ISSN 1750-9653, England, UK
International Journal of Management Science and Engineering Management
Vol. 3 (2008) No. 4, pp. 275-283

A memetic algorithm for symmetric traveling salesman problem

Keivan Ghoseiri[1*], Hassan Sarhadi[2]

[1] School of Railway Engineering, Iran University of Science and Technology, Tehran, Iran
[2] Department of Industrial and System Engineering, Buali Sina University, Hamedan, Iran

(*Received November 13 2007, Accepted February 19 2008*)

Abstract. A Memetic Algorithm (MA) encompasses a class of approaches with proven practical success in variety of optimization problems aiming to make use of benefits of each individual approach. In this paper, we present a special designed Memetic Algorithm to solve the well-known Symmetric Traveling Salesman Problem (STSP).The main feature of the Memetic Algorithm is to use a local search combined with a special designed genetic algorithm to focus on the population of local optima. To check the performance quality of the proposed Memetic Algorithm, some benchmark problems are solved. Experiments on the benchmark set indicate that the Memetic Algorithm is quite efficient and competitive and produces good convergence behaviour and solutions.

Keywords: traveling salesman problem, local search, memetic algorithms, genetic algorithms

1 Introduction

In the Traveling Salesman Problem (TSP), we have a nonnegative integer n and n-dimensional square matrix $C = \{c_{ij}\}$.Any sequence of $p+1$ integers taken from $(1, 2, \ldots, n)$ in which each of n integers appears once and the first and last integers are the same is called a tour if $p = n$ and called subtour if $p < n$. A tour maybe written as $t = (i_1, i_2, \ldots, i_n, i_1)$. In TSP literature, a feasible solution means a tour. An optimal solution is a tour that $Z = \sum c_{ij}$ is minimized where $[(i_1, i_2), (i_2, i_3), \ldots, , (i_{n-1}, i_n), (i_n, i_1)]$. Simply stated, an optimal tour among all possible tours has the minimum length. The usual terminology is that n the integers correspond to cities or nodes, the ordered i, j pairs are links or arcs joining nodes, the c_{ij} is the distance or cost of travel from node i to node j or length of arc and finally the length of tour is summation of the arc i, j lengths included. In addition TSP is symmetric if costs of travel between two cities are identical or $c_{ij} = c_{ji}$ elsewhere it is asymmetric. In graph theory we simply define TSP as finding a minimum Hamiltonian tour in a complete undirected graph with positive costs of travel. Definitely TSP is the most famous and well studied combinatorial optimization problem. Its fame comes from two different origins. The first origin is its broad application domain in variety of science and engineering applications while the second origin is its test bed position for new problem solving strategies and algorithms. Since TSP belongs to class of NP hard problems[11], it has often serves as a touchstone for new problem solving strategies and algorithms. Simplicity to state and define this problem fosters and boosts this important role. So broad range of methods including exact and approximate were tested on TSP. The approximates are methods such as Genetic Algorithms[10, 12], Simulated Annealing[15], Tabu Search[9], Ant colonies[6, 7], Artificial Neural Networks[2] and also some new and unpopular methods like Noising Method[5].

In this paper we present a special designed memetic algorithm to solve STSP. In this memetic algorithm, the main idea is to construct initial solutions by implementing a construction heuristic, then applying local search and genetic algorithm to further improve the quality of solutions. So MA deals with population of

* Corresponding author. Tel: +98-21-77491029; fax: +98-21-77451568. *E-mail address*: ghosiri@iust.ac.ir.

local optima in the entire of algorithm. Reference [10] represents an important background for this paper. This paper is organized as follows: in section 2 we briefly introduce local search concept. Section 3 reviews memetic algorithms and section 4 presents the proposed memetic algorithm for STSP. Section 5 devoted to results and finally in section 6 we will put forward some suggestions to further improve the efficiency and capabilities of the proposed memetic algorithm.

2 Local search

Local search is one of the basic methods used to find approximate solutions for hard combinatorial optimization problems, in particular those known as NP hard. There are numerous combinatorial optimization problems for which computing exact optimal solutions is considered to be intractable. However in practice, we are usually satisfied with good solutions and therefore, approximation algorithms are very important. There are several important tools used to design approximate algorithms. The most common is greedy method. Another important tool is local search which starts from an initial solution x and repeatedly replaces it with a better solution in its neighborhood $N(x)$ until no better solution is found. $N(x)$ is a set of solutions obtainable from x by slight perturbations. The resulting solution x' that cannot be improved further by a solution in $N(x)$ is called local optimum. To simply understand and perceive the role of local search in dealing with NP hard optimization problems, we have to define NP hard optimization problem. Suppose optimization problem $\begin{cases} \min f(x) \\ x \in F \end{cases}$ is defined by introducing a constant k and by asking for a solution $x \in F$ satisfying $F(x) \leq k$. Such a decision problem belongs to NP hard class if enumeration of all solutions like x cannot be done in polynomial time and conditions i.e. $f(x) \leq k$ and can be checked in polynomial time for each x. This NP-hard problem may become tractable if enumeration of all solutions can be replaced by enumeration of good solutions which constitute only small subset of solution space[21].

To design efficient local search algorithms, it is important to carefully define each components of local search. These components are neighborhood, evaluation function, move strategy, and initial solutions. We then briefly explain these components for a typical local search for TSP.

2.1 Neighborhood

In defining neighborhood, it is important to preserve the structure common to good solutions. In TSP, we have following neighborhoods:

- City-swap: the set of solutions obtainable by exchanging the position of two cities in permutation.
- City-insert: the set of solutions obtainable by deleting a city from current position and inserting it into another position.
- Or-opt: the set of solutions obtainable by deleting a subpath of length at most three, and inserting it into another position.
- λ-opt: the set of solutions obtainable by changing at most λ edges of the tour, where $\lambda \neq 2$ is a constant.

The City-insert neighborhood is a subset of Or-opt neighborhood and Or-opt neighborhood is a subset of 3-opt neighborhood. Among these, the λ-opt neighborhood with $\lambda = 2, 3$ and Or-opt neighborhood are known to be effective and City-swap neighborhood is not competitive with others. As the cost of TSP is the summation of lengths of tour edges, the number of different edges between two tours might be the most appropriate distance measure. So, the distance between current tour and subsequent tour in neighborhood is four for City-swap, λ for λ-opt neighborhood and three for Or-opt. this means that the solutions in city swap neighborhood are not close enough to the current solution. Neighborhood size or the number of solutions that can be reached from current solution is also important and the quality of locally optimal solutions usually improves when a larger neighborhood is searched. The size of all mentioned neighborhoods is of order $\bigcirc(N^2)$ when N is the number of cities[14].

2.2 Evaluation function

In traditional local searches for TSP, evaluation function is directly related to objective function. In general, it doesn't need to be directly related to objective function[21].

2.3 Move strategy

The basic move strategy in local search is to choose an improved solution in the neighborhood. In this case, the two strategies called the first admissible move strategy and the best admissible move strategy are usually used. In the first admissible move strategy, solutions in $N(x)$ are scanned according to pre specified order and the first improved solution found is immediately accepted as the next solution. In this strategy, the order of searching the neighborhood may affect the computational time and solution quality. In the best admissible move strategy, the best solution in $N(x)$ is chosen as the next solution[21].

2.4 Initial solutions

Initial solutions in local search may be generated randomly or they may be solutions created by heuristic or even metaheuristics.

2.5 The 2-Opt local search

In 2-opt local search that we will use in our algorithm, each of these components are as follows:
Neighborhood: λ-opt neighborhood with $\lambda = 2$.
Evaluation Function: Evaluation function is the objective function or length of the tour.
Move Strategy: Move Strategy is best admissible move.
Initial Solutions: Initial Solutions are provided with a constructive heuristic for example nearest neighbor.
Fig. 1 demonstrates the 2-opt local search.

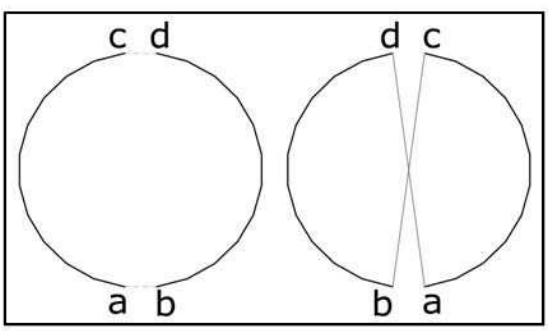

Fig. 1. 2-opt Local Search.

3 Memetic algorithms

The generic denomination of memetic algorithms (MAs) is used to encompass a broad class of metaheuristics. The method is based on a population of agents and proved to be of practical success in variety of problems including NP-hard optimization problems. Unlike traditional evolutionary methods, MAs are concerned with exploiting all available knowledge about problem and this exploitation is fundamental feature that characterizes them. In MAs, heuristics, approximation algorithms, local searches, truncated exact methods are all incorporated. The most crucial and important feature of MAs, exploiting all available knowledge about problem under study, is supported by theoretical results[17]. As "NO FREE LUNCH" Theorem[20], the performance of a search algorithm is closely related to the amount and quality of problem knowledge they are used.

It is not surprising that MAs are also called hybrid evolutionary algorithms or genetic local search, knowledge augmented genetic algorithms. Up to know, albeit unfortunately under different names, MAs have become an important optimization approach with success in variety of NP hard optimization problems.

To give a preliminary idea of MAs, we introduce a pseudo code and describe it in more detail. (Fig. 2 shows the procedure)[16]. Some reserved words in pseudo code take into account many MAs used in practice as well as sub processes that are in generic MAs. In the pseudo code, after initial population has been created, it is locally optimized by Local-Search function which receives as an input an individual and tries to improve it by iteratively using some local search based algorithm. The Local-Search continues to return a better individual until it can no longer improve it or other termination condition has been reached. Then all the individuals are evaluated according to some fitness functions. In the recombination loop, the reserved word *parfor* indicates operations that can be executed in parallel if possible. The *SelectToMerge* function is executed by selecting a subset of individuals i.e. S_{par} to be used as input of recombine function. Some MAs use a random function for selecting input individuals (parents), while others use more complex approaches. The recombine function is responsible for the creation of new individuals which are selected before by *SelectToMerge* function.

```
begin
InitializePopulation Pop using FirstPop();
for each individual i ∈ pop do i= Local-Search (i);
for each individual i ∈ pop do Evaluate(i);
repeat
    parfor j = 1 to #recombinations do
        SelectToMerge a set S_par ⊆ Pop ;
        Offspring= Recombine (S_par , x);
        Offspring: = Local-Search (Offspring);
        Evaluate (Offspring);
        AddInPopulation individual offspring to pop;
    endfor;
    parfor j:=1 to #mutations do
        SelectToMutate an individual i ⊆ Pop
        i_m : = Mutate (i);
        i_m : = Local –Search (i);
        Evaluate (i_m );
        AddInPopulation individual i_m to Pop
    endfor;
    Pop: =SelectPop (Pop);
    If Pop Meets PopConvCriteria then Pop: = RestartPop(Pop);
Until Termination Condition = True;
End;
```

Fig. 2. MA procedure[16]

This recombination can be implemented in a wide variety of ways, ranging from a simple application of traditional recombination operators to more sophisticated and special designed operators. x, as an input to recombination function, stands for necessary information of selected individuals. For example in TSP, x represents the distance matrix between cities. Afterward, the new individual is locally optimized and added to the population according to *AddInPopulation* function. The predefined number of individuals are identified by *SelectToMutate* and mutated. Again, mutation function can be done in a wide variety of ways, ranging from a simple traditional ways to more sophisticated and special designed ones. The *SelectPop* function acts on

the population, having the net effect of reducing its size. The selection of subset to be preserved is not always based on objective function. Other features of individuals maybe considered especially if we want to maximize diversity of the selected set. The convergence of the whole population is checked by *MeetsPopConvCriteria*. These criteria might include the lack of diversity of population. This situation is called diversity crisis. In this case, a *restartPop* function is used and all the processes should be done again from scratch. The termination of MAs like other metaheuristics can be done in many ways. It can be based on time expiration or generation expiration as well as more adaptive procedures, like dynamic monitoring of lack of improvement. This pseudo code and its explanation are for standard MA. Again, there are many special designed MAs that differ from this standard MA.

4 The proposed memetic algorithm

Improving the relatively poor performance of genetic algorithms on TSP has been one of the major dilemmas of researchers. To improve its performance, several approaches are available. The first and definitely the most important approach is to use special designed genetic operators (recombination and mutation) that properly benefited from special features of search space[13, 19]. The second approach is to use a construction heuristic to generate initial population of genetic algorithm rather than simply using random heuristic to generate initial population. The third approach is to use improvement heuristics or local searches to locally optimize solutions. All this approaches are incorporated in proposed memetic algorithm. Fig. 3 sketches the proposed MA.

```
Procedure Memetic Algorithm
 Begin
  Initialize population P with Nearest Neighbor heuristic (NN)
  For each individual belonging to P do
       Individual: =2-opt (individual)
  End;
  Repeat
       For i: = 0 to (recombination rate)* P do
            Select two parents Parent1, Parent2 belonging to P randomly;
            Offspring: = DPX (Parent1, Parent2);
            Offspring: = 2-opt (Offspring);
            With predefined probability (mutation rate) do
                 Offspring: = Mutation (Offspring);
            End;
       End;
            Replace (Offspring)
  Until Converged
 End;
```

Fig. 3. Proposed MA.

At the beginning, nearest neighbor construction heuristic[14] is used to generate initial population of solutions. It starts at an arbitrary city and successively adds the closest unvisited city to the last city until the tour is completed and all the cities were included. Then 2-opt local search is applied on the initial population of solutions and make them locally optimized (minimized). Now this is the turn of genetic algorithm in MA. Genetic algorithm begins with random selection of two solutions as parents in population of solutions. According to[10] other ways of selection that are biased to fitness of solutions will lead to premature convergence and arising similar individuals. Two selected parents then merged and offspring is generated via recombination operator. The design of recombination operator is based on an important feature of TSP search space. According to Boes observation[4], if we define distance between two tours as the number of uncommon edges

that they have, fitness and distance to optimum are correlated for most TSP instances. The average distance between the local optima is similar to the distance to the optimum. Thus the global optimum appears to be more or less central among the local optima. Boes called this feature "big valley", since local optima are closer together if they are closer to optimum and the smaller the tour length, the closer they are to the optimum. The recombination operator that preserves the distance between offspring and each of its parents is called distance preserving recombination or DPX[10]. Fig. 4 sketches the recombination operator and Fig. 5 shows an example of applying DPX on two parents.

```
Procedure DPX (Parent1, Parent2)
Begin
   Offspring: = Parent1;
   Delete all edges in Offspring that are not contained in Parent2;
   Greedy – reconnection (Offspring, d);
End
```

Fig. 4. DPX procedure.

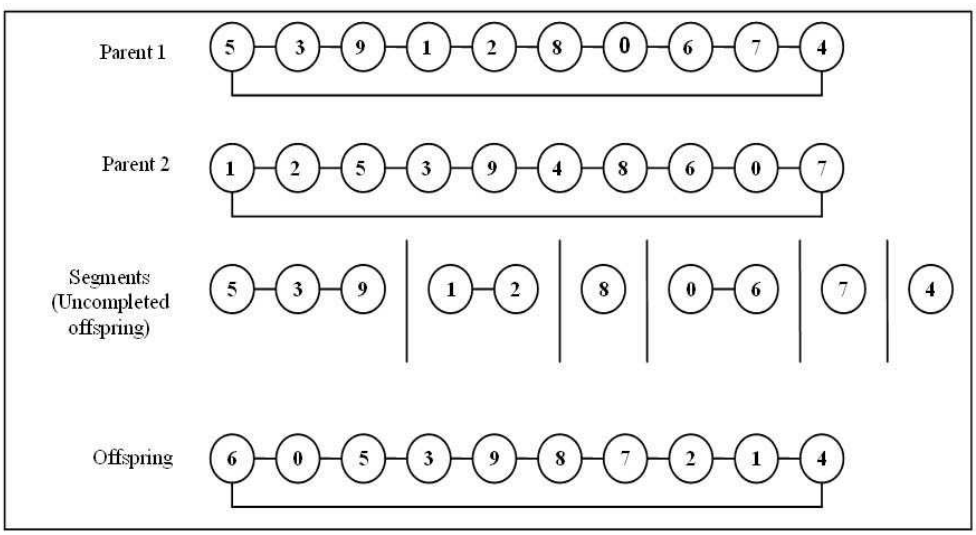

Fig. 5. An Example of DPX.

If we have parent 1 and parent 2 as shown in Fig. 5, at first all the contents of one parent; let us say parent 1; are copied in offspring. Then all the uncommon edges of two parents are deleted. So we have some segments that must be reconnected. At reconnection phase, we connect a given city to all cities so that the distance between parents preserves between each of parents and offspring. For example we cannot connect city 9 to city 4 in reconnection phase, because edge (9, 4) is in parent 2 and not in parent 1 and this reconnection makes the distance between parents and their offspring unequal. So we can add such edges that are not included in both two parents. Now another question arises. Given that we can connect a city to variety of cities that their reconnection preserves the distance, which one of cities must be selected to connect to mentioned city? For example, we can connect city 9 to cities like 2, 5, 6, 7, 8 and 0. To choose among these cities, we lean on greedy method and this is why we need distance matrix d and call this reconnection as "greedy reconnection". By referring to distance matrix, we can determine which city is closest to city 9 and must be chosen to connect to city 9. Mutation is the performed with small probability known as mutation rate. The main purpose of mutation is to slightly modify offspring and maintain the diversity of population. Mutation procedure is done by Non-sequential 3-change. Then again mutated offspring brings to local optimality by 2-opt. Non-sequential

3-change is chosen because it is smallest non sequential change that cannot be neutralized by 2-opt local optimizer. Fig. 6 and Fig. 7 depict the non-sequential 3-change and mutation procedures. In non-sequential 3-change tour is broken in three edges and reconnected in two fashions. Whichever has shorter tour length is accepted while another is discarded.

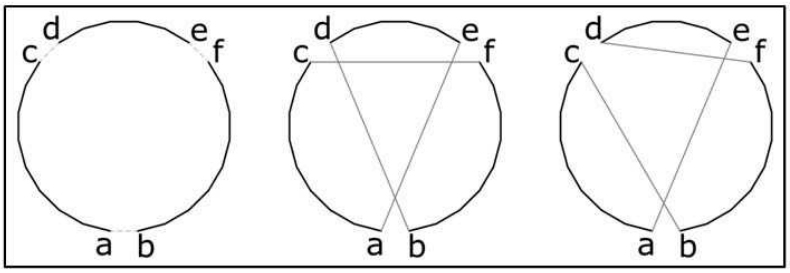

Fig. 6. Non-sequential 3-change.

```
Procedure Mutation (Offspring)
Begin
    Offspring: =Non-Sequential 3-change (Offspring);
    Offspring: =2-opt (Offspring);
End
```

Fig. 7. Mutation procedure.

Finally the offspring must be added to the population. The replacement Scheme is responsible for adding new offspring to the population. This scheme is designed in such a way that maintains the diversity of population and prevents premature convergence. According to [10] we define replacement scheme as follows. At first the nearest individual to new offspring with regard to mentioned definition of distance is identified. If the distance between them is below predefined threshold, the individual is replaced by new offspring. If it is not the case, individual is replaced if new offspring has higher fitness (smaller tour length); otherwise the individual with the lowest fitness (longest tour length) in the population must be replaced by new offspring.

5 Implementation and results

We implemented the mentioned MA in MATLAB 7.0.4 programming language using Intel Dual core 3.4 GHz on some small and medium size instances of TSPLIB[18]. Tab. 1 presents the process of improvement in solutions quality in proposed MA for best results in terms of deviation from global optimum. Tab. 2 summarizes the results and amount of time to obtain these results. In this table, best column presents the best tour found together with its deviation from global optimum. Average column presents the average of results for r runs where $r = 10$ for d198 and $r = 20$ for eil51, eil76, eil101, kroA100. The population sizes, P, for all instances are set to 10 and threshold in replacement scheme is 2. MA parameters i.e. recombination and mutation rates according to [10] are set to 0.5 and 0.2 respectively. Further improvement of solution qualities requires greater computational effort in terms of number of generations. Fig. 8 $\sim$ Fig. 12 depict the improvement of solution qualities when the generations grow up. These figures are good indicators of convergence behavior of this MA.

Table 1. Process of improvement in quality of Solutions for some STSP instances.

Problem	NN	NN + 2-opt	MA (2-opt + DPX)
eil51	22.54%	2.11%	0.82%
eil76	24.456%	11.740%	1.184%
kroA100	16.60%	0.45%	0.014%
eil101	28.320%	11.977%	2.9%
d198	11.57%	4.91%	1.761%

Table 2. Results and amount of time for proposed MA.

Problem Generation	Average (quality)	Best (quality)	Average	time(second)
eil51	100	437.826 (2.776%)	429.53 (0.82%)	18.436
eil76	200	560.0525 (4.098%)	544.37 (1.184%)	201.53
kroA100	200	21593.7 (1.464%)	21285 (0.014%)	224.93
eil101	200	668.1335 (6.221%)	647.27 (2.9%)	473.66
d198	50	16216.1 (2.763%)	16058 (1.761%)	904.6062

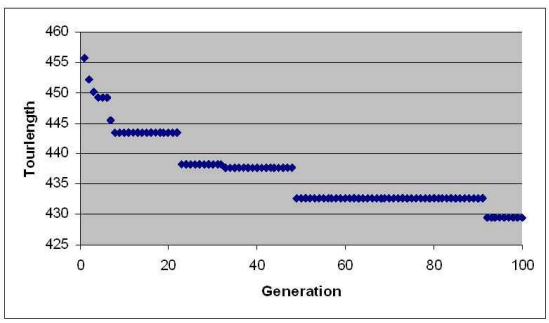

Fig. 8. MA convergence behaviour diagram for eil51 (best result).

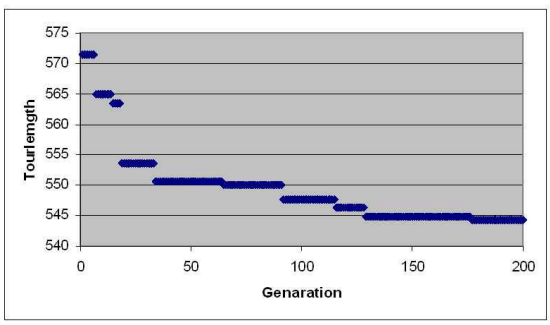

Fig. 9. MA convergence behaviour diagram for eil76 (best result).

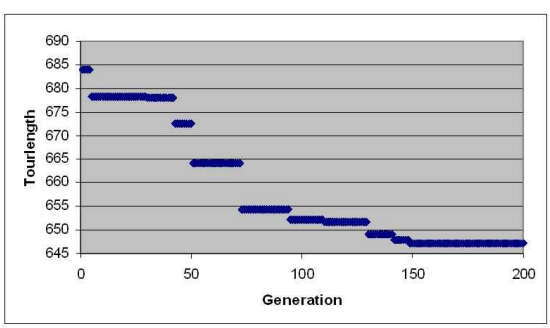

Fig. 10. MA convergence behaviour diagram for eil101 (best result).

6 Conclusion and remarks

In this paper, a memetic algorithm was presented to solve symmetric traveling salesman problem. By using Local Search we focus on population of local optima and then genetic algorithm was responsible for directing search among these local optima to global optima. The efficiency of proposed MA greatly depends on three factors i.e. Local search, Genetic operators and implementation issues. On local search side, we used 2-opt local optimizer for its ease of use, speed and its relative competency. On the second factor, genetic operators completely benefited from special features of TSP search space. On the third factor, we use MATLAB 7.0.4 programming language to implement our algorithm. Finally, there are some remarks for future researches. First, using techniques such as don't look bit[3] and candidate lists[1] can help in further improving speed of the algorithm. Second, we set parameters of genetic algorithm according to [10].Other kinds of tuning approach[8] for finding values of parameters maybe useful if we want to reach reliable solutions.

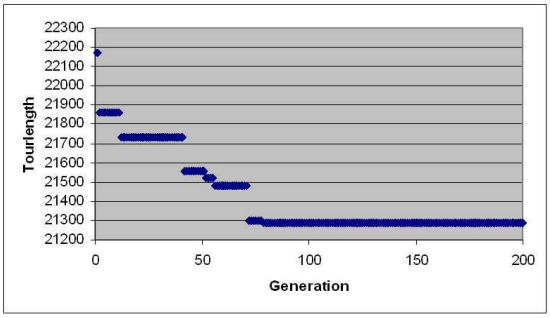

Fig. 11. MA convergence behaviour diagram for kroA100 (best result).

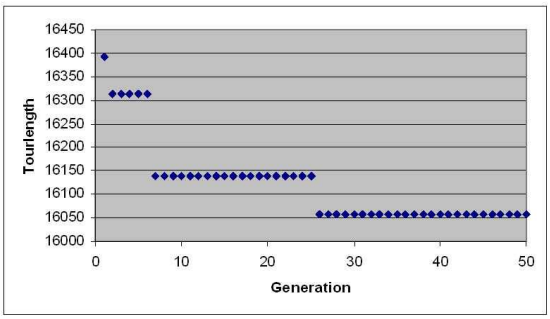

Fig. 12. MA convergence behaviour diagram for d198 (best result).

References

[1] The traveling salesman problem: A guided tour of combinatorial optimization. Wiley and Sons, 1985.
[2] E. Aarts, H. Stehouwer. Neural networks and the travelling salesman problem. **in:** *Proc. of International Conference on Artificial Neural Networks*, 1993, 950–955.
[3] J. Bentley. Fast algorithms for geometric traveling salesman problems. *ORSA Journal on computing*, 1992, **4**: 387–411.
[4] K. Boese. Cost versus distance in the traveling salesman problem. *Tech. Rep. TR-950018*, 1995. UCLA CS Department.
[5] I. Charon, O. Hudry. Application of the noising method to the travelling salesman problem. *European Journal of Operational Research*, 2000, **125**: 266–277.
[6] M. Dorigo, L. Gambardella. Ant colonies for traveling salesman problem. *Biosystems*, 1997, **43**: 73–81.
[7] M. Dorigo, L. Gambardella. Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computations*, 1997, **1**: 53–66.
[8] A. Eiben, etc. Parameter control in evolutionary algorithms. *Computational Intelligence (SCI)*, 2007, **54**: 19–46.
[9] L. Feichter. A parallel tabu search for large traveling salesman problems. *Discrete Applied Mathematics and Combinatorial Operations Research and Computer Science*, 1994, **51**: 234–267.
[10] B. Freisleben, P. Merz. A genetic local search algorithm for solving symmetric and asymmetric traveling salesman problems. **in:** *Proc. of IEEE international conference on evolutionary computation*, 1996.
[11] M. Garey, D. Johnston. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, 1979.
[12] D. Goldberg. *Genetic algorithms in search, optimization and machine learning*. Addison -Wesley, 1989.
[13] L. Homaifar, C. Guan, G. Liepins. A new approach to the traveling salesman problem by genetic algorithms. **in:** *Proc. of 5th International Conference on Genetic Algorithms*, 1993, 460–466.
[14] D. Johnson, L. McGeoch. The traveling salesman problem: A case study in local optimization. **in:** *Local Search in Combinatorial Optimization*, Princeton University Press, 2003, 215–310.
[15] P. Laarhoven, E. Aarts. *Simulated Annealing: Theory and Applications*. Kluwer Academic Publishers, 1987.
[16] P. Moscato. Memetic algorithms. **in:** *Handbook of Applied Optimization*, OXFORD UNIVERSITY PRESS, 2002, 157–165.
[17] P. Moscato. A gentle introduction to memetic algorithms. **in:** *HANDBOOK OF METAHEURISTICS*, KLUWER ACADEMIC PUBLISHERS, 2003, 105–144.
[18] G. Reinelt. Tsplib a traveling salesman problem library. *ORSA Journal on Computing*, 1991, **3**: 376–384.
[19] A. Tang, K. Leung. A modified edge recombination operator for the travelling salesman problem. **in:** *Proc. of Parallel Problem Solving from Nature III* (H. Schwefel, R. Manner, eds.), 1994, 180–188.
[20] D. Wolpert, W. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1997, **1**(1): 67–82.
[21] M. Yaguira, T. Ibaraki. Local search. **in:** *Handbook of Applied Optimization*, OXFORD UNIVERSITY PRESS, 2002, 104–123.