An elite opposition-flower pollination algorithm for a 0-1 knapsack problem

Mohamed Abdel-Basset*
Department of Operations Research,
Faculty of Computers and Informatics,
Zagazig University,
El-Zera Square, Zagazig, Sharqiyah, Egypt
Email: analyst_mohamed@yahoo.com
*Corresponding author

Yongquan Zhou
College of Information Science and Engineering,
Guangxi University for Nationalities,
Nanning 530006, China
Email: yongquanzhou@126.com

Abstract: The knapsack problem is one of the most studied combinatorial optimisation problems with various practical applications. In this paper, we apply an elite opposition-flower pollination algorithm (EFPA), to solve 0-1 knapsack problem, an NP-hard combinatorial optimisation problem. The performance of the proposed algorithm is tested against a set of benchmarks of knapsack problems. Computational experiments with a set of large-scale instances show that the EFPA can be an efficient alternative for solving 0-1 knapsack problems.

Keywords: flower pollination algorithm; meta-heuristics; combinatorial optimisation; NP-hard; optimisation; knapsack problems.


Biographical notes: Mohamed Abdel-Basset received his MS and PhD in Operations Research from Zagazig University. Currently, he is a Head of Operations Research Department, Faculty of Computers and Informatics, Zagazig University. He is also an reviewer/editor in different international journals and conferences. He has published more than 78 refereed articles in national and international journals, edited books, and conference proceedings. His current research interests are mathematics, operations research, statistics, networks, optimisation, intelligent computing, computer theory, artificial intelligence, and decision support systems.

Yongquan Zhou received his MS degree in Computer Science from Lanzhou University, Lanzhou, China in 1993 and PhD in Pattern Recognition and Intelligence System from the Xiandian University, Xi’an, China in 2006. He is currently a Professor in Guangxi University for nationalities. His research interests include computation intelligence, neural networks, and intelligence information processing etc. He has published one book, and more than 150 research papers in journals.

1 Introduction

Knapsack problems have been intensively studied since the pioneering work of Dantzig in the late 50s, both because of their immediate applications in industry and financial management, but more pronounced for theoretical reasons, as knapsack problems frequently occur by relaxation of various integer programming problems (Wäscher et al., 2007; Bas, 2011; Nawrocki et al., 2009; Mavrotas et al., 2008). In such application, we need to solve a knapsack problem each time a bounding function is derived, demanding extremely fast solution techniques (Papadimitriou and Steiglitz, 1998).

The family of knapsack problems all requires a subset of some given items to be chosen such that the corresponding profits sum is maximised without exceeding the capacity of the knapsack(s). Different types of knapsack problems occur, depending on the distribution of the items and knapsacks: In the 0-1 knapsack problem each item may be chosen at most once, while in the bounded knapsack problem we have a bounded amount of each item type (Papadimitriou and Steiglitz, 1998). The multiple-choice...
knapsack problem occurs when the items should be chosen from disjoint classes and, if several knapsacks are to be filled simultaneously, we get the multiple knapsack problem. The most general form is the multi-constrained knapsack problem, which basically is a general integer programming (IP) problem with positive coefficients (Papadimitriou and Steiglitz, 1998).

In recent decades, many heuristic algorithms have been employed to solve 0-1 knapsack problems. Harmony search algorithm (Lee and Geem, 2005) is a recently developed heuristic algorithm which is inspired by the phenomenon of musician attuning. And in 2010, Zou et al. (2011) proposed a novel global harmony search algorithm (NGHS) to solve 0-1 knapsack problems. The proposed algorithm includes two important operations: position updating and genetic mutation with a small probability. The former can effectively prevent the NGHS from trapping into the local optimum. Chu and Beasley (1998) proposed a heuristic based upon genetic algorithms for the multidimensional knapsack problem. The proposed model is capable of obtaining high-quality solutions for problems of various characteristics. Hembecker et al. (2007) presented the application of particle swarm optimisation (PSO) for the problem and the results are very close or equal to the optimal solution known (Trelea, 200). Zhou et al. (2016a) present complex-valued encoding bat algorithm for solving 0-1 knapsack problem. As a result, it is meaningful to explore some new bio-inspired methods to handle the 0-1 knapsack problem.

This paper presents an elite opposition-flower pollination algorithm (EFPA) for solving 0-1 knapsack problems. A previous version of EFPA is presented in Zhou et al. (2016b) validated by a set of multi-modal benchmarks and some structure engineering design problems, which demonstrates that EOFPA has a fast convergence speed, a relatively high degree of stability and much more accurate in precision. Two major optimisation strategies, including global elite opposition-based learning and local self-adaptive greedy strategy are defined in the EFPA. For increasing the probability of obtained a better solution to the problem in global search process and enhances the diversity of the population, global elite opposition-based learning strategy is applied to the proposed EOFPA. For improving the local search, the local self-adaptive greedy strategy is added to the proposed algorithm, which can enhances its exploitation ability. Based on a large number of experiments, the proposed algorithm has demonstrated promising performance on solving 0-1 knapsack problems, and it can find the required optima in some cases when the problem to be solved is too complicated and complex.

A large variety of resource allocation problems can be cast in the framework of a knapsack problem. The general idea is to think of the capacity of the knapsack as the available amount of a resource and the item types as activities to which this resource can be allocated. Two quick examples are the allocation of an advertising budget to the promotions of individual products and the allocation of your effort to the preparation of final exams in different subjects. Formally, suppose we are given the following parameters:

\[ w_k \] the weight of each type-\( k \) item, for \( k = 1, 2, ..., N \)

\[ r_k \] the value associated with each type-\( k \) item, for \( k = 1, 2, ..., N \)

\( c \) the weight capacity of the knapsack.

Then, our problem can be formulated as:

\[
\max \sum_{k=1}^{N} r_k x_k
\]

\[ \text{s.t. } \sum_{k=1}^{N} w_k x_k \leq C \]  

(1)

where \( x_1, x_2, ..., x_N \) are non-negative integer-valued decision variables, defined by \( x_k = \) the number of type-\( k \) items that are loaded into the knapsack (Papadimitriou and Steiglitz, 1998).

This paper is organised as follows: after introduction, the original flower pollination algorithm is briefly introduced. In Section 3, the proposed algorithm is described, while the results are discussed in Section 4. Finally, conclusions are presented in Section 5.

2 The original flower pollination algorithm

Flower pollination algorithm (FPA) was founded by Yang (2012). Inspired by the flow pollination process of flowering plants are the following rules:

\textit{R1} Biotic and cross-pollination can be considered as a process of global pollination process, and pollen-carrying pollinators move in a way that obeys Lévy flights.

\textit{R2} For local pollination, a biotic and self-pollination are used.

\textit{R3} Pollinators such as insects can develop flower constancy, which is equivalent to a reproduction probability that is proportional to the similarity of two flowers involved.

\textit{R4} The interaction or switching of local pollination and global pollination can be controlled by a switch probability \( p \in [0, 1] \), with a slight bias toward local pollination.

In order to formulate updating formulas, we have to convert the aforementioned rules into updating equations. For example, in the global pollination step, flower pollen gametes are carried by pollinators such as insects, and pollen can travel over a long distance because insects can often fly and move in a much longer range (Yang, 2012). Therefore, Rule 1 and flower constancy can be represented mathematically as:
\[
x_{i}^{t+1} = x_{i}^{t} + \gamma L(\lambda)(x_{i}^{t} - B)
\]
(2)

where \( x_{i}^{t} \) is the pollen \( i \) or solution vector \( x_{i} \) at iteration \( t \), and \( B \) is the current best solution found among all solutions at the current generation/iteration. Here \( \gamma \) is a scaling factor to control the step size. In addition, \( L(\lambda) \) is the parameter that corresponds to the strength of the pollination, which essentially is also the step size. Since insects may move over a long distance with various distance steps, we can use a Lévy flight to imitate this characteristic efficiently. That is, we draw \( L > 0 \) from a Lévy distribution:

\[
L \sim \frac{2\Gamma(\lambda)}{\pi} \frac{1}{S^{1+\lambda}}, \quad (S \gg S_0 > 0)
\]
(3)

Here, \( \Gamma(\lambda) \) is the standard gamma function, and this distribution is valid for large steps \( s > 0 \).

**Figure 1** Pseudo code of the flower pollination algorithm

```plaintext
Flower pollination algorithm

Define Objective function \( f(x) \), \( x = (x_1, x_2, \ldots, x_d) \)
Initialise a population of \( n \) flowers/pollen gametes with random solutions
Find the best solution \( B \) in the initial population
Define a switch probability \( p \in [0, 1] \)
Define a stopping criterion (either a fixed number of generations/iterations or accuracy)
while \( t < \text{MaxGeneration} \)
for \( i = 1 : n \) (all \( n \) flowers in the population)
if \( \text{rand} < p \)
Draw a (\( d \)-dimensional) step vector \( L \) which obeys a Lévy distribution
Global pollination via \( x_{i}^{t+1} = x_{i}^{t} + L(B - x_{i}^{t}) \)
else
Draw \( U \) from a uniform distribution in \((0,1)\)
Do local pollination via \( x_{i}^{t+1} = x_{i}^{t} + U(x_{j}^{t} - x_{i}^{t}) \)
end if
Evaluate new solutions
If new solutions are better, update them in the population
Find the current best solution \( B \)
end for
Output the best solution found
```

Then, to model the local pollination, both Rule 2 and Rule 3 can be represented as

\[
x_{i}^{t+1} = x_{i}^{t} + U(x_{j}^{t} - x_{i}^{t})
\]
(4)

where \( x_{j}^{t} \) and \( x_{i}^{t} \) are pollen from different flowers of the same plant species. This essentially imitates the flower constancy in a limited neighbourhood. Mathematically, if \( x_{j}^{t} \) and \( x_{i}^{t} \) comes from the same species or selected from the same population, this equivalently becomes a local random walk if we draw \( U \) from a uniform distribution in \([0, 1]\). Though flower pollination activities can occur at all scales, both local and global, adjacent flower patches or flowers in the not-so-far-away neighbourhood are more likely to be pollinated by local flower pollen than those faraway. In order to imitate this, we can effectively use the switch probability like in Rule 4 or the proximity probability \( p \) to switch between common global pollination to intensive local pollination. To begin with, we can use a naive value of \( p = 0.5 \) as an initially value. A preliminary parametric showed that \( p = 0.8 \) might work better for most applications (Yang, 2012).

The basic steps of FPA can be summarised as the pseudo-code shown in Figure 1.

**Figure 2** Pseudo code of the EFPA

```plaintext
Elite Opposition- Flower Pollination Algorithm

Define Objective function \( f(x) \), \( x = (x_1, x_2, \ldots, x_d) \);
Initialise a population of \( n \) flowers/pollen gametes with random solutions;
Find the best solution \( B \) in the initial population;
Define a switch probability \( p \in (0,1) \);
Define a stopping criterion (either a fixed number of generations/iterations or accuracy)
while \( t < \text{MaxGeneration} \)
for \( i = 1:n \) (all \( n \) flowers in the population)
if \( \text{rand} < p \)
Draw a (\( d \)-dimensional) step vector \( L \) which obeys a Lévy distribution;
Global pollination by \( x_{i}^{t+1} = x_{i}^{t} + L(B - x_{i}^{t}) \);
else
Draw \( U \) from a uniform distribution in \((0,1)\);
Do local pollination by \( x_{i}^{t+1} = x_{i}^{t} + U(x_{j}^{t} - x_{i}^{t}) \) and get new solution \( x_{i}^{t} \);
Use GEOLS to generate the elite opposition-based solution \( x_{i}^{t} \) by \( x_{i j} = x_{i} + k \cdot \Phi \cdot \Delta \) where \( k = 1,2,\ldots, n; j = 1,2,\ldots, D \);
Choose the better one between \( x_{i}^{t} \) and \( x_{i j}^{t} \) as the individual of next generation.
end if
Endif
Evaluate the new solutions;
If new solutions are better, update them in the population;
end for
Find the current best solution \( B \);
Update the switch probability \( p \) by \( p = 0.6 - 0.1 \times (\text{MaxIter} - t)/\text{MaxIter} \);
end while
```

3 Elite opposition – flower pollination algorithm for 0-1 knapsack problem

Flower pollination algorithm can easily solve low-dimensional unimodal optimisation problems. Whereas when handling the high-dimensional and multi-modal optimisation problems, we can clearly discover that the solutions obtained by FPA are not good enough. In order to enhance exploitation and exploration abilities of FPA, in this paper, an elite opposition-based flower pollination algorithm (EFPA) has been applied to 0-1 knapsack problem (Zhou et al., 2016b). The improvement involves two major optimisation strategies. Global elite opposition-based learning enhances the diversity of the population
An elite opposition-flower pollination algorithm for a 0-1 knapsack problem

(Zhou et al., 2016b), and the local self-adaptive greedy strategy enhances its exploitation ability (Zhou et al., 2016b). The basic steps of EFPA can be summarised as the pseudo-code shown in Figure 2.

In global search process, basic flower pollination algorithm use Lévy flight. It is simulated by Lévy distribution. As we know that it is a stochastic process, the probability of getting a good solution is relatively low. For increasing the probability of obtained a better solution to the problem in global search process and expand the searching space, the elite opposition-based strategy is applied to the proposed EFPA. In essence, it is a greedy strategy. Elite opposition-based learning is a new technique in the field of intelligence computation. Its main ideology is: for a feasible solution, calculate and evaluate the opposite solution at the same time, and choose the better one as the individual of next generation. In this paper, individual with the best fitness value in the population is viewed as the elite individual.

Standard flower pollination algorithm use DE to conduct local search. And it is known to us that the searching ability of standard DE is not good enough to solve high-dimensional optimisation problems. For improving the local search, this local self-adaptive greedy strategy is added to the proposed algorithm. By using this strategy, the adjustment of greedy solution is bigger at the beginning of the iterations. It helps to expand the searching space, enhance the ability of jump out of the local minima. And the adjustment of greedy solution is smaller in the end, which helps to enhance the local search and speed up finding the best solution.

4 Experimental results and analysis

In this section, we will carry out numerical simulation based on a set of benchmarks of knapsack problems to investigate the performances of the EFPA algorithm. Here, two sets of small-scale knapsack problems are selected to test the performance of EFPA. For the first set, the dimension and parameters of test problems are presented in Table 1. The initial parameters are set at n = 50, p = 0.8 and the number of iterations is set to t = 1,000. The performance of EFPA algorithm on seven test problems are reported in Table 1. The results have demonstrated the superiority of the EFPA to finding the optimal solution. All computational experiments are conducted with MATLAB 7.0.

<table>
<thead>
<tr>
<th>Test problem</th>
<th>Diminution</th>
<th>Parameters (w, c, r)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test problem #1</td>
<td>4</td>
<td>w = (2, 4, 6, 7)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>c = 11</td>
</tr>
<tr>
<td></td>
<td></td>
<td>r = (6, 10, 12, 13)</td>
</tr>
<tr>
<td>Test problem #2</td>
<td>15</td>
<td>w = (56.358531, 80.874050, 47.987304, 89.596240, 74.660482, 85.894345, 51.353496, 1.498459, 36.445204, 16.589862, 44.569231, 0.466933, 37.788018, 57.118442, 60.716575)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>c = 375</td>
</tr>
<tr>
<td></td>
<td></td>
<td>r = (0.125126, 19.330424, 58.500931, 35.029145, 82.284005, 17.410810, 71.050142, 30.399487, 9.140294, 14.731285, 98.852504, 11.908322, 0.891140, 53.166295, 60.176397)</td>
</tr>
<tr>
<td>Test problem #3</td>
<td>10</td>
<td>w = (30, 25, 20, 18, 17, 11, 5, 2, 1, 1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>c = 60</td>
</tr>
<tr>
<td></td>
<td></td>
<td>r = (20, 18, 17, 15, 15, 10, 5, 3, 1, 1)</td>
</tr>
<tr>
<td>Test problem #4</td>
<td>7</td>
<td>w = (31, 10, 20, 19, 4, 3, 6)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>c = 50</td>
</tr>
<tr>
<td></td>
<td></td>
<td>r = (70, 20, 39, 37, 7, 5, 10)</td>
</tr>
<tr>
<td>Test problem #5</td>
<td>23</td>
<td>w = (983, 982, 981, 980, 979, 978, 488, 976, 972, 486, 486, 972, 972, 485, 485, 969, 966, 483, 964, 963, 961, 958, 959)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>c = 10000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>r = (981, 980, 979, 978, 977, 976, 487, 974, 970, 485, 485, 970, 970, 484, 484, 976, 974, 982, 962, 961, 959, 958, 857)</td>
</tr>
<tr>
<td>Test problem #6</td>
<td>5</td>
<td>w = (15, 20, 17, 8, 31)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>c = 80</td>
</tr>
<tr>
<td></td>
<td></td>
<td>r = (33, 24, 36, 37, 12)</td>
</tr>
<tr>
<td>Test problem #7</td>
<td>20</td>
<td>w = (84, 83, 43, 4, 44, 6, 82, 92, 25, 83, 56, 18, 58, 14, 48, 70, 96, 32, 68, 92)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>c = 879</td>
</tr>
<tr>
<td></td>
<td></td>
<td>r = (91, 72, 90, 46, 55, 8, 35, 75, 61, 15, 77, 40, 63, 75, 29, 75, 17, 78, 40, 44)</td>
</tr>
</tbody>
</table>
The EFPA was tested using seven benchmark problems. The results are compared to different algorithms. The results obtained proved the capability of the EFPA for reaching optimal solution. Referring to Tables 2 and 3 it could be noticed that the EFPA can be an efficient alternative for solving 0-1 knapsack problems.

Table 3  The optimisation results of the EFPA

<table>
<thead>
<tr>
<th>Test problem</th>
<th>Best</th>
<th>Worth</th>
<th>Mean</th>
<th>SD</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>23</td>
<td>23</td>
<td>23</td>
<td>0</td>
</tr>
<tr>
<td>P2</td>
<td>481.07</td>
<td>481.07</td>
<td>481.07</td>
<td>0</td>
</tr>
<tr>
<td>P3</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>0</td>
</tr>
<tr>
<td>P4</td>
<td>107</td>
<td>107</td>
<td>107</td>
<td>0</td>
</tr>
<tr>
<td>P5</td>
<td>9767</td>
<td>9767</td>
<td>9767</td>
<td>0</td>
</tr>
<tr>
<td>P6</td>
<td>130</td>
<td>130</td>
<td>130</td>
<td>0</td>
</tr>
<tr>
<td>P7</td>
<td>1,025</td>
<td>1,025</td>
<td>1,025</td>
<td>0</td>
</tr>
</tbody>
</table>

The second set which contains five instances is taken from http://www.math.mtu.edu/~kreher/cages/Data.html. Number of items in these instances ranging between 8 and 24. For a fair comparison with the results obtained from Zhou et al. (2016c), the individual run time is set as 50. Four algorithms, GGA (He et al., 2007; Houck et al., 1995), DGHS (Xiang et al., 2014), HS (Lee and Geem, 2005; Yang, 2009) and CGMA (Zhou et al., 2016; Zhao and Tang, 2008), are compared to evaluate the performance of EFPA. In addition, the other specific parameters of the algorithms are given as follows: In GGA, crossover rate $P_c = 0.5$, mutate rate $P_x = 0.05$. In HS, harmony memory consideration rate $HMCR = 0.95$, pitch adjusting rate $PAR = 0.3$. In DGHS, the harmony memory consideration rate, $HMCR_{\text{max}} = 0.95$, $HMCR_{\text{min}} = 0.3$ and pitch adjusting rate $PAR = 0.75$.

Table 4 shows the results obtained by the algorithms for the second set. As observed from the results, EFPA performs as best as DGHS and CGMA for test problem 8 and 10. Seen in mean and best, only CGMA can reach the same value with EFPA in test problem 9 and 11. In addition, the performances of EFPA are better than the other four algorithms. It is clear to see the superiority in EFPA compared with GGA, DGHS, HS in terms of ANOVA tests as shown in the figure (Dexuan et al., 2011; Yoshizawa and Hashimoto, 2000; Zhao, 2007). Though the best values of all the five algorithms are same for each of the five test problems, the boxplot for EFPA have greater value and less height than others. It indicates that the proposed method has higher degree stability than other algorithms in solving the given test problems. The curves in the figure (Zhou et al., 2016b; You, 2007; Fayard and Plateau, 1975) show that EFPA has a faster convergence speed in solving each problem instances. As shown in figures, the curves of algorithms are all smooth. Furthermore, the curve of EFPA reach the near-optimal value in the early phase, which means EFPA is the algorithm with high level of searching.
Table 4  Results obtained by the algorithms for the second set

<table>
<thead>
<tr>
<th>Problem</th>
<th>Instance</th>
<th>GGA</th>
<th>DGHS</th>
<th>HS</th>
<th>CGMA (Zhou et al., 2016c)</th>
<th>EFPA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test problem #8</td>
<td>ks_8a</td>
<td>Mean 3,912,735.88</td>
<td>3,924,400</td>
<td>3,908,301.36</td>
<td>3,924,400</td>
<td>3,924,400</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Best 3,924,400</td>
<td>3,924,400</td>
<td>3,924,400</td>
<td>3,924,400</td>
<td>3,924,400</td>
</tr>
<tr>
<td>Test problem #9</td>
<td>ks_12a</td>
<td>Mean 5,652,561.58</td>
<td>5,688,347.48</td>
<td>5,664,773.26</td>
<td>5,688,887</td>
<td>5,688,887</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Best 5,688,887</td>
<td>5,688,887</td>
<td>5,688,887</td>
<td>5,688,887</td>
<td>5,688,887</td>
</tr>
<tr>
<td>Test problem #10</td>
<td>ks_16a</td>
<td>Mean 7,817,542.42</td>
<td>7,850,983</td>
<td>7,877,247.40</td>
<td>7,850,983</td>
<td>7,850,983</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Best 7,850,983</td>
<td>7,850,983</td>
<td>7,850,983</td>
<td>7,850,983</td>
<td>7,850,983</td>
</tr>
<tr>
<td>Test problem #11</td>
<td>ks_20a</td>
<td>Mean 10,662,848.02</td>
<td>10,718,164.62</td>
<td>10,574,213.28</td>
<td>10,727,049</td>
<td>10,727,049</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Best 10,717,569</td>
<td>10,727,049</td>
<td>10,696,244</td>
<td>10,727,049</td>
<td>10,727,049</td>
</tr>
<tr>
<td>Test problem #12</td>
<td>ks_24a</td>
<td>Mean 13,491,781.28</td>
<td>13,489,892.84</td>
<td>13,313,050.40</td>
<td>13,538,097.18</td>
<td>13,546,441.04</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Best 13,549,094</td>
<td>13,549,094</td>
<td>13,514,028</td>
<td>13,549,094</td>
<td>13,549,094</td>
</tr>
</tbody>
</table>

Figure 3  The convergence curves of algorithms for P10
(see online version for colours)

Figure 4  The ANOVA test of algorithms for P10
(see online version for colours)

Figure 5  The convergence curves of algorithms for P11
(see online version for colours)

Figure 6  The ANOVA test of algorithms for P11
(see online version for colours)
5 Conclusions

This paper presented an elite opposition-flower pollination algorithm for solving knapsack problems. Several problems have been used to prove the effectiveness of the presented method. EFPA algorithm is superior to standard FPA in terms of both efficiency and success rate. This implies that EFPA is potentially more powerful in solving NP-hard problems. Table 2 shows the results of EFPA algorithm are privileged compared with the results of other algorithms. It can be concluded that EFPA is an effective and stable algorithm for solving 0-1 knapsack problems. The convergence of the algorithm is very strong, and EFPA can get the optimal solution in less iterations and time.

In the future, we will apply the elite opposition-flower pollination algorithm to various problems found in the real world. Meanwhile, we are interested in extending the method so that it can deal with multi-objective optimisation problems.

References


An elite opposition-flower pollination algorithm for a 0-1 knapsack problem


