An Adaptive Global-Local Memetic Algorithm to Discover Resources in P2P Networks

Ferrante Neri[1,2], Niko Kotilainen[1], and Mikko Vapa[1]

[1] Department of Mathematical Information Technology, Agora, University of Jyväskylä, FI-40014, Finland, {neferran, npkotila, mikvapa}@jyu.fi
[2] Dipartimento di Elettrotecnica ed Elettronica, Politecnico di Bari, Via E. Orabona 4, 70125, Italy, neri@deemail.poliba.it

Abstract. This paper proposes a neural network based approach for solving the resource discovery problem in Peer to Peer (P2P) networks and an Adaptive Global Local Memetic Algorithm (AGLMA) for performing the training of the neural network. This training is very challenging due to the large number of weights and noise caused by the dynamic neural network testing. The AGLMA is a memetic algorithm consisting of an evolutionary framework which adaptively employs two local searchers having different exploration logic and pivot rules. Furthermore, the AGLMA makes an adaptive noise compensation by means of explicit averaging on the fitness values and a dynamic population sizing which aims to follow the necessity of the optimization process. The numerical results demonstrate that the proposed computational intelligence approach leads to an efficient resource discovery strategy and that the AGLMA outperforms two classical resource discovery strategies as well as a popular neural network training algorithm.

1 Introduction

During recent years the use of peer-to-peer networks (P2P) has significantly increased and thus demand of high performance peer-to-peer networks is constantly growing. In order to obtain proper functioning of a P2P network a crucial point is to efficiently execute the P2P resource discovery, since an improper resource discovery strategy would lead to overwhelming query traffic and consequently to a waste of bandwidth for each single user.

This problem has been intensively analyzed and several solutions have been proposed in commercial packages and scientific literature. The solutions so far proposed can be classified into two categories: breadth-first search (BFS) and depth-first search (DFS). BFS strategies forward a query to multiple neighbors at the same time whereas DFS strategies forward only to one neighbor.

BFS strategies have been used in Gnutella, where the query is forwarded to all neighbors and the forwarding is controlled by a time-to-live parameter. This parameter is defined as the amount of hops required to forward the query. Two nodes are said to be n hops apart if the shortest path between them has length n [1]. The main disadvantage of the Gnutella's mechanism is that it generates a massive traffic of query messages when the time-to-live parameter is high.

In order to reduce query traffic, Lv et al. [2] proposed the *Expanding Ring*. This strategy establishes that the time-to-live parameter is gradually increased until enough resources have been found. Although use of the *Expanding Ring* is beneficial in terms of query packet reduction, it introduces some delay to resource discovery and thus implies a longer waiting time for the user. Kalogeraki et al. [3] and Menascé [4] proposed that only a subset of neighbors are selected randomly for forwarding. While in [3] a mechanism is proposed which stores the performance of the queries previously done for each neighbor and then uses this memory to direct subsequent queries, in [4] the earlier replies are cached in directory entries and queried prior to using broadcast probability. Yang and Garcia-Molina [1] proposed to heuristically select the first neighbor and further uses BFS for forwarding the query. In Gnutella2 a trial query is sent to the neighbors and estimates how widely the actual query should be forwarded.

In the DFS strategies, selection of the neighbor for query forwarding is performed by means of heuristics. Lv et al. [2] studied the use of multiple random walkers which periodically check the query originator in order to verify whether the query should be forwarded further. Tsoumakos and Roussopoulos [5] proposed using the feedback from previous queries in order to tune probabilities for further forwarding of random walkers. Crespo and Garcia-Molina [6] proposed routing indices, which provide shortcuts for random walkers in locating resources. Sarshar et al. [7] proposed replicating a copy of resources and thus ensure that resource discovery strategy locates at least one replica of the resource.

The main limitation of the previous studies, for both BFS and DFS strategies, is that all the approaches are restricted to only one search strategy. On the contrary, for the same P2P network, in some conditions it is preferable to employ both BFS and DFS strategies. In order to obtain a flexible search strategy, which intelligently takes into account the working conditions of the P2P network, Vapa et al. [8] proposed a neural network based approach (NeuroSearch) which adaptively combines BFS and DFS. In NeuroSearch, a trained neural network is able to map a specific input set to forward decisions in an if-then logic. Thanks to this logic, the resource discovery strategy can be applied also in devices with limited computing power. On the other hand, training neural networks to adapt to various conditions is challenging since it requires training in multiple topological scenarios thus leading to complicated computational requirements. It is therefore fundamental to investigate efficient training algorithms which lead to high performance in a short training time.

2 Problem Description

NeuroSearch [8] is a neural network-based approach which combines different local information units together as an input to multi-layer perceptron (MLP) neural network [9]. The neural network employed in NeuroSearch contains two hidden layers, both having 10 neurons and two different transfer functions in hidden and output layers. The structure of this neural network (see Fig. 1) has been selected on the basis of previous studies carried out by means of the

P2PRealm simulation framework [10]. Details regarding the functioning of this neural network are given in [8] and [10]. We characterize the query forwarding situation with a model consisting of 1) the previous forwarding node, 2) the currently forwarding node and 3) the receiver of the currently forwarding node. Upon receiving a query, the currently forwarding node selects the first of its neighbors and determines the inputs, related to that neighbor, of the neural network. The neural network output is then calculated. This output establishes whether or not the query will be forwarded to the neighbor. Next, all other neighbors including the previous forwarding node, are processed in a similar manner by means of the same neural network. Fig. 2, shows an example of the functioning of a P2P network with neural network based forwarding. The circles shown in the figure represent peers of the P2P network. The arcs between the peers represent the Transmission Control Protocol communication links between the peers. The rectangles represent a neural network evaluation for different neighbors. This paper addresses the problem in the training of a neural network

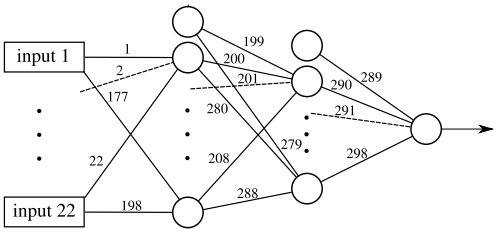

Fig. 1: MLP Neural Network

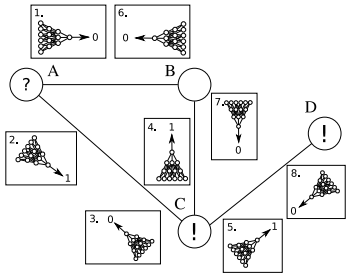

Fig. 2: Query Forwarding.

(i.e. the determination of the set of weight coefficients W) of the kind in Fig. 1 with the aim summarized in Fig 2. As shown in Fig. 1, the weights can be divided into three categories on the basis of the layer to which they belong to. There are 22 input neurons and 10 neurons on both the hidden layers. Since one input is constant (*Bias*, see [8]) the total amount of weights is $22 * 9 + 10 * 9 + 10 = 298$. The weights can take values within the range $(-\infty, \infty)$. In order to estimate the quality of a candidate solution, the performance of the P2P network is analyzed with the aid of a simulator whose working principles are described in [10] and a certain number n of queries are performed. For each query, the simulator returns two outputs: the number of query packets P used in the query and the number of found resource instances R during the query. At each j^{th} query, these outputs are combined in the following way and F_j is determined:

$$F_j = \begin{cases} 0 & \text{if } P > 300 \\ 1 - \frac{1}{P+1} & \text{if } P \leq 300 \text{ AND } \quad R = 0 \\ 50 * R - P & \text{if } P \leq 300 \text{ AND } 0 < R < \frac{AR}{2} \\ 50 * \frac{AR}{2} - P & \text{if } P \leq 300 \text{ AND } \quad \frac{AR}{2} < R \end{cases} \tag{1}$$

In (1), the amount of Available Resources (AR) instances is constant at each query and the constant values 300 and 50 have been set according to the criterion explained in [8]. It must be noted that due to its formulation each F_j could likely contain several plateaus (see (1)). The total fitness over the n queries is given by $F = \sum_{j=1}^{n} F_j(W)$. It is important to remark that multiple queries ($n = 10$) are needed in order to ensure that the neural network is robust in different query conditions. The querying peer and the queried resource need to be changed to ensure that the neural network is not only specialized for searching resources from one part of the network or one particular resource alone. Therefore, two consecutive fitness evaluations do not produce the same fitness value for the same neural network. Since n queries are required and, for each query, the first forwarding node is chosen at random, fitness F is noisy. This noise is not Gaussian. Let us indicate with $PN(n)$ the distribution of this noise and thus formulate the optimization problem addressed in this paper:

$$\max\left(F\left(W\right)+Z\right)\text{in}\left(-\infty,\infty\right)^{298};Z\sim PN\left(n\right) \tag{2}$$

3 The Adaptive Global-Local Memetic Algorithm

In order to solve the problem in (2), the following Adaptive Global-Local Memetic Algorithm (AGLMA) has been implemented.

Initialization. An initial sampling made up of S_{pop}^{i} individual has been executed pseudo-randomly with a uniform distribution function over the interval $[-0.2, 0.2]$. This choice can be briefly justified in the following way. The weights of the initial set of neural networks must be small and comparable among each other in order to avoid one or a few weights dominating with respect to the others as suggested in [11], [12].

Parent Selection and Variation Operators. All individuals of the population S_{pop} undergo recombination and each parent generates an offspring. The variation occurs as follows. Associated with each candidate solution i is a self-adaptive vector h_i which represents a scale factor for the exploration. More specifically, at the first generation the self-adaptive vectors h_i are pseudo-randomly generated with uniform distribution within $[-0.2, 0.2]$ (see [11], [12]).

At subsequent generations each self-adaptive vector is updated according to [11], [12]:

$$h_i^{k+1}\left(j\right)=h_i^k\left(j\right)e^{\left(\tau N_j(0,1)\right)}\ \text{ for }\ j=1,2...n \tag{3}$$

where k is the index of generation, j is the index of variable ($n = 298$), $N_j(0,1)$ is a Gaussian random variable and $\tau = \frac{1}{\sqrt{2\sqrt{n}}} = 0.1659$. Each corresponding candidate solution W_i is then perturbed as follows [11], [12]:

$$W_i^{k+1}\left(j\right)=W_i^k+h_i^{k+1}\left(j\right)N_j\left(0,1\right)\ \text{ for }\ j=1,2...n \tag{4}$$

Fitness Function. In order to take into account the noise, function F is calculated n_s times and an *Explicit Averaging* technique is applied [13]. More specifically, each set of weights for a neural network (candidate solution) is evaluated

by means of the following formula:

$$\hat{F} = F^i_{mean} - \frac{\sigma^i}{\sqrt{n_s}} \tag{5}$$

where F^i_{mean} and σ^i are respectively the mean value and standard deviation related to the n_s samples performed to the i^{th} candidate solution.

The penalty term $\frac{\sigma^i}{\sqrt{n_s}}$ takes into account distribution of the data and the number of performed samples [14]. Since the noise strictly depends on the solution under consideration, it follows that for some solutions the value of σ^i is relatively small (stable solutions) and so penalization is small. On the other hand, other solutions could be unstable and score 0 during some samples and give a high performance value during other samples. In these cases σ^i is quite large and the penalization must be significant.

Local Searchers. Two local searchers with different features in terms of search logic and pivot rule have been employed. These local searchers have the role of supporting the evolutionary framework, offering new search directions and exploiting the available genotypes [15].

1)Simulated Annealing The Simulated Annealing (SA) metaheuristic [16] has been chosen since it offers an exploratory perspective in the decision space which can choose a search direction leading to a basin of attraction different from starting point W_0 and, thus, prevents an undesired premature convergence. The exploration is performed by using the same mutation scheme as was described in equations (3) and (4) for an initial self-adaptive vector h_0 pseudo-randomly sampled in $[-0.2, 0.2]$.

The main reason for employing the SA in the AGLMA is that the evolutionary framework should be assisted in finding better solutions which improve the available genotype while at the same time exploring areas of the decision space not yet explored. It accepts, with a certain probability, solutions with worse performance in order to obtain a global enhancement in a more promising basin of attraction. In addition, the exploratory logic aims to overcome discontinuities of the fitness landscape and to "jump" into a plateau having better performance. For these reasons the SA has been employed as a "global" local searcher.

2)Hooke-Jeeves Algorithm The Hooke-Jeeves Algorithm (HJA) [17] is a deterministic local searcher which has a steepest descent pivot rule. The HJA is supposed to efficiently exploit promising solutions enhancing their genotype in a meta-Lamarckian logic and thus assist the evolutionary framework in quickly climbing the basin of attractions. In this sense the HJA can be considered as a kind of "local" local searcher integrated in the AGLMA.

Adaptation. In order to design a robust algorithm [15], at the end of each generation the following parameter is calculated:

$$\psi = 1 - \left| \frac{\hat{F}_{avg} - \hat{F}_{best}}{\hat{F}_{worst} - \hat{F}_{best}} \right| \tag{6}$$

where $\hat{F}_{worst}$, $\hat{F}_{best}$, and $\hat{F}_{avg}$ are the worst, best, and average of the fitness function values in the population, respectively. As highlighted in [18], ψ is a

fitness-based measurement of the population diversity which is well-suited for flat fitness landscapes. The employment of this parameter, taking into account the presence of plateaus in the fitness landscape (i.e. areas with a very low variability in the fitness values.) ψ, efficiently measures the population diversity even when the range of variability of all fitness values is very small. The population has high diversity when $\psi \approx 1$ and low diversity when $\psi \approx 0$. A low diversity means that the population is converging (possibly in a suboptimal plateau). We remark that the absolute diversity measure used in [14], [19], [20] and [21] is inadequate in this case, since, according to this, the population diversity would be very low most of the time.

Coordination of the local searchers. The SA is activated by the condition $\psi \in [0.1, 0.5]$. This adaptive rule is based on the observation that for values of $\psi > 0.5$, the population diversity is high and therefore the evolutionary framework needs to have a high exploitation of the available genotypes (see [19], [18] and [21]). On the other hand, if $\psi < 0.5$ the population diversity is decreasing and application of the SA can introduce a new genotype in the population which can prevent a premature convergence. In this sense, the SA has been employed as a local searcher with "global" exploratory features. The condition regarding the lower bound of usability of the SA ($\psi > 0.1$) is due to the consideration that if $\psi < 0.1$ application of the SA is usually unsatisfactory since it most likely leads to a worsening in performance.

Moreover, the SA, in our implementation, is applied to the second best individual. This gives a chance at enhancing a solution with good performance without possibly ruining the genotype of the best solution. The initial temperature $Temp^0$ has been adaptively set $Temp^0 = \left|\hat{F}_{avg} - \hat{F}_{best}\right|$. This means that the probability of accepting a worse solution depends on the state of the convergence. In other words, the algorithm does not accept worse solutions when the convergence has practically occurred.

The HJA is activated when $\psi < 0.2$ and is applied to the solution with best performance. The basic idea behind this adaptive rule is that the HJA has the role of quickly improving the best solution while staying in the same basin of attraction. In fact, although evolutionary algorithms are efficient in detecting a solution which is near the optimum, they are not so efficient in "ending the game" of optimization. In this light, the action of the HJA can be seen as purely "local". The condition $\psi < 0.2$ means that the HJA is employed when there are some chances that optimal convergence is approaching. An early application of this local searcher can be inefficient since a high exploitation of solutions having poor fitness values would not lead to significant improvements of the population.

It should be noted that in the range $\psi \in [0.1, 0.2]$ both local searchers are applied to the best two individuals of the population. This range is very critical for the algorithm because the population is tending towards a convergence but still has not reached such a condition. In this case, there is a high risk of premature convergence due to the presence of plateaus and suboptimal basins of attraction or false minima introduced by noise. Thus, the two local searchers are supposed to "compete and cooperate" within the same generation, merging

the "global" search power of the SA and the "local" search power of the HJA. An additional rule has been implemented. When the SA has succeeded in enhancing the starting solution, the algorithm attempts to further enhance it by the application of the HJA under supervision of the evolutionary framework.

Dynamic population size in survivor selection. The population is resized at each generation and the S_{pop} individuals having the best performance are selected for the subsequent generation:

$$S_{pop} = S_{pop}^{f} + S_{pop}^{v} \cdot (1 - \psi)\,, \tag{7}$$

where S_{pop}^{f} and S_{pop}^{v} are the fixed minimum and maximum sizes of the variable population S_{pop}, respectively.

The dynamic population size has two combined roles. The first is to massively explore the decision space and thus prevent a possible premature convergence (see [19]), the second is to *Implicitly Average* in order to compensate for noise by means of the evaluations of similar individuals [13]. According to the first role, when $\psi \approx 0$ the population is converging and a larger population size is required to increase the exploration and possibly inhibit premature convergence by offering new search directions. On the other hand, if the population is spread out in the decision space it is highly desirable that the most promising solution leads the search and that the algorithm exploits this promising search direction. According to the second role, it is well-known that large population sizes are helpful in defeating the noise [22]. Furthermore, recent studies [14], [23] have noted that the noise jeopardizes functioning of the selection mechanisms especially for populations made up of individuals having similar performance, since the noise introduces a disturbance in pair-wise comparison. Therefore, the AGLMA aims to employ a large population size in critical conditions (low diversity) and a small population size when a massive averaging is unnecessary. The algorithm stops when either a budget condition on the number of fitness evaluations is satisfied or ψ takes a value smaller than 0.01.

4 Numerical Results

For the AGLMA 30 simulation experiments have been executed. Each experiment has been stopped after 1.5×10^6 fitness evaluations. At the end of each generation, the best fitness value has been saved. These values have been averaged over the 30 experiments available. The average over the 30 experiments defines the Average Best Fitness (ABF). Analogously, 30 experiments have been carried out with the Checkers Algorithm (CA) described in [11], [12] according to the implementation in [8], and the proposed here Adaptive Checkers Algorithm (ACA) which is the CA with the fitness as shown in (5) and the adaptive population size as shown in (7). For the same P2P network, the BFS according to the implementation in Gnutella and the random walker DFS proposed in [2] have been applied. Table 1 shows the parameter settings for the three algorithms and the optimization results. The final fitness $\hat{F}^b$ obtained by the most successful experiment (over the 30 sample runs), the related number of query packets

P used in the query and the number of found resource instances R during the query are given. In addition the average best fitness at the end of the experiments $< \hat{F} >$, the final fitness of the least successful experiment $\hat{F}^w$ and the related standard deviation are shown. Since the BFS follows a deterministic logic, thus only one fitness value is shown. On the contrary, the DFS under study employs a stochastic structure and thus the same statistic analysis as that of CA, ACA and AGLMA over 30 experiments has been carried out.

Table 1: Parameter setting and numerical results

PARAMETER	AGLMA	CA	ACA	BFS	DFS
EVOLUTIONARY FRAMEWORK					
S^i_{pop}	30	30	30	–	–
S_{pop}	$\in [20, 40]$	30	$\in [20, 40]$	–	–
sample size n_s	10	–	10	–	–
SIMULATED ANNEALING					
initial temperature $Temp^0$	adaptive	–	–	–	–
temperature decrease	hyperbolic	–	–	–	–
maximum budget per run	600	–	–	–	–
HOOKE-JEEVES ALGORITHM					
exploratory radius	$\in [0.5, 0.01]$	–	–	–	–
maximum budget per run	1000	–	–	–	–
NUMERICAL RESULTS					
P	350	372	355	819	514
R	81	81	81	81	81
$\hat{F}^b$	3700	3678	3695	3231	3536
$< \hat{F} >$	3654	3582	3647	–	3363
$\hat{F}^w$	3506	3502	3504	–	3056
std	36.98	37.71	36.47	–	107.9

Numerical results in Table 1 show that the AGLMA and ACA outperform the CA and that the AGLMA slightly outperformed the ACA in terms of the final solution found. Moreover, the AGLMA clearly outperforms the BFS employed in Gnutella and the DFS.

Figures 3 and 4 show the comparison of the performance. As shown, the AGLMA has a slower convergence than the CA and the ACA but reaches a final solution having better performance. It is also clear that the ACA has intermediate performance between the CA and AGLMA. The ACA trend, in early generations, has a rise quicker than the AGLMA but slower than the CA. On the other hand, in late generations, the ACA outperforms the CA but not the AGLMA. Regarding effectiveness of the noise filtering components, Fig. 4 shows that the ACA and the AGLMA are much more robust with respect to noise than the CA. In fact, the trend of the CA performance contains a high amplitude and frequency ripple, while the ACA and AGLMA performance are roughly monotonic. Regarding effectiveness of the local searchers, the comparison between the

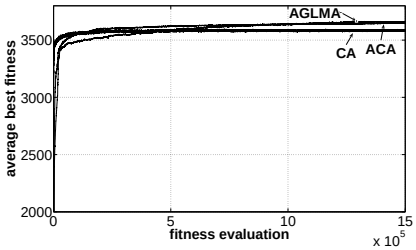

Fig. 3: Algorithmic Performance

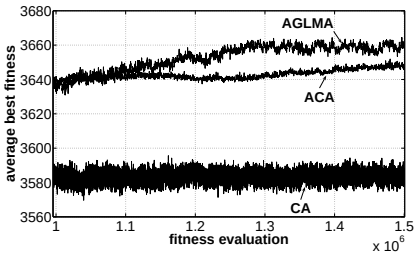

Fig. 4: Performance (zoom detail)

ACA and the AGLMA shows that the AGLMA slightly outperforms the ACA tending to converge to a solution having a better performance.

5 Conclusion

This paper proposes an AGLMA for performing the training of a neural network, which is employed as computational intelligence logic in P2P resource discovery. The AGLMA employs averaging strategies for adaptively executing noise filtering and local searchers in order to handle the multivariate fitness landscape. These local searchers execute the global and local search of the decision space from different perspectives. The numerical results show that the application of the AGLMA leads to a satisfactory neural network training and thus to an efficient P2P network functioning. The proposed neural network along with the learning strategy carried by the AGLMA allows the efficient location of resources with little query traffic. Thus, with reference to classical resource discovery strategies (Gnutella BFS and DFS), the user of the P2P network obtains plentiful amounts of information about resources consuming a definitely smaller portion of bandwidth for query traffic. Regarding performance during the optimization process, comparison with a popular metaheuristic present in literature shows the superiority of the AGLMA in terms of final solution found and reliability in a noisy environment.

References

1. Yang, B., Garcia-Molina, H.: Improving search in peer-to-peer networks. In: Proc. of the 22nd Intern. Conf. on Distributed Computing Systems. (2002) 5–14
2. Lv, Q., Cao, P., Cohen, E., Li, K., Shenker, S.: Search and replication in unstructured peer-to-peer networks. In: Proc. of the 16th ACM Intern. Conf. on Supercomputing. (2002) 84–95
3. Kalogeraki, V., Gunopulos, D., Zeinalipour-Yazti, D.: A local search mechanism for peer-to-peer networks. In: Proc. 11th ACM Intern. Conf. on Information and Knowledge Management. (2002) 300–307
4. Menascé, D.A.: Scalable p2p search. IEEE Internet Computing **7**(2) (2003) 83–87

5. Tsoumakos, D., Roussopoulos, N.: Adaptive probabilistic search for peer-to-peer networks. In: Proc. 3rd IEEE Intern. Conf. on P2P Computing. (2003) 102–109
6. Crespo, A., Garcia-Molina, H.: Routing indices for peer-to-peer systems. In: Proc. of the 22nd IEEE Intern. Conf. on Distributed Computing Systems. (2002) 23–33
7. Sarshar, N., Boykin, P.O., Roychowdhury, V.P.: Percolation search in power law networks: Making unstructured peer-to-peer networks scalable. In: Proc. of the IEEE 4th Intern. Conf. on P2P Computing. (2004) 2–9
8. Vapa, M., Kotilainen, N., Auvinen, A., Kainulainen, H., Vuori, J.: Resource discovery in p2p networks using evolutionary neural networks. In: Intern. Conf. on Advances in Intelligent Systems - Theory and Applications, 067-04. (2004)
9. Engelbrecht, A.: Computational Intelligence-An Introduction. J. Wiley (2002)
10. Kotilainen, N., Vapa, M., Keltanen, T., Auvinen, A., Vuori, J.: P2prealm - peer-to-peer network simulator. In: IEEE Intern. Works. on Computer-Aided Modeling, Analysis and Design of Communication Links and Networks. (2006) 93–99
11. Chellapilla, K., Fogel, D.: Evolving neural networks to play checkers without relying on expert knowledge. IEEE Trans. Neural Networks, **10**(6) (1999) 1382–1391
12. Chellapilla, K., Fogel, D.: Evolving an expert checkers playing program without using human expertise. IEEE Trans. Evol. Computation, **5**(4) (2001) 422–428
13. Jin, Y., Branke, J.: Evolutionary optimization in uncertain environments - a survey. IEEE Transactions on Evolutionary Computation **9**(3) (2005) 303–317
14. Neri, F., Cascella, G.L., Salvatore, N., Kononova, A.V., Acciani, G.: Prudent-daring vs tolerant survivor selection schemes in control design of electric drives. In Rothlauf, F. et al., ed.: Applications of Evolutionary Computing, LNCS. Volume 3907., Springer (2006) 805–809
15. Krasnogor, N.: Toward robust memetic algorithms. In W. E. Hart et al., ed.: Recent Advances in Memetic Algorithms, Springer (2004) 185–207
16. Cerny, V.: A thermodynamical approach to the traveling salesman problem. Journal of Optimization, Theory and Applications **45**(1) (1985) 41–51
17. Hooke, R., Jeeves, T.A.: Direct search solution of numerical and statistical problems. Journal of the ACM, **8** (1961) pp. 212–229
18. Neri, F., Toivanen, J., Cascella, G.L., Ong, Y.S.: An adaptive multimeme algorithm for designing hiv multidrug therapies. IEEE/ACM Transactions on Computational Biology and Bioinformatics, Special Issue on Computational Intelligence Approaches in Computational Biology and Bioinformatics (2007) to appear.
19. Caponio, A., Cascella, G.L., Neri, F., Salvatore, N., Sumner, M.: A fast adaptive memetic algorithm for on-line and off-line control design of PMSM drives. IEEE Trans. on System Man and Cybernetics-part B (Feb 2007) to appear.
20. Neri, F., Toivanen, J., Mäkinen, R.A.E.: An adaptive evolutionary algorithm with intelligent mutation local searchers for designing multidrug therapies for HIV. Applied Intelligence, Springer (2007) to appear.
21. Neri, F., Mäkinen, R.A.E.: Hierarchical evolutionary algorithms and noise compensation via adaptation. In S. Yang et al., ed.: Evolutionary Computation in Dynamic and Uncertain Environments, Springer (2007) to appear.
22. Miller, B.L., Goldberg, D.E.: Genetic algorithms, selection schemes, and the varying effects of noise. Evolutionary Computation **4**(2) (1996) 113–131
23. Schmidt, C., Branke, J., Chick, S.E.: Integrating techniques from statistical ranking into evolutionary algorithms. In F. Rothlauf et al., ed.: Applications of Evolutionary Computing. Volume LNCS 3907., Springer (2006) 752–763