Hybrid Self Organising Migrating Algorithm – Scatter Search for the Task of Capacitated Vehicle Routing Problem

Donald Davendra[a], Ivan Zelinka[a], Roman Senkerik[b], Roman Jasek[b] and Magdalena Bialic-Davendra[b]

[b]*Department of Computer Science, Faculty of Electrical Engineering and Computer Science, VŠB - Technical University of Ostrava, 17. Listopadu 15, Ostrava-Poruba, Czech republic.*
donald.davendra@vsb.cz

[b]*Tomas Bata University in Zlin, Faculty of Applied Informatics, nám. T. G. Masaryka 5555, 760 01 Zlín, Czech republic.*

Abstract. One of the new emerging application strategies for optimization is the hybridization of existing meta-heuristics. The research combines the unique paradigms of solution space sampling of SOMA and memory retention capabilities of Scatter Search for the task of capacitated vehicle routing problem. The new hybrid heuristic is tested on the Taillard sets and obtains good results.

Keywords: SOMA, Scatter search, evolutionary heuristics, vehicle routing problem.
PACS: 87.55.de

INTRODUCTION

Metaheuristics are algorithms, which are used for the optimization of complex systems. The vital attribute for these heuristics is that they have to operate without apriori information of the system.

Metaheuristics operate on two ideological frameworks, firstly that a group of solutions provide a better platform to find optimal solution, and secondly that certain guiding concept leads the solutions towards the optimal solution, or nearby regions.

A number of different transformation concepts have evolved within the scope of Metaheuristics [1].

Ant Colony (ACO), Genetic Algorithms (GA), Differential Evolution (DE), Particle Swarm Optimization (PSO) and Self Organising Migration Algorithm (SOMA) are some of the most potent heuristics available. Most of these algorithms have mimicked naturally occurring phenomena.

One of the recent advancements is the hybridization of different algorithms in order to merge their unique methodology under one platform.

Two promising algorithms, which have a proven track record but based on different philosophy, are Scatter Search (SS) and SOMA. SS is based on the memory adaptive paradigm whereas SOMA is on a swarm paradigm. This research is centered on the merging of these unique paradigms in a single framework of a hybrid Discrete SOMA – SS (DSOMA-SS).

The paper is organized as follows. Section 2 introduces SOMA, whereas SS is described in Section 3. The hybrid framework is given in Section 4. Section 5 outlines the vehicle routing problem after which Section 6 presents the results which is analyzed in Section 7.

SELF ORGANISING MIGRATING ALGORITHM

SOMA [2], is based on the competitive-cooperative behavior of intelligent creatures solving a common problem.

In SOMA, individual solutions reside in the optimized model's hyperspace, looking for the best solution. It can be said, that this kind of behavior of intelligent individuals allows SOMA to realize very successful searches.

Because SOMA uses the philosophy of competition and cooperation, the variants of SOMA are called strategies. They differ in the way as to how the individuals affect all others. The best operating strategy is called 'AllToAll' and consists of the following steps:

1. *Definition of parameters*. Before execution, the SOMA parameters (PathLength, Step, PRT, Migrations see Table 1) are defined.
2. *Creating of population*. The population SP is created and subdivided into clusters.
3. *Migration loop*.
   a. Each individual is evaluated by the cost function
   b. For each individual the PRT Vector is created.
   c. All individuals, perform their run towards the randomly selected solution in the opposing cluster according to Equation 2. Each solution is selected from individual cluster piecewise. The movement consists of jumps determined by the Step parameter until the individual reaches the final position given by the PathLength parameter. For each step, the cost function for the actual position is evaluated and the best value is saved. Then, the individual returns to the position, where it found the best-cost value on its trajectory.

The schematic of SOMA with clustered population is given in Figure 1. SOMA, like other evolutionary algorithms, is controlled by a number of parameters, which are predefined. They are presented in Table 1.

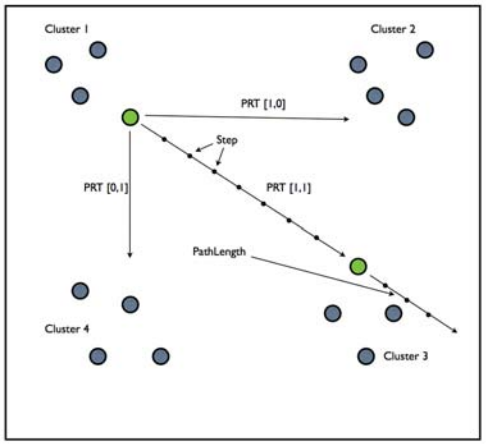

FIGURE 1. SOMA migration utilizing clustered population

TABLE 1. SOMA Parameters

Name	Range	Type
PathLength	(1.1 - 3)	Control
StepSize	(0.11 - PathLength)	Control
PRT	(0 - 1)	Control

Mutation

Mutation, the random perturbation of individuals, is applied differently in SOMA compared with other ES strategies. SOMA uses a parameter called PRT to achieve perturbation. It is defined in the range [0, 1] and is used to create a perturbation vector (PRT Vector) as follows:

$$\begin{aligned} &\text{if } rnd_j < PRTVector_j = 1 \\ &\text{else} \quad 0, j = 1, ..., n_{param} \end{aligned} \tag{1}$$

The novelty of this approach is that the PRT Vector is created before an individual starts its journey over the search space. The PRT Vector defines the final movement of an active individual in search space.

The randomly generated binary perturbation vector controls the allowed dimensions for an individual. If an element of the perturbation vector is set to zero, then the individual is not allowed to change its position in the corresponding dimension.

Crossover

In standard metaheuristics, the Crossover operator usually creates new individuals based on information from the previous generation. Geometrically speaking, new positions are selected from an N dimensional hyper-plane. In SOMA, which is based on the simulation of cooperative behaviour of intelligent beings, sequences of new positions in the N-dimensional hyperplane are generated. The movement of an individual is thus given as follows:

$$\vec{r} = \vec{r}_0 + \vec{m}tP\vec{R}TVector \tag{2}$$

where:

$\vec{r}$: new candidate solution

$\vec{r}_0$: original individual

m: difference between leader and start position of individual

t: $\in$ [0, PathLength]

PRTVector: control vector for perturbation

It can be observed from Equation 2, that the PRT vector causes an individual to move toward the leading individual (the one with the best fitness) in N–k dimensional space. If all N elements of the PRT vector are set to 1, then the search process is carried out in an N dimensional hyperplane (i.e. on a N+1 fitness landscape). If some elements of the PRT vector are set to 0 then the second terms on the right-hand side of Equation (2) equals 0. This means that the number of frozen parameters, *k*, is simply the number of dimensions that are not taking part in the actual search process.

For each individual, once the final placement is obtained, the values are re-converted into integer format through rounding and repairement process.

SCATTER SEARCH

SS is an evolutionary method initially proposed by [3] in the 1960s for combining decision rules and problem constraints. SS encompasses the principles that underlie the surrogate constraint design. It is devised to (1) capture information about the solutions not contained within, (2) use auxiliary heuristics to generate better solutions from the current solutions.

The basic template of SS routines is given as:

1. A Diversification Generator: to generate a diverse population from a trial solution or a seed solution
2. An Improvement Method: to transform a trial solution into an enhanced solution
3. A Reference Set Update Method: to build and maintain a Reference Set consisting of b solutions, where half the solutions are best and the other half diverse.
4. A Subset Generation Method: to operate on the reference set, to produce a subset of its solutions as a basis for creating combined solutions.
5. A Solution Combination Method: to transform a given subset of solutions produced by the above routine into a combined solution.

The diversification generator generates the population from the initial solution, which is supplied as seed or using random generation. The main attribute is that there is replication of solutions within the population; each solution is unique. This is useful since having a diverse population will also have more solutions within the solution space instead of multiple solutions occupying the same space. Using the permutation generator in [3], the population is generated permutatively from the single seed solution.

The improvement routine in SS is an integral part of the heuristic. Each solution generated is improved using either a surrogate heuristic like tabu search (TS) or an improvement routine like 2 – OPT local search.

The reference set or refset is a select group of solutions that are isolated from the main population for manipulation by the SS heuristic. Unlike other evolutionary heuristics, SS does not operate on the complete population, but rather on a select group of solutions. This group of solutions is selected based on two criteria;

1. Intensified Solutions: solutions, which have the "best" objective values in the population.
2. Diversified Solutions: solutions, which have the "farthest" objective functions from the intensified solutions.

These two criteria are collectively known as memory adaptive programming (MAP) [3]. MAP is currently regarded as the critical factor in evolutionary heuristics. The ability to recognize the path and facilitate movement towards the global minima is highly desirable. For the refset build routine in SS, initially the first half is selected on the basis of their objective functions. The other half is selected on the min-max criteria given by [4]. These solutions are the furthest away from the "best" solutions.

The subset generation method is utilized for the combination of the solutions in the refset. [5] outlined a two-point crossover for genetic algorithms. The solution combination method in SS was modified to use this approach. Taking two solutions, two crossover points are isolated as the values in between the regions are swapped with each other. The resulting solution is checked for repetition, and if repetition is detected, a new unique value is generated in its place. This solution is checked against the worst performing solution in the refset, and if there is improvement, then the solution is replaced in refset. This operation iterates throughout the refset solutions, till as specified cutoff limit is reached. If during one iteration, no improvement is seen in the refset, the refset is considered stagnated and taking the best performing solution as seed a new population is generated according to the diversification generator.

HYBRID DISCRETE SELF ORGANISING MIGRATING ALGORITHM – SCATTER SEARCH

Hybrid DSOMA-SS is the hybrid variant of DSOMA, which utilizes the refset structure of SS in order to enhance the population propagation in

DSOMA. This algorithm has been developed to solve permutation based combinatorial optimization problem. The same ideology of the sampling of the space between two individuals is retained. The generic outline is given as:

1. The Population is initialized and the fitness of all the solutions is evaluated.
2. Based on the refset size *b*, half the solutions are selected based on the fitness.
3. The other half of the refset solutions is selected based on the min-max criteria [4].
4. The jump sequence is calculated on the refset.
5. After each migration, the best solution is updated to the refset.

Initialization

The initial population is initialized as a permutative schedule representative of the size of the problem at hand. Each element within the individual is unique.

$$P \supseteq X_i^t \supseteq x_{i,j}^t = \begin{cases} 1 + rand(\ ) \cdot (IS - 1) \\ \text{if } \ x_{i,j}^t \notin \left\{ x_{i,1}^t, .., x_{i,j-1}^t \right\} \end{cases}$$

Reference Set Selection

Based on the fitness evaluation, the best refset/2 solutions are accepted from the population. Based on the min-max criteria from [4], the most diverse solutions are accepted in the refset. The refset is now the operational population.

Jump Sequences

DSOMA operates by calculating the number of discrete jump steps that each individual has to circumnavigate. In
DSOMA, the parameter minimum jumps (J_{min}) is used in lieu of PathLength, which states the minimum number of individuals or sampling between the two individuals. Taking two individuals in the refset, one as the incumbent $\left(X_i^t\right)$and the other as the leader $\left(X_L^t\right)$, the possible number of jump solutions J_{max} is the mode of the difference between the adjacent values of the elements in the individual (3).

$$\begin{aligned} J &= \left\{ \left| x_{i,j}^{t-1} - x_{L,j}^{t-1} \right| \right\} \\ J_{\max} &= \text{mode}\left[J\right] \end{aligned} \tag{3}$$

The step size *(s)* can now be calculated as the integer fraction between the required jumps and possible jumps. The jump matrix **J**, which contains all the possible jump positions, can be calculated as:

$$J_{j,l}^P = \begin{cases} x_{i,j}^{t-1} + sl & \text{if } x_{i,j}^{t-1} + sl < x_{L,j}^{t-1} \\ & \text{and } x_{i,j}^{t-1} < x_{L,j}^{t-1}; \\ x_{i,j}^{t-1} - sl & \text{if } x_{i,j}^{t-1} - sl < x_{L,j}^{t-1} \\ & \text{and } x_{i,j}^{t-1} > x_{L,j}^{t-1}; \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

$$j = 1, ..., IS; \ l = 1, .., J_{\min}$$

New Jump Individual Selection

A total of J_{min} new individuals can now be constructed from the jump positions. Each new individual Y_w^t where $w = 1, ..., J_{\min}$ is constructed piecewise from the jump matrix $\mathbf{J}$. Each element y_w^t in the individual, indexes its values from the corresponding j^{th}array in the jump matrix $\mathbf{J}_{i\ j}^P$ Each l^{th} $\left(l = 1..., J_{\min}\right)$position for a specific $j^{th}\left(j = 1..., IS\right)$element is sequentially checked to ascertain if it already exists in the current individual Y_w^t.If this is a new element, it is then accepted in the individual, and the corresponding l^{th}value in the jump matrix is set to zero$\mathbf{J}_{i\ j}^P = 0$.This iterative procedure can be given as in (5).

$$Y_{w,j}^t = \begin{cases} \mathbf{J}_{j,l}^P \begin{cases} \text{if } \mathbf{J}_{j,l}^P \notin \left\{ y_{w,1}^t, ..., y_{w,j-1}^t \right\} \\ \text{and } \mathbf{J}_{j,l}^P \neq 0; l = 1, .., J_{\min} \\ \text{then } \mathbf{J}_{j,l}^P = 0 \end{cases} \\ 0 \qquad \text{otherwise} \end{cases} \tag{5}$$

$$w = 1, .., J_{\min}$$

RefSet Update

After each individual is evaluated for its fitness value as in (6).

$$C_w^t = f\left(Y_w^t\right); w = 1,..,J_{\min} \tag{6}$$

2-OPT local search is applied to the best individual obtained with the minimum fitness value. After the local search routine, the new individual is compared with the fitness of the incumbent individual, and if it improves on the fitness, then the new individual is accepted in the population (7).

$$X_i^t = \begin{cases} Y_{best}^t & \text{if } f\left(Y_{best}^t\right) < C_i^{t-1} \\ X_i^{t-1} & \text{otherwise} \end{cases} \tag{7}$$

Iteration

Sequentially, each individual X_{i+1}^{t-1} is selected from the refset, and it begins its own sampling towards the designated leader X_L^{t-1}. It should be noted that the leader does not change during the evaluation of one migration.

Repairement Procedure

The repairment process [6] is given in a number of routines. The first routine is to check the entire solution for repeated values. These repeated values and their positions are isolated in a replicated array $x_{repl} = \left\{x_j, x_{j+n}, ..., x\right\}$. The second routine is to find which values are missing from the solutions given as $x_{\text{mis}} = \left\{1,..,n\right\} \cap \left\{x_1, x_2, ..., x_n\right\}$.

Since, the replicated array contains a number of sequences of replicated solutions, randomly one solution in each sequence is labeled as feasible and repatriated back into the main solution. This leaves the replicated array containing only infeasible values.

Randomly each value is selected from the missing array and inserted in the position of a replicated value in the replicated array $x_{\text{mis}} \xrightarrow{random} x_{repl}$.

Finally, the replicated array is reinserted in the solution array with all values now feasible $x_{repl} \rightarrow x$.

Iteration

When all the individuals have completed their run towards the leader, the refset is evaluated for its viability. If during the last migration, no new solution was obtained, a new population is generated with the best solution from the current refset included, and the refset genereated from the new population.

If a new solution was obtained during the migration, the migration continues with the current refset, till the termination criteria is reached.

CAPACITATED VEHICLE ROUTING PROBLEM

The Vehicle Routing Problem (VRP) introduced for the first time by [7] is a complex combinatorial optimization problem, which can be seen as a merge of two well-known problems: the Traveling Salesperson Problem (TSP) and the Bin Packing Problem (BPP).

It can simply be described as follows: given a fleet of vehicles with uniform capacity, a common depot, and several costumer demands, find the set of routes with overall minimum route cost which service all the demands.

Assume a quantity d_i of a single commodity which is to be delivered to each customer $i \in N = \left\{1,...,n\right\}$ from a central depot {0} using *k* independent delivery vehicles of identical capacity *C*. Delivery is to be accomplished at minimum total cost, with $c_{i,j} \geq 0$ denoting the transit cost from *i* to *j*, for $0 \leq i, j \leq n$. The cost structure is assumed symmetric, i.e., $c_{i,j} = c_{j,i}$ and $c_{j,i} = 0$.

Combinatorially, a solution for this problem consists of a partition of *N* into *k* routes $\left\{R_1,..,R_k\right\}$, each satisfying $\sum_{j \in R_i} d_j \leq C$, and a corresponding permutation σ_i of each route specifying the service ordering. This problem is naturally associated with the complete undirected graph consisting of nodes $N \cup \left\{0\right\}$, edges E, and edge-traversal costs $c_{i,j} \left\{i, j\right\} \in E$. In this graph, a solution is the union of *k* cycles whose only intersection is the depot node. Each cycle corresponds to the route serviced by one of the *k* vehicles. By associating a binary variable with each edge in the graph, the following integer programming formulation is obtained:

$$\min \sum_{e \in E} c_e x_e$$

$$\sum_{e=\{0,j\} \in E} x_e = 2k \tag{8}$$

$$\sum_{e=\{0,j\} \in E} x_e = 2 \forall i \in N \tag{9}$$

$$\sum_{\substack{e=\{0,j\}\in E,\\ i\in S, j\notin S}} x_e = 2b(S) \forall S \subset N, |S| \tag{10}$$

$$0 \le x_e \le 1 \forall e = \{i,j\} \in E, i,j \ne 0 \tag{11}$$

$$0 \le x_e \le 1 \forall e = \{i,j\} \in E \tag{12}$$

$$x_e \text{ integral } \forall e \in E \tag{13}$$

For ease of computation, $b(S) = \left\lceil \frac{\left(\sum_{i\in S} d_i\right)}{C} \right\rceil$ is defined as an obvious lower *C* bound on the number of trucks needed to service the customers in set S. Constraints 8 and 9 are the degree constraints. Constraints 10 is a generalization of the subtour elimination constraints from the TSP and serves to enforce the connectivity of the solution, as well as to ensure that no route has total demand exceeding the capacity *C*. A (possibly) stronger inequality may be obtained by computing the solution to a Bin Packing Problem (BPP) with the customer demands in set S being packed into bins of size *C*. Equation 10 is the capacity constraints.

It is clear from the description that the VRP is closely related to two difficult combinatorial problems. By setting C = ∞, the Multiple Traveling Salesman Problem (MTSP) is obtained. An MTSP instance can be transformed into an equivalent TSP instance by adjoining to the graph *k − 1* additional copies of node *0* and its incident edges (there are no edges among the *k* depot nodes). On the other hand, the question of whether there exists a feasible solution for a given instance of the VRP is an instance of the BPP. The decision version of this problem is conceptually equivalent to a VRP model in which all edge costs are taken to be zero (so that all feasible solutions have the same cost). Hence, the first transformation can be seen as the relaxing the underlying packing (BPP) structure and the second transformation as relaxing the underlying routing (TSP) structure. A feasible solution to the full problem is a TSP tour (in the expanded graph) that also satisfies the packing constraints (i.e., that the total demand along each of the *k* segments joining successive copies of the depot does not exceed *C*).

Because of the interplay between the two underlying models, instances of the Vehicle Routing Problem can be extremely difficult to solve in practice. In fact, the largest solvable instances of the VRP are two orders of magnitude smaller than those of the TSP. Exact solution of the VRP thus presents an interesting challenge.

RESULTS

A total of 12 problems of the Taillard sets have been experimented. Three different sets exist of four instances of size 75, 100 and 150. The results of HDSOMA-SS is given in Tables 3 - 5. The bolded values are the best results for that particular instance. The average and standard deviation values are also provided. The *t-test* values for the different data sets are given in Table 6.

The operating parameters of DSOMA-SS are given in Table 2.

TABLE 2. DSOMA-SS operating parameters

Parameter	Value
Strategy	All-to-One
Population	100
Refset size	20
Migration	50

TABLE 3. VRP 75 tour result

Instant	n	Optimal	DSOMA	Hybrid
tai75a	75	1618.36	0.928	**0.621**
tai75b	75	1344.62	0.754	**0.532**
tai75c	75	1291.01	1.181	**0.721**
tai75d	75	1365.24	0.950	**0.652**
Average			1.064	**0.631**
StdDev			0.122	**0.067**

TABLE 3. VRP 100 tour result

Instant	n	Optimal	DSOMA	Hybrid
tai100a	100	2401.34	1.144	0.976
tai100b	100	1940.61	1.467	1.143
tai100c	100	1406.2	1.414	0.932
tai100d	100	1581.25	1.459	1.092
Average			1.371	1.035
StdDev			0.152	0.085

TABLE 3. VRP 150 tour result

Instant	n	Optimal	DSOMA	Hybrid
tai150a	150	3055.23	1.772	1.432
tai150b	150	2656.47	2.217	1.843
tai150c	150	2341.84	1.962	1.532
tai150d	150	2645.39	1.743	1.342
Average			1.924	1.537
StdDev			0.218	0.188

TABLE 3. VRP 75 tour result

Instant	*t*-value	*p*-value	significant	algorithm
tai50	6.4517	0.0076	yes	Hybrid
tai100	5.1606	0.0141	yes	Hybrid
tai150	20.123	0.0003	yes	Hybrid

ANALYSIS AND CONCLUSION

The comparison of the new hybrid algorithm was done with the previous experiment done by DSOMA. In each of the experimentations, the new hybrid algorithm improved on the previous algorithm.

The t-test results conducted on the two algorithms validate the claim that HDSOMA-SS is a valid and significant improvement on DSOMA.

Further improvements will include the tuning of this hybrid algorithm and more application in other logistical problem domains.

ACKNOWLEDGMENTS

This work was supported by the Grant Agency of the Czech Republic GACR 102/09/1680 and by the European Regional Development Fund under the project CEBIA-Tech No. CZ.1.05/2.1.00/03.0089.

REFERENCES

1. J. Dreo, A. Petrowski, P. Siarry, and E. Taillard, *Metaheuristics for Hard Optimization*. Germany: Springer-Verlag, 2006.
2. I. Zelinka, "SOMA - self organizing migrating algorithm", edited by G. Onwubolu and B. Babu. *New Optimization Techniques in Engineering*, Germany: Springer-Verlag, 2004.
3. F. Glover, "Scatter Search and Path Relinking", edited by D. Corne, M. Dorigo and F. Glover. *New Ideas in Optimization*. Wiley, 1999.
4. V. Campos,M. Laguna and R. Marti. *Scatter Search for the Linear Ordering Problem*. Technical Report. University of Valencia. 2001, TR03-2001
5. D. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Publishing Company, Inc. 1989
6. G. Onwubolu and D. Davendra, *Differential Evolution: A Handbook for Global Permutation-based Combinatorial Optimization*, Springer-Verlag, Germany, 2009.
7. G. Dantzigand R. Ramser. "The truck dispatching problem". *Management Science*, **6**, 80–91, 1959.