A shuffled complex evolution algorithm for the multidimensional knapsack problem using core concept

Marcos Daniel Baroni

Vitória, 24 de agosto de 2015

Sumário

The core concept for MKP

The multidimensional knapsack problem

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:

Modelagem matemática

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

The core concept for MKP

The multidimensional knapsack problem

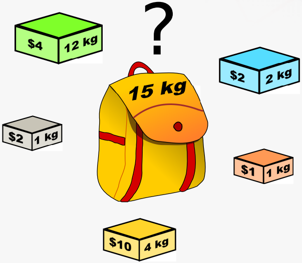
The multidimensional knapsack problem (MKP) is a strongly NP-hard combinatorial optimization problem which can be viewed as a resource allocation problem and defined as follows:

Modelagem matemática

$$\text{maximize} \sum_{j=1}^{n} p_j x_j$$

$$\text{subject to} \sum_{j=1}^{n} w_{ij} x_j \leqslant c_i \quad i \in \{1, \ldots, m\}$$

$$x_j \in \{0, 1\}, \quad j \in \{1, \ldots, n\}.$$



The core concept for MKP

The **core** is a set of items which are hard to decide if they are or not selected in good solutions.

efficiency: $e_i = \dfrac{p_i}{w_i}$

The core concept for MKP

The **core** is a set of items which are hard to decide if they are or not selected in good solutions.

efficiency: $e_i = \frac{p_i}{w_i}$

high efficiency → $x_i = 1$
low efficiency → $x_i = 0$
mid efficiency → $x_i = ?$

split item ↓

	1	2	3	4	5	6	7	8	9	10	11	12	13
efficiency	1.8	1.8	1.7	1.7	1.6	1.4	1.3	**1.2**	1.1	0.9	0.5	0.4	0.2
LP-value	1.0	1.0	1.0	1.0	1.0	1.0	1.0	**0.6**	0.0	0.0	0.0	0.0	0.0

core = {6,7,..,10}

	1	2	3	4	5	6	7	8	9	10	11	12	13
variable fixing (KPC)	1.0	1.0	1.0	1.0	1.0	*	*	*	*	*	0.0	0.0	0.0

The core concept for MKP

The core for KP extended to MKP:

$$e_j(simple) = \frac{p_j}{\sum_{i=1}^{m} w_{ij}}$$

Problem: orders of magnitude of the constraints are not considered.

The core concept for MKP

The core for KP extended to MKP:

$$e_j(simple) = \frac{p_j}{\sum_{i=1}^{m} w_{ij}}$$

Problem: orders of magnitude of the constraints are not considered.
Solution: Introduce a relevance factor for each constraint:

$$e_j(relevance) = \frac{p_j}{\sum_{i=1}^{m} r_i.w_{ij}}$$

Values of an optimal solution to the dual problem (relates to the importance of each resource).

The core concept for MKP

The KP Core problem (KPC) is defined as

Modelagem matemática

$$\begin{aligned}
\text{maximize} & \sum_{j \in C} p_j x_j + \tilde{p} \\
\text{subject to} & \sum_{j \in C} w_j x_j \leqslant c - \tilde{w} \\
& x_j \in \{0, 1\}, \quad j \in C.
\end{aligned}$$

The shuffled complex evolution (SCE)

The shuffled complex evolution is a population based evolutionary optimization algorithm that regards a natural evolution happening simultaneously in independent communities:

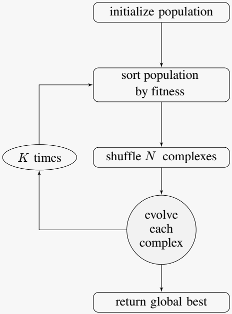
The shuffled complex evolution (SCE)

The shuffled complex evolution is a population based evolutionary optimization algorithm that regards a natural evolution happening simultaneously in independent communities:

The whole population is partitioned in N **complexes**. Each complex has M individuals and evolves independently.

Complexes is periodically *balanced* through **shuffling**.

The shuffled complex evolution (SCE)



initialize population

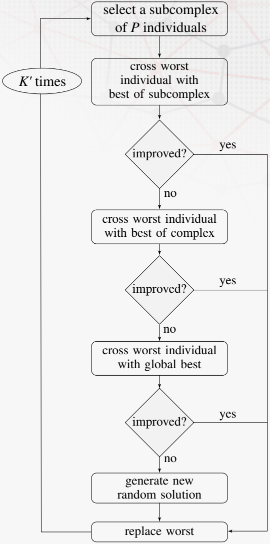
sort population by fitness

shuffle N complexes

evolve each complex

return global best

K times



select a subcomplex of P individuals

cross worst individual with best of subcomplex

improved?

yes

no

cross worst individual with best of complex

improved?

yes

no

cross worst individual with global best

improved?

yes

no

generate new random solution

replace worst

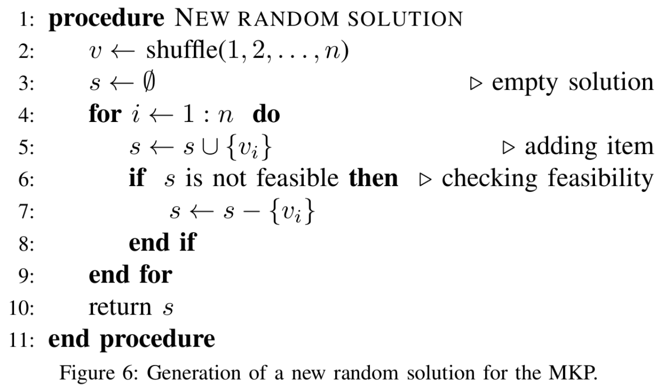
K' times

SCE for the MKP

As it can be noted in its description the SCE is easily applied to any optimization problem.

The only steps needed to be specified is:
  (a) creation of a new random
  (b) the crossing procedure of two solutions

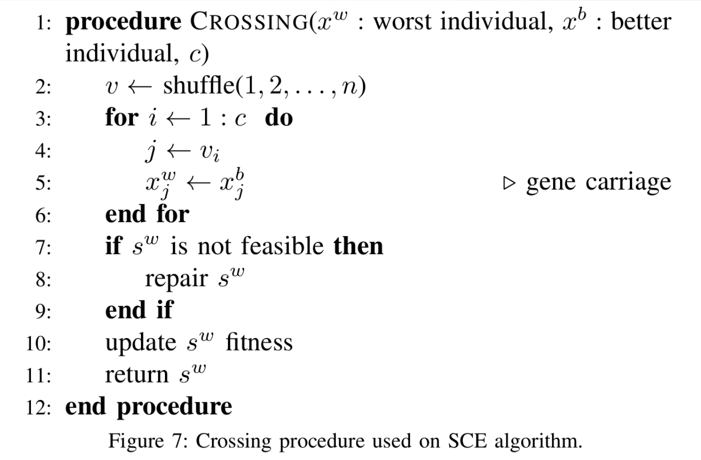
The shuffled complex evolution (SCE)



```
procedure NEW RANDOM SOLUTION
    $v \leftarrow \text{shuffle}(1, 2, \ldots, n)$
    $s \leftarrow \emptyset$                              ▷ empty solution
    for $i \leftarrow 1 : n$ do
        $s \leftarrow s \cup \{v_i\}$                     ▷ adding item
        if $s$ is not feasible then         ▷ checking feasibility
            $s \leftarrow s - \{v_i\}$
        end if
    end for
    return $s$
end procedure
```

Figure 6: Generation of a new random solution for the MKP.

The shuffled complex evolution (SCE)



```
procedure CROSSING($x^w$ : worst individual, $x^b$ : better individual, $c$)
    $v \leftarrow$ shuffle$(1, 2, \ldots, n)$
    for $i \leftarrow 1 : c$ do
        $j \leftarrow v_i$
        $x^w_j \leftarrow x^b_j$                                ▷ gene carriage
    end for
    if $s^w$ is not feasible then
        repair $s^w$
    end if
    update $s^w$ fitness
    return $s^w$
end procedure
```

Figure 7: Crossing procedure used on SCE algorithm.

Computational experiments

Parameter	**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 for each evolving process
c	$n/5$	# of genes carried (crossing)

Computational experiments

n	**m**	α	**SCE time (s)**	**quality (%)**
100	5	0.25	0.15	99.73
		0.50	0.15	99.86
		0.75	0.13	99.91
	10	0.25	0.22	99.53
		0.50	0.22	99.76
		0.75	0.22	99.96
	30	0.25	1.00	99.02
		0.50	0.79	99.21
		0.75	0.77	99.52
			average quality	**99.50**

Table : SCE performance on Chu-Beasley problems with 100 items.

Computational experiments

n	**m**	α	**SCE time (s)**	**quality (%)**
250	5	0.25	0.54	99.87
		0.50	0.55	99.94
		0.75	0.51	99.95
	10	0.25	0.67	99.57
		0.50	0.62	99.80
		0.75	0.64	99.88
	30	0.25	1.16	99.46
		0.50	0.96	99.36
		0.75	1.03	99.59
	average quality			**99.60**

Table : SCE performance on Chu-Beasley problems with 250 items.

Computational experiments

n	**m**	α	**SCE time (s)**	**quality (%)**
500	5	0.25	0.85	99.77
		0.50	0.86	99.87
		0.75	0.84	99.92
	10	0.25	0.99	99.49
		0.50	0.97	99.78
		0.75	0.95	99.83
	30	0.25	1.53	99.75
		0.50	1.48	99.42
		0.75	1.42	99.68
			average quality	**99.61**

Table : SCE performance on Chu-Beasley problems with 500 items.

Computational experiments

#	**n**	**m**	**SCE time (s)**	**quality (%)**
01	100	15	0.31	99.68
02	100	25	0.47	99.51
03	150	25	0.79	99.60
04	150	50	1.61	99.10
05	200	25	0.83	99.73
06	200	50	1.67	99.30
07	500	25	1.27	99.72
08	500	50	2.06	99.62
09	1500	25	1.83	99.32
10	1500	50	5.25	99.76
11	2500	100	11.94	99.77
			average quality	**99.46**

Table : SCE performance on Glover-Kochenberger problems.

Conclusions and future remarks

The SCE algorithm proved to be able to achieve fast convergence ratio, finding good quality near optimal solutions, demanding small amount of computational time.

The application of the core concept for MKP proved to be efficient to reduce the size of the problems which provided fast execution time yet producing high quality solutions.

The heuristic could achieved 99.61% on average of quality of the best known solution for the 270 Chu-Beasley instances and 99.46% on average for the Glover-Kochenberger instances.

Future works includes the investigation of different crossing procedures and the use of local search in the process of evolving complexes.

Thank you!