Transaction of Azerbaijan National Academy of Sciences, Series of Physical-Technical and Mathematical Sciences: Informatics and Control Problems, Vol. XXXV, No.3, 2015
www.icp.az/2015/3-13.pdf

N.K. GURSOY, U.G. NURİYEV

GREEDY ALGORITHMS WITH GUARANTEE VALUE FOR 0/1 MINIMIZATION KNAPSACK PROBLEM

In this paper, we propose two new algorithms for 0-1 Minimization Knapsack problem by mentioning the well-known algorithms with guaranteed estimate. We present computational results and compare the proposed algorithms with the previous ones. Finally, we show that the second algorithm achieves an approximation guaranteed value 2.

Keywords: 0/1 knapsack problem, minimization, greedy algorithm, approximation, guarantee value

1. Introduction. The goal of the NP-hard 0/1 Knapsack Problem (KP) is to fill a knapsack with maximum capacity $c \in \mathbb{N}$ such that the total weight of all assigned items isn't over the capacity while the total profit is maximized with items having weight $w_i \in \mathbb{N}$ and profit $p_i \in \mathbb{N}$ $(i = 1, \dots, n)$ [1, s. 5].

$$KP = \max \{ \sum_{i=1}^{n} p_i x_i \mid \sum_{i=1}^{n} w_i x_i \leq c \, , x_i \in \{0,1\}, i = 1, \dots, n \}$$

The decision variables x_i point out which items are put into the knapsack, that is if $x_i = 1$ then item *i* is inserted. Without loss of generality, we suppose $w_i \leq c, \ i = 1, \dots, n$ and $\sum_{i=1}^{n} w_i > c$.

It can be indicated that the KP as the problem to minimize the profit of items not inserted in the knapsack subject to condition that their combined weight has to be at least $d = \sum_{i=1}^{n} w_i - c$. This definition is mentioned as Minimization Knapsack Problem (KM) [1, s. 412.].

$$KM = \min \{ \sum_{i=1}^{n} p_i y_i \mid \sum_{i=1}^{n} w_i y_i \geq d \, , y_i \in \{0,1\}, i = 1, \dots, n \}$$

The KM variables y_i indicate which items are inserted in the minimization knapsack. This means that they are not inserted in the maximization knapsack. As a result, to a solution x_i of the KP we determine the corresponding solution of the KM as $y_i := 1 - x_i$ (i =1,…,n), and conversely.

2. Greedy algorithms and their performance analysis. Greedy algorithms always make the best choice at the moment on each step and are optimization procedure with having many steps which also stay unchanged on next steps. This method first sorts the items according to some criteria and then extends the solution set starting with empty set. Only one item is decided at each step. If final solution is feasible then the item is inserted in current solution, otherwise it is eliminated permanently.

Running time of a greedy algorithm is $O(n \log n) + O(n) = O(nlogn)$ because of sorting of n items ($O(n \log n)$) and feasibility test for n items ($O(n)$) [1, s. 16].

The approximation of an algorithm which is designed for a problem and the required time and memory of this algorithm is determined according to the results of some analysis. The common method used for evaluating an algorithm's performance is the worst-case analysis [2]. In this method, the maximum difference between the solution, which the algorithm has found for a given problem class, and the optimum solution is determined. Here, such a Δ value, which is independent from the problem's input parameters, is found that for each individual maximization[minimization] problem in given class $\Delta \geq A/OPT$ ($\Delta \leq A/OPT$); where A is the solution value that the algorithm has found and OPT is the optimum solution. The variable Δ is generally called as the *guarantee value* of the algorithm.

ε variable which equals to $(1 - \Delta)$ for maximization problem and $(1 + \Delta)$ for minimization problem is used in order to evaluate quality of algorithms. The algortihms which find approximate solution with a formerly given ε are called *ε-approximation algorithms*. For maximization [minimization] problem with an optimal objective value OPT the approximate algorithm yields a solution A as $A \geq (1 - \varepsilon) \cdot OPT$ $[A \leq (1 - \varepsilon) \cdot OPT]$ [1, s. 33].

3. Algorithm with guarantee value for KP. First approximation schemas for KP are established by Ibara and Kim [3, s.464-465], Sahni [4, s.123-124]. All of these algorithms are based on dynamic programming and are separated from each other on "scaling" technique of input data.

Greedy algorithms with guarantee value $\Delta \geq 1/2$ are known for KP [5, s. 15]. One of these algorithm is given as following.

To simplify the definition, without loss of generality, we assume that the items are sorted in decreasing order according to their profit per weight, i.e.,

$$\frac{p_1}{w_1} \geq \frac{p_2}{w_2} \geq \cdots \geq \frac{p_n}{w_n}$$

A_1 Algorithm

1. Find the index k with $\sum_{i=1}^{k} w_i \leq b \leq \sum_{i=1}^{k+1} w_i$

2. $A_1 = max\left\{\sum_{i=1}^{k} p_i, p_{k+1}\right\}$

3. End

***Theorem 1*:** Guarantee value of A_1 algorithm (Δ_1) is equal to $1/2$, that is $\frac{1}{2}KP < A_1 \leq KP$.

4. Algorithm with guarantee value for KM. $\Delta = 2$ guarantee value algorithm for KM is proposed first in [5, s. 16-18]. To simplify the definition, without loss of generality, we assume that the items are sorted in ascending order according to their profit per weight i.e.,

$$\frac{p_1}{w_1} \leq \frac{p_2}{w_2} \leq \cdots \leq \frac{p_n}{w_n} \tag{4.1}$$

and let $J = \{1,2, \dots, n\}$.

A_2 Algorithm

1. Let $A_2 = \sum_{i=1}^{n} p_i$
2. Find $k = \min\left\{l \middle| \sum_{i=1}^{l} w_i \geq d\right\}$
3. $L = \sum_{i=1}^{k} p_i$
4. $A_2 = \min\{A_2, L\}$, $J = J \setminus k$
5. If $\sum_{i \in J} w_i \geq d$ then Goto 2
6. End

***Theorem 2*:** Guarantee value of A_2 algorithm is (Δ_2), is equal to 2 that is,

$$KM \leq A_2 \leq 2KM \tag{4.2}$$

5. Algorithms with guarantee value for KP and KM. A minimization Algorithm (A_3) is proposed in [6, s. 57-58] for solving KP and KM. In analogy to the Greedy Algorithm a solution is constructed by insertion of the items according to their profit per weight. The algorithm starts with a current solution by setting all decision variables to 0 and by marking all items as unprocessed. Then, the algorithm repeats two phases, the insertion and the completion phase, until all items are processed.

The insertion phase starts by inserting the unprocessed items one by one in the given order until the insertion of the heaviest unprocessed item would lead to a feasible solution. In the following completion phase, all items whose addition to the current solution would lead to a feasible solution are considered. Every new solution constructed with this way is compared to the best solution so far and stored if it is better. Whenever an item is used for insertion or completion it is marked as processed.

Finally, the algorithm has to store the best solution so far implicitly by storing its total weight and its completion item. To determine the final solution, the completion item is inserted in the knapsack and the other items are added one by one in the given order whenever the insertion does not exceed the final weight.

A_3 Algorithm (MinGreedy)

1. Initialization
2. $w = 0, p = 0$
3. $\bar{w} = \sum_{i=1}^{n} w_i; \bar{p} = \sum_{i=1}^{n} p_i$
4. $k = 1$
5. $S = \{1,2, \dots, n\}$
6. *while* $S \neq 0$ *and* $p < \bar{p}$ *do*
7. *while* $\max\{w_i : i \in S\} < d - w$
8. $j = \min S, w = w + w_j, p = p + p_j$, S: = S\{j}
9. *od*
10. *while* $\max\{w_i : i \in S\} \geq d - w$ *do*
11. Find j with $w_j = \max\{w_j : j \in S\}; S = S \setminus \{j\}$
12. *İf* p+p_j<$\bar{p}$ *then* $\bar{w} := w + w_j$; $\bar{p} := p + p_j; k = j$ fi
13. *od*
14. *od*

***Theorem 3*:** A_3 algorithm is 1-approximate for KM and $1/2$- approximate for KP.

It is important note that the connection between the solutions of the KP and KM problems are first made in [7]. KM problem is defined as complement problem of KP.

This case cannot be applied to ε-approximate algorithms whereas exact algorithms for one of two versions also give an exact solution for the another.

Let $P = \sum_{i=1}^{n} p_i$, $KP = \alpha_1 P$ and $KM = \alpha_2 P$. Then, the following Theorems are true [7, s.68-69]:

***Theorem 4*:** A_2 value founded by A_2 algorithm provides the following conditions:

$$A_2 \leq \begin{cases} 2KM & , \quad if\ \alpha_1 < \dfrac{1}{3} \\ \dfrac{\alpha_1 + 1}{2\alpha_1} KM & , \quad if\ \alpha_1 \geq \dfrac{1}{3} \end{cases}$$

***Theorem 5*:** $P - A_2$ value founded by A_2 algorithm provides the following conditions:

$$P - A_2 \geq \begin{cases} \dfrac{1}{2} KP & , \quad if\ \alpha_2 < \dfrac{2}{3} \\ \dfrac{2\alpha_2 - 1}{\alpha_2} KP & , \quad if\ \alpha_2 \geq \dfrac{2}{3} \end{cases}$$

In [8], complement problem concept is improved for integer programming problems and suitable theorems are proved for suggested algorithms.

6. Algorithm based on Fractional Algebra for KM. New operations are defined on fractions to solve Linear Fractional Boole minimization programming problems in [9, s. 112-113] and it is shown that these operations on fractions set constitute an algebra. Inversion condition is determined according to sum operation defined in this algebra. In the following, a Greedy algorithm is suggested for KM considering inversion condition and assuming condition (4.1). In this algorithm, the priority is given to elements satisfy inversion condition at the time of selection.

A_4 Algorithm

1. Initialization
2. $pp = 0,\ ww = 0,\ i = 1,\ j = 2$
3. *while* $ww < d$ *do*
4. if $p_i < p_j$ then $k = i,\ i = j$
5. else $k = j$
6. $pp = pp + p_k,\ \ ww = ww + w_k$
7. $j = j + 1$
8. *od*
9. $pr = pp,\ wr = ww$
10. $j = j - 1$
11. *while* 1 *do*
12. $pp = pp + p_k,\ \ ww = ww + w_k$
13. *while* $ww < d$ *do*
14. $j = j + 1$
15. if $j > n$ then Goto 22
16. else if $p_i < p_j$ then $k = i,\ \ i = j$
17. else $k = j$
18. $pp = pp + p_k,\ \ ww = ww + w_k$
19. *od*
20. if $pp < pr$ then $wr = ww,\ \ pr = pp$
21. *od*
22. $A_4 = pr$

7. The new algorithm with guarantee value for KM. In this section, A_2 algorithm is improved with the assumption condition (4.1).

A_5 Algorithm

1. Initialization
2. $k = \min\left\{ l \,\middle|\, \sum_{i=1}^{l} w_i \geq d \right\}$
3. $p = \sum_{i=1}^{k} p_i\,, w = \sum_{i=1}^{k} w_i\,, pr = p, wr = w, i = i + 1, j = i$
4. *while* 1 *do*
5. $p = p - p_k,\ \ w = w - w_k$
6. *while* 1 *do*
7. $j = j + 1$
8. if $j > n$ then Goto 25
9. if $p_i < p_j$ then
10. if $w_i \leq d - w$ &$w_j \leq d - w$ then $k = i,\ i = j$, Goto 20
11. if $w_i < d - w$ &$w_j > d - w$ then $k = j$
12. if $w_i > d - w$ &$w_j \leq d - w$ then $k = i,\ i = j$
13. if $w_i \geq d - w$ &$w_j > d - w$ then $k = i,\ i = j$
14. *exit do*
15. else
16. if$w_i \leq d - w$ then $k = i,\ i = j$
17. if $w_i > d - w$ &$w_j \leq d - w$ then $k = i,\ i = j$
18. if $w_i > d - w$ &$w_j > d - w$ then $k = j$
19. *exit do*
20. $p = p + p_k,\ \ w = w + w_k$

21. *od*
22. $p = p + p_k,\ \ w = w + w_k$
23. if $p \le p_r$ then $pr = p,\ \ wr = w$
24. *od*
25. $A_5 = pr$

Theorem 6: Guarantee value of A_5 algorithm is $\Delta_5 = 2$, that is $KM \le A_5 < 2KM$

Proof: Solution which is found by A_5 algorithm is never worse than solution which find A_2 algorithm. Because $p_i < p_j,\ \ w_i < w_j,\ w_i < d - w$, for $j > i \ge k$ and A_2 algorithm chooses first *i*th item and then *j*th item while $w_j \ge d - w$. But A_5 algorithm chooses only *j*th item and A_2 algorithm can never do this choice. A_5 algorithm can find a better solution (that is $A_5 \le A_2$) since $p + p_i + p_j > p + p_j$.

Out of this case, both of algorithm chooses the same items. Hence, since the condition (4.2) is satisfied for the solution found by A_2 algorithm condition (4.2) is also satisfied for the solution found by A_5 algorithm, i.e., $\Delta_5 = 2$.

8. The results of computational experiments. To research efficiency of algorithms for Knapsack Problem with Boolean variable, a library, named as the KNAPSACK Test Problems Library, have been prepared (http://fen.ege.edu.tr/~urfat/knapsack/). These test problems are created by using the method in [11. s.1430] and setting values as $= 1,\ n = 100,\ \pi = 3.14,\ \theta = 1.33,\ l = 0,\ r \in \{3, 5\},\ \beta \in \{0.3, 0.4, 0.5, 0.6, 0.7, 0.8\},\ t = 1, \dots, 120$, where *n* is the number of variables of the problem, *t* is the problem ID and β is the right hand side parameter. Besides, $c = \lfloor \beta \cdot \sum_{i=1}^{n} w_i \rfloor$, $d = \sum_{i=1}^{n} w_i - c$. It has been done computational experiments on totally 120 problems, that is 10 test problems for each value of r and β.

Optimal solutions of the test problems are obtained by WinQSP [12]. In the library, problem parameters, the optimal solutions and solutions of the algorithms are given in detail.

Results of the computational experiments are given in Table1 for most 3-digit numbers (1-999) and Table2 for most 5-digit numbers (1-99999). The optimal solutions in Table1 and Table2 are colored as gray. These results summarized in Table 3 are given to compare the algorithms.

Table1.

Computational Experiments for most three-digit numbers (r=3)

	β=0.3										β=0.4										β=0.5									
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
A2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	0	0
A4	0	0	0	0	0	1	0	1	0	0	1	0	1	1	0	1	1	1	1	0	0	1	1	0	1	0	1	1	1	1
A3	1	0	0	1	0	1	1	1	0	1	1	1	1	1	0	1	1	1	1	0	1	1	1	1	1	0	0	1	0	0
A5	1	1	1	1	0	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	0	0
	β=0.6										**β=0.7**										**β=0.8**									
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
A2	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1
A4	0	0	1	1	0	1	0	0	0	0	0	1	1	1	1	1	0	1	0	0	1	1	1	1	1	0	0	0	0	0
A3	1	0	1	1	0	1	0	0	1	1	0	1	1	0	0	1	0	1	0	1	0	1	1	1	1	1	0	1	1	0
A5	1	0	1	1	0	1	1	0	1	1	0	1	1	0	0	1	0	1	0	1	0	1	1	1	1	1	0	1	1	0

Table2.

Computational Experiments for most five-digit numbers (r=5)

	β=0.3										β=0.4										β=0.5									
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
A2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
A4	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	1	0	1	1	0	0	0	0	0	0	0	1	0	0	1
A3	0	1	1	0	1	1	0	1	1	1	1	1	1	0	1	1	1	1	1	1	0	1	0	1	0	0	1	0	1	1
A5	0	1	1	0	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	0	1	1	0	1	0	1	1
	β=0.6										**β=0.7**										**β=0.8**									
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
A2	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
A4	0	1	0	0	0	0	1	0	1	0	1	1	1	1	0	0	1	0	1	0	0	0	0	1	0	1	0	1	1	1
A3	0	0	1	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	0	1	1	1	1	1	1	1
A5	0	0	1	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1

Table3.

Comparison of Algorithms

Algorithm	The number of the best solution	%*	The number of the optimal solution	%*	Average approximation ratio	The worst ratios				
A2	113	94%	50	42%	0,005	0,074	0,038	0,032	0,032	0,026
A3	83	69%	35	29%	0,008	0,074	0,069	0,062	0,055	0,038
A4	51	43%	17	14%	0,017	0,093	0,074	0,067	0,065	0,062
A5	93	78%	38	32%	0,007	0,074	0,062	0,038	0,032	0,034

* The number of related solutions per 120

9. Conclusion. As seen above algorithms, to design a guarantee value algorithm for KM is more difficult according to KP. In this study, we have discussed and developed the algorithms A2 and A3. So, A4 and A5 algorithms based on Fractional Algebra are proposed. Using these four algorithms on the test problems, computational experiments and comparisons are made. Results of the computational experiments can be accessed in detail at http://fen.ege.edu.tr/~urfat/knapsack/min/. It is obvious from Table1 and Table2 that A5 algorithm has always better results than A2 algorithm. Also, it is demonstrated that the guarantee value of A5 algorithm is (Δ_5) is equal to 2. Besides, it is observed in tables, that algorithms A2 and A4 are complementary to each other. Therefore, better results can be obtained by a combination of these two algorithms.

References

1. Kellerer H., Pferschy U., Pisinger D. *Knapsack Problems.* – Berlin: Springer, 2004.- 546 p.
2. Fisher, M. L., *Worst-case Analysis of Heuristic Algorithms*, Management Science, 26(1):1-17, 1980.
3. Ibarra, O. H and Kim, C. E., *Fast approximation algorithms for the knapsack and sum of subset problems,* Journal of ACM, 22(4):463-468, 1975.

4. Sahni, S., *Approximate algorithms for the 0/1 knapsack problems*, Journal of ACM, 22(1), 115-124, 1975.
5. Gens G.V., Levner E.V., *Efficient Approximate Algorithms for Combinatorial Problems*, Moscow, Central Institute of Economics and Mathematics of Academy of Sciences of the USSR, Preprint, 1981 - 66p.
6. Güntzer, M. M. and Jungnickel, D., *Approximate minimization algorithms for the 0/1 knapsack and subset-sum problem*, Operations Research Letters, 26, 55-66, 2000.
7. Nuriyev, U. G., *On the solution of the knapsack problem with guaranteed estimate*, Problems of Computing Mathematics and Theoretical Cybernetics, VII, 65-69, Baku, Elm, 1986.
8. Guler, A., Nuriyev, U., Berberler, M. E. and Nuriyeva, F., *Algorithms with Guarantee Value for Knapsack Problems***,** Optimization, 61(4), 477-488, 2012**.**
9. Nikitin, A.I. and Nuriev, U.G., *A heuristic algorithm for solving a linear-fractional Boolean programming problem,* Izvestiya Akad. Nauk Az. SSR, Ser. Fiz.-Tekn.-Math. Nauk, 3(5), 112 – 117, 1982.
10. Gürsoy N. K., Firat A. and Nuriyev U., *On the Algebra of Fractions*, Ege University. Journal of Faculty of Science., 35(2), 73-84, 2011.
11. Babayev, D. A., Mamedov, K. Sh., Mextiev, M. G., *Methods for the suboptimal solution of the multidimensional knapsack problem,* Computational Mathematics and Mathematical Physics, 18(6), 1443-1453, 1978.
12. WinQSB, http://www.wiley.com/college/tech/winqsb.htm.

UOT 519.852.6

N.K. Gürsoy, Ü.G. Nuriyev

Binar dəyişənli minimallaşdırma çanta məsələsi üçün qarantili algoritmlər

Məqalədə çanta məsələsi üçün məlum qarantili algoritmlər analiz edilərək binar dəyişənli minimimallaşdırma çanta məsələsi üçün iki yeni alqoritm hazırlanmış, hesablama eksperimentləri aparılaraq bu alqoritmlərin əvvəlki alqoritmlərlə müqayisəsi verilmiş və ikinci algoritmin qaranti dəyərinin $\Delta = 2$ olduğu göstərilmişdir.

Açar sözlər: 0/1 çanta məsələsi, minimallaşdırma, acgöz alqoritm, ε–yaklaşım, qaranti dəyər

УДК 519.852.6

Н.К. Гюрсой, У.Г. Нуриев

Алгоритмы с гарантированными оценками для минимизационной булевой задачи о ранце

Предложены два гриди-алгоритма для минимизационной булевой задачи о ранце. Приведены результаты численных экспериментов и, для второго алгоритма доказано, что априорная гарантированная оценка равна двум.

Ключевые слова: 0/1 задача о ранце, минимизация, жадный алгоритм, ε-приближенный алгоритм, гарантированная оценка

Ege University, Izmir, TURKEY

Presented 20.05.2015