CLUSTERED SELF ORGANISING MIGRATING ALGORITHM FOR THE QUADRATIC ASSIGNMENT PROBLEM

Donald Davendra, Ivan Zelinka and Roman Senkerik
Department of Applied Informatics
Tomas Bata University in Zlin
Nad Stranemi 4511, Zlin 76001, Czech Republic
E-mail: {davendra,zelinka,senkerik}@fai.utb.cz

Abstract

An approach of population dynamics and clustering for permutative problems is presented in this paper. Diversity indicators are created from solution ordering and its mapping is shown as an advantage for population control in metaheuristics. Self Organising Migrating Algorithm (SOMA) is modified using this approach and vetted with the Quadratic Assignment Problem (QAP). Extensive experimentation is conducted on benchmark problems in this area.

Key words: SOMA, QAP, optimization, evolutionary heuristics

1. 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, Genetic Algorithms, Differential Evolution, Particle Swarm Optimization and SOMA are some of the most potent heuristics available. Most of these algorithms have mimicked naturally occurring phenomena.
The principle concept of the population is to provide a pool of solutions from which the optimal solution evolves from, however during the subsequent transformation, the solutions cluster together in nearby neighbourhoods. This leads to the loss of diversity of the solutions, and later generations are usually mutating within a cloned gene pool. This phenomenon is generally termed as stagnation and the two most current correcting tools are local search within the hierarchical population and forfeiting of some better-placed solutions in lieu of worst ones [2].
Stagnation is a major concern for evolutionary heuristics, since evolution is principally based on diversity of the existing population. The preconception that extended generations of a heuristic will lead to better solutions is nonviable for a stagnated population.
This paper is devoted to the concept of the population and its controlling dynamics. It is shown through experiment that population control is an effective tool in sustaining the diversity of the population, which in turn leads to more viable regions of search space for exploration.

2. Initial Population

Random population provides an initial loose mapping of the solution space. For permutative problems, where solution ordering is strict, it is often the case that adjacent values are required. This also holds true for flow shop scheduling problems and vehicle routing problems. A typical approach of using local search heuristics to search in the neighbourhood of the solutions usually yields closely aligned solutions.
The initial population *P,* for this heuristic is partially stochastic and partly deterministic. The population is divided into two sub-populations, *SP's*, one randomly generated (SP_{rand}) and the other structurally generated (SP_{struct}) as given in Fig 1.

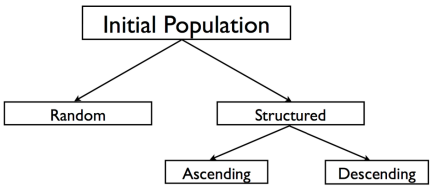

Fig. 1: Initial Population

The formulation for SP_{rand} is fairly simple. A random permutative string is generated for each solution till a specified number given as P_{size}.
The structured population SP_{struct} is somewhat more complex. It is made of two parts. In the first part an initial solution is generated with ascending values given as $x_{ascending} = \{1,2,...,n\}$, where n is the size of the problem. In order to obtain a structured solution, the first solution is segmented and recombined in different orders to produce different combinations. The first segmentation occurs at $n/2$, and the two half's are swapped to produce the second solution. The second fragmentation occurs by the factor 3; $n/3$. Three regions of solutions now exist. The number of possible

CP1159, *Power Control and Optimization,* edited by Nader Barsoum, Abdul Halim Hakim, and Pandian Vasant

recombination's that can exist is 3! = 6. At this point there are nine solutions in the SP_{struct}. The general representation is given as:

$$k \geq 1 + 2! + 3! + \ldots + z! \quad (1)$$

where z is the total number of permutations possible and k is $P_{size}/2$. The second part of SP_{struct} is made from solution in descending order $x_{desending} = \{n,..,2,1\}$. The end result is that two separate populations exist and are used independently by the underlying heuristic.

3. Solution Dynamics

A solution represented as $x = \{x_1, x_2, x_3, \ldots, x_n\}$, where n is the number of variables, within a population has a number of attributes. Usually the most visible is its fitness value, by which it is measured within the population. This approach is not so viable in order to measure the diversity of the solution in the population. In retrespect, a single solution is assigned a number of attributes for measure, as given in Table 1.

Table 1. Solution Parameters

Parameter	Description	Activity
Deviation	Measure of the deviation of the solution	Control
Spread	Alignment of solution	Control
Life	Number of generation	Selection
Offspring	Number of successful offspring's produced	Selection

The most important attribute is the *deviation* (the difference between successive values in a solution). Since we are using only permutative solutions, deviation or *ordering* of the solution is important. Each value in the solution has a unique footprint in the search space. The formulation for deviation is given as:

$$\varsigma = \left(\frac{\sum_{i=1}^{n-1} |x_i - x_{i+1}|}{n} \right) \quad x_i \in \{x_1, x_2, \ldots, x_n\} \quad (2)$$

Spread of a solution gives the alignment of the solution. Each permutative solution has a specific ordering, whether it is *forward* aligned or *reverse* aligned. Whereas deviation measures the distance between adjacent solutions, spread is the measure of the hierarchy of subsequent solutions given as:

$$\partial = \begin{cases} +1 & if\ (x_{i+1} - x_i) \geq 1 \\ -1 & if\ (x_{i+1} - x_i) \leq 1 \end{cases}$$
$$where\ \ i \in \{1,2,\ldots,n\} \quad (3)$$

The generalisation of *spread* is given in Table 2.

Table 2. Spread generalization

Spread	Generalization
> 0	Forward spread
0	Even spread
< 0	Reverse spread

Life is the number of generations the solution has survived in the population and *Offspring* is the number of viable solutions that have been created from that particular solution. These two variables are used for evaluating the competitiveness of different solutions.

3.1 Clustering

Within the population, certain solutions exhibit attracting features. These points are usually local optima regions, which draw the solutions together. The approach utilized, is to subdivide the population in clusters, each cluster a distinct distance from another. Two controlling parameters are now defined which control the clusters.

Cluster Attractor C_A: The distance that each segment of solution has to differ from each other. The C_A is given as:

$$C_A \in [0.1,\ 1+) \quad (4)$$

Within the population indexed by the *deviation,* solutions with similar deviation are clustered together, and each cluster is separated by at least a single C_A

$$\left(\delta_1, \delta_2, .. \delta_{k/5}\right) \overset{C_A}{\leftrightarrow} \left(\delta_{(k/5)+1}, \delta_{(k/5)+2}, \ldots \delta_{2(k/5)}\right)$$
$$\overset{C_A}{\leftrightarrow} \ldots . \overset{C_A}{\leftrightarrow} \left(\delta_{4(k/5)+1}, \delta_{4(k/5)+2}, \ldots, \delta_k\right) \quad (5)$$

An illustrative example is presented to showcase how this approach works. Assume a group of solutions as presented in Table 3.

Table 3. Illustrative example

Solution	Deviation	Spread	Cluster	C_A	Fitness
1 2 3 4 5 6 7 8 9 10	0.9	+9	1	1.2	1592
1 2 5 6 9 10 3 4 7 8	2.1	+7	2	1.1	1559
10 9 8 7 6 5 4 3 2 1	0.9	-9	1	1.2	1567
10 9 6 5 2 1 8 7 4 3	2.1	-7	2	1.1	1547
6 4 8 2 3 9 1 5 7 10	3.7	+3	3	0.4	1765
9 4 3 7 5 2 10 1 6 8	4.1	-1	4	-	1788
8 7 4 1 3 6 5 2 10 9	2.5	-3	2	1.1	1678
2 5 7 9 1 8 10 3 6 4	3.6	+3	3	0.4	1686
6 1 3 9 7 10 5 2 8 4	3.6	-1	3	0.4	1654
5 9 2 6 8 3 1 7 10 4	3.9	+1	4	-	1545
C_E					90.13379

Each solution is evaluated for its deviation and spread as given in columns 2 and 3. Column 4 gives the cluster to which the solution belongs, and column 5 gives the value of C_A by which each cluster is separated from each other. The graphical representation is given in Fig 2.

The second controlling factor is the *Cluster Edge* C_E. Whereas C_A is the mapping of individual solutions, C_E is the measure of the entire population. C_E is the measure of the deviation of the fitness of the population and to prevent the population from stagnating to any fitness minima.

3.2. Selection and Deletion

Selection of the next generation is based on a tier-based system. If the new solution improves on the global minima, it is then accepted in the solution. Otherwise, competing clusters jokey for the new solution. Initially the solution is mapped for its deviation. This deviation is then mapped to the corresponding cluster.
Within the cluster, the placement of the solution is evaluated. If the new solution corresponds to an existing solution, or reduces the threshold C_A value of the cluster, then it is discarded.

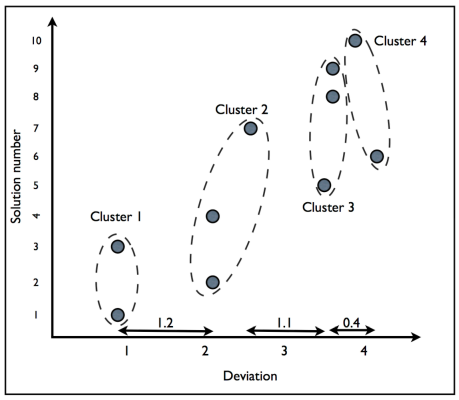

Fig. 2: Graphical representation of population

The solution is accepted if it improves on the C_A value of the cluster (hence improving diversity) and also to balance the C_E. If the cluster has less than average solutions, then the new solution is admitted.
Table 4 gives the selection criteria.

Table 4. Selection criteria

Variables	Criteria
Fitness	Improves clusters best solution
C_A	Increases the value of C_A
C_E	Decreases

Once the solution is added to the cluster, another solution can be discarded. This solution is usually elected from the middle placed solutions in the cluster, whose fitness is not in the top 5% of the population. If no such solutions exist, then the average rated solution is removed. Solution with high *Life* and low *Offspring* are discarded, since they are considered dormant within the cluster.

Table 5. Deletion criteria

Variables	Criteria
Life	High
Offspring	Low
C_A	Decreases

4. Self Organising Migrating Algorithm

SOMA [3], is based on the competitive-cooperative behaviour 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 behaviour 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 6) are defined.
2. *Creating of population*. The population *SP* is created and subdivided into clusters.
3. *Migration loop*.

3.1. Each individual is evaluated by the cost function
3.2. For each individual the PRT Vector is created.
3.3. All individuals, perform their run towards the randomly selected solution in the opposing cluster according to equation (7). 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 Fig 3. SOMA, like other evolutionary algorithms, is controlled by a number of parameters, which are predefined. They are presented in Table 6.

Table 6. SOMA Parameters

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

4.1. 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:

$$if\ rnd_j < PRT\ then\ PRTVector_j = 1$$
$$else\ 0, \quad j = 1,\ldots,n_{param} \tag{6}$$

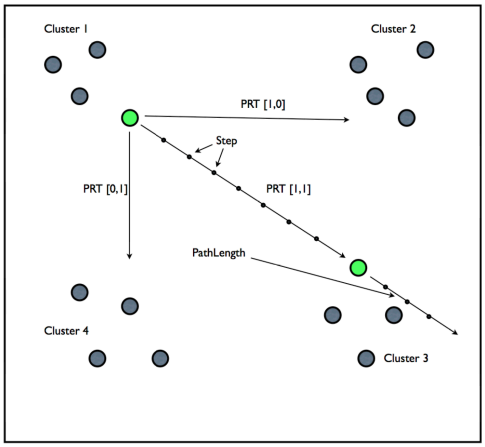

Fig. 3: SOMA migration utilizing clustered population

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.

4.2. 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:

$$\vec{r} = \vec{r}_0 + \vec{m}\, t \vec{PRTVector} \tag{7}$$

where:
$\vec{r}$: new candidate solution
$\vec{r}_0$: original individual
m: difference between leader and start position of individual
t: ∈ [0 , Path length]
PRTVector: control vector for perturbation

It can be observed from Equation (7) 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 (7) equal 0. This means those parameters of an 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.

4.3. Repairment procedure

The repairment process [4] 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 $x_{repl} = \{x_j, x_{j+n}, \ldots, x\}$. The second routine is to find which values are missing from the solutions given as $x_{mis} = \{1,\ldots,n\} \cap \{x_1, x_2, \ldots, x_n\}$.
Since, the replicated array contains a number of sequences of replicated solutions, randomly one solution in each sequence is labelled 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 $x_{mis} \overset{random}{\rightarrow} x_{repl}$.
Finally, the replicated array is reinserted in the solution array with all values now feasible $x_{repl} \rightarrow x$.

5. Dynamic Clustering

The selection and crossover criteria have now been outlined. After each migration, the clusters are reconfigured. Since, in all heuristics, there is a tendency to converge, it is imperative to keep the solutions unique.
The procedure is to calculate the deviation of the new solutions. Since a mesh may exist, it is feasible to reconfigure certain boundary solutions. Fig 4 can be a representation of a SP.
A mutation routine is used to reconfigure a solution. By altering certain positions within the solution, it is possible to realign the deviation and spread of the solution. Boundary values within the solutions (usually represented by the upper and lower bound of the solution) are swapped. Another approach is to have two random positions generated and the values in these positions swapped. An illustration is given to describe this process in Table 7 and Fig 5-6.
Once the boundary values are re-aligned, the second migration loop occurs.

Table 7. Swap of boundary values

Solution	Deviation	Spread

10 9 6 5 2 **1** 8 7 4 3	2.1	-7
1 9 6 5 2 **10** 8 7 4 3	3.0	-5

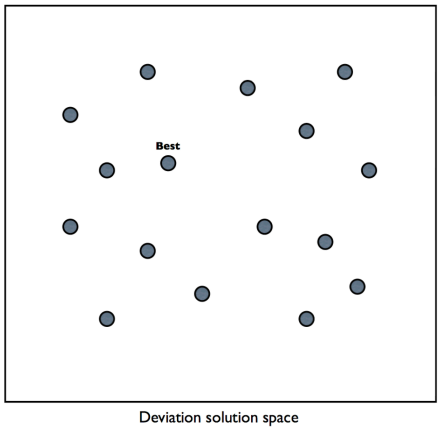

Fig. 4 Solution space after migration

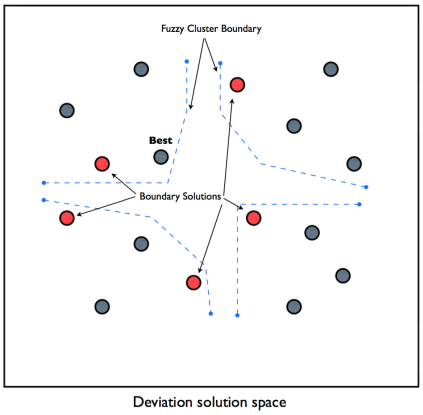

Fig. 5 Fuzzy clustering and boundary solution isolation.

6. General Template

Collating all the piecewise explanation, a general generic template is now described.

1. Generate: Randomly create SP_{rand}, half the size of P_{size}, and then structurally create SP_{struc}. These two form the basis of the population.

2. Calculate the *deviation* and *spread* of each solution in the population. Taking take *deviation* values, configure the population into four clusters. The minimal separation value between the clusters is assigned as C_A. Taking the entire *SP*, the standard deviation of the *fitness* is computed. This is labelled as the C_E.

3. Generation/Migration

   4.1 Taking each *SP* in turn, the selected heuristic of SOMA is applied to the population.

   4.2 The new solution is calculated for its *deviation* and *spread.*

   4.3 Using the selection criteria, the solution is placed within the cluster corresponding to its deviation. If replicated solutions exist, then it is discarded. Selection is based on *fitness* and the move of the C_A and C_E.

5. The *SP* is re-calculated for its cluster boundaries.

6. If the value of C_A has deceased, then the boundary solutions are reconfigured. The C_E value is calculated for the new population.

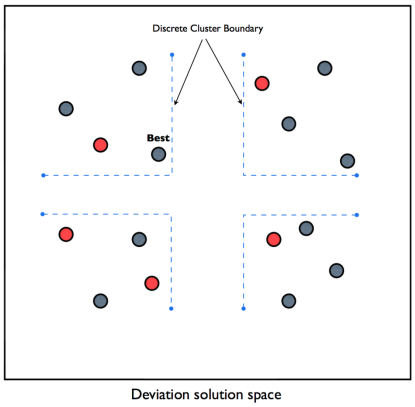

Fig. 6 Realigned solutions into discrete clusters.

The graphical representation is given in Fig 7.

7. Quadratic Assignment Problem

QAP is a *NP*-hard optimization problem [5]. It is considered as one of the hardest optimization problems as general instances of size $n \geq 20$ cannot be solved to optimally [6].

It can be described as follows: Given two matrices

$$A = (a_{ij}) \text{ and } B = (b_{ij})$$

find the permutation π^* minimizing

$$\min_{\pi \in \Pi(n)} f(\pi) = \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} \bullet b_{\pi(i)\pi(j)} \qquad (8)$$

where $\Pi(n)$ is a set of permutations of *n* elements.

The problem instances selected for the QAP are from the OR Library and reported in [7]. There are two separate problem modules; *regular* and *irregular*.

The difference between regular and irregular problems is based on the *flow-dominance (fd).* Irregular problems have a flow-dominance statistics larger than 1.2. Most

of the problems come from practical applications or have been randomly generated with non-uniform laws, imitating the distributions observed on real world problems.

Fd is defined as a coefficient of variation of the flow matrix entries multiplied by 100. That is:

$$fd = 100\sigma / \mu \tag{9}$$

where:

$$\mu = \frac{1}{n^2} \bullet \sum_{i=1}^{n} \sum_{i=1}^{n} b_{ij} \tag{10}$$

1. Input: $n, P_{size}, SP_{struct}, SP_{rand}, C_A \in (0.1,1+), C_E, Gen$

2. Initialize: $SP_{rand} = \begin{cases} \forall i \le P_{size}/2 \wedge \forall j \le n : x_{i,j,G=0} = rand_j[0,1] \bullet \left(x_j^{(hi)} - x_j^{(lo)}\right) \\ i = \left\{1,2,..,P_{size}/2\right\}, j = \{1,2,...,n\}, G = 0, rand_j[0,1] \in [0,1] \end{cases}$

$$SP_{struct} = \begin{cases} k = P_{size}/2 ; z \ni \max\left(\sum_{i=1}^{z}(1+2!+...+z!)\right) \le k; x_{ascend} = \left\{x^{(lo)},..,x^{(hi)}\right\} \\ x_{descend} = \left\{x^{(hi)},..,x^{(lo)}\right\} \\ \forall i \le z, \left(x_{ascend}, x_{descend}\right) \subseteq i \xrightarrow{permutate} append\left(SP_{struct}\right) \end{cases}$$

3. Calculate

$$\begin{cases} \text{Deviation } \delta = \left(\frac{\sum_{j=1}^{n-1}|x_j - x_{j+1}|}{n}\right) : x_j \in \{x_1, x_2, ..., x_n\} \\ \text{Spread } \partial = \begin{cases} +1 \ if\left(x_{j-1} - x_j\right) \ge 1 \\ -1 \ if\left(x_{j-1} - x_j\right) \ge 1 \end{cases} \\ C_A = \left(\delta_1, \delta_2, .., \delta_{k_5}\right) \overset{C_A}{\leftrightarrow} \left(\delta_{(k_5)+1}, \delta_{(k_5)+2}, ..., \delta_{2(k_5)}\right) \overset{C_A}{\leftrightarrow} ... \overset{C_A}{\leftrightarrow} \left(\delta_{4(k_5)+1}, \delta_{4(k_5)+2}, ..., \delta_k\right) \\ C_E = std\left(f(x_i)\right) : x_i \in \{x_1, x_2, ..., x_{P_{size}}\} \end{cases}$$

4. While $G < G_{max}$ for each SP

$\forall i \le P_{size}$:

5. Mutate and recombine:

$$\vec{u}_{i,G+1} \overset{DE}{\leftarrow} \{x_1, x_2, ..., x_i\}$$

5.1 Calculate δ and ∂ of $\vec{u}_{i,G+1}$

6. Select

$$\vec{x}_{i,G+1} = \begin{cases} \vec{u}_{i,G+1} & \text{if } f\left(\vec{u}_{i,G+1}\right) \le f\left(\vec{x}_{best}\right) \\ \vec{u}_{i,G+1} & \text{if } > C_A \\ \vec{x}_{i,G} & \text{otherwise} \end{cases}$$

7. Calculate C_A, C_E

8. Dynamic clustering

$G = G + 1$

Fig. 7 Graphical representation of clustering.

$$\sigma = \sqrt{\frac{1}{n^2} \bullet \sum_{i=1}^{n} \sum_{i=1}^{n} \left(b_{ij} - \mu\right)^2} \tag{11}$$

8. Results

The operating parameters of SOMA are given in Table 8. The results of clustered SOMA applied to the QAP problems as given in Tables 9 and 10. The main comparison is done with SOMA without clustering of [8].

In all problem instances, SOMA with clustering improved on the values obtained for SOMA without clustering.

Table 8. SOMA operating parameters

Parameter	Value
Strategy	All-to-All
Step Size	0.21
PathLength	3
Population	500 - 1000
Migration	500 - 1000

Table 9: $SOMA_{clust}$ Irregular QAP comparison

Instant	fd	n	Optimal	SOMA[8]	$SOMA_{clust}$
bur26a	2.75	26	5246670	**0**	**0**
bur26b	2.75	26	3817852	**0**	**0**
bur26c	2.29	26	5426795	**0**	**0**
bur26d	2.29	26	3821225	**0**	**0**
bur26e	2.55	26	5386879	**0**	**0**
bur26f	2.55	26	3782044	0.03	**0.01**
bur26g	2.84	26	10117172	**0**	**0**
bur26h	2.84	26	7098658	**0**	**0**
chr25a	4.15	26	3796	0.129	**0.10**
els19	5.16	19	17212548	**0**	**0**
kra30a	1.46	30	88900	**0.002**	**0.002**
kra30b	1.46	30	91420	0.03	**0.027**
tai20b	3.24	20	122455319	0.004	**0**
tai25b	3.03	25	344355646	**0**	**0**
tai30b	3.18	30	637117113	0.043	**0**
tai35b	3.05	35	283315445	**0**	**0**
tai40b	3.13	40	637250948	0.02	**0**
tai50b	3.1	50	458821517	**0.2**	**0.2**
tai60b	3.15	60	608215054	0.5	**0.2**
tai80b	3.21	80	818415043	0.8	**0.4**

Table 10: $SOMA_{clust}$ Regular QAP comparison

Instant	fd	n	Optimal	SOMA[8]	$SOMA_{clust}$
nug20	0.99	20	2570	**0**	**0**

nug30	1.09	30	6124	0.02	**0**
sko42	1.06	42	15812	0.01	**0**
sko49	1.07	49	23386	0.005	**0**
sko56	1.09	56	34458	0.01	**0**
sko64	1.07	64	48498	0.06	**0.02**
sko72	1.06	72	66256	0.2	**0.04**
sko81	1.05	81	90998	0.35	**0.05**
tai20a	0.61	20	703482	**0**	**0**
tai25a	0.6	25	1167256	**0**	**0**
tai30a	0.59	30	1818146	0.01	**0**
tai35a	0.58	35	2422002	0.03	**0**
tai40a	0.6	40	3139370	0.623	**0.58**
tai50a	0.6	50	4941410	0.645	**0.42**
tai60a	0.6	60	7208572	0.62	**0.62**
tai80a	0.59	80	13557864	1.05	**0.95**
wil50	0.64	50	48816	**0**	**0**
nug20	0.99	20	2570	**0**	**0**
nug30	1.09	30	6124	0.02	**0**
sko42	1.06	42	15812	0.01	**0**

9. Analysis and Conclusion

Comparison of the obtained results is done with some published heuristics. The first comparison is done with the irregular QAP instances. $SOMA_{clust}$ is compared with the Improved Hybrid Genetic Algorithm (GA_1) of [9] and the highly refereed Ant Colony approach (HAS) of [7] given in Table 11.

The best performing algorithm is $SOMA_{clust}$ which obtains the best comparitive result in 13 out of 20 problem instances. The hybrid Genetic Algorithm appproach however is able to find the optimal result in the two instances that it is applied, where the other heuristics are not so effective.

The second set of comparison is done with the regular QAP instances. Comparison of the clustered SOMA is done with the GA (GA_1) approach of [9], greedy GA (GA_{Greedy}) of [10], GA (GA_2) of [11], Simulated Annealing algorithm (TB2M) of [12], Robust Tabu Search (RTS) of [13], Combined Simulated Annealing and Tabu Search (IASA-TS) of [14] and Ant Colony (HAS) of [7]. The results are given in Table 12.

As with the previous instances, $SOMA_{clust}$ is the best performing heuristic with 10 best solutions, all of which are optimal values of those particular problems.

In terms of population dynamics, consider the initial population clustering of a sample population of "bur26a" instance as given in Fig 8.

The final population clustering is given in Fig 9. The deviation of the solutions is from 1 - 2.75 in the initial population and 4 - 9 in the final population. This shows a drift of the solutions in the deviation space. Another point of interest is that the solutions are still diversified in their sructure. The solutions within the clusters have converged, however the overall diversity is maintained within the population. This opens more oppertunity to obtain better solutions in next generations.

The spread of the solutions in given in Fig 10.

The Cluster Edge. C_E of the population throughout the population generation (in this case, 100 generations) is given in Fig 11. A general decline of the spread of the clusters and fitness values is seen. This is typical for a minimising function. The final graph of the best individual is seen in Fig 12.

Table 11. Irregular QAP comparison

Instant	fd	n	Optimal	GA_1	HAS	$SOMA_{clust}$
bur26a	2.75	26	5246670	-	**0**	**0**
bur26b	2.75	26	3817852	-	**0**	**0**
bur26c	2.29	26	5426795	-	**0**	**0**
bur26d	2.29	26	3821225	-	**0**	**0**
bur26e	2.55	26	5386879	-	**0**	**0**
bur26f	2.55	26	3782044	-	**0**	0.01
bur26g	2.84	26	10117172	-	**0**	**0**
bur26h	2.84	26	7098658	-	**0**	**0**
chr25a	4.15	26	3796	-	3.082	0.10
els19	5.16	19	17212548	-	**0**	**0**
kra30a	1.46	30	88900	**0**	0.629	0.002
kra30b	1.46	30	91420	**0**	0.071	0.027
tai20b	3.24	20	122455319	-	0.091	**0**
tai25b	3.03	25	344355646	-	**0**	**0**
tai30b	3.18	30	637117113	-	**0**	**0**
tai35b	3.0	3	28331544	-	0.02	**0**

tai40b	5 3.1 3	5 4 0	5 63725094 8	-	5 **0**	**0**	tai80b	3.2 1	8 0	81841504 3	-	0.66 7	0.4
tai50b	3.1	5 0	45882151 7	-	0.19 2	0.2							
tai60b	3.1	6	60821505	-	0.04	0.2							

Table 12. Regular QAP comparison

Instant	fd	n	Optimal	GA_1	GA_{Gre}	GA_2	TB2M	RTS	IASA - TS	HAS	$SOMA_{clust}$
nug20	0.99	20	2570	-	-		-		-	**0**	**0**
nug30	1.09	30	6124	**0**	0.07	**0**	0.94	0.73	0.52	0.098	**0**
sko42	1.06	42	15812	**0**	0.250	**0**	0.66	1.03	0.46	0.076	**0**
sko49	1.07	49	23386	0.038	0.210	0.009	0.67	0.54	0.46	0.141	**0**
sko56	1.09	56	34458	**0**	0.02	0.001	0.66	0.53	0.50	0.101	**0**
sko64	1.07	64	48498	**0**	0.22	**0**	0.57	0.93	0.45	0.504	0.02
sko72	1.06	72	66256	0.042	0.29	0.014	0.60	0.52	0.48	0.702	0.04
sko81	1.05	81	90998	0.067	0.2	0.014	0.46	0.41	0.40	0.493	0.05
tai20a	0.61	20	703482	-	-	-	-	-	-	0.675	**0**
tai25a	0.6	25	1167256	-	-	-	-	-	-	1.189	**0**
tai30a	0.59	30	1818146	-	-	-	-	-	-	1.311	**0**
tai35a	0.58	35	2422002	-	-	-	-	-	-	1.762	**0**
tai40a	0.6	40	3139370	-	-	-	-	-	-	1.989	0.58
tai50a	0.6	50	4941410	-	-	-	-	-	-	2.8	0.42
tai60a	0.6	60	7208572	-	-	-	-	-	-	0.313	0.62
tai80a	0.59	80	13557864	-	-	-	-	-	-	1.108	0.95

A direct correlation is seen between the graphs of Cluster Edge and Best Individual. The Edge is a prelude to a shift in solution space. A shift generally signifies a region of new solutions, and possibility of further improvement.

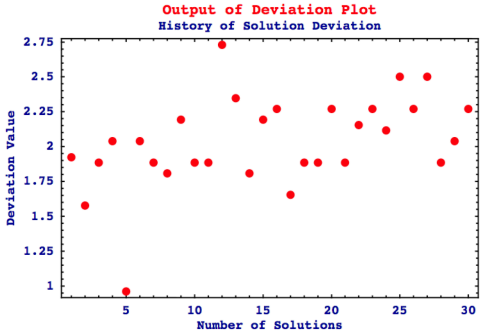

Fig. 8 Initial Population

Acknowledgement

This work was supported by grant No. MSM 7088352101 of the Ministry of Education of the Czech Republic and by grants of the Grant Agency of the Czech Republic GACR 102/09/1680.

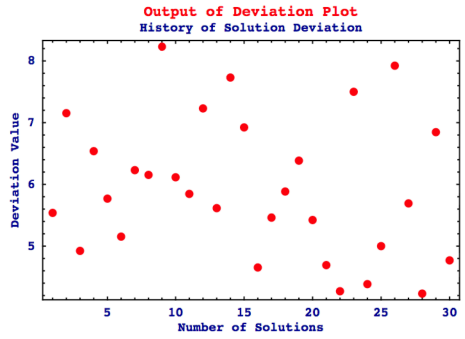

Fig. 9 Final Population

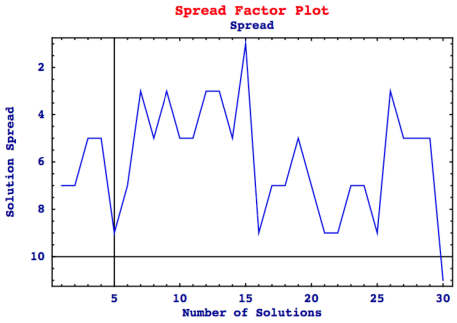

Fig. 10 Spread

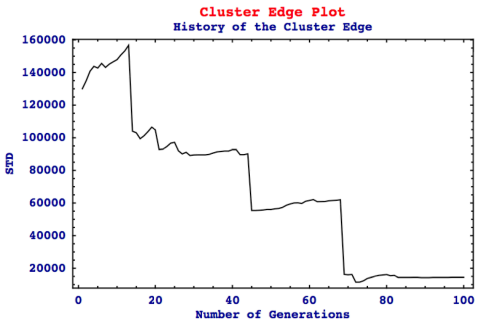

Fig. 11 Cluster Egde plot

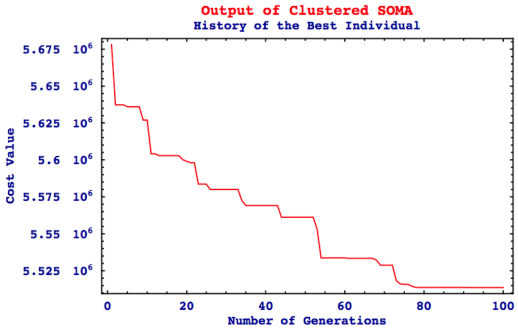

Fig. 12 Best Solution

References

[1] J. Dreo, A. Petrowski, P. Siarry, and E. Taillard, "Metaheuristics for Hard Optimization". Germany: Springer-Verlag, 2006.

[2] G. Onwubolu, "Emerging Optimization Techniques in Production Planning and Control". London, England: Imperial Collage Press, 2002.

[3] I. Zelinka, "Soma self organizing migrating algorithm", in New Optimization Techniques in Engineering, G. Onwubolu and B. Babu, (Eds). Germany: Springer-Verlag, 2004.

[4] G. Onwubolu and D. Davendra, "Differential Evolution: A Handbook for Global Permutation-based Combinatorial Optimization", Springer-Verlag, Germany, 2009.

[5] S. Sahni and T. Gonzalez, "P-complete approximation problems", J ACM 1976, 23:555–565.

[6] M. Dorigo, V. Maniezzo and A. Colorni, "The Ant System: optimisation by a colony of co-operating agents." IEEE Trans Syst Man Cybern B Cybern, 1996, 26(1):29-41.

[7] L. Gambardella, E. Thaillard and M. Dorigo, "Ant Colonies for the Quadratic Assignment Problem", Int J Oper Res, 1999, 50:167–176

[8] D. Davendra and I Zelinka "Optimization of Quadratic Assignment Problem using Self Organising Migrating Algorithm". Journal of Computing and Informatics, Accepted, In press.

[9] P. Ji, W. Yongzhong and L. Haozhao, "A solution method for the Quadratic Assignment Problem (QAP)". Proceeding of the Sixth International Symposium on Operations Research and Its Applications (ISORA'06), August 8-12, 2006, Xinjiang, China 106-117.

[10] R. Ahuja, J. Orlin, A. Tiwari, "A descent geentic algorithm for the quadratic assignment problem". Computers and Operations Research, 2000 27:917-934

[11] Z. Drezne, "A new genetic algorithm for the quadratic assignment problem". INFORMS Journal on Computing, 2003, 115:320-330

[12] A. Boelte and U. Thonemann, "Optimizing simulated annealing schedules with genetic programming". Eur J Oper Res , 1996, 92:402-416

[13] E. Taillard, "Robust taboo search for the quadratic assignment problem", Parallel Comput , 1991, 17: 443–455

[14] A. Misevicius, "An Improved Hybrid Optimization algorithm for the Quadratic Assignment Problem". Mathematical Modelling and Analysis, 2004, 9(2):149-168