CORE PROBLEMS IN KNAPSACK ALGORITHMS

DAVID PISINGER

Department of Computer Science, University of Copenhagen, Universitetsparken 1, DK-2100 Copenhagen, Denmark
pisinger@diku.dk

(Received July 1994; revision received April 1996; accepted March 1997)

Since Balas and Zemel in the 1980s introduced the so-called core problem as an efficient tool for solving the Knapsack Problem, all the most successful algorithms have applied this concept. Balas and Zemel proved that if the weights in the core are uniformly distributed then there is a high probability for finding an optimal solution in the core. Items outside the core may be fathomed because of reduction rules.

This paper demonstrates that generally it is not reasonable to assume a uniform distribution of the weights in the core, and it is experimentally shown that the heuristic proposed by Balas and Zemel does not find as good solutions as expected. Also, other algorithms that solve some kind of core problem may be stuck by difficult cores. This behavior has apparently not been noticed before because of unsufficient testing.

Capacities leading to difficult problems are identified for several categories of instance types, and it is demonstrated that the hitherto applied test instances are easier than the average. As a consequence we propose a series of new randomly generated test instances and show how recent algorithms behave when applied to these problems.

1. INTRODUCTION

Assume that n items with corresponding *profits* p_j and *weights* w_j are given. The *Knapsack Problem* is the task of packing some of these items in a knapsack of *capacity* c, such that the profit sum of the included items is maximized.

In the solution process it is suitable to order the items according to nonincreasing profit-to-weight ratios, because in this way tight upper and lower bounds may be derived quickly. Balas and Zemel (1980) noted that for easy instances this ordering takes up the majority of the computational time, proposing to avoid the sorting by considering a sufficiently small subset of the items known as the *core*. The core problem is a usual Knapsack Problem defined on a small subset of the available items, such that there is a high probability for finding a global optimum within the core. The objective value of the core problem is then used as a lower bound in a global reduction algorithm, which tries to fix decision variables at their optimal values by applying some bounding rules. The remaining problem is finally solved exactly through enumeration.

Balas and Zemel (1980) used an approximate algorithm for solving the core problem, showing that the probability for the heuristic to find an optimal solution grows with the size of the instance. The proof is based on the assumption that the weights w_j in a core are uniformly distributed, making it quite easy to obtain a filled knapsack by repeatedly removing an item and replacing it with some others.

Martello and Toth (1988) improved the technique by solving the core problem to optimality through branch-and-bound, thus obtaining a better lower bound. In a comprehensive test Martello and Toth (1990) demonstrate that their code MT2 is the best of the codes present at that moment.

However, Pisinger (1995b) detected some situations where the variance of the weights in a core is very small, contradicting the assumption by Balas and Zemel. This means that the heuristic solution found in a core becomes worse than expected, and for the MT2 algorithm it may result in extremely long computational times.

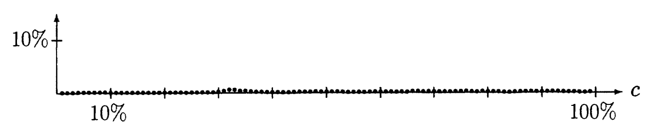
In this paper we prove in § 2 that Balas and Zemel's assumption about the uniform distribution of the weights in the core does not hold in practice, and actually deviates progressively with the instance size. Thus even for randomly generated instances, the core problems are generally difficult to solve. We use these observations to test the quality of a lower bound found in the core as a function of different capacities in § 3. The tests show that we seldom obtain as good lower bounds by solving a core problem as claimed by Balas and Zemel, and it is observed that the hitherto applied test instances are easier than the average. Thus in § 4, we present a new method for testing knapsack algorithms where the capacities are varied in order to obtain cores with different properties. Computational times are reported for several algorithms from the literature, showing that it is most beneficial to solve a core problem if it is solved through dynamic programming or by use of an expanding core.

2. DEFINITIONS

The Knapsack Problem (KP) is defined as the following optimization problem:

Subject classifications: Programming, integer, algorithms: Knapsack Problem. Analysis of algorithms: expected hardness. Statistics, cluster analysis: applied to Knapsack Problem.
Area of review: OPTIMIZATION.

Figure 1. Probability for uniform distribution of weights as function of the capacity c (average of 500 uncorrelated instances, $n = 3000$, $R = 100$, $|C| = 100$).



$$\text{maximize} \sum_{j=1}^{n} p_j x_j$$
$$\text{subject to} \sum_{j=1}^{n} w_j x_j \leq c, \tag{1}$$
$$x_j \in \{0, 1\}, \quad j = 1, \ldots, n,$$

where p_j, w_j, and c are positive integers, and x_j is a binary variable indicating whether or not item j was included in the knapsack.

The *Linear Knapsack Problem*, which is defined by relaxing the integrality constraint on x_j to $0 \leq x_j \leq 1$, may be solved by using the *greedy principle*: Order the items according to nonincreasing *efficiencies* $e_j = p_j/w_j$ and then include the most efficient items in the knapsack until the *break item* $b = \min \{j: \sum_{i=1}^{j} w_i > c\}$ is reached. The LP-optimal solution is then $x_j = 1$ for $j = 1, \ldots, b - 1$, and $x_j = 0$ for $j = b + 1, \ldots, n$, while we set $x_b = (c - \sum_{j=1}^{b-1} w_j)/w_b$. Having found the break item, we may derive upper and lower bounds on KP, which again may be used for reducing the problem size (Ingargiola and Korsh 1973, Dembo and Hammer 1980, Martello and Toth 1988). The reduced problem is then solved exactly using enumerative techniques.

Balas and Zemel (1980) noticed that the sorting according to efficiencies often takes up a majority of the computational time, thus proposing to focus the search on a limited number of items, the so-called *core problem*—an ordinary KP defined on the core C with reduced capacity $\bar{c}$. An *approximate core* may be found in $O(n)$ time by partitioning the items in three sets A, C, B according to their efficiencies, such that the break item is in the core. Thus

$$e_i \geq e_j \geq e_k, \quad i \in A, j \in C, k \in B,$$
$$\sum_{j \in A} w_j \leq c < \sum_{j \in A \cup C} w_j. \tag{2}$$

Figure 2. Probability for uniform distribution of weights as function of the capacity c (average of 500 weakly correlated instances, $n = 3000$, $R = 100$, $|C| = 100$).

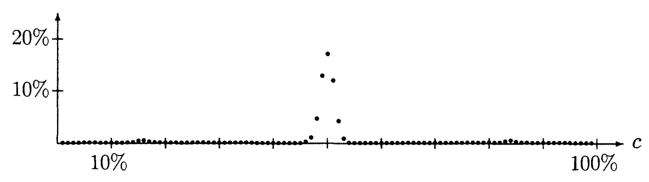
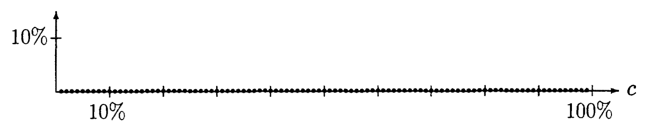


Figure 3. Probability for uniform distribution of weights as function of the capacity c (average of 500 strongly correlated instances, $n = 3000$, $R = 100$, $|C| = 100$).



Because all items in the core will have similar efficiencies, the core problem basically consists of finding an optimally *filled* knapsack. Balas and Zemel proved the following for a 2-optimal heuristic H.

PROPOSITION 1. *Assuming that the weights in the core are uniformly distributed in* $[1, R]$ *then the heuristic* H *will find a completely filled knapsack solution, with a probability that increases with larger core size* $|C|$, *and smaller range* R.

PROPOSITION 2. *If a filled solution is found by algorithm* H, *then the probability that it is optimal increases with* n.

The size of the core should be chosen sufficiently large to find an optimal solution but also small enough to avoid unnecessary enumeration. Balas and Zemel proposed the size $|C| = 50$, while Martello and Toth (1990) chose $|C| = 2\sqrt{n}$ for $n \geq 200$.

3. THE WEIGHTS ARE NOT UNIFORMLY DISTRIBUTED

Although solving the core problem means that a smaller problem is enumerated, it does not necessarily mean that the problem is easier to solve. Pisinger (1995b) presented a difficult core that was almost impossible to solve for the MT2 algorithm by Martello and Toth. Thus we will investigate what may have caused the problem.

Proposition 1 assumes that the weights in the core are uniformly distributed, which makes it easy for a 2-optimal heuristic to find a near-optimal solution in the core. To test the hypothesis that the weights in the core actually are uniformly distributed, we consider four different types of randomly generated instances from the literature. In each case the weights w_j are randomly distributed in $[1, R]$ while the choice of p_j is varied as follows: (1) *uncorrelated data instances*—the profits p_j are randomly distributed in $[1, R]$; (2) *weakly correlated data instances*—p_j randomly distributed in $[w_j - R/10, w_j + R/10]$ such that $p_j \geq 1$; (3)

Figure 4. Probability for uniform distribution of weights as function of the capacity c (average of 500 subset-sum instances, $n = 3000$, $R = 100$, $|C| = 100$).

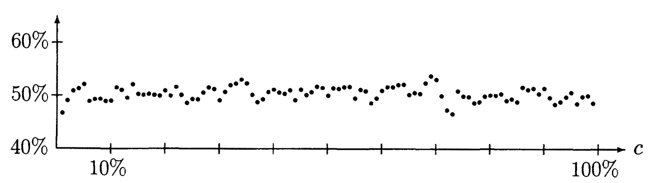
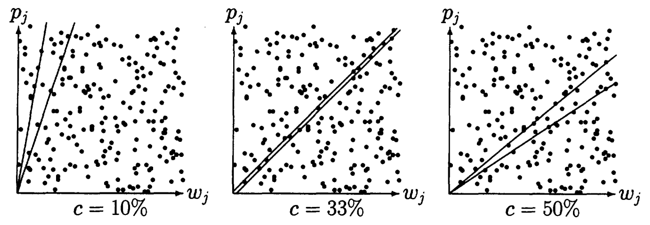


Figure 5. Three possible cores for an uncorrelated instance (the sector chosen depends on the capacity c).



strongly correlated data instances—profits are set to $p_j = w_j + 10$; and (4) *subset-sum data instances*—profits $p_j = w_j$. We test the hypothesis for a small *range* $R = 100$ because this should lead to the best heuristic solutions according to Proposition 1. Instances with $n = 3000$ items are considered because these are moderately large, even for Balas and Zemel (1980). The core is chosen so that $|C| = 100$ symmetrically around the break item. The size conforms with the choice by Martello and Toth (1990).

We use a χ^2 test to determine whether it is reasonable to assume that the weights in the core are uniformly distributed in $[1, R]$. The interval is divided in $m = 10$ equally sized sections, and N_i determines the number of weights in interval i. According to the hypothesis we expect $\mu = 10$ weights in each section. The test value becomes

$$\chi^2 = \sum_{i=1}^{m} (N_i - \mu)^2/\mu, \tag{3}$$

where small values of χ^2 indicate that the hypothesis is correct. As the expected number of weights in each section is large ($\mu \geqslant 10$) we may assume that the values follow a chi-square distribution with 9 degrees of freedom. Thus for each value of χ^2 we may derive a probability for that entry under the assumption that our hypothesis holds.

Figures 1 to 4 show the probabilities for a uniform distribution of the weights in the interval $[1, R]$, when c is varied from 1% to 99% of the total weight sum. Each entry is the average values of 500 different cores. It is seen that for uncorrelated instances and strongly correlated instances, we may discard the hypothesis of a uniform distribution. For the weakly correlated instances, we also discard the hypothesis apart from situations where the capacity is close to $c = 50\%$ of the total weight sum. Only for the subset-sum instances, we may conclude that the hypothesis holds.

Figure 6. $\mathcal{H} = \bar{w}\chi$ as function of the capacity c (weakly correlated instances, $n = 3000$, $R = 100$, $|C| = 100$).

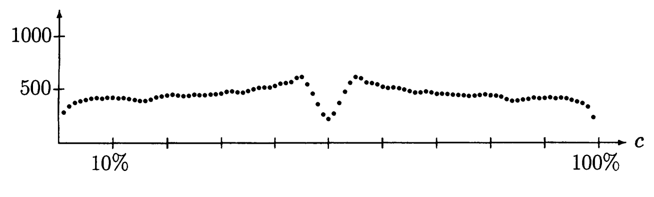
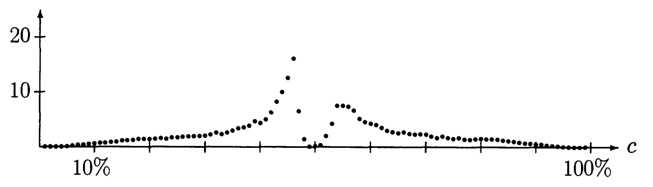


Figure 7. Average residual capacity for heuristic H, as function of the capacity c (weakly correlated instances, $n = 3000$, $R = 100$, $|C| = 100$).



There is a simple explanation to the bad distribution of the weights. Figure 5 shows the core for different values of c. The items in the core have a profit-to-weight ratio close to that of the break item, thus the core consists of a sector of the plane. Because the sector is triangular, we cannot expect to get a uniform distribution of the weights. Moreover, the individual profits and weights are integers, meaning that certain values of p_j/w_j can be obtained only as a unique fraction, while $p_j/w_j = 1$, for example, can be obtained in plenty of ways.

Although Proposition 1 in general does not hold for large problems, we cannot exclude that a good lower bound may be found in the core. The first core in Figure 5 comprise relatively small weights, and thus it should be easy to combine them to a filled knapsack. Pisinger (1994) argues that the hardness $\mathcal{H}$ of a core depend on the product of the average weight $\bar{w}$ and the clustering χ from (3). For weakly correlated instances, $\mathcal{H}$ is shown in Figure 6. The instance has $n = 3000$ items, range $R = 100$, and the core size is $|C| = 100$. To smooth out the stochastical variation of each instance, we give the average values of 500 different data instances for each value of c.

In Figure 7 we compare the expected hardness with the solution quality of heuristic H, measured as the residual capacity of the best filling. If heuristic H is able to find a filled solution and thus a good lower bound, obviously the core problem will be easy to solve. The two figures generally have the same characteristics.

Finally, we compare the expected core hardness with the actual running times of the MT2 algorithm in Figure 8. MT2 solves the core problem to optimality, using a considerable amount of the solution time for this step. Because of the exponential growth of computing times, we show the average values of the logarithm to the solution times. It is seen

Figure 8. Average log computational times for MT2 in seconds, as function of the capacity c (weakly correlated instances, $n = 3000$, $R = 100$, $|C| = 100$).

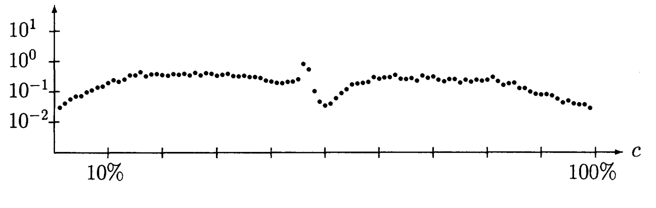
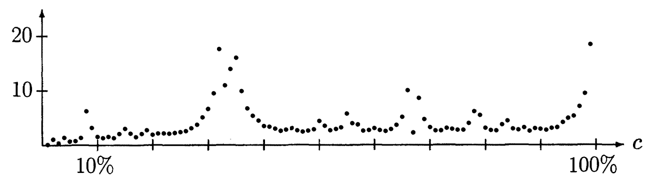


Figure 9. Residual capacity, uncorrelated instances, $n = 100000$, $R = 100$, $|C| = 100$.



that actual running times generally have the same behavior as the two previous figures, although the hardness of a core for MT2 is more complex to analyze theoretically, as it depends on the branching strategy.

4. THE HARD PROBLEMS

Since Proposition 2 will ensure that filled core solutions also are optimal solutions to the original problem, the main concern is to obtain a sufficiently filled knapsack. Thus we will use heuristic H to determine for which values of c we may derive good lower bounds in a core. Traditionally, only the correlation and instance size was varied in computational experiments, while the capacity was fixed at half of the weight sum (Balas and Zemel 1980; Martello and Toth 1977, 1988; Pisinger 1995a, 1996).

Figures 9 to 12 give the residual capacity of the best filling found by heuristic H for uncorrelated, weakly correlated, strongly correlated, and subset-sum instances as a function of the capacity c for very large instances $n = 100000$. All figures are average values of 500 instances generated with data range $R = 100$ and core $|C| = 100$.

For the weakly correlated instances, choosing $c = 50\%$ of the total weight sum leads to the easiest problems, while other choices of c will lead to bad lower bounds in the core. For the uncorrelated instances, the capacity $c = 50\%$ does not show the real hardness of the problems, because the interesting problems are found for capacities close to $c = 35\%$. The strongly correlated problems are the most difficult of the considered problems, and the quality of the lower bound in a core will decrease as c is increased. Finally, the subset-sum problems are easy, because the heuristic solution found in the core generally is optimal.

It should be noted that in those cases where the residual capacity of the heuristic solution is close to $R/2 = 50$, it means that no improvement was done by the 2-optimal heuristic compared to the LP-solution.

We may draw several conclusions from these observations. First of all, the comprehensive comparative tests in Martello and Toth (1990) are not representative, because all tests are based on the capacity $c = 50\%$. Thus we do not know how the algorithms will behave for different choices of c. Moreover, we may now explain the anomalous behavior of MT2 for weakly correlated problems as reported by Martello and Toth (1990), because although $c = 50\%$ is applied, small variations in the weights may place the instance among the most difficult ones.

All the presented graphs have been given for the data range $R = 100$, as Proposition 1 mainly says how heuristic H will behave when the weights are distributed in a small interval. For weights distributed in larger intervals, it will be more difficult to obtain a completely filled solution, meaning that for $R > 10000$ it may not be beneficial to solve a core problem at all.

Figure 10. Residual capacity, weakly correlated instances, $n = 100000$, $R = 100$, $|C| = 100$.

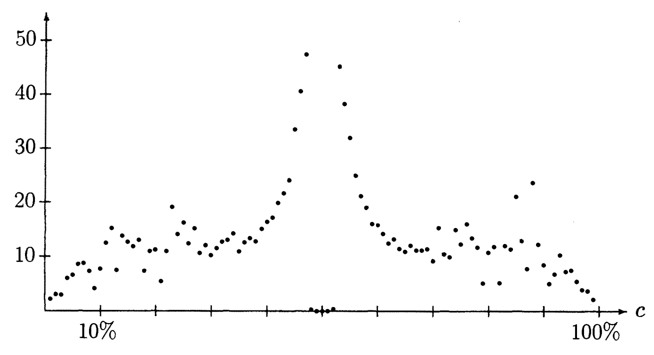
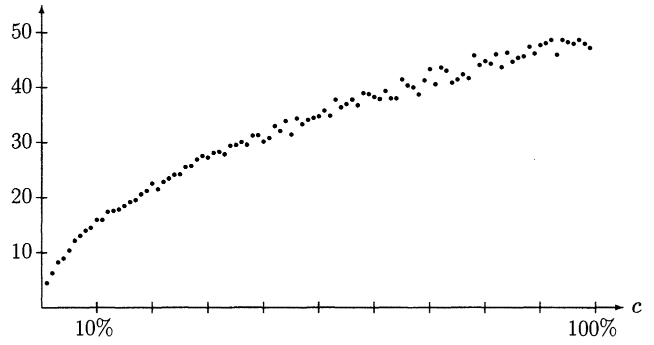


Figure 11. Residual capacity, strongly correlated instances, $n = 100000$, $R = 100$, $|C| = 100$.



5. NEW METHOD FOR TESTING ALGORITHMS

In order to test algorithms for KP more thoroughly, we propose a new way of constructing test instances. First we notice that because of the stochastic nature of KP, the applied series of test instances should be large—much larger than the previously used 10 or 20 instances (Balas and Zemel 1980; Martello and Toth 1990; Pisinger 1995a, 1996). Thus we propose a series of $S = 1000$ instances for each type of problem. Moreover the capacities should be uniformly scattered among the possible weight sums, so

Figure 12. Residual capacity, subset-sum instances, $n = 100000$, $R = 100$, $|C| = 100$.

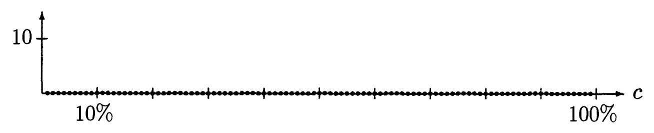


Table 1. Total computing time in seconds, FPK79 (average of 1000 instances).

	Uncorrelated			Weakly Corr.			Strongly Corr.			Subset-Sum		
n\R	100	1000	10000	100	1000	10000	100	1000	10000	100	1000	10000
100	0.00	0.00	0.00	0.00	0.00	0.00	28.78	28.13	56.02	0.00	0.00	0.01
300	0.00	0.00	0.00	0.00	0.00	0.00	—	—	—	0.01	0.01	0.02
1000	0.00	0.00	0.00	0.08	0.00	0.01	—	—	—	0.06	0.08	0.11
3000	0.01	0.01	0.01	—	0.01	0.01	—	—	—	0.49	0.62	0.86
10000	0.50	0.03	0.04	—	0.16	0.05	—	—	—	5.26	5.68	8.08
30000	—	0.10	0.11	—	—	0.14	—	—	—	—	—	—
100000	—	0.47	0.42	—	—	3.01	—	—	—	—	—	—

Table 2. Total computing time in seconds, MT2 (average of 1000 instances).

	Uncorrelated			Weakly Corr.			Strongly Corr.			Subset-Sum		
n\R	100	1000	10000	100	1000	10000	100	1000	10000	100	1000	10000
100	0.00	0.00	0.00	0.00	0.00	0.00	8.07	8.31	16.24	0.00	0.00	0.01
300	0.00	0.00	0.00	0.00	0.00	0.01	—	—	—	0.00	0.00	0.02
1000	0.00	0.00	0.01	0.00	0.01	0.02	—	—	—	0.00	0.00	0.02
3000	0.00	0.01	0.02	0.07	0.01	0.06	—	—	—	0.00	0.00	0.02
10000	0.02	0.03	0.06	—	0.02	0.14	—	—	—	0.01	0.01	0.03
30000	—	0.06	0.17	—	0.19	0.23	—	—	—	0.03	0.03	0.05
100000	—	0.27	0.52	—	—	0.44	—	—	—	0.12	0.13	0.14

Table 3. Total computing time in seconds, EXPKNAP (average of 1000 instances).

	Uncorrelated			Weakly Corr.			Strongly Corr.			Subset-Sum		
n\R	100	1000	10000	100	1000	10000	100	1000	10000	100	1000	10000
100	0.00	0.00	0.00	0.00	0.00	0.00	42.21	45.14	126.66	0.00	0.00	0.02
300	0.00	0.00	0.00	0.00	0.01	0.01	—	—	—	0.00	0.00	0.02
1000	0.00	0.00	0.01	0.00	0.02	0.06	—	—	—	0.00	0.00	0.02
3000	0.00	0.01	0.02	0.00	0.01	0.25	—	—	—	0.00	0.00	0.03
10000	0.01	0.03	0.11	0.01	0.02	0.65	—	—	—	0.01	0.01	0.05
30000	0.04	0.06	0.39	0.04	0.05	0.63	—	—	—	0.04	0.04	0.05
100000	0.17	0.19	0.91	0.17	0.18	0.39	—	—	—	0.15	0.15	0.15

Table 4. Total computing time in seconds, MINKNAP (average of 1000 instances).

	Uncorrelated			Weakly Corr.			Strongly Corr.			Subset-Sum		
n\R	100	1000	10000	100	1000	10000	100	1000	10000	100	1000	10000
100	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.03	0.25	0.00	0.00	0.05
300	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.14	1.44	0.00	0.00	0.05
1000	0.00	0.00	0.00	0.00	0.00	0.01	0.06	0.61	7.43	0.00	0.01	0.05
3000	0.00	0.01	0.01	0.00	0.01	0.03	0.22	2.33	29.10	0.00	0.01	0.06
10000	0.01	0.01	0.02	0.01	0.01	0.05	1.00	10.47	125.12	0.01	0.01	0.06
30000	0.03	0.03	0.06	0.04	0.04	0.07	3.22	38.04	—	0.03	0.03	0.08
100000	0.10	0.10	0.16	0.15	0.14	0.15	14.22	152.51	—	0.11	0.12	0.16

that the capacity-dependent behavior is annihilated. For test instance i we choose the capacity as

$$c = \frac{i}{S+1} \sum_{j=1}^{n} w_j, \tag{4}$$

meaning that a variety of different capacities are tested as i runs from 1 to S. An algorithm for generating the instances is available from www.diku.dk/~pisinger/codes.html.

We consider four different algorithms for KP. Although they are all based on solving a core problem exactly, they differ essentially in several respects. The FPK79 algorithm (Fayard and Plateau 1982) is based on a heuristic solution of the core problem, while the MT2 algorithm by Martello and Toth (1990) solves a core problem of fixed size to optimality through branch-and-bound. The EXPKNAP algorithm (Pisinger 1995a) is using a depth-first branch-and-bound algorithm for solving the core problem, but the core is expanded by need as the branching propagates. Finally the MINKNAP algorithm (Pisinger 1996) is using dynamic programming for solving an expanding core. The codes used are obtained from the mentioned references. Algorithm FPK79 was modified a few places since the reduction part did not behave properly. The code by Balas and Zemel (1980) was not available for this test.

Tables 1 to 4 give the total running times for the four algorithms on a HP-9000/730 computer. The instances marked with a "—" were stopped after 10 hours. It is seen that FPK79 and MT2 have substantial problems for all low-ranged weakly correlated instances of medium and large size. Also some low-ranged uncorrelated instances cause problems. On the other hand both EXPKNAP and MINKNAP show a stable behavior, with all running times within one second (the strongly correlated instances excepted). Only MINKNAP is able to solve large strongly correlated instances.

Although MT2 and EXPKNAP both are using depth-first branch-and-bound techniques, EXPKNAP is not affected of the difficulty of a core because it may expand the core by need. The MINKNAP algorithm is using dynamic programming for solving the core, so similarly weighted states are simply fathomed through dominance.

6. CONCLUSION

We have seen that the weights in a core generally are *not* uniformly distributed, and thus it may be difficult to obtain a good heuristic solution by only considering the items in the core. Experiments with the heuristic H by Balas and Zemel have shown that choosing the capacity as 50% of the total weight sum does not show the real hardness of a core problem. A new and more thorough experimental evaluation of several algorithms from the literature have been performed, showing that fixed-core algorithms generally have unstable behavior.

ACKNOWLEDGMENTS

The author wishes to thank the associate editor and the referees for their detailed comments, which have improved the presentation of this paper.

REFERENCES

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

Dembo, R. S., P. L. Hammer. 1980. A reduction algorithm for knapsack problems. *Methods of Oper. Res.* **36** 49–60.

Fayard, D., G. Plateau. 1982. An algorithm for the solution of the 0-1 knapsack problem. *Computing* **28** 269–287.

Ingargiola, G. P., J. F. Korsh. 1973. A reduction algorithm for zero-one single knapsack problems. *Management Sci.* **20** 460–463.

Martello, S., P. Toth. 1977. An upper bound for the zero-one knapsack problem and a branch and bound algorithm. *European J. Oper. Res.* **1** 169–175.

——, ——. 1988. A new algorithm for the 0-1 knapsack problem. *Management Sci.* **34** 633–644.

——, ——. 1990. *Knapsack Problems: Algorithms and Computer Implementations*. Wiley, Chichester, England.

Pisinger, D. 1994. Core problems in knapsack algorithms. Report 94/26. DIKU, University of Copenhagen, Denmark.

——. 1995a. An expanding-core algorithm for the exact 0-1 knapsack problem. *European J. Oper. Res.* **87** 175–187.

——. 1995b. Avoiding anomalies in the MT2 algorithm by Martello and Toth. *European J. Oper. Res.* **82** 206–208.

——. 1997. A minimal algorithm for the 0-1 knapsack problem. *Oper. Res.* **45** 758–767.