A Hybrid Genetic Scatter Search Algorithm for Solving Optimization Problems

Saber M. El-Sayed  
Faculty of Computers and Informatics  
Operations Research Department  
Zagazig, EL-Sharkia, Egypt.  
saber_is@yahoo.com

Wael F. Abd EL-Wahed  
Faculty of Computers and Informatics  
Professor of Operations Research  
Decision Support  
Shiben EL-Kom, Menoufia, Egypt.  
waelf@yahoo.com

Nabil A. Ismail  
Faculty of Computers and Informatics  
Professor of Computer Science  
Shiben EL-Kom, Menoufia, Egypt.  
nabil_is@hotmail.com

Abstract

The search methods that have been developed to investigate the optimum point of any optimization problem are suffering from some shortcomings especially that are complex and NP-hard problems. The progress that have been achieved in the filed of information technology leads to developing a new search methods called "intelligent search methods". This paper highlights hybrid genetic scatter search algorithm that replace two steps in scatter search (combination and improvement) with two steps in genetic (crossover and mutation). This algorithm leads to increases the efficiency and exploration of the solution process. The algorithm has been coded in C# and tested by solving a set of knapsack problems.

Index Terms — Scatter search, genetic algorithms, combinatorial optimization, meta-heuristics, knapsack problem.

1. Introduction

Meta-heuristic is a set of concepts that can be used to define heuristic methods that can be applied to a wide set of different problems. i.e., a metaheuristic can be considered as a general algorithmic framework which is applied to different optimization problems with relatively few modifications to make them adapted to a specific problem. Scatter search (SS) is one of these methods. SS is an evolutionary method that exploits the knowledge of the problem for creating new better solutions, from the combination of the existing ones [7]. Another heuristic search method is genetic algorithms (GA) are adaptive heuristic search algorithm premised on the evolutionary ideas of natural selection and genetic [22]. The basic concept of GAs is designed to simulate processes in natural system necessary for evolution, specifically those that follow the principles first laid down by Charles Darwin of survival of the fittest [21]. The algorithm is tested with large instance of knapsack problems.

Due to the practical importance of CO optimization many algorithms to tackle them have been developed. Combinatorial optimization algorithms solve instances of problems that are believed to be hard in general (NP-hard problems). Combinatorial optimization algorithms achieve this by reducing the effective size of the space, and by exploring the space efficiently. These algorithms can be classified as either complete algorithm that is guaranteed to find for every finite size instance of a CO problem an optimal solution in bounded time, or, approximate we sacrifice the guarantee of finding optimal solutions for the sake of getting good solutions in a significantly reduced amount of time [1].

There are many difficulties of combinatorial optimization problems: Computationally difficult, Solution space is large, difficult in modeling, Large computation time, Are typically not well handled by cutting plane and branch and bound, Technical and modeling expertise needed. According to these problems, a new kind of approximate algorithm has emerged which basically tries to combine heuristic methods in higher framework, these methods called "Meta-heuristics".

This paper developed a hybrid genetic scatter search algorithm that combines both scatter search and genetic algorithm in anew one. So, this algorithm leads to increases the efficiency and exploration of the solution process. This algorithm is coded into C#.

The paper is organized such that the next section provides a brief overview of combinatorial optimization problems and meat-heuristics methods. Section 3 describes the basics of the SS metaheuristic. Section 4 describes the basics of genetic algorithms (GA). Section 5 discusses the knapsack problem. The hybrid GSS algorithm is analyzed in section 6. An example is
solved in section 7 GSS algorithm. Computational results and conclusions are given in sections 8 and 9 respectively.

2. Combinatorial Optimization

Combinatorial Optimization is the process of finding one or more of best (optimal) solutions in a well-defined discrete problem space [11]. Such problems occur in almost all fields of management (e.g. finance, marketing, production, scheduling, inventory control, facility location and layout, data-base management), as well as in many engineering disciplines.

Combinatorial Optimization problem \( P = (S, f) \) can be defined by:

- A set of variables \( x = \{x_1, \ldots, x_n\} \).
- Variable domains \( D_1, \ldots, D_n \).
- Constraints among variables.
- An objective function \( f \) to be minimized.
- (Maximize \( f \) is the same as minimize \( -f \)).

Note, \( S \) is usually called a search (or solution) space [1].

Metaheuristic is a set of concepts that can be used to define heuristic methods that can be applied to a wide set of different problems. In other words, a metaheuristic can be seen as a general algorithmic framework which can be applied to different optimization problems with relatively few modifications to make them adapted to a specific problem.”

There are different ways to classify and describe meta-heuristic algorithms: Nature-inspired vs. non-nature inspired, population-based vs. single point search, dynamic vs. static objective function, one vs. various neighborhood structures, memory usage vs. memory-less methods [1]. There are many Heuristic search methods such as Local Search, Simulated Annealing, Greedy Randomized Adaptive Search Procedure (GRASP), Swarm Intelligence, Tabu Search, Genetic Algorithms, Ant Colony Optimization, and Scatter Search. This paper highlights the hybrid genetic scatter search meta-heuristic.

3. Scatter Search Meta-Heuristic

According to Fred Glover, Scatter Search is a population-based metaheuristic that uses a reference set to combine its solutions and construct others. The method generates a reference set from a population of solutions. Then a subset is selected from this reference set. The selected solutions are combined to get starting solutions to run an improvement procedure. The result of the improvement can motivate the updating of the reference set and even the updating of the population of solutions. From the standpoint of meta-heuristic classification, SS may be viewed as an evolutionary algorithm (EA) because it builds, maintains and evolves a set of solutions throughout the search.

The advantages of scatter search are demonstrated as follows: first, it solves a diverse array of optimization problems from both classical and real world settings. Second, scatter search contrasts with other evolutionary procedures, such as genetic algorithms, by providing unifying principles for joining solutions based on generalized path constructions in Euclidean space and by utilizing strategic designs where other approaches tend to randomization. Third, additional advantages are provided by intensification and diversification mechanisms that exploit adaptive memory drawing on foundations that link scatter search to tabu search which is a metaheuristic that guides a local search heuristic procedure to explore the solution space beyond local optimality. The main difference between intensification and diversification is that during an intensification stage the search focuses on examining neighbors of elite solutions. The diversification stage on the other hand encourages the search process to examine unvisited regions and generate solutions that are different in various significant ways from those seen before.”

Scatter search differs from genetic algorithms in that; the first use heuristic rule in generation of the initial population but genetic generates it randomly, also, scatter search updates the population using deterministic rules which combines diversity and quality, on the other hand, genetic algorithms update population using the principle of the “survival of the fittest” [5].

There are many application of scatter search such as vehicle routing, quadratic assignment, job shop scheduling, optimization simulation, Non-linear optimization, graph coloring, integer programming, and unconstrained optimization.

The process of solving a problem by means SS is organized in the following five components [8]:

3.1 Diversification method.
3.2. Reference set update method.
3.3 Subset generation method.
3.4 Combination method.
3.5 Improvement method.

The combination method and the improvement method are two of the main components of SS that decisively affect the trade-off between the exploration and the exploitation maintained by this search.
1.4 DSS - procedure [21]. The meta-heuristic strategy includes the decision on how to update the reference set having into account the state of the search. The algorithm must also realize when the reference set does not change and seek to diversify the search by generating a new set of solutions for the population. In addition, the metaheuristic involves the stopping criterion for the whole search procedure. Then the best solution used in the reference set is provided by the method.

Begin
Repeat
Create Population;
Repeat
Generate Reference Set;
Repeat
Select Subset;
Combine Solutions;
Improve Solution;
Until (StoppingCriterion1);
Update Reference Set (RefSet);
Until (StoppingCriterion2);
Until (StoppingCriterion3);
End;

Figure 1. Pseudo-code of sequential scatter search

4. Genetic Algorithms

According Goldberg, genetic algorithms are search algorithms based on the mechanics of natural selection and natural genetics”.

One advantage of GA is efficient means of investigating large combinatorial problems, can solve combinatorial problems, optimizes with continuous or discrete variables, can jump out of a local minimum, may encode the variables so that the optimization is done with the encoded variables, Deals with a large number of variables [25]. Disadvantages are: Computationally expensive, Blind, undirected search, Can be sensitive to initial parameters.

The process of solving a problem by means of GA is organized in the following steps [21] [22] [24]:
1. [Start] Generate random population of n chromosomes (suitable solutions for the problem)
2. [Fitness] Evaluate the fitness f(x) of each chromosome x in the population
3. [New population] Create a new population by repeating following steps until the new population is complete
4. [Replace] Use new generated population for a further run of algorithm
5. [Test] if the end condition is satisfied, stops, and returns the best solution in current population

Figure 2. Flow chart of basic genetic algorithm iteration

5. Knapsack Problem

The knapsack problem is a problem in Combinatorial Optimization. It is a classic application problem of integer programming. This problem involves filling a knapsack with a given capacity (volume or weight) by selecting certain items from a given number of available articles. Usually, the available quantity of each item is limited to one unit and each item has certain characteristics in terms of weight, volume, and relative benefits (utilities). The objective of the problem is to determine the optimal combination of items that should be packed in the knapsack to maximize the total benefit, keeping to a minimum the weight of this set. That is, the profit-to-weight ratio or profit density $\delta_i$, can be used as a measure of the utility of any given item [Cotta and Troya, 1998, Pisinger, 1995]. Many techniques have been developed to solve 0-1 knapsack problems.
**Definition**

In the following, we have:
- \( n \) kinds of items
- \( c \) is a \( 1 \times n \) vector of the items value (integer).
- \( w \) is a \( 1 \times n \) vector of the items weight (integer).
- \( x \) is a \( 1 \times n \) vector of the decision variables.
- \( W \) is the knapsack capacity (integer).

Mathematically, the 0-1 single-objective knapsack problem can be formulated as:

**Max**

\[
Z = \sum_{j=1}^{n} c_{j} x_{j}
\]  
(1)

**Subject to**

\[
\sum_{j=1}^{n} w_{j} x_{j} \leq W
\]  
(2)

\( x_{j} = 0 \text{ or } 1 \) for \( j = 1, 2 \ldots n \)  
(3)

6. The Hybrid Genetic Scatter Search Algorithm

The hybrid genetic scatter search replaces two steps in SS with crossover and mutation respectively. GSS starts with the initialization of start points (population) which are generated. These start points must satisfy the diversification property. The second step is to update the reference set. The third step is to create subsets from the reference set. The fourth step is for each subset make the crossover process. The fifth step is to mutate solutions obtained from crossover process. In the last, test if the reference is stable then stop else re-update the reference set and repeat the previous steps. Note that the reference set contains 20 solutions. Figure 3 summarizes these steps.

Begin
Repeat
Create Population;
Repeat
Generate Reference Set;
Repeat
Select Subset;
Crossover;
Mutation;
Until (StoppingCriterion1);
Update Reference Set (RefSet);
Until (StoppingCriterion2);
Until (StoppingCriterion3);
End;

**Figure 3. Pseudo-code of hybrid genetic scatter search**

7. Solving Knapsack Problem by the Hybrid Genetic Scatter Search.

Let us solve the following example by the proposed algorithm:

Let us solve the following example by the proposed algorithm:

\[
Z = 4X_{1} + 7X_{2} + 3X_{3} + 5X_{4} + 4X_{5}
\]

**St.**

\[
5X_{1} + 3X_{2} + 2X_{3} + 2X_{4} + 1X_{5} \leq 8
\]

\( X_{j} = 0 \text{ or } 1 \) for \( j = 1, 2 \ldots 5 \)

The first step is to initiate start points which are generated according to items priority (c/w).

\[
\begin{align*}
0 & : 01001 \\
0 & : 00001 \\
0 & : 11111 \\
0 & : 01111 \\
0 & : 10111 \\
0 & : 00011 \\
0 & : 00101 \\
0 & : 00000 \\
\end{align*}
\]

**Start Population**

\[0 1 0 0 1 2 0 0 0 0 1 1 1 1 1 0 1 1 1 0 0 0 1 0 0 0 0 0\]

**Figure 4. Start population points.**

Now update the reference set, this done by taking the solution with the largest profit and put it as the first one, then the remaining solutions are ordered according to non decreasing values of the dissimilarity formula (4)

\[
d_{X}^{ref} = \left\| \frac{ref}{X} - \frac{ref}{X} \right\| \leq \left\| \sum_{j=1}^{n} \frac{ref}{X} - x_{j} \right\|, k=1, \ldots , n
\]  
(4)

, where \( x_{j} \) is the solution with the highest value in \( z \) [5]. The reference set obtained is shown in table 1.

**Table 1. Reference Set Update**

<table>
<thead>
<tr>
<th>Z</th>
<th>Decision Space</th>
<th>Dissimilarity</th>
<th>After Ordering</th>
</tr>
</thead>
<tbody>
<tr>
<td>23</td>
<td>( X_{1} ) : 11111</td>
<td>0</td>
<td>( X_{1} ) : 11111</td>
</tr>
<tr>
<td>4</td>
<td>( X_{2} ) : 00001</td>
<td>4</td>
<td>( X_{4} ) : 01111</td>
</tr>
<tr>
<td>11</td>
<td>( X_{3} ) : 01001</td>
<td>3</td>
<td>( X_{5} ) : 01011</td>
</tr>
<tr>
<td>19</td>
<td>( X_{4} ) : 01111</td>
<td>1</td>
<td>( X_{3} ) : 01001</td>
</tr>
<tr>
<td>16</td>
<td>( X_{5} ) : 01011</td>
<td>2</td>
<td>( X_{4} ) : 00011</td>
</tr>
<tr>
<td>9</td>
<td>( X_{6} ) : 00011</td>
<td>3</td>
<td>( X_{7} ) : 00010</td>
</tr>
<tr>
<td>5</td>
<td>( X_{7} ) : 00001</td>
<td>4</td>
<td>( X_{2} ) : 00001</td>
</tr>
</tbody>
</table>
Now let's create subsets of the existing reference.

\[
\begin{align*}
X^1 &: 11111 \\
X^2 &: 00001 \\
X^3 &: 01001 \\
X^4 &: 01111 \\
X^5 &: 01011 \\
X^6 &: 00011 \\
X^7 &: 00010 \\
X^8 &: 00000
\end{align*}
\]

There are seven pairs of the solutions to be combined, each subset contains two solutions, for each subset crossover the solutions with each other. The crossover rate here is 60%. For example the offsprings of the first subset:

- Offspring 1: 11101
- Offspring 2: 00011

Now, mutate each offspring obtained from the previous step. The mutation rate used was 1%. Re-update the reference set with formula (4); if the reference is still stable stop else do the previous steps. The final reference is:

<table>
<thead>
<tr>
<th>$Z$</th>
<th>Decision Space</th>
</tr>
</thead>
<tbody>
<tr>
<td>19</td>
<td>01111</td>
</tr>
<tr>
<td>16</td>
<td>01011</td>
</tr>
<tr>
<td>15</td>
<td>01110</td>
</tr>
<tr>
<td>14</td>
<td>01101</td>
</tr>
<tr>
<td>13</td>
<td>10011</td>
</tr>
<tr>
<td>12</td>
<td>00111</td>
</tr>
<tr>
<td>11</td>
<td>01001</td>
</tr>
<tr>
<td>10</td>
<td>01100</td>
</tr>
<tr>
<td>9</td>
<td>00011</td>
</tr>
<tr>
<td>8</td>
<td>10001</td>
</tr>
<tr>
<td>7</td>
<td>01000</td>
</tr>
<tr>
<td>5</td>
<td>00010</td>
</tr>
<tr>
<td>4</td>
<td>00001</td>
</tr>
<tr>
<td>3</td>
<td>00100</td>
</tr>
<tr>
<td>0</td>
<td>00000</td>
</tr>
</tbody>
</table>

The final best solution is $z = 19$, $x1 = 0$; $x2 = x3 = x4 = x5 = 1$.

6. Computational Results

The algorithm has been coded in C#, and has been tested with different instances of knapsack problem. The computational experiments have been performed with a pentium 4 processor with 384 MB RAM and 80GB hard disk.

<table>
<thead>
<tr>
<th>Size</th>
<th>Hybrid GSS</th>
<th>SS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Time</td>
<td>Z</td>
</tr>
<tr>
<td>5</td>
<td>46.125</td>
<td>19</td>
</tr>
<tr>
<td>8</td>
<td>125</td>
<td>280</td>
</tr>
<tr>
<td>10</td>
<td>140.625</td>
<td>443</td>
</tr>
<tr>
<td>15</td>
<td>187.5</td>
<td>569</td>
</tr>
<tr>
<td>20</td>
<td>453.125</td>
<td>576</td>
</tr>
<tr>
<td>30</td>
<td>1359.375</td>
<td>460</td>
</tr>
<tr>
<td>50</td>
<td>3750</td>
<td>965</td>
</tr>
<tr>
<td>100</td>
<td>34015.625</td>
<td>4266</td>
</tr>
<tr>
<td>250</td>
<td>234734.375</td>
<td>10242</td>
</tr>
</tbody>
</table>

Figure 5. Problems size versus time in milliseconds in the hybrid algorithm.

Figure 6. Problem size versus time in milliseconds in Sequential scatter search.
7. Conclusion

Because of the shortcomings and complexity of the combinatorial optimization problems, a hybrid genetic scatter search algorithm has been developed by combining scatter search with genetic algorithm in order to increase the diversification of solutions as much as possible. It appears from the results that the proposed algorithm performs consistently better than the existing methods. Future work will be required to apply the proposed algorithm for solving multi-objective problems and to overcome the drawbacks of time by developing parallel hybrid genetic scatter search algorithm.

References