A Neuro-Ethological Approach for the TSP: Changing Metaphors in Connectionist Models

Orazio Miglino

Department of Psychology
University of Palermo
90100 Palermo PA, Italy
(orazio@caio.irmkant.rm.cnr.it)

Filippo Menczer

Department of Computer Science and Engineering
University of California, San Diego
La Jolla, CA 92093-0114, USA
(fil@cs.ucsd.edu)

Pierre Bovet

Department of Psychology and Educational Sciences
University of Geneva
CH-1227 Carouge/Geneve, Switzerland
(bovet@uni2a.unige.ch)

ABSTRACT

Biological systems often offer solutions to difficult problems which are not only original but also efficient. Connectionist models have been inspired by neural systems and successfully applied to the formulation of algorithms for solving complex problems such as the traveling salesman problem. In this paper we extend the connectionist metaphor to include an ethological account of how problems similar to the traveling salesman problem are solved by real living systems. A model is presented in which a population of neural networks with simple sensory-motor systems evolve genetically in simulated environments which represent the problem instances to be solved. Preliminary results are discussed, showing how the ethological metaphor allows to overcome some shortcomings of other connectionist models, such as their time and space complexity.

Keywords: traveling salesman problem, computational neuro-ethology, genetic algorithms, connectionism, computational complexity.

1 Introduction

Combinatorial optimization problems in the important NP-complete class have no efficient and general way to be solved. NP-complete problems are such that the number of solutions grows faster than any power of the number of parameters [8, 17]. A widely studied NP-complete problem, the traveling salesman problem (TSP), can be expressed as follows: given N points scattered in a bi-dimensional plane, compute the shortest path that connects all of them. Algorithms based on heuristics have been proposed to solve the TSP and they produce good solutions [13, 14].

In contrast to the mathematical computational style of these algorithms, computer science and artificial intelligence often use metaphorical representations which produce insight for many difficult problems. Metaphors underlie some features of a problem while hiding many others. However, the problem-solver (artificial or natural system) is biased by the chosen metaphor toward particular solution strategies.

In this context, Hopfield and Tank [12] suggested an approach for the TSP based on biological computation (artificial neural networks), and found reasonable solutions on limited problem sets. But, as they themselves stressed, the importance of their work was in suggesting a new, biologically oriented approach for solving complex formal problems. Cuykendall and Reese [5] and Xu and Tsai [19] improved the performance obtained by of Hopfield and Tank's neural networks by adjusting some technical aspects of the original model.

Another branch of these studies is represented by several authors who prefer to work on different biological networks. They use algorithms based on self-organizing networks such as elastic nets [6] and feature maps [1, 7]. Favata and Walker [7], in particular, were able to combine biological plausibility with good performance. The methodology of building artificial systems with a biologically-based computation seems promising for approaching many formal theoretical problems such as the TSP.

Unfortunately, in the TSP case, the two above connectionist approaches produce neural networks with sizes which relate inefficiently to the problem size. For example, for N points, Hopfield and Tank used a network with $O(N^2)$ units and $O(N^3)$ connections; Durbin and Willshaw [6] and Favata and Walker [7] used networks with $O(N)$ units. This feature of neural models leads to two unpleasant effects: (i) the computational complexity (space and, consequently, time) does not allow for the building of networks able to solve problems with arbitrarily large sizes; (ii) each network cannot generalize to different problem instances (e.g., with different sizes).

We propose that these limitations are due in part to implicit assumptions underlying the neural TSP metaphors. For instance, the above algorithms imply that all the available information about the problem parameters (positions of the points) is used at the same time and in a global way. No consideration is given to the process by which such information is collected and accessed in natural systems.

In this paper, we will suggest a way to overcome the limitations of these connectionist methods by suggesting a broader metaphor, in which the concept of biological computation is enlarged from the neural view to include sensory and motor features of real organisms.

Since early cybernetics there have been many authors who have suggested artificial systems simulating animal sensory-motor features. For example, Braitenberg's book [3] is seminal in this field. More recently, many researchers have tried to join artificial neural networks and life-like artificial systems. Cliff [4] suggested a new field called computational neuro-ethology. Parisi et al. [18] have used feed-forward neural networks called "econets" to simulate the sensory-motor systems of very simple organisms which move in environments. Other authors have used similar approaches analyzing more complex sensory systems [2, 15, 16]. Our present work falls within the framework of these models.

We imagine the TSP problem in terms of an organism that is inside an environment and must reach all target points scattered in it. At any given time, the organism only accesses (local) information about its surroundings by means of a sensory system, and produces a motor action based on such information.

2 The Model

The environment in which an artificial organism acts is a two-dimensional square divided up into cells. Some cells contain targets which correspond to the TSP points. An environment represents a random TSP map in the unit square. Organisms' networks are trained to reach all the targets, starting from a given one, using the shortest possible path. Figure 1 shows an example. This task does not require completing the tour by returning to the starting-point. The reason is that completing the cycle would require some form of memory, while our goal is to solve a difficult problem using a simple two-layered feed-forward neural network. It should also be added here that the problem considered in this paper is a TSP variation in which the points are distributed on the unit square, with Euclidean distances, and with the additional constraint that they lie on a grid -- the graining is chosen such that the number of cells is proportional to the number of points, and thus the density is constant. In spite of the above restrictions, however, the problem of finding the shortest path is easily shown to remain *NP*-complete.

The network's input contains information elaborated by the sensory apparatus of the organism; this is fed to the units in the hidden layer; and finally, the activations of the units in the output layer are used by the motor apparatus. All units in each layer are connected with all units in the next layer. Hidden and motor units have a Heaviside activation function.

The sensory apparatus in our model computes information about the position of targets in the following way. The environment is mapped onto 8 receptive fields, each a 45-degree sector centered at the organisms' current position. For each of these fields there is an input unit. An input value consists of the distance (non-linearly inverted and normalized) between the organism and its nearest target within the receptive field corresponding to that input.

A network has three binary output units, for a total of 8 possible output patterns. Each output pattern encodes a motor action by means of Gray binary code. The motor action lets the organism move into the neighboring cell in the direction determined by the output pattern. If the new cell contains a target, this is removed from the environment. The final path is determined by the order in which targets are reached. Figure 2 illustrates the sensory and motor systems of an individual.

A simple genetic algorithm [9, 11] is used for finding the weights of the neural network's connections. One creates an initial generation of networks with random weights. For each generation, each network is placed in each of a set of randomly generated environments, where it can make at most K moves.

The fitness criterion has two components. In order to account for learning to approach targets, it is incremented by one for each reached target. In order to account for learning to complete paths efficiently, it is incremented proportionally to $(K-T)$ when all targets have been reached after T moves ($T<K$).

Networks with higher fitness values are selected from each generation for reproduction. The next generation is created by making copies of the weight matrices of these organisms, with mutations (by additive uniform deviate) on a fraction of the weights.

3 Results

3.1 Genetic training

All the TSP instances used in both training and testing have points distributed *uniformly* in the unit square. Running experiments with many different seeds for the random number generator showed no appreciable differences in training performance, proving the model quite robust. Experiments with different numbers of hidden units showed that, as expected, larger hidden layers afford better performances. Table 1 summarizes the values used for all of the relevant parameters in our genetic training algorithm.

During training, the best networks gradually become able to reach all targets in each of their environments. Figure 3 shows, for each generation, the percentage of completed paths of the best organism in the population. Figure 4 plots the best time-performance obtained for each generation, measured by the number of steps required to complete the path. (Note that in the first few generations, when no paths are completed, the measure of time-performance is meaningless.) All data represent averages over the environments used during the training phase. Figures 3 and 4 also illustrate the convergence of the genetic training algorithm.

3.2 Performance

The best network evolved by our genetic training algorithm was tested on a number of random TSP instances (different from those used in training). While the training took place with fixed N (32 points), the test set included maps with both smaller and larger N, in order to see how the performance of the evolved network scales with problem size. Table 2 outlines the results, also showing the path length relative excess over the optimal length. The latter is computed based on the Held-Karp lower bound for best tours in the unit square [10].

A graphical representation of how the quality of paths found by the evolved networks scale with problem size is given in Figure 5. For sake of comparison, the lengths of paths found by a greedy heuristic algorithm are also plotted. This corresponds to the strategy of always moving toward the closest sensed point. For random maps with uniformly-distributed points and for N not too large (in the range of our experiments), such strategy is known to perform quite well compared with other local strategies. While the performance of the evolved network is worse in absolute terms, we find it particularly encouraging that its relative excess path length scales well with problem size, without any degradation for large N values in the observed range -- in fact, there is an improvement for N=1024. Figure 6 shows the quality of a typical path found by the evolved network.

3.3 Complexity

Time and space complexity are perhaps the best results of our neuro-ethological model. The space occupied by our evolutionary networks (or even by a population of them during training, since population size is kept constant) is $O(N)$. In our model this is because the size of the data structure used for the environment is linear in the number of points. Linear space complexity is asymptotically as good as the best neural models applied to the TSP (those inspired to feature maps and elastic nets [1, 6, 7]), and much better than the Hopfield and Tank networks [12] where space is $O(N^3)$. Linear space is the best possible because we need at least to represent the problem instance; furthermore, the size of the neural networks in our model is *constant*.

Time complexity is harder to compute for the average case in our stochastic model. However, by measuring the number of steps required by an evolved network to find a path, we get a natural measure of time, as shown in Figure 7. Time appears to scale *linearly* with N. Such $O(N)$ time complexity is better than that of all the other connectionist approaches to the TSP, which require at least $O(N^3)$ time. The reason for this is that in other neural models training has to be repeated for different problem sizes, while in our model the genetic training is done once and then its time can be amortized over many applications of the evolved network, for different values of N. Linear time complexity is a very promising result, particularly important for practical TSP problems, whose size can be very large ($N \sim 10^6$).

4 Conclusion

Extending the biological metaphor of neurally-inspired problem solvers has allowed us to apply artificial life techniques to a real, hard problem. One advantage of the neuro-ethological approach outlined in this paper is the space and time complexity of our algorithm: a genetically trained network can solve TSP instances of variable N with little or no loss of efficiency, while its size remains unchanged.

It is known that a greedy strategy cannot find globally optimal solutions to the TSP. For this reason, most heuristic algorithms for this problem choose sub-optimal paths under certain circumstances. This is also the case in our model, which does not impose a greedy strategy (see Figure 6). Since an organism can see different targets at once, it can choose a sub-optimal sub-path (e.g., not move toward the closest target) in order to obtain an optimal over-all path. Yet, since the information available to an organism is local, only strategies that are somehow greedy can evolve, and thus the performance obtained with our model must be compared with other greedy methods. The problem becomes harder for a greedy strategy when the density of points increases. In this light, we would hope to see the performance of the evolved networks improve to equal or exceed that of the greedy heuristic algorithm illustrated in the previous section for large problem sizes. In fact, there is partial evidence from our results that this happens for N exceeding the values explored in this paper (see the trend in Figure 5 for $N>512$). Expanding the range of N is the object of further experiments.

A number of improvements to the sensory system, neural organization, motor system, environment representation, genetic algorithm (e.g., training with varying problem sizes), and other features of the model are expected to lead to better quality of solutions. While the results presented in this paper are very preliminary, performance will be another direction for future work.

The appeal of biological computation has produced several artificial systems that solve complex formal problems efficiently. But the classical connectionist models have serious limitations. They are static and are built like pattern recognizers, while real biological computers (organisms) are agents situated in their environment, and build their perceptions through their actions. In this paper we have shown how a neuro-ethological viewpoint can be integrated with traditional models of neural computation and help to build artificial systems that are able to produce solutions for a particular formal problem. We also intend to apply the techniques of biological computation to other hard problems. In particular, for problems in which locality appears to be more important than in the TSP (i.e., those difficult for global heuristics but not so localized as to make a greedy strategy optimal), we expect the neuro-ethological approach to lead to efficient solutions.

Acknowledgments

We wish to thank Domenico Parisi, Elias Koutsoupias, Curtis Padgett, and Andrea Clementi for helpful discussions and suggestions.

References

[1] Angeniol B, de La Croix Vauban G and Le Texier J-Y, Self-Organizing Feature Maps and the Travelling Salesman Problem. *Neural Networks* 1 (1988) 289-293.

[2] Bovet P and Maurer R, Modeles de l'apprentissage spatial chez le robot et chez l'animal. In *Intelligence naturelle et intelligence artificielle,* ed. by Le Ny J-F (Paris: Presses Universitaires de France, 1993) 269-278.

[3] Braitenberg V, *Vehicles: Experiments in Synthetic Psychology* (Cambridge, MA: MIT press, 1984)

[4] Cliff D, Computational Neuroethology: A Provisional Manifesto. In *From animals to animats. Proceedings of the First International Conference on Simulation of Adaptive Behavior,* ed. by Meyer J and Wilson S (Cambridge, MA: MIT press, 1990) 29-39.

[5] Cuykendall R and Reese R, Scaling the neural TSP algorithm. *Biol Cybern* 60 (1989) 365-371.

[6] Durbin R and Willshaw D, An analog approach to the travelling salesman problem using an elastic net method. *Nature* 326 (1987) 689-691.

[7] Favata F and Walker R, A study of the application of Kohonen-type neural networks to the Travelling Salesman Problem. *Biol Cybern* 64 (1991) 463-468.

[8] Garey MR and Johnson DS, *Computer and Intractability* (San Francisco, CA: Freeman, 1979)

[9] Goldberg DE, *Genetic Algorithms in Search, Optimization and Machine Learning* (Reading, MA: Addison Wesley, 1989)

[10] Held M and Karp RM, The traveling-salesman problem and minimum spanning trees: Part 2. *Math. Programming* 1 (1971) 6-25

[11] Holland JH, *Adaptation in natural and artificial systems* (Ann Arbor, Michigan: University of Michigan Press, 1975)

[12] Hopfield JJ and Tank D, "Neural" computation of decisions in optimization problems. *Biol Cybern* 52 (1985) 141-152.

[13] Kirkpatrick S, Gelatt CD, and Vecchi MP, Optimization by simulated annealing. *Science* 220 (1983) 671-680.

[14] Lawler EL, Lenstra JK, Rinnooy Kan AHG, and Shmoys DB, *The Traveling Salesman Problem* (Chichester, NY: John Wiley, 1985)

[15] Menczer F and Belew RK, Evolving sensors in environments of controlled complexity. To appear in *Artificial Life IV* (1994)

[16] Menczer F and Parisi D, Evidence of Hyperplanes in the Genetic Learning of Neural Networks. *Biol Cybern* 66 (1992) 283-289.

[17] Papadimitriou CH and Steiglitz K, *Combinatorial Optimization: Algorithms and Complexity* (Englewood Cliffs, NJ: Prentice-Hall, 1982)

[18] Parisi D, Cecconi F, and Nolfi S, Econets: neural networks that learn in an environment. *Network* 2 (1990) 149-168.

[19] Xu X and Tsai WT, Effective Neural Algorithms for the Travelling Salesman Problem. *Neural Networks* 4 (1991) 193-205.

Figure Captions

Figure 1 The environment. Targets are scattered randomly on a grid. All cells reached by the individual are marked in grey. A path is defined by the order in which an organism reaches the cells containing targets.

Figure 2 Sensory-motor system. For each 45-degree receptive field, the distance of the closest target (if there is one within the sensory range) is passed to the corresponding input unit. The activation of the three output units is interpreted as one of eight possible translation moves.

Table 1 Genetic training parameters.

Figure 3 Percentage of complete paths during the training phase.

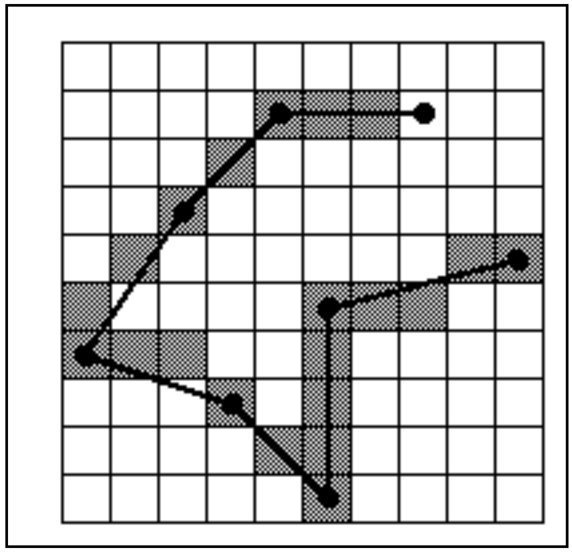
Figure 4: Number of moves used to reach all targets during the training phase.

Table 2 Performance results. Errors correspond to one standard deviation.

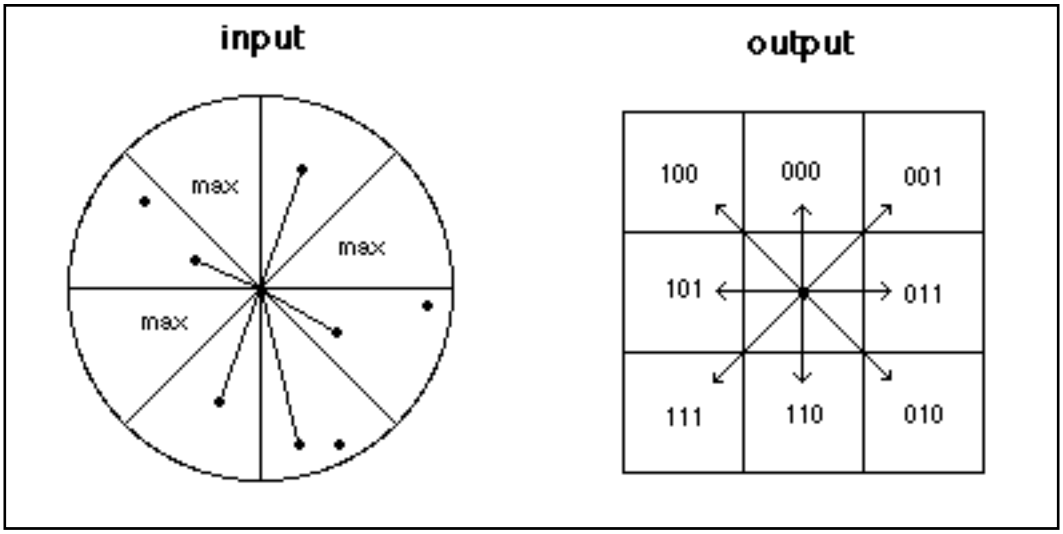
Figure 5: Scaling of performance with problem size. Points correspond to averages (error bars are negligible: see Table 2). Evolved networks are compared with both the Held-Karp lower bound and a greedy heuristic algorithm.

Figure 6 A typical path found for N=1024.

Figure 7 Experimental time complexity (with error bars corresponding to one standard deviation). The log-log plot allows to visualize that time scales linearly with problem size.



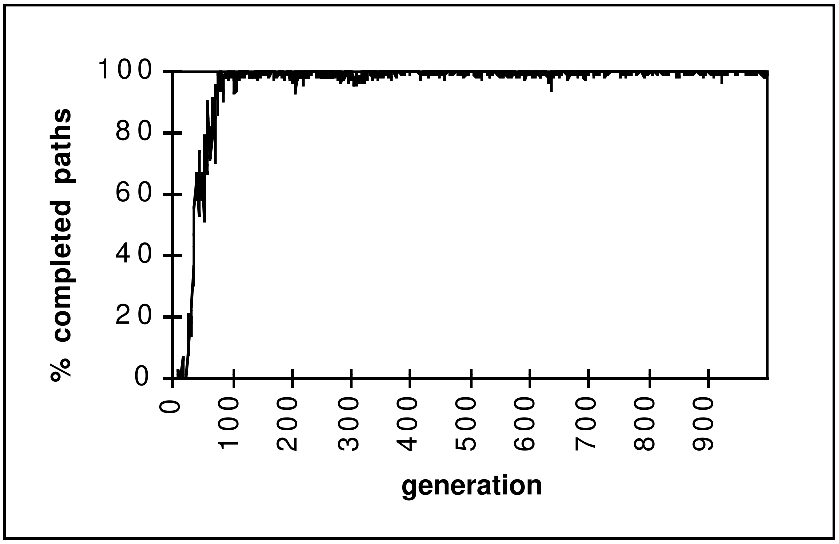
Miglino, Menczer and Bovet - Figure 1



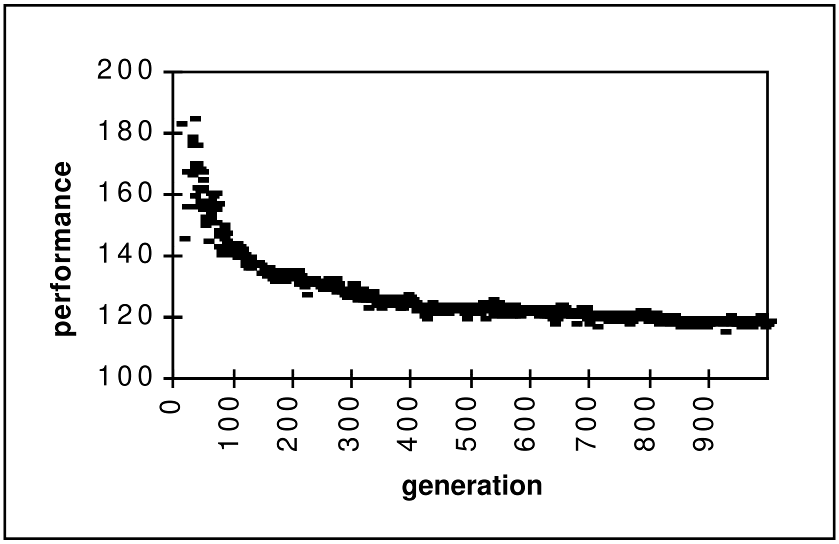
Miglino, Menczer and Bovet - Figure 2

N (number of points during training)	32
D (number of cells)	10*N
K (maximum number of steps)	D/2
number of environments per net per training	100
population size	100
percentage of parents	20%
number of generations for training	1000
percentage of weights mutated	10%
initial weight range	[-1,+1]
mutations range	[-1,+1]
number of hidden units	11

Miglino, Menczer and Bovet - Table 1



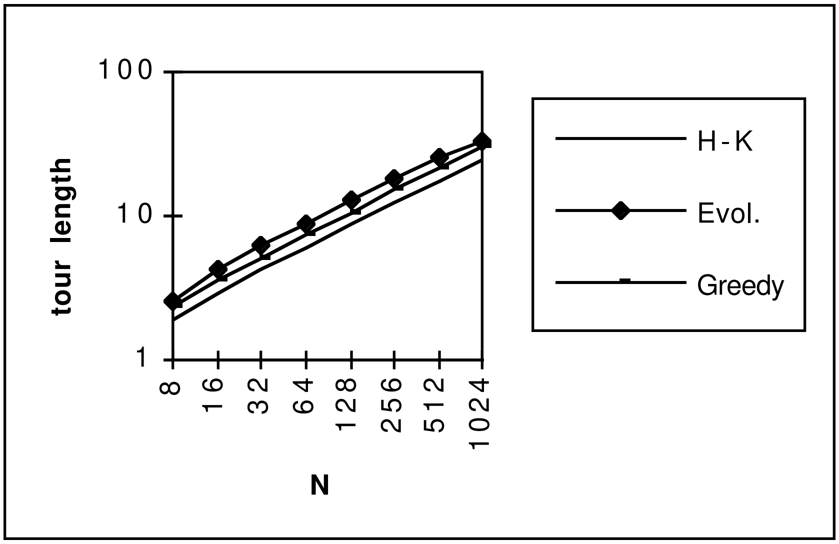
Miglino, Menczer and Bovet - Figure 3



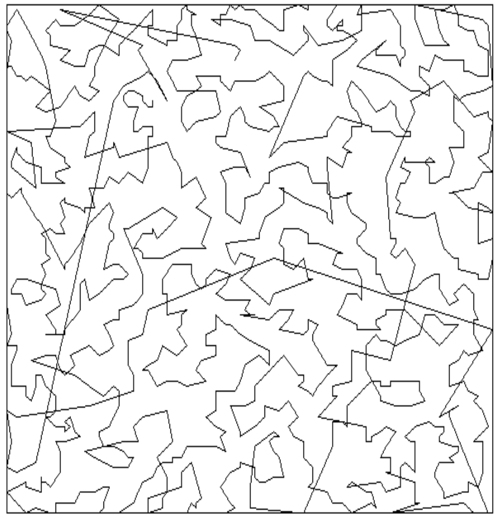
Miglino, Menczer and Bovet - Figure 4

N	Optimal	Evolved	Excess
8	1.9	2.5±0.1	32%
16	2.9	4.2±0.3	45%
32	4.2	6.1±0.1	45%
64	6.0	8.8±0.1	47%
128	8.6	12.7±0.1	48%
256	12.2	18.2±0.1	49%
512	17.3	25.7±0.1	49%
1024	24.5	32.1±0.1	31%

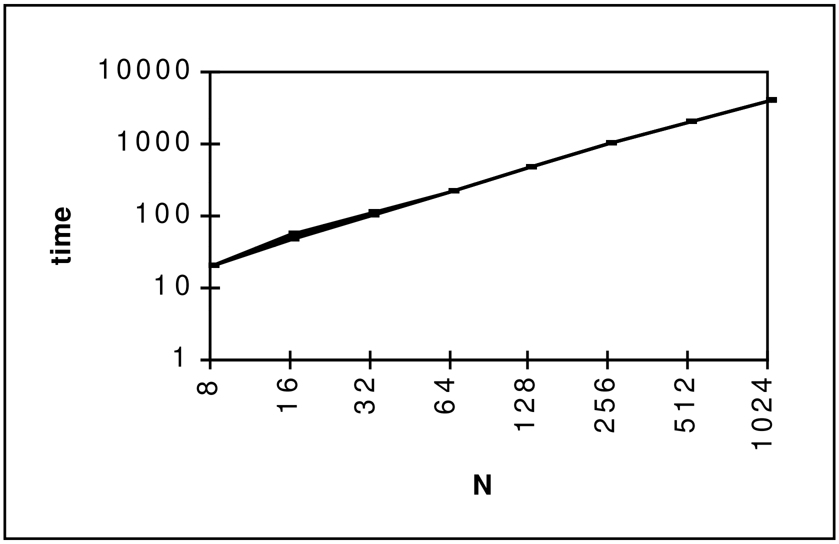
Miglino, Menczer and Bovet - Table 2



Miglino, Menczer and Bovet - Figure 5



Miglino, Menczer and Bovet - Figure 6



Miglino, Menczer and Bovet - Figure 7