HYBRID GENETIC ALGORITHM AND KNAPSACK PROBLEM IN MATLAB ENVIRONMENT

Radek Matoušek
Institute of Automation and Computer Science
Brno University of Technology

Abstract
This paper discusses applications of specific heuristic methods for solving an optimization problem known as the 0/1 Knapsack problem (KS). A Hill Climbing algorithm (HC) and a Genetic Algorithm (GA) was used for design of special heuristic hybrid algorithm denoted as a GA-HC. In GA algorithm a fuzzy control by FIS was used. The KS problem tested tasks differ in numbers of items and size of restrictions. A different behavior of heuristic algorithms related to size of KS restriction is shown.
Keywords: Hill Climbing algorithm, Genetic Algorithm, Knapsack problem

1 Introduction

This paper deals with short empirical tests of specific heuristic algorithms. A Matlab® environment for implementation of these algorithms was used.

- CW algorithm (greedy strategy)
- HC algorithm (Hill Climbing with h_1-transformation)
- GA algorithm (Genetic Algorithm and GA-FIS)
- GA-HC algorithm (hybrid of GA and HC algorithms)

We must note that the heuristics are algorithms for which we are not able to guarantee the computation of a correct result in a reasonable time. Their basic advantages are simplicity and robustness. Genetic Algorithms and Hill Climbing are well-known representatives of heuristics.

The 0/1 Knapsack problem was chosen as a test problem for these heuristics. The problem is easy to formulate, yet, the given version of it belongs to a family of NP-complete problems. It is an interesting exercise to evaluate the advantages and disadvantages of constraint handling techniques on this particular problem with a single constraint: the conclusions might be applicable to many constrained combinatorial optimization problems [1,4]. For this reason we provide the optimum solutions up to 20 items for the test cases. A brute force for searching for optimal KS problem solutions was used. We do not provide the optimal solutions for more than 20 items for the test cases: we make only some comparisons between presented heuristic methods for time complexity reasons of the KS problem and some restrictions in the MatLab® environment.

2 Test Problem Description

The knapsack problem (KP) is an NP-hard optimization problem [1]. Input consists of a number of objects (items) with given weights and costs. One has a knapsack whose weight capacity is bounded by positive number $b \in \mathbb{R}_+$ (frequently by positive integer b), n items of weights w_i, $i=1,2,\ldots n$ and costs c_i for every item i.

The objective is to maximize the common cost of objects (the profit) packed into the knapsack under the constraint that the common weight of the objects in the knapsack is not above b. Thus, our instance of the knapsack problem (KP) is as fallows:

Input: The vectors w_i and c_i are generated by uniform random generator, parameter b reflects different size [%] of restriction of the KS problem.

$$w_i \in (0,n],\ c_i \in (0,n],\ \text{(uniform random, uncorrelated)}$$

$$b \ldots \{10\%, 50\%, 90\%\} \text{ of } \sum_{i=1}^{n} w_i$$

$$b \in \mathbb{R}_+,\ w_i \in \mathbb{R}_+,\ c_i \in \mathbb{R}_+,\ i=1,2,\ldots n \tag{1}$$

Solution vector: $\mathbf{x} \in \{0,1\}^n$ where

$$x_i = \begin{cases} 1 & \text{if the } i\text{-th object (item) is accepted} \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

Goal: To maximalize

$$f(\mathbf{x}) = \sum_{i=1}^{n} c_i x_i \tag{3}$$

$$\textit{Constraint}: \sum_{i=1}^{n} w_i x_i \le b \tag{4}$$

Table 1 demonstrates complexity of the KS problem brute force solutions in Matlab® environment. We noted, that cited heuristic methods reduce memory usage and are suitable for KS problem with number of items $n>20$. The KS problem can be solved by means of classical mathematical approximate or exact methods e.g. FPTAS [1], dynamic programming or branch and bound method [8].

n	$2^n \times n$ *(array)*	*memory [Bytes]* [**]	*time[s]*
1	2×1	16 B ≡ 16 B	0
10	1024×10	81920 B ≡ 80 kB	0
20	1048576×20	167772160 B ≡ 160 MB	3.5
21	2097152×21	352321536 B ≡ 336 MB	7.4
22	4194304×22	738197504 B ≡ 704 MB	15.2
23	8388608×23	1543503872 B ≡ 1472 MB	31.9

[*] Depends on algorithm and computer hardware (AMD Athlon XP1800+, 768MB, KT266).
[**] Environment: Matlab® R12, max. memory size for variables: ~250MB, used data type: double (8bytes), OS environment: MS Windows™ 98 SE.

Table 1: The Complexity of the KS problem brute force solutions in Matlab® environment.

3 Formal Description of The Heuristic Methods

The heuristic algorithms were implemented as m-function and tested. In all algorithms a binary string of the length n represents a solution $\mathbf{x}$ to the problem: the i-th item is selected for the knapsack iff $x(i) = 1$. The fitness $eval(\mathbf{x})$ of each string is determined as:

$$eval(\mathbf{x}) = \sum_{i=1}^{n} c_i x_i - pen(\mathbf{x}), \qquad (5)$$

where penalty function $pen(\mathbf{x})$ is zero for each feasible solution $\mathbf{x}$, i.e., solutions such that $\sum_{i=1}^{n} w_i x_i \le b$, and is greater than zero otherwise.

3.1 CW algorithm

A simple and quick algorithm [1,2] (simple greedy strategy) destined for KS problem. Coefficients $d_i = c_i / w_i$ are sorted and corresponding items are put to the knapsack until they fill it up (4).

3.2 HC algorithm

Hill climbing techniques, such as gradient descent, use local information about the search space to find optimum. For unimodal function, basic HC algorithms easily outperform most other methods, but for more complicated functions HC algorithms are apt to get stuck in local optima. We use a basic hill climbing algorithm, with following specifications:

- The initial solution vector $\mathbf{x}$ (2) is generated randomly (uniform).
- Next solutions are deterministic and determined by transformation function $h_1(\mathbf{x})$.

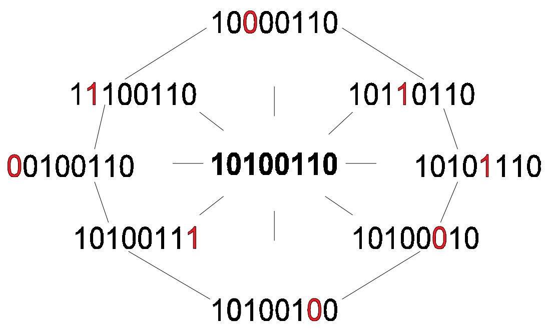

Fig. 1: HC, h_1 transformation, example for n=8.

- The transformation function h_1 produces n neighbor of kernel vector $\mathbf{x}$ [2] with Hamming distance (6), $\rho_H = 1$.

$$\rho_H(a,b) = \sum_{i=1}^{n} |a_i - b_i| \qquad (6)$$

- The objective function including penalization is given by equation (5).

3.3 GA (GA-FIS) algorithm

The concept of genetic algorithm is well known as well as its advantages and disadvantages [3,5,6]. Genetic algorithms are heuristic algorithms whose search methods model some natural phenomena: Mendelian genetic inheritance and Darwinian strife for survival. From the optimization point of view we can denote GA as a robust optimization method.

An objective function (fitness) including penalization is given by (5). The population size of 50 and 200 generations was used. GA parameters were set by GA-FIS. GA-FIS is improving GA with self-control mechanism based on fuzzy inference system (FIS) [4,7].

3.4 GA-HC algorithm

A concept of the hybrid GA-HC algorithm is based on classical genetic algorithms with special HC mutation operator. GA represents global optimization technique and HC is local search estimator. From this conjunction we get optimization method with good attributes from both methods (GA and HC).

Fig. 2 shows the principle of implementation of HC down to the GA individual. The GA optimization process is a classical (GA or GA-FIS) with selection, crossover and mutation operator. Next, a special HC mutation operator is applied with probability

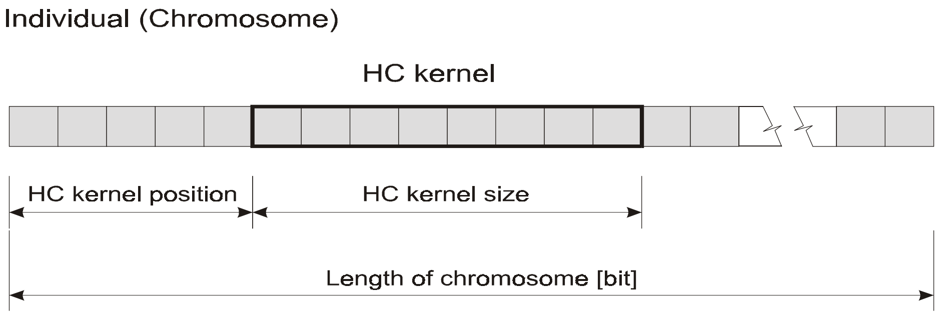

Fig. 2: The principle of implementation of HC down to the GA individual.

$p_{\mathrm{mHC}} \approx 0.05$ on each individual (chromosome). HC mutation is performed by selecting a sub-string (HC kernel) of a specified size and using the HC algorithm to it (Fig. 2).

4 Experimental Results

We used following in experiments:

- Inputs (1) for given set of *n*-items and 90%, 50% and 10% size of restriction.
- The number of experiments (instances with different *n*) for a given restriction is 100.

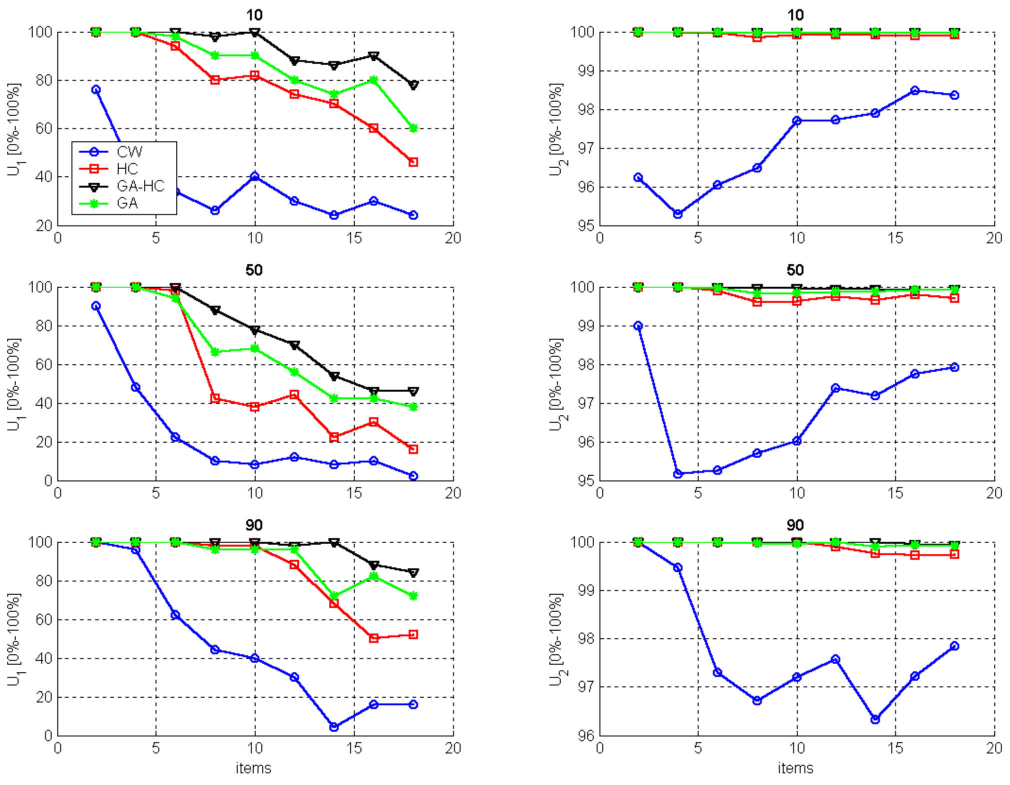

Fig. 3: Comparison of CW, HC, GA and GA-HC methods for different KS restrictions.

The U_1 and U_2 characteristics (7) were designed to compare results of the heuristics:

$$U_1 = \frac{\text{number of runs with optimal heuristic search}}{\text{number of all runs}} \times 100\ [\%]$$

$$U_2 = \frac{\text{sum of costs of all solutions found by heuristic}}{\text{sum of costs of all optimal solutions}} \times 100\ [\%] \tag{7}$$

5 Conclusion

Genetic algorithms and hill climbing techniques have been applied to a wide variety of optimization problems. In this paper we combined the strengths of these two algorithms into a single GA-HC system that outperforms both the GA and HC on a given test suite (KS problem). The performance of a given heuristics (CW, HC, GA and GA-HC) in comparison to optimal solution is shown in Fig. 3 (by means of coefficient U_1 and U_2). The coefficient U_1 denotes the global successfulness of the given heuristics and coefficient U2 denotes practical aspects of given methods. As one can see the GA-HC algorithm is the best for all instances of knapsack problem and for all restrictions of knapsack size. The change of behavior of heuristic algorithms in relation to size of restriction is obvious. The diagrams in Fig.4 clearly show the difference of HC, GA and GA-HC algorithms to the CW algorithm. Optimal solution for numbers of items greater than 20 cannot be found by brute force in Matlab® environment due to memory limitations, so these diagrams contain only differences between reported methods and the CW algorithm.

Presented results show the performance of used methods for KS problem of a given size. For analysis of the promising GA-HC algorithm for KS problem of greater size (>100 items) a change of strategy will be needed due to increasing requirements for system resources, e.g. paralleled or distributed computations. This will be the objective of future work.

Acknowledgements

The paper was supported by research designs CEZ: J22/98: 261100009 “Non-traditional methods for investigating complex and vague systems”, VZ MŠMT No: 260000013, 2211406 and by grant from Grant Agency of the Czech Republic reg. No. 103/01/0675.

Mailto: matousek@uai.fme.vutbr.cz

References

1. Hromkovič, J.: Algorithmic for Hard Problems (Introduction to Combinatorial Optimization, Randomization, Approximation and Heuristics). Spriger-Verlag (2001). ISBN 3-540-66860-8
2. Matoušek, R.: Hill Climbing and 0/1 KSP. In: MENDEL 02’ 8th International Conference on Soft Computing. Brno (2002), p. 325-328, ISBN 80–214-2135-5
3. Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs. Second, Extended Edition, Springer-Verlag (1994).
4. Matoušek, R., Ošmera, P.: A Fuzzy Setting of GA Parameters. Proceeding of 7th Zittau Fuzzy Colloquium (1999), Germany, p.154-160
5. Popela, P., Dvořák, J.: Global optimization and genetic algorithms. Mendel’96, Brno (1996), Czech Republic, p. 205-214
6. Bäck, T.: Evolutionary Algorithms in Theory and Practice. Oxford UP, Oxford (1996).Matoušek, R., Ošmera, P., Roupec, J.: GA with Fuzzy Inference System. Proceedings of the 2000 Congress on Evolutionary Computation - CEC00, USA 2000, p. 646-652
7. Matoušek, R., Ošmera, P., Roupec, J.: GA with FIS. Proceedings - Congress on Evolutionary Computation - CEC00, USA 2000, p. 646-652
8. Klapka, J., Dvořák, J., Popela, P.: Methods of Operation Research, (in Czech), Vutium Brno (2001).

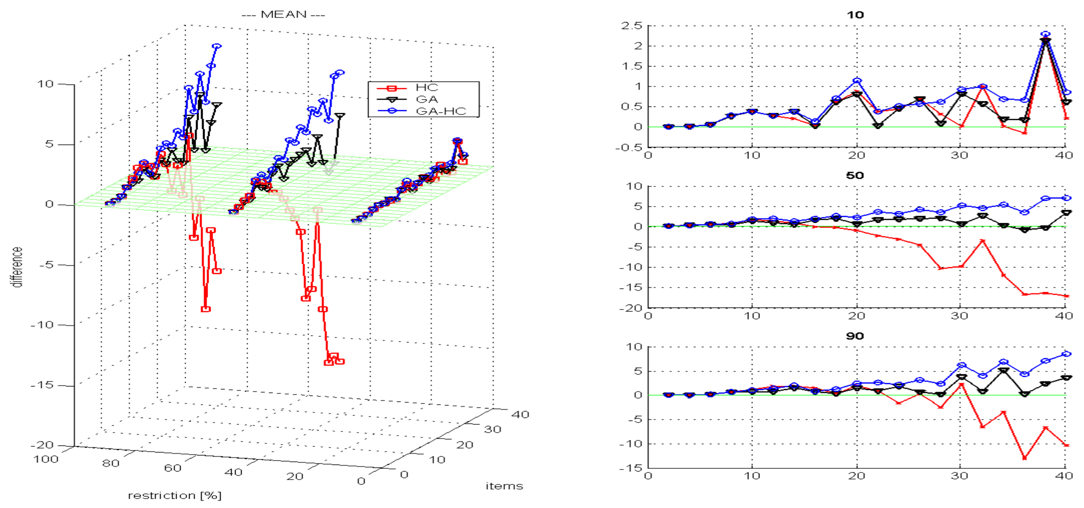

Fig. 4: Comparison (mean values) of HC, GA and GA-HC methods for different KS restriction.