A shuffled complex evolution algorithm for the multidimensional knapsack problem

CIARP 2015
XX Iberoamerican Congress on Pattern Recognition

Marcos Baroni & Flávio Varejão
Universidade Federal do Esprito Santo - Departamento de Informática
marcos.baroni@aluno.ufes.br

Abstract
This work addresses the application of a population based evolutionary algorithm called shuffled complex evolution (SCE) in the multidimensional knapsack problem. The SCE regards a natural evolution happening simultaneously in independent communities. The performance of the SCE algorithm is verified through computational experiments using well-known problems from literature and randomly generated problem as well. The SCE proved to be very effective in finding good solutions demanding a very small amount of processing time.
Keywords: Multidimensional knapsack problem, Meta-heuristics, Artificial Intelligence

Introduction

The multidimensional knapsack problem (MKP) is a strongly NP-hard combinatorial optimization problem which can be viewed as a resource allocation problem and defined as follows:

$$\begin{aligned} &\text{maximize} \sum_{j=1}^{n} p_j x_j \\ &\text{subject to} \sum_{j=1}^{n} w_{ij} x_j \leqslant c_i \quad i \in \{1, \ldots, m\} \\ &x_j \in \{0, 1\}, \quad j \in \{1, \ldots, n\}. \end{aligned}$$

The multidimensional knapsack problem can be applied on budget planning scenarios and project selections [7], cutting stock problems [6], loading problems [9], allocation of processors and databases in distributed computer programs [5].

The problem is a generalization of the well-known knapsack problem (KP) in which $m = 1$. However it is a NP-hard problem significantly harder to solve in practice than the KP. Due its simple definition but challenging difficulty of solving, the MKP is often used to to verify the efficiency of novel metaheuristics. A good review for the MKP is given by [4].

In this paper we address the application of a metaheuristic called shuffled complex evolution (SCE) to the multidimensional knapsack problem. The SCE is a metaheuristic, proposed by Duan in [3], which combines the ideas of a controlled random search with the concepts of competitive evolution and shuffling.

The shuffled complex evolution

The shuffled complex evolution is a population based evolutionary optimization algorithm that regards a natural evolution happening simultaneously in independent communities. The algorithm works with a population partitioned in N complexes, each one having M individuals. In the next Subsection the SCE is explained in more details. In the later Subsection the application of SCE to the multidimensional knapsack problem is considered.

In the SCE a population of $N * M$ individuals is randomly taken from the feasible solution space. After this initialization the population is sorted by descending order according to their fitness and the best global solution is identified. The entire population is then partitioned (shuffled) into N complexes, each containing M individuals. In this shuffling process the first individual goes to the first complex, the second individual goes to the second complex, individual N goes to N-th complex, individual $M + 1$ goes back to the first complex, etc.

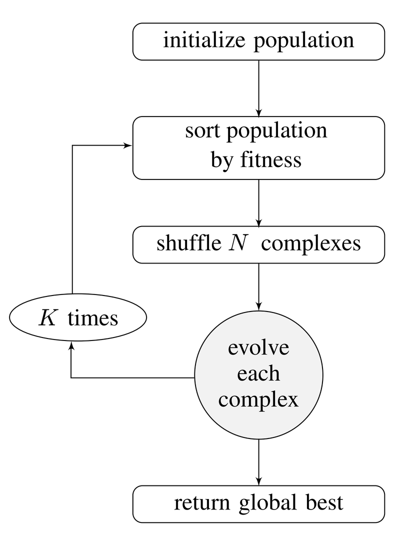

Figure 1: The SCE algorithm overview.

The next step after shuffling the complexes is to evolve each complex through a given fixed amount of K' steps. The individuals in each complex is sorted by descending order of fitness quality. In each step a subcomplex of P individuals is selected from the complex using a triangular probability distribution, where the i-th individual has a probability $p_i = \frac{2(n+1-i)}{n(n+1)}$ of being selected. The use of triangular distribution is intended to prioritize individuals with better fitness, supporting the algorithm convergence rate.

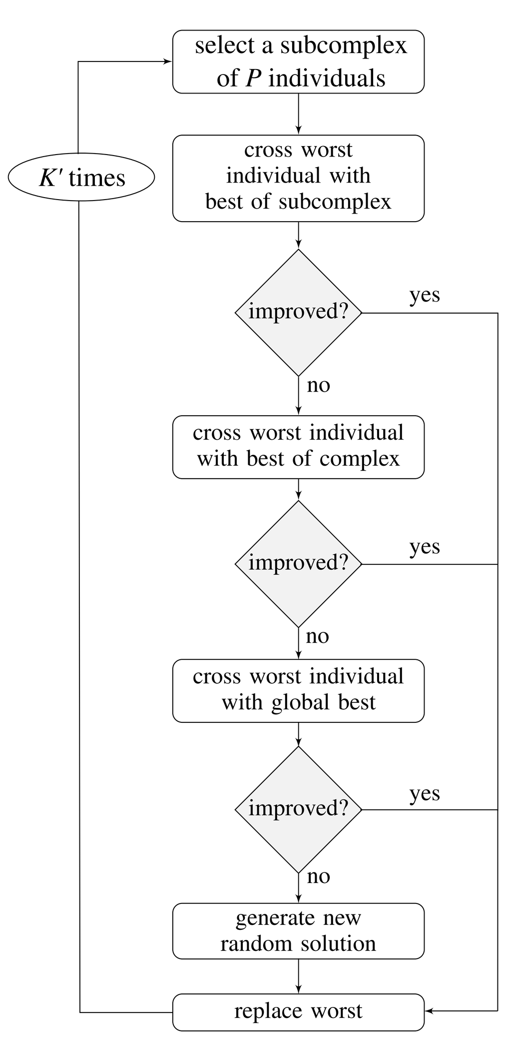

Figure 2: Evolving stage for a single complex.

After the selection of the subcomplex, its worst individual is identified to be replaced by a new generated solution. This new solution is generated by the crossing of the worst individual and an other individual with better fitness. At first the best individual of the subcomplex is considered for the crossing. If the new solution is not better than the worst one, the best individual of the complex is considered for a crossing. If the latter crossing did not result in any improvement, the best individual of whole population is considered. Finally, if all the crossing steps couldn't generate a better individual, the worst individual of the subcomplex is replaced by a new random solution taken from the feasible solution space. This last step is important to prevent the algorithm becoming trapped in local minima. Fig. 2 presents the evolving procedure described above in a flowchart diagram.

After evolving all the N complexes the whole population is again sorted by descending order of fitness quality and the process continues until a stop condition is satisfied. Fig. 1 shows the SCE algorithm in a flowchart diagram.

The SCE for the MKP

As it can be noted in its description the SCE is easly applied to any optimization problem. The only steps needed to be specified is (a) the creation of a new random solution and (b) the crossing procedure of two solutions. These two procedures are respectively presented by Algorithm. 3 and Algorithm **??**.

To construct a new random solution (Algorithm 3) the items are at first shuffled in random order and stored in a list (line 2). A new empty solution is then defined (line 3). The algorithm iteratively tries to fill the solution's knapsack with the an item taken from the list (lines 4-9). The feasibility of the solution is then checked: if the item insertion let the solution unfeasible (line 6) its removed from knapsack (line 7). After trying to place all available items the new solution is returned.

Computational experiments

For the computational experiments a batch of tests was driven to find the best parameters for the problem. Afterwards two main tests was considered: (a) using the well-known set of problems defined by Chu and Beasley [2] and (b) a large set of randomly generated instances using uniform distribution.

Algorithm 1 Generation of a new random solution.

```
 1: procedure NEW RANDOM SOLUTION
 2:     v ← shuffle(1, 2, ..., n)
 3:     s ← ∅                                  ▷ empty solution
 4:     for i ← 1 : n do
 5:         s ← s ∪ {v_i}                      ▷ adding item
 6:         if s is not feasible then          ▷ checking feasibility
 7:             s ← s − {v_i}
 8:         end if
 9:     end for
10:     return s
11: end procedure
```

Algorithm 2 Crossing procedure used on SCE algorithm.

```
 1: procedure CROSSING(x^w : worst individual,
            x^b : better individual, c : n. of genes to be carried)
 2:     v ← shuffle(1, 2, ..., n)
 3:     for i ← 1 : c do
 4:         j ← v_i
 5:         x^w_j ← x^b_j                      ▷ gene carriage
 6:     end for
 7:     if s^w is not feasible then
 8:         repair s^w
 9:     end if
10:     update s^w fitness
11:     return s^w
12: end procedure
```

Figure 3: figure caption

The set of MKP instances provided by Chu and Beasley was generated using a procedure suggested by Freville and Plateau [4], which attempts to generate instances hard to solve. The number of constraints m varies among 5, 10 and 30, and the number of variables n varies among 100, 250 and 500.

The w_{ij} were integer numbers drawn from the discrete uniform distribution $U(0, 1000)$. The capacity coefficient c_i were set using $b_i = \alpha \sum_{j=1}^{n} w_{ij}$ where α is a tightness ratio and varies among 0.25, 0.5 and 0.75. For each combination of (m, n, α) parameters, 10 random problems was generated, totaling 270 problems. The profit p_j of the items were correlated to w_{ij} and generated as follows:

	Value	Description
N	20	# of complexes
M	20	# of individuals in each complex
P	5	# of individuals in each subcomplex
K	300	# of algorithm iterations
K'	20	# of iterations of evolving process
c	$n/5$	# of genes carried from parent in crossing

Table 1: Parameters used in SCE algorithm.

$$p_j = \sum_{i=1}^{m} \frac{w_{ij}}{m} + 500 q_j \qquad j = 1, \ldots, n$$

The second set of instances is composed by problems generated using a similar setup. The only differences is that the profit p_j is also drawn from a discrete uniform distribution $U(0, 1000)$. For each combination of (m, n, α) parameter, 600 random problems was generated, totaling 16200 problems.

All the experiments was run on a Intel Core i5-3570 CPU @3.40GHz computer with 4GB of RAM. The SCE algorithm for MKP was implemented in C programming language. For the set of random instance all best known solution was found by the solver SCIP 3.0.1 running for at least 10 minutes. SCIP [1] is an open-source integer programming solver which implements the branch-and-cut algorithm [8].

n	m	SCE t (s)	gap (%)
100	5	0.81	97.6
	10	0.86	97.0
	30	1.02	96.7
	average gap		**97.1**
250	5	1.75	95.3
	10	1.83	95.0
	30	2.24	94.7
	average gap		**95.0**
500	5	3.23	93.7
	10	3.40	93.7
	30	3.91	93.3
	average gap		**93.6**

Table 2: SCE performance on Chu-Beasley problems.

n	m	α	SCIP t (s)	SCE t (s)	gap (%)
500	10	0.25	278.85	2.23	96.1
		0.50	177.32	2.14	98.4
		0.75	8.47	1.87	99.6
				average gap	**98.0**
	20	0.25	284.11	2.30	96.7
		0.50	275.68	2.16	98.6
		0.75	33.67	1.90	99.7
				average gap	**98.3**
	30	0.25	283.78	2.50	96.9
		0.50	283.54	2.32	98.7
		0.75	71.66	1.96	99.7
				average gap	**98.3**

Table 3: SCE performance on random generated problems.

Conclusions

In this work we addressed the application of the shuffled complex evolution (SCE) to the multidimensional knapsack problem and investigated it performance through several computational experiments.

The SCE algorithm, which combines the ideas of a controlled random search with the concepts of competitive evolution proved to be very effective in finding good solution for hard instances of MKP, demanding a very small amount of processing time to reach high quality solutions for MKP.

Future remarks

Future work includes the investigation of different crossing procedures, the use of local search in the process of evolving complexes and the application of problem reduction procedures for the MKP.

References

[1] Tobias Achterberg. Scip: solving constraint integer programs. *Mathematical Programming Computation*, 1(1):1–41, 2009.

[2] P. C. Chu and J. E. Beasley. A genetic algorithm for the multidimensional knapsack problem. *Journal of Heuristics*, 4(1):63–86, June 1998.

[3] Qingyun Duan, Soroosh Sorooshian, and Vijai Gupta. Effective and efficient global optimization for conceptual rainfall-runoff models. *Water resources research*, 28(4):1015–1031, 1992.

[4] Arnaud Freville and Gérard Plateau. An efficient preprocessing procedure for the multidimensional 0–1 knapsack problem. *Discrete Applied Mathematics*, 49(1):189–212, 1994.

[5] B. Gavish and H. Pirkul. Allocation of data bases and processors in a distributed computing system. 1982.

[6] P. C. Gilmore and R. E. Gomory. The Theory and Computation of Knapsack Functions. *Operations Research*, 14:1045–1074, 1966.

[7] C McMillan and DR Plaine. Resource allocation via 0–1 programming. *Decision Sciences*, 4:119–132, 1973.

[8] Manfred Padberg and Giovanni Rinaldi. A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. *SIAM review*, 33(1):60–100, 1991.

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

Acknowledgements

Research supported by Fundação de Amparo à Pesquisa do Espírito Santo.