Zuzana Čičková
Ivan Brezina
Juraj Pekár

Alternative Method for Solving Traveling Salesman Problem by Evolutionary Algorithm

Article Info:

Management Information Systems, Vol. 3 (2008), No. 1, pp. 017-022

Received 12 Januar 2008
Accepted 24 April 2008

UDC 519.863

Summary

This article describes the application of Self Organizing Migrating Algorithm (SOMA) to the well-known optimization problem - Traveling Salesman Problem (TSP). SOMA is a relatively new optimization method that is based on Evolutionary Algorithms that are originally focused on solving non-linear programming problems that contain continuous variables. The TSP has model character in many branches of Operation Research because of its computational complexity; therefore the use of Evolutionary Algorithm requires some special approaches to guarantee feasibility of solutions. In this article two concrete examples of TSP as 8 cities set and 25 cities set are given to demonstrate the practical use of SOMA. Firstly, the penalty approach is applied as a simple way to guarantee feasibility of solution. Then, new approach that works only on feasible solutions is presented.

Key words

Traveling Salesman Problem, Evolutionary Algorithms, Self Organizing Migrating Algorithm

Introduction

The Traveling Salesman Problem (TSP) is well known in optimization. A traveling salesman has number of *n* cities to visit. A tour (the sequence in which the salesman visits different cities) should be such that every city on the list is visited once and only once, except that salesman returns to the city from which he starts. The goal is to find a tour that minimizes the cost (usually the total distance), among all the tours that satisfy this criterion. The problem can be visualized on graph. Each city becomes a node. Edge lengths correspond to the distance between the attached cities (we assume complete weighted graph). Then, TSP can be formulated as finding a Hamiltonian cycle with the minimal length.

TSP is one of the most discussed problems in literature. Many algorithms were applied with more or less success. There are various ways to classify algorithms, each with its own merits. One way to classify algorithms is by implementation principle (Zelinka, 2002).

- **Explicit enumeration**. It leads to reconnaissance all possible solutions of problems, therefore is applicable only for small problem size.
- **Deterministic methods**. These algorithms base only on rigorous methods of „classical" mathematics. Some additional information, such as gradient, convexity etc. is usually needed (Branch and Bound Algorithm, Cutting Plane Method, Dynamic Programming etc.).
- **Stochastic methods**. Those algorithms work on probabilistic methods to solve problems. Stochastic algorithms work slowly and are usually applicable only for „guessing"(Monte Carlo, Random search Walk, Evolutionary Computation etc.).
- **Combined methods**. Combined methods are comprised by stochastic and deterministic composition. Various metaheuristics algorithm has been devised (Ant Colony Optimization, Memetic Algorithms, Genetic algorithms etc.). Metaheuristics consist of general search procedures whose principles allow them to escape local optimality using heuristics design. Evolutionary algorithms are significant part of metaheuristics.

1. Evolutionary Algorithms

Evolutionary Algorithms (EA) are relatively new optimization techniques that use mechanisms inspired by biological evolution, such as reproduction, mutation, recombination and natural selection. EA indicates a subset of evolutionary computation, which is a part of artificial intelligence. EA differ from more traditional optimization techniques in that they involve a search from a "population" of solutions, not from a single one. Each iteration of EA involves a competitive selection that carried out poor solutions. The solutions with high "fitness" are "recombined" with other solutions by swapping parts of a solution with another. Solutions are also "mutated" by making a small change to a single element of the solution. Recombination and mutation are used to generate new solutions that are biased towards regions of the search space for which good solutions have already been seen. Different main schools of EA evolved during the last 30 years: Genetic algorithms (GA), Evolutionary Programming (EP), Evolutionary Strategy (ES), Differential Evolution (DE), Ant Colony Optimization (ACO), Immunology System Method (ISM), Scatter Search (SS), Particle Swarm (PS), Self Organizing Migrating Algorithm (SOMA) etc.

EA work well on solving unconstrained problems, but several specific methods have been proposed to solve constrained problems. Both main objectives of constrained optimization using evolutionary techniques are to bring the individuals into feasible domain or exploring efficiently the feasible domain to find a solution as close as possible to optimum. Several methods have been proposed to handle such problems. They have been classified by Michalewicz and Schoenauer (1996) into four categories:

1. Methods based on penalties functions which penalize unfeasible solutions
2. Methods which make a clear distinction between feasible and unfeasible solutions
3. Methods using special reproduction operators to preserve feasibility of solutions
4. Hybrid methods

In this article penalty approach will be used as way to guarantee feasibility of solutions (1) and then simply approach that works only on feasible solution (3) will be proposed.

2. Principles of Soma

Self Organizing Migrating Algorithm (SOMA) was created in 1999 by Zelinka (2002). It can be classified as an evolutionary algorithm, despite the fact that no new individuals are created during the computation, and only the position of individuals in the search place is changed. SOMA is based on the self–organizing behavior of groups of individuals in a “social environment”. Such a behavior can be observed anywhere in the world (e.g. a group of animals looking for the food). Even through SOMA is not based on the philosophy of evolution, the final result, after one migration loop, is equivalent to the result from one generation derived by EA algorithms. SOMA was tested on test functions e.g. Onwubolu and Babu (2004), Zelinka (2002) that are:

- none–fractal type
- defined at real, integer or discrete argument space
- constrained, multiobjective, nonlinear etc.

In each case of the function tested, SOMA was searching for the global minimum of the given function. This algorithm was also used on various examples usually from the engineering domain (active compensation in RF–driven plasmas, neural network learning, inverse fractal problem, static optimization of chemical reactor etc.) e.g. Onwubolu and Babu (2004), Zelinka (2002).

SOMA, as well other EA algorithms, is working on a population of individuals, $i = 1,2,...,np$ (*np*– number of individuals in the population). A population can be viewed as a $np \times (d+1)$ matrix, where the columns represent individuals. Each individual represents one candidate solution for the given problem, i.e. a set of arguments of objective function, $j = 1,2,...,d$. Associated with each individual is also the fitness $f_c(\mathbf{x}_j)$, $j = 1,2,...,d$ which represents the relevant value of objective function. The fitness does not take part in the evolutionary process itself, but only guides the search.

Table 1 Population

	$f_c(\mathbf{x}_i)$	1	2	....	d
$\mathbf{x}_1$	$f_c(\mathbf{x}_1)$	x_{11}	x_{12}		x_{1d}
$\mathbf{x}_2$	$f_c(\mathbf{x}_2)$	x_{21}	x_{22}		x_{2d}
. . .					. . .
$\mathbf{x}_{np}$	$f_c(\mathbf{x}_{np})$	x_{np1}	x_{np2}	....	x_{npd}

A population $P^{(0)}$ is usually randomly initialized at the beginning of evolutionary process used the Specimen, which is defined for each parameter.

$$\text{Specimen} = \{\{\text{type},\{\text{Lo,Hi}\}\}_1, \{\text{type},\{\text{Lo,Hi}\}\}_2 ..., \{\text{type},\{\text{Lo,Hi}\}\}_d\} \quad (1)$$

where

- type (integer, real, discrete etc.)
- Lo – lower border
- Hi – upper border

The borders define the allowed range of values for each parameter of individuals at the beginning and also during migration loops. The initial population $P^{(0)}$ is generated as follows:

$$P^{(0)} = x_{i,j}^{(0)} = rnd\left(x_{i,j}^{(Hi)} - x_{i,j}^{(Lo)}\right) + x_{i,j}^{(Lo)}$$
$$i = 1,2,...,np \quad j = 1,2,...,d \qquad (2)$$

If during the migration loop some parameters of individual exceed specimen's borders, that parameters are changed according to the rule:

$$x_{i,j}^{t+1} = \begin{cases} rnd\left(x_{i,j}^{(Hi)} - x_{i,j}^{(Lo)}\right) + x_{i,j}^{(Lo)}, \text{ if } x_{i,j}^{t+1} < x_{i,j}^{(Lo)} \text{ or } x_{i,j}^{t+1} > x_{i,j}^{(Hi)} \\ x_{i,j}^{t+1}, \text{ otherwise} \end{cases} \qquad (3)$$

SOMA, as other EA algorithms, is controlled by a special set of parameter. Recommended values for the parameters are usually derived empirically from a great number of experiments e.g. Onwubolu and Babu (2004), Zelinka (2002):

- *d* - dimensionality. Number of arguments of objective function.
- *np* – population size. It depends of user and his hardware.
- *m* – migrations. Represent the maximum number of iteration.
- *mass* – path length, $mass \in \langle 1,1;3 \rangle$. Represents how far an individual stops behind the leader.
- *step* – $step \in \langle 0,11; mass \rangle$. Defines the granularity with what the search space is sampled.
- *prt* – perturbation, $prt \in \langle 0,1 \rangle$. Determines whether an individual travel directly towards the leader or not.

SOMA was inspired by the competitive–cooperative behavior of intelligent creatures solving a common problem. SOMA works in migration loops. Basic principle is shown in Figure 1.

Each individual is evaluated by cost function and the individual with the highest fitness – *Leader* **Figure 1** is chosen for the current migration loop. According to the *step*, other individuals begin to jump towards the Leader according the rule:

$$x_{i,j}^{mk+1} = x_{i,j,start}^{mk} + (x_{L,j}^{mk} - x_{i,j,start}^{mk})tprt_j$$
$$t \in \langle 0, \text{by } step \text{ to } mass \rangle \qquad (4)$$

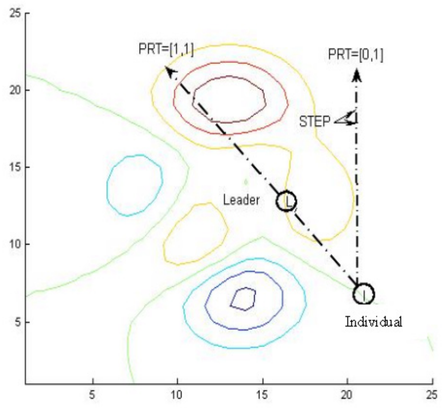

Figure 1 The principle of individual motion

Each individual is evaluated after each jump by objective function. The jumping continues, until new position defined by the *mass* is reached. Then the individual returns to that position, where the best fitness was found:

$$x_{i,j}^{mk+1} = \min\left\{f_c(x_{i,j}^{mk}), f_c(x_{i,j,start}^{mk})\right\} \qquad (5)$$

Before individual begins to jump, a random number for each individual component is generated and is compared with *prt*. If the random number is larger then *prt*, then the associated component of the individual is set to 0. Hence, the individual moves in *d–k* dimensional subspace. This fact establishes a higher robustness of the algorithm. Vector **prt** is created before the individual begin to move in the search space.

$$\text{prt}_j = \begin{cases} 1, \text{ ak } rand_j \langle 0,1 \rangle > prt \\ 0, \text{ otherwise} \end{cases} \qquad (6)$$

This general convention is known as *AllToOne* strategy. In literature Onwubolu and Babu (2004), Zelinka (2002) can be found different working strategies of SOMA. All versions are fully comparable with each other in the sense of finding optimum of objective function.

In its canonical form, SOMA is only capable of handling continuous variables. Two methods have been proposed to handle integer variables by Zelinka (2002):

1) Parameter of individual is converted to an integer value in population

2) Integer variables are used only for evaluation of cost function and in population continuous variables are preserved. This is essential for maintaining the diversity of the population and the robustness of the algorithm.

Formally 2):

$$f_c\left(y_j\right) \quad j = 1,2,\ldots,d \tag{7}$$

where

$$y_j = \begin{cases} x_j, \text{ if } x_j \in \mathrm{Z} \\ \\ \mathrm{INT}\left(x_j\right), \text{ if } x_j \in \mathrm{R} \end{cases} \tag{8}$$

where INT( ) is a function for converting a real value to an integer value by truncation.

For solving constrained problems by penalty approach two ways can be used e.g. Onwubolu and Babu (2004):

1. Soft constraint (penalty) approach. The constraint function introduces a distance measure from the feasible region and the search space remains continuous.
2. Hard constraint approach. Unfeasible solutions are rejected and it often leads to splitting the search space into many separated parts of feasible solution. For this reason this approach is not considered essential.

3. Soma for Solving TSP

TSP is a discrete optimization problem. By solving a natural representation of individual that is known from genetic algorithm is used. Using this representation, the cities are listed in the order in which they are visited. Each city is assigned with a different integer value 1 to *n* that represent sequence of visited cities. Then, number of parameters of individual is $d = n$. The initial population $P^{(0)}$ is generated as follows:

$$\mathrm{P}^{(0)} = x_{i,j}^{(0)} = \mathrm{INT}\left(\mathrm{rnd}\left(x_{i,j}^{(Hi)} - x_{i,j}^{(Lo)}\right) + x_{i,j}^{(Lo)}\right) \tag{9}$$

where

INT( ) a function for converting a real value to an integer value

$x_{i,j}^{(Hi)}$ represents upper border, in case of natural representation $x_{i,j}^{(Hi)} = d$

$x_{i,j}^{(Lo)}$ represents lower border, in case of natural representation $x_{i,j}^{(Lo)} = 1$

Fitness represents the cost of corresponding tour. Firstly, example of 8 cities set (Slovak region cities) is given. The penalty approach is used as a simple way to guarantee feasibility of solutions (In case of duplicity, the penalty constant, which is greater as longest distance between couple of cities, is added as many times as duplicity appears). The values of *d* is fixed according to problem size to 8, parameters *np* = 80 and *mig* = 300, *mass* = 3 are used during the simulation. To choose the other parameters efficiently, 168 simulations were carried out and some statistical methods (ANOVA, Kruskal – Wallis etc.) were applied. The best parameter values: *prt* = 0,8; *step* = 0,9 (Results of 8 simulations gives Table 2.)

Table 2 Final results of simulations (penalty approach)

	1	2	3	4	5	6	7	8	Mean	Variance
Tour length	848	848	857	848	848	857	857	848	851,375	21,67

The tour length 848 was obtained also by system GAMS (model tsp42 from model library).

In order to use penalty approach for solving 25 cities set the parameters *np* = 150 and *mig* = 3000 was used. The complexity is increased, because of the penalty approach works also on infeasible domain. An example of infeasible individual formation gives Figure 2.

For that reasons, new approach that works only on feasible solutions was proposed. At the beginning, a population of individuals is randomly initialized as follows:

$$\mathrm{P}^{(0)} = x_{i,j}^{(0)} = \mathrm{randperm}\left(d\right)$$
$$i = 1,2,\ldots,np \quad j = 1,2,\ldots,d \tag{10}$$

This function is assigning each individual with a random permutation vector of integers size *d* (random permutation of cities index on the salesman route). If during the SOMA's migration loop some parameters of individual exceed specimen's borders, only a valid part of individual is preserved, and that part is completed to a permutation size *d*, following idea of validity of each individual. The procedure:

1. let **m** be a vector of parameters of individual size *d* with k different elements. If *d-k* = 0, go to step 4), otherwise 2)
2. **p** is a *d-k* size vector of rand permutation of *d-k* elements that don't contain in vector **m,** if number of nonzero elements of vector **p** = 0, go to step 4), otherwise 3)
3. m_c is the first repetitive element of vector **m,** p_k is the firth nonzero element of vector p, then $m_c = p_k$, let p_k , *k* = 0 and back to 2)
4. return **m**

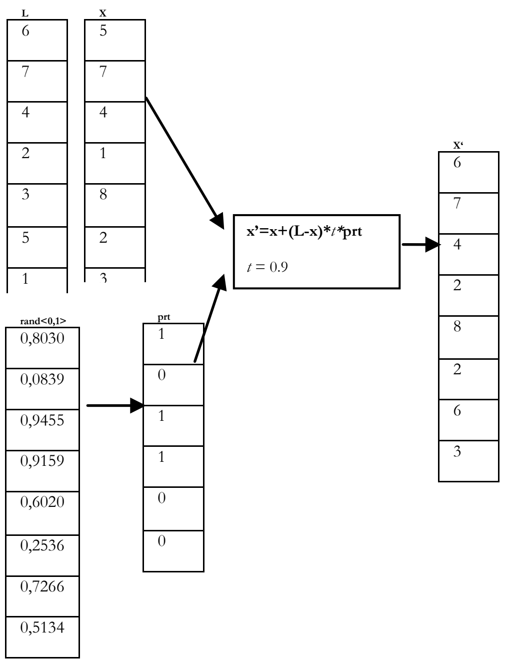

Figure 2 Infeasible individual formation

An example of feasible individual formation gives Figure 3.

This approach was tested on 8 cities set problem with number of migration loops *mig* = 50. Results that include number of migration loop in which the tour length 848 was appeared for the first time gives table 3.

With increasing number of migration loops more alternative solutions are a part of final population. In this case, all 16 cyclic permutations are components of final migration loop.

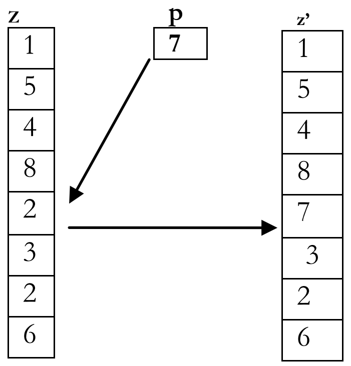

Figure 3 Feasible individual formation

Table 3 Final results of simulations (8 cities set)

	1	2	3	4	5	6	7	8
Tour length	848	848	848	848	848	848	848	848
mig	6	2	7	10	11	7	7	7

For solving problem of 25 cities set, parameters *np* = 150, *mig* = 2000, *prt* = 0,8, *step* = 0,9, *mass* =2 are used. The results are compared with optimal solution obtained by GAMS (14118 km tour length). A result gives table 4.

Table 4 Final results of simulations (25 cities set)

SOMA	Tour length	%dev.opt.
1	15077	6,79%
2	14760	4,54%
3	15077	6,79%
4	15311	8,45%
5	15311	8.45%
Mean	15107	7,00%

Convergence of cost function to final result that depends on number of migration loops in case (2) from Table 4 is shown in Figure 5.

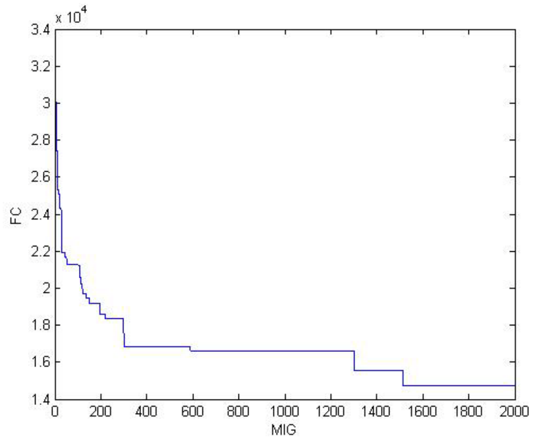

Figure 4 Convergence of cost function

The results of this article show that presented approach is completely used for small size problems, for problems of larger size SOMA gives relatively good solution. Problems in solving large instance problems are similar in appearance to other algorithms. In spite of it, SOMA is usable for larger size problems. Perhaps, other special variants of SOMA for solving TSP could lead to increasing efficiency of computation and this research is open for future work.

4. Conclusion

The Traveling salesman problem is a famous problem in the field of optimization not only for its practical relevance, but also for the practical application. Many algorithms were devised to solve that problem, but TSP belongs to NP-hard problems, so no algorithm has been known to solve this problem in the polynomial time. The most used exact algorithms give exact solution but due to the computational complexity are applied only for solving the relatively small problems. The alternative is the use of evolutionary algorithm, which can give after finite number of iteration „effective" solution. This article presents the application of Self Organizing Migrating Algorithm and two examples of TSP (8 cities, 25 cities) are given to demonstrate the practical use of SOMA.

References

Brezina, I. (2003). Kvantitatívne metódy v logistike. Bratislava: EKONÓM.

Čičková, Z. (2005). Aplikácia evolučných algoritmov na riešenie úlohy obchodného cestujúceho, Dizertačná práca. Bratislava: KOVE FHI.

Haupt, L. R. (2004). Haupt, S. E.: Practical Genetic algorithm. New Jersey: John Wiley & Sons.

Michalewicz, Z. (1996). Genetic Algorithms + Data Structures = Evolution Programs. USA: Springer – Verlag Berlin Heidelberg.

Michalewicz, Z., & Schoenauer, M. (1996). Evolutionary Algorithms for constrained parameter optimization problems. Evolutionary Computation , 4 (1), 1-32.

Onwubolu, G. C., & Babu, B. (2004). New Optimization Techniques in Engineering (Studies in Fuzziness and Soft Computing). News Previews , 710.

Zelinka, I. (2002). Umělá intelligence v problémech globální optimalizace. Praha: BEN-technická literatúra.

Zuzana Čičková

Department of Operations Research and Econometrics,
Faculty of Economic Informatics
Dolnozemská cesta 1/b
852 35 Bratislava
Slovak Republic

Email: zuzana.cickova@euba.sk

Ivan Brezina

Department of Operations Research and Econometrics,
Faculty of Economic Informatics University of Economics in Bratislava
Dolnozemská cesta 1/b
852 35 Bratislava
Slovak Republic

Email: ivan.brezina@euba.sk

Juraj Pekár

Department of Operations Research and Econometrics,
Faculty of Economic Informatics University of Economics in Bratislava
Dolnozemská cesta 1/b
852 35 Bratislava
Slovak Republic

Email: juraj.pekar@euba.sk