Simulated Annealing for the 0/1 Multidimensional Knapsack Problem[1]

QIAN Fubin[2] DING Rui[3]

[2]School of Management, University of Science & Technology of Suzhou, Suzhou 215008, China

[2] Institute of Economics, Molde University College, Molde 6411, Norway

[3]School of Math. Sci. ,Soochow University, Suzhou 215006, China

Abstract: In this paper a simulated annealing (SA) algorithm is presented for the 0/1 multidimensional knapsack problem. Problem-specific knowledge is incorporated in the algorithm description and evaluation of parameters, in order to look into the performance of finite-time implementation of SA. Computational results show that SA performs much better than a genetic algorithm in term of solution time, whilst requiring only a modest loss of solution quality.

Keywords: simulated annealing algorithm, multidimensional knapsack, static cooling schedule, finite-time implementation
AMS Classification: 65K05, 90C10, 90C59

0. Introduction

The NP-hard 0/1 multidimensional knapsack problem (MKP) represents an optimization model, which can formulate many practical problems such as capital budgeting problem, cargo loading (Shih 1979) and cutting stock problems (Gilmore and Gomory 1966).

Several metaheuristics have been developed for MKP, including tabu search (Glover and Kochenberger, 1996, and Hanafi and Freville, 1998) and genetic algorithm (Chu and Beasley, 1998). These metaheuristic approaches have yielded very good results so far.

In this paper, an SA is implemented to solve the 0/1 MKP with static cooling schedule. Here, a reasonable strategy is used to generate random solution in the neighborhood, the relevant parameters concerning cooling schedule are tested in order to obtain finite-time implementation of the algorithm.

This introduction is followed in section 1 by the 0/1 multidimensional knapsack problem formulation. Section 2 describes the approach and preliminary testing for identifying parameters. The computational results are presented in section 3. The conclusions are summarized in section 4.

1. Problem Formulation

The MKP is a classical $0-1$ combinatorial optimization problem that can be applied to various fields such as economics. A set of n items and a set of m resources are given. Each item $j (j = 1, \cdots, n)$ has assigned a profit p_j and a resource consumption value r_{ij} for each resource $i (i = 1, \cdots, m)$. The problem is to identify a subset of all items that leads to the highest total profit and does not exceed the resource

[1]Supported by the National Natural Science Foundation of China (No. 10201026, 10672111)
[2]**Biography:** Qian Fubin(1978-), male, was born in Yancheng, Jiangsu, majored in quantitative logistics, master.
[3]Corresponding author

upper bound b_i. The MKP can be formulated as:

$$\text{maxmize} \quad f = \sum_{j=1}^{n} p_j x_j \tag{1.1}$$

subject to

$$\sum_{j=1}^{n} r_{ij} x_j \le b_i, i = 1, \cdots, m, \quad x_j \in \{0, 1\}, j = 1, \cdots, n. \tag{1.2}$$

The variable x_j is an indicator of item j, if x_j is set to 1, it means item j is selected, or 0 means item j is not selected for $j = 1, \cdots, n$. Equation 1 represents the total profit of selection items and Eq.2 the m resource constraints. A well stated 0/1 MKP assumes that $p_j > 0$ and $r_{ij} \le b_i \le \sum_{j=1}^{n} r_{ij}$ for all $i = 1, \cdots, m, \quad \text{and } j = 1, \cdots, n.$

2. Simulated annealing

Simulated annealing is a local search algorithm capable of escaping from local optima. Its ease of implementation, convergence properties, and its capability of escaping from local optima has made it a popular algorithm over the past decades. Simulated annealing is so named because of its analogy to the process of physical annealing with solids, in which a crystalline solid is heated and then allowed to cool very slowly until it achieves its stable state, i.e. its minimum lattice energy state, and thus is free of crystal defects. Simulated annealing mimics this type of thermodynamic behavior in searching for global optima for discrete optimization problems (DOP).

At each iteration of a simulated annealing algorithm applied to a DOP, the objective function values for two solutions (the current solution and a newly generated neighboring solution) are compared. Better solutions are always accepted, while a fraction of inferior solutions are accepted in the hope of escaping local optima in search of global optima. The probability of accepting non-improving solutions depends on a temperature parameter, which is non-increasing with each iteration of the algorithm.

The key algorithmic feature of simulated annealing is that provides a means to escape local optima by allowing worse moves (i.e. moves to a solution that corresponds to a worse objective function value). As the temperature is decreased to zero, worse moves occur less frequently, and the solution distribution associated with the inhomogeneous Markov chain that models the behavior of the algorithm converges to a distribution in which all the probability is concentrated on the set of globally optimal solutions, which means that the algorithm is asymptotically convergent.

To formally describe simulated annealing algorithm for MKP, some definitions are needed. Let Ω be the solution space; define $\eta(\omega)$ to be the neighborhood function for $\omega \in \Omega$. Simulated annealing starts with an initial solution $\omega \in \Omega$. A neighboring solution $\omega' \in \eta(\omega)$is then generated randomly in most cases. Simulated annealing is based on the Metropolis acceptance criterion, which models how a thermodynamic system moves from its current solution $\omega \in \Omega$ to a candidate solution $\omega' \in \eta(\omega)$, in which the energy content is being minimized. The candidate solution,ω' , is accepted as the current solution based on the acceptance probability

$$P\{\text{Accept } \omega' \text{as next solution}\} = \begin{cases} \exp[(f(\omega') - f(\omega))/t_k] & \text{if } f(\omega') - f(\omega) < 0 \\ 1 & \text{if } f(\omega') - f(\omega) \ge 0 \end{cases} \tag{1.3}$$

Define t_k as the temperature parameter at iteration k, such that

$$t_k > 0 \text{ for all } k \text{ and } \lim_{k\to\infty} t_k = 0. \tag{1.4}$$

In this paper, finite-time implementations of simulated annealing algorithm are considered, which can no longer guarantee to find an optimal solution, but may result in faster executions without losing too much on the solution quality. Simulated annealing algorithm with static cooling schedule (Kirkpatrick et al. 1983) for MKP is outlined in pseudo-code:

```
1. Select an initial solution $\omega = (x_1, \cdots, x_n) \in \Omega$; an initial temperature $t = t_0$ ; control parameter value $\alpha$; final temperature $e$ ; a repetition schedule, $M$ that defines the number of iterations executed at each temperature;
2. Incumbent solution $\leftarrow f(\omega)$ ;
3. Repeat;
4.   Set repetition counter $m = 0$;
5.   Repeat;
6.     Select an integer $i$ from the set $\{1, 2, \cdots, n\}$ randomly;
7.     If $x_i = 0$, pick up item $i$, i.e. set $x_i = 1$, obtain new solution $\omega'$, then
8.       While solution $\omega'$ is infeasible, do
9.         Drop another item $\omega'$ from randomly; denote the new solution as $\omega'$;
10.       Let $\Delta = f(\omega') - f(\omega)$;
11.       While $\Delta \geq 0$ or Random $(0,1) < e^{\frac{\Delta}{t}}$, do $\omega \leftarrow \omega'$;
12.     Else
13.       Drop item $i$, and pick up another item randomly, get new solution $\omega'$;
14.       Let $\Delta = f(\omega') - f(\omega)$;
15.       While $\Delta \geq 0$ or Random $(0,1) < e^{\frac{\Delta}{t}}$, do $\omega \leftarrow \omega'$;
16.     End If
17.     If incumbent solution $< f(\omega)$, Incumbent solution $\leftarrow f(\omega)$;
18.     $m = m + 1$;
19.   Until $m = M$
20.   Set $t = a * t$;
21. Until $t < e$.
```

A set of parameters needs to be specified that govern the convergence of the algorithm, i.e. initial temperature t_0, temperature control parameter α, final temperature e, and Markov chain length M, in order to study the finite-time performance of simulated annealing algorithm. Here t_0 should be the maximal difference in cost between any two neighboring solutions, let t_0 be 500 by rough estimation of our test instances[1] since exact calculation is quite time consuming. Final temperature e is 1.0e-5 due to the magnitude of the test instances. The first test instance from OR-library is selected to tune the rest two parameters. Figure 1 shows the solution quality on the selected test case, with different Markov

[1]The test instances are from OR library, which are available at: http://people.brunel.ac.uk/ mastjjb/jeb/orlib/files/.

chain length M ranging from 10 to 200, i.e. M increases from $0.1n$ to $2n$, where α is equal to 0.85, while Figure 2 illustrates the corresponding solution time. Figure 3 presents the solution quality with different temperature control parameter α, where α ranges from 0.8 to 0.975 with $M = n$; meanwhile Figure 4 gives the relevant solution time. So let temperature control parameter α be 0.85 and Markov chain length M be n, which is the reasonable value combination of these parameters for the following tests for all instances in order to evaluate finite-time implementation of SA.

Figure 1 Solution with different Markov chain length

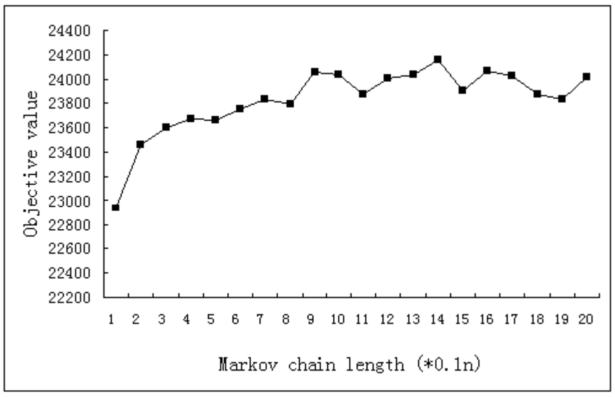


Figure 2 CPU time with different Markov chain length

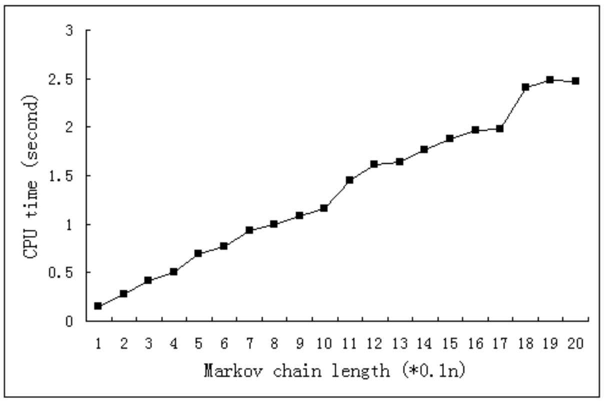


Figure 3 Solution with different α

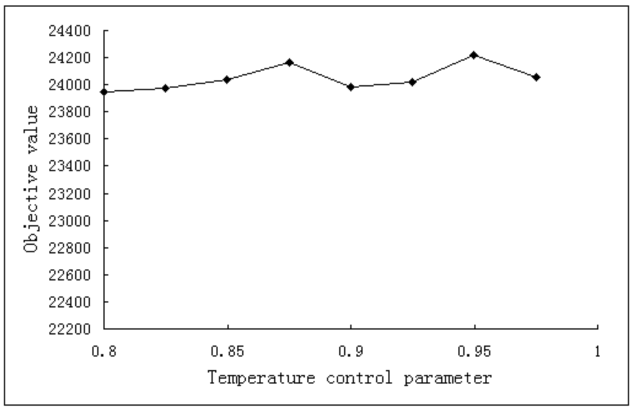
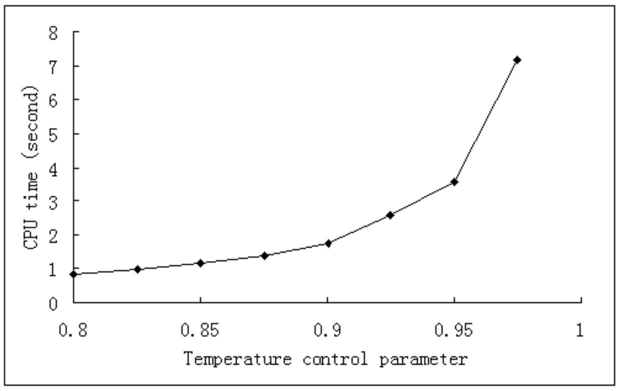


Figure 4 CPU time with different α



3. Computational Results

The standard MKP test data proposed by Chu and Beasley (1998) were used to examine the simulated annealing algorithm. These data contain 10 instances for each combination of and with the tightness ratio . The quality of a solution is measured by the percentage gap between the objective value and the optimal value of the LP-relaxed problem, i.e. 100(optimal LP value- best solution value)/(optimal LP value), since the optimal solution values for most of these problems are not known.

In this paper, the SA code was implemented in Matlab 6.1, running on a 1.4 GHz Pentium M laptop with Microsoft Windows XP, while GA (Chu and Beasley 1998) was run on Silicon Graphics Indigo workstation (R4000, 100 MHz). (A simple whetstone test deemed the laptop to be about 30 times faster than the workstation.). In Table 1, all values are average values determined from runs for 10 different problems. Each test problem was run twenty times and the best solution with its CPU time was recorded, due to random mechanism inherent in SA.

Table 1 directly compares the performance of the SA with other well-known heuristic methods by means of relative percentage gaps. The performance of SA have be compared with the heuristic of Magazine and Oguz (1984) (M&O), the heuristic of Volgenant and Zoon (1990) (V&Z), the heuristic of Pirkul (1987) (MKHEUR) and the heristic of Chu and Beasley (1998) (GA).

Table 1 indicates that SA has the medium performance in term of the quality of the solution obtained in all heuristics mentioned above. SA has better performance than M&O and V&Z on all test instances. For small test problems, MKHEUR is better than SA, but for the more important large problems, SA has better performance in terms of solution quality. The results of GA are one of the best solutions that have been reported so far in term of the solution quality. SA runs much faster than GA even when the test environments are considered; SA also produces very competitive results even for the larger test cases. When the time requirement is a decisive factor in practical problems, SA is a better alternative.

4. Conclusions

In this paper, a heuristic algorithm based on SA is presented for solving multidimensional knapsack problems. Problem-specific knowledge is incorporated in algorithm description and parameters identification, which coordinates tradeoffs between CPU time and solution quality, in order to evaluate the finite-time implementation attributes of SA. The SA algorithm outperforms GA proposed by Chu in terms of solution time, at very limited sacrifice of solution quality.

References

[**1**] Chu P.C. and Beasley J.E. A Genetic Algorithm for the Multidimensional Knapsack Problem[J]. Journal of Heuristic, 1998, 4(1):63-86.

[**2**] Gilmore P.C. and Gomory, R.E. The Theory and Computation of Knapsack Functions[J]. Operations Research, 1966, 14:1045-1075.

[**3**] Glover F. and Kochenberger G.A. Critical Event Tabu Search for Multididensional Knapsack Problems[M]. Metaheuristics: The Theory and Applications, pages: 407-427. Kluwer Academic Publisher, 1996.

[**4**] Hanafi S. and Freville A. An Efficient Tabu Search Approach for the 0-1 Multidimensional Knapsack Problem[J]. European Journal of Operational Research, 1998, 106(2): 659-675.

[**5**]Kirkpatrick S., Gelatt Jr. C.D. and Vecchi M.P. Optimization by Simulated Annealing[J]. Science, 1983, 220(4598): 671-680.

[**6**]Magazine M.J. and Oguz O. A Heuristic Algorithm for the Multidimensional Zero-One Knapsack Problem[J]. European Journal of Operational Research, 1984, 16: 319-326.

[**7**]Pirkul H. A Heuristic Solution Procedure for the Multiconstraint Zero-One Knapsack Problem[J]. Naval Research Logistics, 1987,34(2):161-172.

[**8**] Shih W. A Branch and Bound Method for the Multiconstraint Zero-one Knapsack Problem[J]. Journal of the Operational Research Society, 1979,30:369-378.

[**9**]Volgenant A. and Zoon J.A. An Improved Heuristic for Multidimensional 0-1 Knapsack Problems[J]. Journal of the Operational Research Society, 1990, 41(10): 963-970.

Table1. Performance comparison of the SA with other heuristic methods

			Average % gap and A.E.T [1]						
m	n	α	M& O	V& Z	SA	A.E.T	MKHEUR	GA	A.E.T
5	100	0.25	13.69	10.30	2.95	1.17	1.59	0.99	345.9
		0.50	6.71	6.90	1.67	1.06	0.77	0.45	347.3
		0.75	5.11	5.86	0.87	1.10	0.48	0.32	361.7
		Average	8.50	7.63	1.83	1.11	0.95	0.59	351.6
5	250	0.25	6.64	5.85	3.09	2.91	0.53	0.23	682.0
		0.50	5.22	4.40	1.89	2.73	0.24	0.12	709.4
		0.75	3.56	3.59	0.84	3.34	0.16	0.08	763.3
		Average	5.14	4.61	1.94	3.00	0.31	0.14	718.2
5	500	0.25	4.93	4.11	3.41	5.82	0.22	0.09	1271.9
		0.50	2.96	2.53	2.04	6.02	0.08	0.04	1345.9
		0.75	2.31	2.41	0.92	6.43	0.06	0.03	1412.6
		Average	3.40	3.02	2.13	6.09	0.12	0.05	1343.5
10	100	0.25	15.88	15.55	3.93	1.83	3.43	1.56	384.1
		0.50	10.41	10.72	2.09	1.71	1.84	0.79	418.9
		0.75	6.07	5.67	1.06	1.62	1.06	0.48	462.6
		Average	10.79	10.65	2.36	1.72	2.11	0.94	421.9
10	250	0.25	11.73	10.53	3.21	4.59	1.07	0.51	870.9
		0.50	6.83	5.92	2.14	4.44	0.57	0.25	931.5
		0.75	4.42	3.77	1.01	4.62	0.33	0.15	1011.2
		Average	7.66	6.74	2.21	4.55	0.66	0.30	937.9
10	500	0.25	8.81	7.90	3.67	10.28	0.52	0.24	1504.9
		0.50	5.71	4.14	2.50	9.90	0.22	0.11	1728.8
		0.75	3.62	2.93	1.17	9.78	0.14	0.07	1931.7
		Average	6.05	4.99	2.45	9.99	0.29	0.14	1721.8
30	100	0.25	17.39	17.21	4.94	5.07	9.02	2.91	604.5
		0.50	11.82	10.19	2.62	4.46	3.51	1.34	782.1
		0.75	6.58	5.92	1.29	4.21	2.03	0.83	904.2
		Average	11.93	11.11	2.95	4.58	4.85	1.69	763.6
30	250	0.25	13.54	12.41	3.60	12.63	3.70	1.19	1499.5
		0.50	8.64	7.12	2.25	11.03	1.53	0.53	1980.0
		0.75	4.49	3.91	0.96	10.74	0.84	0.31	2441.4
		Average	8.89	7.81	2.27	11.47	2.02	0.68	1973.6
30	500	0.25	9.84	9.62	3.75	25.18	1.89	0.61	2437.7
		0.50	7.10	5.71	2.30	22.66	0.73	0.26	3198.9
		0.75	3.72	3.51	1.07	20.99	0.48	0.17	3888.2
		Average	6.89	6.28	2.37	22.94	1.03	0.35	3174.9

[1]A.E.T. =average execution time (CPU seconds).