0.000	10	39	0.000	10	71
		0.50	116296	36.00	0.001	9	33	0.000	10	56	0.000	10	66	0.000	10	43	0.000	10	73
		0.75	150236	6.49	0.000	10	45	0.000	10	62	0.000	10	71	0.000	10	61	0.000	10	75
100	10	0.25	25581	28.60	0.673	1	0	0.201	5	3	0.038	8	20	0.000	10	62	0.000	10	95
		0.50	44562	35.82	0.194	3	0	0.003	8	2	0.000	10	16	0.000	10	56	0.000	10	79
		0.75	58563	0.77	0.031	8	9	0.027	9	27	0.002	9	41	0.000	10	64	0.000	10	79
Average			66971	17.40	0.121	6.1	17	0.034	8.4	35	0.004	9.7	50	0.000	10.0	55	0.000	10.0	78

20 % negative weights, 20 % negative profits																			
			no core		$\delta = 0.1n$			$\delta = 0.15n$			$\delta = 0.2n$			$\delta = 0.1n + 2m$			$\delta = 0.2n + 2m$		
n	m	α	$\bar{z}$	t[s]	$\bar{\%}_{opt}$	#	$\%t$	$\bar{\%}_{opt}$	#	$\%t$	$\bar{\%}_{opt}$	#	$\%t$	$\bar{\%}_{opt}$	#	$\%t$	$\bar{\%}_{opt}$	#	$\%t$
100	5	0.25	34071	0.70	0.117	4	13	0.008	9	42	0.000	10	67	0.000	10	67	0.000	10	93
		0.50	46970	0.14	0.008	9	24	0.000	10	48	0.000	10	58	0.000	10	58	0.000	10	77
		0.75	56538	0.03	0.029	7	40	0.006	8	63	0.006	8	60	0.006	8	60	0.006	8	77
250	5	0.25	85206	33.14	0.005	8	30	0.000	10	48	0.000	10	70	0.000	10	45	0.000	10	81
		0.50	118236	8.60	0.000	10	38	0.000	10	55	0.000	10	72	0.000	10	52	0.000	10	86
		0.75	142169	0.21	0.000	10	39	0.000	10	61	0.000	10	64	0.000	10	60	0.000	10	75
100	10	0.25	29090	13.36	0.604	1	0	0.190	5	5	0.048	7	18	0.001	9	64	0.000	10	77
		0.50	45267	2.68	0.203	6	2	0.000	10	13	0.000	10	40	0.000	10	76	0.000	10	97
		0.75	55821	0.06	0.033	7	24	0.021	8	48	0.000	10	57	0.000	10	67	0.000	10	83
Average			68152	6.55	0.111	6.9	23	0.025	8.9	43	0.006	9.4	56	0.001	9.7	61	0.001	9.8	83

Table 3: Solving different sized cores for various percentages of negative values to optimality. (All values shown are averaged over 10 problem instances)

Time Limit = 5 Seconds														
			original problem			$\delta = 0.1n$			$\delta = 0.15n$			$\delta = 0.2n$		
n	m	α	$\%_{LP}$	#	Nnodes	$\%_{LP}$	#	Nnodes	$\%_{LP}$	#	Nnodes	$\%_{LP}$	#	Nnodes
500	5	0.25	$0.120_{0.021}$	2	16421	$\mathbf{0.114}_{0.025}$	4	31847	$0.116_{0.022}$	3	26955	$0.117_{0.014}$	4	24337
		0.50	$0.067_{0.011}$	2	15976	$\mathbf{0.051}_{0.012}$	9	31901	$0.061_{0.011}$	4	27587	$0.061_{0.017}$	3	23958
		0.75	$0.041_{0.004}$	5	18598	$\mathbf{0.041}_{0.004}$	8	32791	$0.042_{0.006}$	6	29194	$0.042_{0.005}$	6	27454
500	10	0.25	$0.438_{0.024}$	0	8061	$\mathbf{0.352}_{0.048}$	6	23208	$0.383_{0.051}$	3	15496	$0.364_{0.037}$	6	13707
		0.50	$0.171_{0.029}$	2	8262	$\mathbf{0.163}_{0.025}$	4	23548	$0.165_{0.023}$	5	15865	$0.175_{0.036}$	3	13743
		0.75	$0.097_{0.018}$	3	9765	$0.093_{0.011}$	6	22537	$0.094_{0.013}$	4	17021	$\mathbf{0.092}_{0.012}$	5	15421
500	30	0.25	$1.220_{0.115}$	2	2977	$\mathbf{1.191}_{0.105}$	3	10262	$1.230_{0.044}$	3	7881	$1.214_{0.113}$	2	6014
		0.50	$0.562_{0.040}$	0	3088	$0.507_{0.042}$	6	10649	$0.536_{0.018}$	1	8181	$\mathbf{0.495}_{0.047}$	5	6101
		0.75	$0.285_{0.021}$	2	3627	$0.282_{0.036}$	2	11431	$0.283_{0.015}$	2	8715	$\mathbf{0.273}_{0.021}$	4	6593
Average			$0.333_{0.031}$	2.0	9642	$\mathbf{0.310}_{0.034}$	5.3	22019	$0.323_{0.023}$	3.4	17433	$0.315_{0.034}$	4.2	15259
Time Limit = 50 Seconds														
			original problem			$\delta = 0.1n$			$\delta = 0.15n$			$\delta = 0.2n$		
n	m	α	$\%_{LP}$	#	Nnodes	$\%_{LP}$	#	Nnodes	$\%_{LP}$	#	Nnodes	$\%_{LP}$	#	Nnodes
500	5	0.25	$0.103_{0.016}$	3	172471	$0.100_{0.015}$	5	330280	$0.100_{0.015}$	6	272642	$\mathbf{0.099}_{0.015}$	7	249998
		0.50	$0.049_{0.011}$	6	181177	$\mathbf{0.046}_{0.008}$	9	328066	$0.048_{0.011}$	8	287341	$0.047_{0.009}$	8	260145
		0.75	$0.038_{0.004}$	6	188492	$0.038_{0.004}$	7	322998	$\mathbf{0.038}_{0.004}$	8	289610	$0.038_{0.004}$	9	275409
500	10	0.25	$0.331_{0.026}$	2	85187	$0.301_{0.019}$	4	231031	$\mathbf{0.296}_{0.024}$	6	150726	$0.312_{0.031}$	4	132856
		0.50	$0.144_{0.019}$	3	88829	$0.133_{0.014}$	4	235044	$0.135_{0.014}$	5	156361	$\mathbf{0.131}_{0.016}$	4	136481
		0.75	$0.082_{0.010}$	4	107407	$\mathbf{0.077}_{0.010}$	8	233931	$0.078_{0.012}$	5	171427	$0.079_{0.012}$	5	157293
500	30	0.25	$1.108_{0.076}$	1	33855	$\mathbf{1.045}_{0.066}$	6	102129	$1.098_{0.077}$	2	78250	$1.094_{0.080}$	2	62556
		0.50	$0.477_{0.030}$	2	34741	$\mathbf{0.458}_{0.034}$	5	104797	$0.467_{0.026}$	2	80025	$0.470_{0.035}$	3	62627
		0.75	$0.270_{0.019}$	0	38640	$0.254_{0.024}$	3	111507	$\mathbf{0.254}_{0.027}$	6	87269	$0.255_{0.024}$	1	65981
Average			$0.289_{0.023}$	3.0	103422	$\mathbf{0.272}_{0.022}$	5.7	222198	$0.279_{0.023}$	5.3	174850	$0.281_{0.025}$	4.8	155927
Time Limit = 500 Seconds														
			original problem			$\delta = 0.1n$			$\delta = 0.15n$			$\delta = 0.2n$		
n	m	α	$\%_{LP}$	#	Nnodes	$\%_{LP}$	#	Nnodes	$\%_{LP}$	#	Nnodes	$\%_{LP}$	#	Nnodes
500	5	0.25	$\mathbf{0.092}_{0.011}$	10	1468306	$\mathbf{0.092}_{0.011}$	10	2156859	$\mathbf{0.092}_{0.011}$	10	2038741	$0.092_{0.011}$	9	1844037
		0.50	$0.045_{0.008}$	9	1474390	$\mathbf{0.044}_{0.007}$	10	2282445	$\mathbf{0.044}_{0.007}$	10	2184819	$0.045_{0.008}$	9	1968420
		0.75	$\mathbf{0.038}_{0.004}$	10	1051111	$\mathbf{0.038}_{0.004}$	10	1170739	$\mathbf{0.038}_{0.004}$	10	1193311	$\mathbf{0.038}_{0.004}$	10	1133267
500	10	0.25	$0.291_{0.022}$	1	752103	$\mathbf{0.266}_{0.027}$	4	1973083	$0.269_{0.024}$	5	1190391	$0.274_{0.018}$	4	1070741
		0.50	$0.125_{0.018}$	3	804076	$0.120_{0.011}$	5	2064695	$\mathbf{0.119}_{0.015}$	6	1247341	$0.124_{0.014}$	3	1131538
		0.75	$0.077_{0.011}$	7	1013516	$0.076_{0.011}$	8	2359838	$0.075_{0.010}$	7	1530181	$\mathbf{0.075}_{0.010}$	9	1409720
500	30	0.25	$0.982_{0.088}$	3	312874	$0.982_{0.026}$	3	990696	$0.974_{0.025}$	2	759324	$\mathbf{0.952}_{0.073}$	3	610611
		0.50	$0.431_{0.013}$	3	318599	$0.428_{0.031}$	2	1024538	$\mathbf{0.427}_{0.017}$	3	772248	$0.429_{0.021}$	3	620238
		0.75	$0.238_{0.017}$	2	368143	$\mathbf{0.228}_{0.018}$	5	1081771	$0.235_{0.020}$	2	858274	$0.233_{0.021}$	3	670913
Average			$0.258_{0.021}$	5.3	840346	$0.253_{0.016}$	6.3	1678296	$0.253_{0.015}$	6.1	1308292	$\mathbf{0.251}_{0.020}$	5.9	1162165

Table 4: Fixed time runs of larger benchmark instances. Various core sizes are shown for 10% negative coefficients. All values shown are averaged over 10 problem instances.

References

Balas, E. & Zemel, E. (1980), 'An algorithm for large zero-one knapsack problems', *Operations Research* **28**(5), 1130–1154.

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

Chu, P. & Beasley, J. (1998), 'A genetic algorithm for the multidimensional knapsack problem', *Journal of Heuristics* **4**(1), 63–86.

Danna, E., Rothberg, E. & Le Pape, C. (2005), 'Exploring relaxation induced neighborhoods to improve MIP solutions', *Mathematical Programming, Series A* **102**, 71–90.

Fischetti, M. & Lodi, A. (2003), 'Local Branching', *Math. Programming Series B* **98**, 23–47.

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

Gilmore, P. & Gomory, R. (1966), 'The theory and computation of knapsack functions', *Operations Research* **14**, 1045–1074.

Glover, F. & Kochenberger, G. (1996), Critical event tabu search for multidimensional knapsack problems, *in* I. Osman & J. Kelly, eds, 'Metaheuristics: Theory and Applications', Kluwer Academic Publishers, pp. 407–427.

Gomes da Silva, C., Clímaco, J. & Figueira, J. (2005), Core problems in bi-criteria 0,1-knapsack: new developments, Technical Report 12/2005, INESC-Coimbra.

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

Martello, S. & Toth, P. (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**(1), 175–187.

Pisinger, D. (1997), 'A minimal algorithm for the 0-1 knapsack problem', *Operations Research* **45**(5), 758–767.

Puchinger, J., Raidl, G. & Pferschy, U. (2006), The core concept for the multidimensional knapsack problem, *in* 'Evolutionary Computation in Combinatorial Optimization - EvoCOP 2006', Vol. 3906 of *LNCS*, Springer, pp. 195–208.

Puchinger, J., Raidl, G. & Pferschy, U. (2007), The multidimensional knapsack problem: Structure and algorithms, Technical Report 006149, National ICT Australia, Melbourne, Australia. submitted for publication.

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

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. & Hao, J. (2001), 'A hybrid approach for the 0–1 multidimensional knapsack problem', *Proceedings of the 17th International Joint Conference on Artificial Intelligence* pp. 328–333.

Vasquez, M. & Vimont, Y. (2005), 'Improved results on the 0-1 multidimensional knapsack problem', *European Journal of Operational Research* **165**(1), 70–81.

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