Computing 40, 1–8 (1988)

Computing

A Simulated Annealing Approach to the Multiconstraint Zero-One Knapsack Problem

A. Drexl, Darmstadt

Received October 22, 1987

Abstract — Zusammenfassung

A Simulated Annealing Approach to the Multiconstraint Zero-One Knapsack Problem. The multiconstraint 0-1 knapsack problem encounters when deciding how to use a knapsack with multiple resource constraints. The problem is known to be NP-hard, thus a "good" algorithm for its optimal solution is very unlikely to exist. We show how the concept of simulated annealing may be used for solving this problem approximately. 57 data sets from literature demonstrate, that the algorithm converges very rapidly towards the optimum solution.

Key words: Binary knapsack problem, multiple constraints, simulated annealing.

Ein Verfahren der simulierten Abkühlung zur Lösung mehrfach beschränkter binärer Knapsack-Probleme. Mehrfach beschränkte binäre Knapsack-Probleme erfordern Engpaßentscheidungen unter Berücksichtigung zahlreicher Ressourcenbeschränkungen. Angesichts der NP-Schwierigkeit ist die Existenz eines „guten" Optimierungsverfahrens zu seiner Lösung sehr unwahrscheinlich. Wir zeigen, wie das Problem mit Hilfe des Konzeptes der simulierten Abkühlung näherungsweise gelöst werden kann. 57 Datensätze aus der Literatur verdeutlichen, daß das Verfahren sehr schnell gegen die optimale Lösung konvergiert.

1. Introduction

This paper is concerned with the multiconstraint 0-1 knapsack problem (MCKP):

$$\text{Maximize } z = \sum_{j=1}^{n} c_j x_j$$

$$\text{Subject to } \sum_{j=1}^{n} a_{ij} x_j \leq b_i \qquad (i = 1, \ldots, m)$$

$$x_j \in \{0, 1\} \qquad (j = 1, \ldots, n).$$

Without loss of generality all c_j, a_{ij} and b_i are assumed to be nonnegative.

MCKP is a special case of general 0-1 programming. It may be used to model many practical problems. Frequently it is encountered in the areas of capital budgeting and resource allocation. Indeed, the paper by Lorie and Savage [14] on capital budgeting is one of the earliest references to this problem.

The problem is known to be NP-hard [11], thus a "good" algorithm for its optimal solution is very unlikely to exist. Only when $m=1$ problems with thousands of variables can be quickly solved to optimality [3]. When $m \leq 3$ problems with up to 300 variables may be solved to optimality within a reasonable amount of time by relaxation methods [12]. For larger m there does not exist any reasonable fast exact algorithm.

Several heuristics have been proposed for solving MCKP approximately. For example Senyu/Toyoda [20, 23] developed primal and dual effective gradient methods, Loulou/Michaelides [15] presented four greedy heuristics, Balas/Martin [2] proposed to solve the LP-relaxation and to look for integral solutions accordingly, Frieze/Clark [10] use the dual simplex algorithm of linear programming, Magazine/Oguz [16] combine the Senyu/Toyoda with a Lagrangian multiplier approach, and Freville/Plateau [8] present a greedy algorithm within a reduction framework.

In the following we use the concept of simulated annealing in order to derive an efficient approximate solution method for MCKP. Recently, simulated annealing has been found to be a highly suitable method for solving hard combinatorial optimization methods; see [4, 5] for quadratic assignment problems, [19] for euclidean traveling salesman problems, [24] for euclidean matching problems, [6] for minimization of waiting times in transit networks, [22] for reconstruction of polycristalline structures, as well as [13] for (a description of simulated annealing by Marcov chains and) other applications.

2. Probabilistic Exchange Algorithm

The use of deterministic exchange algorithms for solving combinatorial optimization problems has the main drawback that – heavily depending on chosen starting solutions – the algorithms quite often only find bad local optima.

Simulated annealing tries to overcome this deficiency by allowing temporary deteriorations of actual solutions, where the deteriorations are controlled by a parameter t (called temperature due to the analogy and inspiration from statistical physics [17]); t determines the mobility of the system.

The parameter t is reduced by a positive factor φ less than 1 in the course of the algorithm such that the likelihood of accepting a deteriorated solution decreases as the algorithm progresses (i.e. the temperature t is changed according to a socalled annealing or cooling schedule). For any given value of t some exchange trials r (repititions) are performed. r is multiplied with a positive constant ρ when t is decreased.

For a formal statement of the algorithm we will use the following symbols regarding any feasible solution $\mathbf{x}$:

$$I0 := \{j \mid x_j = 0 \text{ for } j = 1, \ldots, n\}$$
$$I1 := \{j \mid x_j = 1 \text{ for } j = 1, \ldots, n\}$$

Procedure PROEXC

```
logical change
real t, φ, ρ
integer r
select a feasible solution x randomly;
evaluate z(x)
change:=.true.
while change do
begin
    change:=.false.
    repeat r times
    begin
        choose h∈I0 randomly
        if Σ_{j∈I1∪h} a_ij x_j ≤ b_i ∀ i then
        begin
            I0:=I0−h; I1:=I1∪h; z(x):=z(x)+c_h;
            change:=.true.; goto M2
        end
        choose k∈I1 randomly
        if Σ_{j∈I1∪h−k} a_ij x_j ≤ b_i ∀ i then
        begin
            δ:=c_h−c_k
            if δ<0 then
            begin
                P(δ):=exp(δ/t)
                generate random variable γ uniformely distributed in (0, 1)
                if γ>P(δ) then goto M1
            end
            I0:=I0−h∪k; I1:=I1∪h−k; z(x):=z(x)+δ;
            change:=.true.; goto M2
        end
    M1: P(c_k):=exp(−c_k/t)
        generate random variable γ uniformely distributed in (0, 1)
        if γ<P(c_k) then
        begin
            I0:=I0∪k; I1:=I1−k; z(x):=z(x)−c_k;
            change:=.true.
        end
M2:
    end
    t:=t·φ; r:=r·ρ
end
```

It is quite clear that the quality of solutions obtainable with PROEXC as well as the required computational times heavily depend upon parameters t, φ, r and ρ. Before evaluating suitable parameter configurations some remarks concerning the above algorithm should be made.

Remark 1:

The interchange of variable $h \in I0$ with variable $k \in I1$ constitutes the main part of the algorithm. If such an interchange is impossible due to resource restrictions we propose to exclude or drop variable k from feasible solution in some cases controlled by the temperature t (see $M1$). On the other hand, separate dropping of variables leads itself to separate addition of variables in cases of sufficient resources. Preliminary computational results demonstrated, that such a flexible "add-interchange-drop" concept gives the most promising framework for an annealing algorithm.

Remark 2:

Manipulation of sets $I0$ and $I1$ constitutes the most time consuming part of the algorithm. We implemented a cyclic linked list representation [1] of both sets in one compact array of length n. Thus updates of $I0$ and $I1$ can be done by some simple operations. Accordingly it is not necessary to perform a random search for $h \in I0$ ($k \in I1$) over the whole cycle. Rather we choose these variables only out of the "first" five elements of the sets in order to save computational time.

3. Evaluation of Parameters

Obviously the initial temperature t should depend upon the range of cost coefficients. In order to find an appropriate t we initialized with $t := \alpha \cdot \beta$ where $\alpha := \max\{c_j \mid \forall j\} - \min\{c_j \mid \forall j\}$ and β ranges from 0.1, 0.2 to 0.9. For all these initial values of t (and $\varphi := 0.5$, $r := n$ as well as $\rho := 1.1$) we repeatedly solved some of the larger problems cited in detail below. We found that PROEXC produces relatively best solutions for $\beta := 0.5$.

In order to investigate the impact of φ, we increased φ from 0.2, 0.3 to 0.9 and solved (once more for $r := n$ and $\rho := 1.1$) the above subset of test problems. It turned out to be the best strategy using $\varphi := 0.6$.

In a similar way we tested different values of (repetitions) r ranging from $n/2$ to $3n/2$; we found that $r := n$ is a suitable way of initializing the number of repetitions (balancing solution quality and speed). Last but not least we made some experiments with ρ ranging from 0.9, 1.0 to 1.3 (the other parameters initialized as indicated above). For $\rho := 1.3$ in all cases the optimum solution has been found at least in one of ten repeated applications of PROEXC to the same data set. The computational times being relatively high for $\rho \geq 1.3$ we propose to use $\rho := 1.2$, where we found "good" solutions very rapidly.

Finally we made a two-dimensional search over φ and ρ and found out, that the above observations for the "one-parametric case" also hold for the "two-parametric one".

To sum it all up we will use $\beta := 0.5$ for determining t, $\varphi := 0.6$, $r := n$ as well as $\rho := 1.2$ for initialization in the following computational study.

4. Computational Results

PROEXC has been coded in FORTRAN (using the VSFOR-Compiler with vector processing capabilities) and implemented on an IBM 3090/200. In order to see how PROEXC works we solved 57 test problems from literature (which are fully reproduced in [7, 9]) with well-known optimum solutions.

Table 1. *Results of 30 test problems from literature* [7, 21]

Problem		m	n	PROEXC		DETEXC
				Gap to optimal solution (%)	CPU time in milliseconds	Gap to optimal solution (%)
Wei Shih	1	5	30	0	11.0	1.69
	2			0	6.9	0.71
	3			0	8.7	0
	4			0	5.6	1.97
	5			0	6.2	0
	6		40	0	14.4	0.32
	7			0	11.4	0
	8			0.04	12.4	1.57
	9			0	15.5	0
	10		50	0.58	14.9	2.26
	11			0	14.5	0.34
	12			0.02	15.3	0.85
	13			0	15.9	0.31
	14		60	0	24.8	1.35
	15			0	18.1	0.49
	16			0	28.5	0.03
	17			0	26.9	2.10
	18		70	0	33.0	1.10
	19			0.17	25.3	4.78
	20			0.10	28.1	0.52
	21			0.43	28.0	1.32
	22		80	0.62	31.5	1.98
	23			0.59	38.9	2.26
	24			0.05	36.0	0.20
	25			0.40	29.6	1.30
	26		90	1.00	65.3	1.52
	27			0	68.2	3.03
	28			0.26	64.6	0.73
	29			0.98	33.7	1.36
	30			0.48	35.7	2.53

Table 2. *Results of 27 test problems from literature* [7, 9, 18, 20, 25]

Problem		m	n	PROEXC Gap to optimal solution (%)	PROEXC CPU time in milliseconds	DETEXC Gap to optimal solution (%)
Petersen	1	10	6	0	0.7	0
	2		10	0	5.4	0.64
	3		15	0	4.1	0.25
	4		20	0.16	1.0	1.47
	5		28	0.08	10.5	0.24
	6	5	39	0.32	31.3	1.06
	7		50	0.56	36.8	8.86
Hansen, Plateau	1	4	28	0.37	9.3	2.87
	2		35	0.42	19.0	4.52
Weingartner	1	2	28	0	17.8	0.67
	2			0.63	22.8	4.76
	3			0.27	17.6	1.05
	4			0	12.9	2.95
	5			0.49	25.0	3.05
	6			0.13	15.6	0
	7	2	105	0.10	127.8	0.38
	8			0.68	71.1	1.96
Senju, Toyoda	1	30	60	0	31.4	1.16
	2			0.25	49.4	0.23
Fleisher		10	20	0.78	3.4	2.01
Pb	1	4	27	0.79	9.2	3.53
	2		34	0.41	15.5	5.48
	3	2	19	0.17	10.1	0
	4		29	0.41	10.1	4.48
	5	10	20	0.79	3.0	2.01
	6	30	40	0	12.2	7.73
	7		37	0.10	10.5	2.13

Each problem has been solved 10 times by calling PROEXC. Tables 1 and 2 reproduce the results. In the first column the problems are identified with their authors' name (Pb 1 to Pb 7 are taken from Freville/Plateau [9]). Columns two and three give the dimensions of each problem. Column four shows the percentage deviation of the best found objective function value from the optimum one. Column five gives the average CPU time per calling of PROEXC (input and output excluded) in milliseconds.

For comparative purposes the last columns of both tables give the computational results of a variant of PROEXC, called DETEXC. DETEXC is equivalent with PROEXC with the exception, that only changes improving the objective function value will be performed. The gaps ranging up to at most 9% stress the fact, that deterministic exchange algorithms in general don't produce as good results as probabilistic variants. On the other hand DETEXC is on average twenty times faster than PROEXC.

First of all it should be stressed that PROEXC is a very fast approximation algorithm regarding the overall very low computational times. Then we see that PROEXC found in 23 cases (40.35%) the optimum solution. Furthermore it is remarkable that the gap to optimal solutions never exceeds 1%.

At present a restricted branch and bound approach developed by Gavish/Pirkul [12] (called early termination heuristic) seems to be the best one for solving MCKP with $m \leq 5$. This heuristic has been tested in [12] on random generated problems (which are in general not so hard to solve; see [9]); it generated in all cases solutions within half a percent from the optimum.

Our heuristic works equally fast and equally good on problems with arbitrary relations of m and n. Furthermore it is very easy to implement. Thus it is supposed to be a highly attractive alternative to existing methods.

References

[1] Aho, A. V., Hopcroft, J. E., Ullman, J. D.: Data Structures and Algorithms. Reading, Mass.: Addison-Wesley Publishing Company 1983.

[2] Balas, E., Martin, C. H.: Pivot and complement – a heuristic for 0-1 programming. Management Science *26*, 86–96 (1980).

[3] Balas, E., Zemel, E.: Solving large zero-one knapsack problems. Operations Research *28*, 1130–1154 (1980).

[4] Bonomi, E., Lutton, J.-L.: The asymptotic behaviour of quadratic sum assignment problems: a statistical mechanics approach. European J. of Operational Research *26*, 295–300 (1986).

[5] Burkard, R. E., Rendl, F.: A thermodynamically motivated simulation procedure for combinatorial optimization problems. European J. of Operational Research *17*, 169–174 (1984).

[6] Domschke, W.: Minimierung der Wartezeiten für Umsteiger im öffentlichen Nahverkehr. Forthcoming in OR Spektrum.

[7] Freville, A., Plateau, G.: Méthodes heuristiques performantes pour les problèmes en variables 0-1 à plusieurs constraintes en inegalité. Publication ANO-91, Université des Sciences et Techniques de Lille 1982.

[8] Freville, A., Plateau, G.: Heuristics and reduction methods for multiple constraints 0-1 linear programming problems. European J. of Operational Research *24*, 206–215 (1986).

[9] Freville, A., Plateau, G.: Hard 0-1 multiknapsack test problems for size reduction methods. Prepublications informatique 72, Université Paris-Nord 1987.

[10] Frieze, A. M., Clarke, M. R. B.: Approximation algorithms for the m-dimensional 0-1 knapsack problem. Worst-case and probabilistic analyses. European J. of Operational Research *15*, 100–109 (1984).

[11] Garey, M., Johnson, D.: Computers and Intractability: a Guide to the Theory of NP-Completeness. San Francisco: Freeman 1979.

[12] Gavish, B., Pirkul, H.: Efficient algorithms for solving multiconstraint zero-one knapsack problems to optimality. Math. Programming *31*, 78–105 (1985).

[13] van Laarhoven, P. J. M., Aarts, E. H. L.: Simulated Annealing: Theory and Applications. Dordrecht: D. Reidel Publishing Company 1987.

[14] Lorie, J., Savage, L. J.: Three problems in capital rationing. J. of Business *28*, 229–239 (1955).

[15] Loulou, R., Michaelides, E.: New greedy-like heuristics for the multidimensional 0-1 knapsack problem. Operations Research *27*, 1101–1114 (1979).

[16] Magazine, M. J., Oguz, O.: A heuristic algorithm for the multidimensional zero-one knapsack problem. European J. of Operational Research *16*, 319–326 (1984).

[17] Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., Teller, E.: Equation of state calculation by fast computing machines. J. of Chemical Physics *21*, 1087–1092 (1953).

[18] Petersen, C. C.: Computational experience with variants of the Balas algorithm applied to the selection of R & D projects. Management Science *13*, 736–750 (1967).
[19] Rossier, Y., Troyon, M., Liebling, Th. M.: Probabilistic exchange algorithms and euclidean traveling salesman problems. OR Spektrum *8*, 151–164 (1986).
[20] Senyu, S., Toyoda, Y.: An approach to linear programming with 0-1 variables. Management Science *15*, B196–B207 (1968).
[21] Shih, W.: A branch and bound method for the multiconstraint zero-one knapsack problem. J. Opl. Res. Soc. *30*, 369–378 (1979).
[22] Telley, H., Liebling, Th. M., Mocellin, A.: Reconstruction of polycristalline structures: a new application of combinatorial optimization. Computing *38*, 1–11 (1987).
[23] Toyoda, Y.: A simplified algorithm for obtaining approximate solutions to zero-one programming problems. Management Science *21*, 1417–1427 (1975).
[24] Weber, M., Liebling, Th. M.: Euclidean matching problems and Metropolis algorithm. Zeitschrift für Operations Research *30*, A85–A110 (1986).
[25] Weingartner, H. M., Ness, D. N.: Methods for the solution of the multi-dimensional 0/1 knapsack problem. Operations Research *15*, 83–103 (1967).

A. Drexl
Institut für Betriebswirtschaftslehre
Technische Hochschule Darmstadt
Hochschulstrasse 1
D-6100 Darmstadt
Federal Republic of Germany