Fulfilling the Vision of Fully Autonomic Peer-to-Peer Systems

Michele Amoretti
Department of Information Engineering
University of Parma
Viale Usberti 181/a, Parma, Italy
michele.amoretti@unipr.it

ABSTRACT

A peer-to-peer (P2P) distributed system is a network of collectors, providers and consumers of resources. In order to be autonomic, peers may be provided with an internal structure being dynamically adjusted by an adaptive plan which determines successive re-configurations in response to the environment. To address this challenging problem, we have introduced the Adaptive Autonomic Framework, that allows to turn any P2P network in a complex adaptive system. In this paper we present the performance analysis of an AEF-based architecture for Grid Computing.

KEYWORDS: autonomic computing, peer-to-peer, genetic algorithms.

1. INTRODUCTION

A peer-to-peer system is a complex system, because it is composed of several interconnected parts that as a whole exhibit one or more properties (*i.e.* behavior) which are not easily inferred from the properties of the individual parts. The environment in which P2P networks operate is the set of users, that directly or indirectly act on peers and on resources that each node can use (running environment) and share with other nodes.

In the last decade, many considerable peer-to-peer protocols (pre-established patterns) have been proposed. On the other side, there is a lack of common understanding about adaptiveness mechanisms. Our attempt to fill this gap has produced the Adaptive Evolutionary Framework (AEF) [1], where autonomicity is defined in terms of an adaptive plan τ that evolves the peers' internal structure, or at least its configuration, in order to adapt to the environment (fig. 1). In details, the adaptive plan τ is based on an evolutionary algorithm, which utilizes a *population* of individuals (*i.e.* a set of possible structures or configurations for a peer), where each individual represents a candidate solution to the considered problem. In this way, autonomic peer-to-peer networks should be able to emulate biological systems to cope with unforeseen scenarios, variations in the environment or presence of deviant peers.

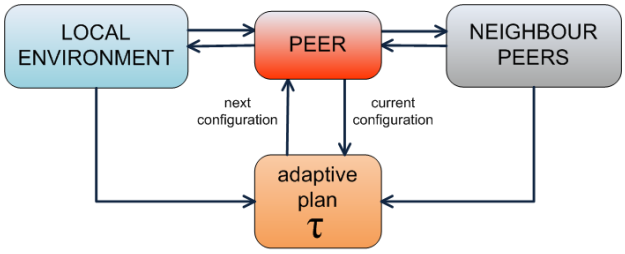

Figure 1. Interactions Between a Peer, Local Environment and Neighbors in an Adaptive P2P Architecture.

The research illustrated in this paper focuses on fixed structures, each one consisting of the same p elements, and evolving configurations. The ith element ($i = 1, .., p$) of the structure has l_i possible values. The set of possible values for the ith element is defined as $A_i = \{a_{i1}, .., a_{il_i}\}$. Each structure configuration $\vec{C}$ represents a point in the search space $\mathcal{A}$ (*configuration space*), which is the domain of action of the adaptive plan. In other words, $\mathcal{A}$ is the set of all combinations of values for the elements of the structure:

$$\mathcal{A} = A_1 \times A_2 \times ... \times A_p \quad (1)$$

The adaptive plan τ produces a sequence of configurations, *i.e.* a trajectory through $\mathcal{A}$, following an evolutionary pro-

cess. A very important step in the design of an evolutionary algorithm is to find an appropriate structure representation.

One case of particular importance is that in which the adaptive plan receives direct indication of the performance of each configuration it tries. That is, a part of the input I is the *fitness function*, which maps a structure representation into a scalar value:

$$F : \mathcal{A}^+ \rightarrow \mathbb{R} \tag{2}$$

The fitness function quantifies the quality of a configuration, *i.e.* how close it is to optimality, with respect to the environment. It is therefore extremely important that the fitness function accurately models the problem, including all criteria to be optimized. Frequently the fitness function does not directly evaluate the elements of a structure, but their consequences at a higher level of abstraction. If the fitness function represents a cost (as usual), it should return lower values for better structures.

If the adaptive plan is implemented by a *genetic algorithm (GA)*, the peer's configuration is the code that maps the way the peer looks and/or acts (*phenotype*) [6]. In other words, the configuration is the *genotype*, which is also called *chromosome* in evolutionary computing (in genetics this simplification is not correct). Moreover, the p elements of the structure are called *genes*. The ith gene ($i = 1, .., p$) of the structure has l_i possible *alleles* (previously called "values"); the set of possible alleles for the ith element is defined as $A_i = \{a_{i1}, .., a_{il_i}\}$, assuming finite values in limited specific ranges. Each configuration $\vec{C}$ represents a point in the search space $\mathcal{A}$, which is the set of all combinations of alleles (*i.e.* the *genome*).

GAs make use of operators like reproduction (cross-over, mutation) and elitism, which act on genotypes. Throughout the adaptation process, the selection operator is also used, in order to emphasize best configurations (chromosomes) in a population. The interplay of these operators leads to system optimization. Whenever the entities reproduce they create a surplus, variation amounts to innovation of novel entities (*i.e.* reproduction is not simply cloning), and finally selection takes care of promoting the right variants by discarding the poor ones [3].

The fitness function, in the context of GAs, is something like

$$F(\vec{C}) = a_1 F_1(\vec{C}) + a_2 F_2(\vec{C}) + .. \tag{3}$$

i.e. a function which rates the chromosomes according to a number of criteria, the influence of each criterion being controlled by the related constant $a \in \mathbb{R}$. Function F represents a cost and returns lower values for better chromosomes. In general, also the a_i factors may be set as evolving parameters, adapting the weight of each F_i to the environment. This aspect is out of the scope of this paper and will be considered in future work.

The following section illustrates an AEF-based architecture for sharing consumable resources such as disk space, CPU cycles, etc. in a Grid Computing environment. Simulation results illustrating the effectiveness of the AEF approach are presented in section . A discussion of related work is provided in section . Finally, section concludes the paper summarizing its main results and proposing future work.

2. GRID COMPUTING ARCHITECTURE BASED ON AEF

Grid Computing systems should enable efficient and scalable sharing of resources that can be discovered and used upon contracting with their hosts [11]. An example is disk space, which can be partitioned and allocated to requestors for the duration of a task, or in general for an arranged time.

Here we illustrate an AEF implementation, based on GAs, that enables peer-to-peer Grids. The envisioned overlay networks can be represented as undirected graphs whose arcs stand for mutual knowledge among peers. The *node degree* k is the number of neighbors of each node. Its statistical distribution depends on the history and on the dynamics of the network, and may affect the performance of the distributed algorithms which are executed.

Resource discovery is based on *epidemic* propagation of queries [4]. Query messages generated by a peer are matched with local resources and eventually propagated to neighbors. Each peer has a cache which contains advertisements of resources previously discovered in other peers. The cache is used as a preferred alternative to random (*i.e.* blind) query propagation. Each query message has a time-to-live (TTL) which is the remaining number of hops before the query message itself expires (*i.e.* it is no longer propagated). Moreover, each peer caches received queries, in order to drop subsequent duplicates. In general, bio-inspired epidemic protocols have considerable benefits as they are robust against network failures, are scalable and provide probabilistic reliability guarantees [10].

The resource discovery process is affected by the following parameters, representing the phenotype of each peer: f_k (fraction of neighbors targeted for query propagation), TTL_{max} (max number of hops for messages), and D_{max} (max size of the cache).

Traditional epidemic algorithms use fixed values for pa-

rameters, set in advance according to the results of some tuning process which should guarantee good performance with high probability. Actually, if all nodes would be configured with $f_k = 1$ and the same TTL value, the resulting protocol would be like Gnutella [5]. In our scheme, parameters are functions of the chromosome, thus randomly initialized when a peer is created and joins the network. Moreover, adaptive tuning of parameters is based on GAs, with fitness function $F(\Phi, \langle QHR \rangle)$ to be minimized, where $\Phi = \Phi(f_k, TTL_{max}, D_{max})$ and

$$\langle QHR \rangle = \frac{1}{k+1} \sum_{j=0}^{k} QHR_j \tag{4}$$

QHR_j is the query hit ratio, *i.e.* the number of query hits QH versus the number of queries Q, assuming index $j = 0$ for current peer and $j = 1, .., k$ for its neighbors.

Since shared resources (*e.g.* CPU, RAM, disk space) are consumable, the discovery process required by a job to start and complete its execution may be penalized by an increasing request rate. In general, the fitness function must be chosen in order to make each peer adapt to the rate of user requests which directly or indirectly affect its running environment.

If the values of the parameters which define the phenotype increase, the search process becomes more expensive in terms of time/space, but at the same time the discovery probability (which may be approximated by QHR) should result to be improved. In its most simple form

$$\Phi = \phi_0 f_k + \phi_1 TTL_{max} + \phi_2 D_{max} \tag{5}$$

where constant weights $\phi_0, \phi_1, \phi_2 \in \mathbb{R}$ control the influence of each parameter. In this case, the fitness function should reward peers with smaller Φ value, having the same *sufficiently high* $\langle QHR \rangle$ value.

The genotype $\vec{C}$ of each peer is given by three genes $\{C_0, C_1, C_2\}$ with values in a limited subset of $\mathbb{N}$. The relationship between the phenotype and the genotype is given by the following equations:

- $f_k = c_0 C_0$
- $TTL_{max} = c_1 C_1$
- $D_{max} = c_2 C_2$

where $c_i \in \mathbb{R}$, (with $i = 0, 1, 2$) are constants.

The pseudocode in Algorithm 1 describes the adaptation process which is periodically executed by each peer.

Algorithm 1 Adaptation

```
1: Let g = 0
2: while not converged do
3:    Evaluate the fitness of own chromosome
4:    g = g + 1
5:    Select neighbor with best fitting chromosome
6:    Perform cross-over with best neighbor to generate offspring O_g = {O⃗_g1, O⃗_g2}
7:    Mutate offspring in O_g with probability 1 − ⟨QHR⟩
8:    Select the new generation C_g from the previous generation C_{g−1} and the offspring O_g
9: end while
```

The best neighbor is chosen using proportional selection, *i.e.* the chance of individuals (neighbors' chromosomes) of being selected is inversely proportional to their fitness values. Cross-over is always performed, with a randomly-generated crosspoint. Mutation depends on $\langle QHR \rangle$, being highly improbable if the average query hit ratio of the peer and its neighbors tends to 1. The final selection between current peer's chromosome and the mutated offspring is random, with probability that is inversely proportional to their fitness values.

3. SIMULATION EXPERIMENTS

To compare the performance of the Grid Computing system with and without GA-based adaptation, we implemented it within the Discrete Event Universal Simulator (DEUS) [2]. Such a tool provides a simple Java API for the implementation of nodes, events and processes, and a straightforward but powerful visual tool for configuring simulations of complex systems.

Each simulated peer is characterized by three kinds of consumable resources: CPU, RAM, and disk space. Their values are randomly generated (with uniform distribution) as multiple of, respectively, 512 MHz, 256 MB, and 10 GB. The maximum amount of resources per peer is 2 GHz, 1 GB for the RAM, and 100 GB for the disk space.

We assumed $C_i \in [1, 6]$, and $f_k = C_0/6$, $TTL_{max} = C_1$, $D_{max} = 2C_2$.

We evaluated the performance of the adaptation scheme with respect to three different fitness functions:

$$F1(\Phi, \langle QHR \rangle) = \begin{cases} 1/\Phi & \text{if } \langle QHR \rangle < Th \\ \Phi & \text{if } \langle QHR \rangle > Th \end{cases}$$

$$F2(\Phi, \langle QHR \rangle) = \frac{1}{\delta^2}(1 - \langle QHR \rangle)\frac{1}{\Phi} + \langle QHR \rangle \Phi$$

$$F3(\Phi, \langle QHR \rangle) = \frac{1}{\delta^2}(\frac{1}{\langle QHR \rangle + \delta} - 1)\frac{1}{\Phi} + \langle QHR \rangle \Phi$$

where Th is a threshold value, Φ is given by eq. 5, with $\phi_0 = 100$, $\phi_1 = 10$, and $\phi_2 = 5$, for which f_k has more weight than TTL_{max} and D_{max} in the computation of the fitness value. Of course $\langle QHR \rangle$ is given by eq. 4. We assumed that each peer at the beginning has $QHR = 0.5$.

We performed all the experiments with two different networks. The first one (called "fixed" in the following) is completely created before resource discovery starts, it is made of $N = 10000$ nodes and does not change during the simulation. This network is grown without preferential attachment, with N_0 completely connected nodes, and each other joining peer choosing $m \in [1, N_0]$ existing peers, with even probability. All connections are bidirectional, *i.e.* if node n_a has node n_b in its peerview, then n_b has n_a in its peerview. The resulting degree distribution is exponential:

$$P(k) = (1 - e^{-\frac{1}{m}})e^{1-\frac{k}{m}} \forall k \geq m \qquad (6)$$

In these experiments, we used $m = 3$. The second network (called "noisy" in the following) is given by the modification of the fixed network, throughout the simulation, by node departures and arrivals regulated by the same Poisson process with $\lambda = 10^{-1}s{-}1$. Since arrivals and departures have the same rate, the average network size is still $N = 10000$. As shown in figure 2, the node degree distribution is the same of the fixed network.

We simulated about 56 h of the life of a network of peers, with resource queries scheduled according to a Poisson process with fixed rate $\lambda = 0.25$ s, *i.e.* one query every 4 seconds, associated to a randomly chosen node. Each query is a request for a randomly generated amount of resource (no more than 2 GHz of CPU, 1 GB of RAM, and 100 GB of disk space, respectively). Once a resource is found, it is consumed for a randomly distributed time interval (Poisson with 14 h average).

We firstly studied the performance of the system without adaptation, with peers having constant values for their genes. In details, we considered the three settings $C_i = 1, 3, 5$, and we measured the evolution over time of the average QHR, considering peers which have sent at least one query (fig. 3). In this case, the performance depends on how parameters are pre-fixed. Setting $C_i = 1$ for all peers is not sufficient for covering the network in search for resources, while setting $C_i = 5$ is not so much better than setting $C_i = 3$. But we must consider that for other networks, with different size, connection strategy and evolution, this result could not be true anymore. With respect to prefixed strategies, the autonomic approach makes the network adaptable to changing workloads.

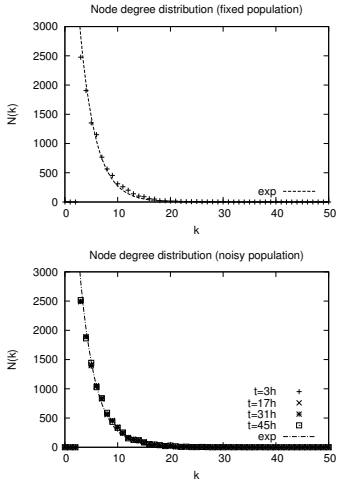

Figure 2. Node Degree Distribution for the Fixed (Left) and Noisy (Right) Networks.

We simulated periodic adaptation performed by all peers every 5000 s $\simeq$ 83 min. In this case the evolution over time of the average QHR (considering only those peers which have sent at least one query) is different, as shown in fig. 4 (fixed population) and 5 (noisy population).

Fig. 6 and 7 illustrate the dynamics of the genes, averaged *considering the whole network*, respectively with fixed and noisy peer population. Considering the whole network means that initially $\langle QHR \rangle = 0.5$ for all peers, and only after a while the average QHR of the whole network and the average QHR of the searchers converge, because all peers have performed at least one search.

In general, if QHR decreases for most peers, the system reacts in a way that chromosomes with high gene values survive and proliferate. When the network is fixed, all three fitness function make the genes C_i increase over time, in order to maximize the QHR. It is interesting that, when the network is noisy, the values of the genes converge (to 4 for F1, to 3 for F2, and to 2.5 for F3). The reason of the quick stabilization is that joining and leaving nodes (*i.e.* the network noise) introduce random chromosomes that have the same effect of mutation.

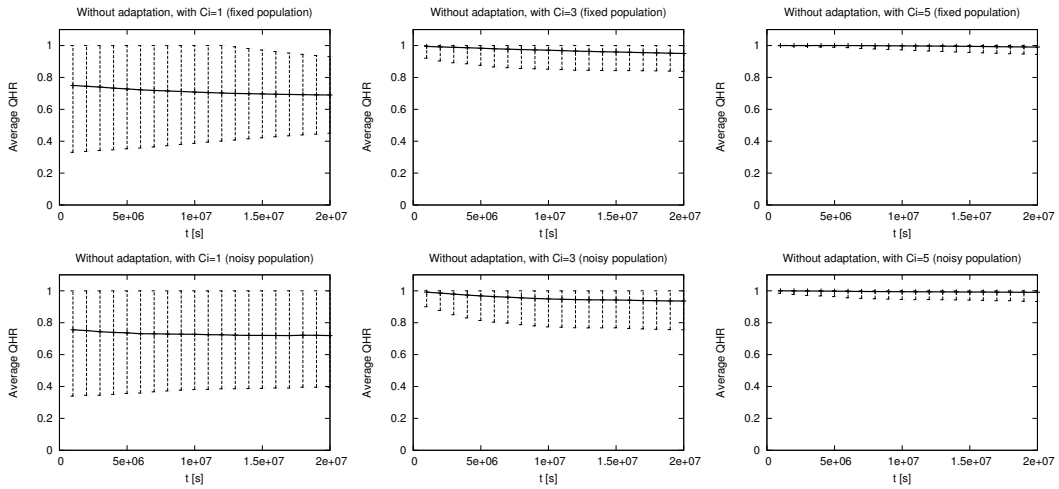

Figure 3. Average QHR over Time for Networks With Fixed (Top) and Noisy (Bottom) Population, for Different Configurations of Peers' Genes.

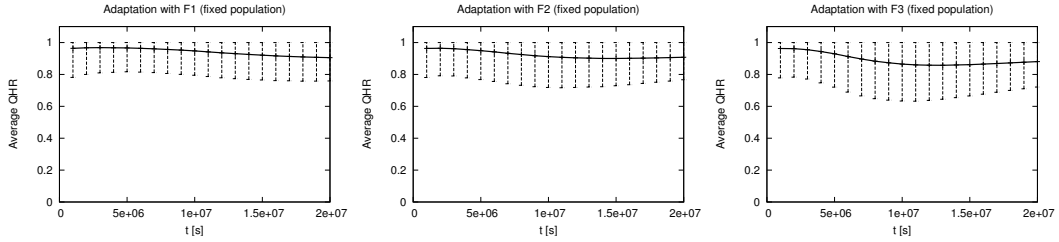

Figure 4. Average QHR over Time for Networks with Fixed Population, Comparing Fitness Functions F1, F2, F3.

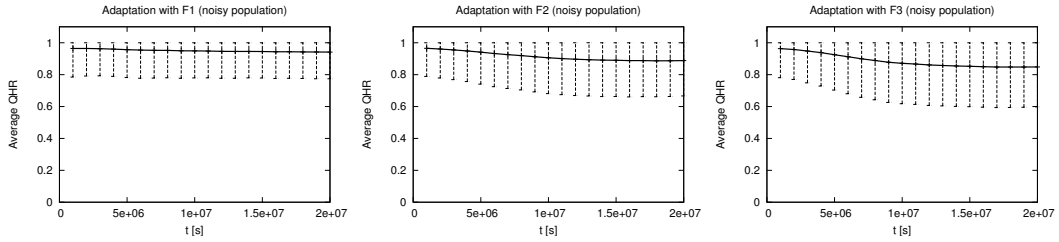

Figure 5. Average QHR over Time for Networks with Noisy Population, Comparing Fitness Functions F1, F2, F3.

4. RELATED WORK

An almost complete survey on peer-to-peer resource sharing models for Grid systems is proposed in a recent work of Trunfio *et al.* [12]. Among others, the approach proposed by Iamnitchi and Foster [7] is interesting because it compares four strategies for epidemic request propagation, including one based on epigenetic (*i.e.* learning-based) evolution of the internal structure of peers.

A constructive criticism to both epidemic algorithms and mechanisms based on spanning trees (Chord-like) is provided by Merz and Gorunova [9]. Being epidemic algorithms easy to implement, fault-tolerant, scalable, but slow, and spanning trees much more efficient but unresilient, the authors propose an hybrid approach. Epidemic dissemination is used to maintain the overlay, with an efficient mechanism for multicasting information. Simulation results show that the combination of the two methods is more efficient than the methods alone. Scalability up to 100000 peers is also shown. The authors defer to future work a detailed evaluation considering failures, high churn and presence of malicious peers.

The feasibility of a fully decentralized evolutionary algo-

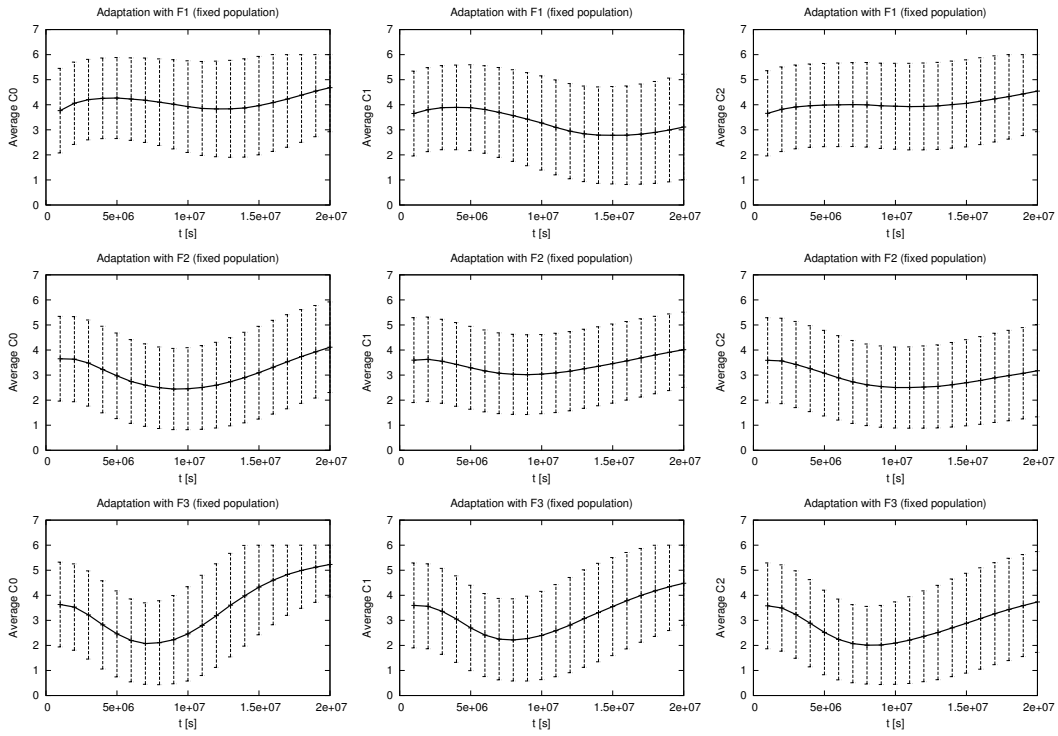

Figure 6. Evolution over Time of Genes C_0, C_1, C_2 for Networks with Fixed Population, Comparing Fitness Functions F1, F2, F3.

rithm, which is the basic idea on which AEF is designed, is demonstrated in a recent work of Wickramasinghe *et al.* [13]. The authors describe and evaluate a fully distributed P2P evolutionary algorithm where decisions regarding survival and reproduction are taken by the individuals themselves independently, without any central control. This allows for a fully distributed evolutionary algorithm, where not only reproduction (crossover and mutation) but also selection is performed at local level. An unwanted consequence of adding and removing individuals in a non-synchronized manner is that the population size gets out of control too. This problem is solved by adding an adaptation mechanism allowing individuals to regulate their own selection pressure. The key to this is a gossiping algorithm that enables individuals to maintain estimates on the size and the fitness of the population.

Kessler *et al.* [8] propose to use genetic algorithms to find optimal network structures, by means of offline simulations. Each individual population member (chromosome) encodes a network design with connected superpeers and clusters of leaf nodes forming superpeer groups. The fitness of a population member is directly related to how successful the particular network configuration is in answering resource queries. Our GA-based AEF architecture is more appealing, since re-configuration is an online process whose cost is distributed among peers.

In the future we would like to quantitatively compare these approaches with the scheme we illustrated in section , using the DEUS simulation tool.

5. CONCLUSIONS

To evaluate the usefulness of the Adaptive Evolutionary Framework (AEF) based on genetic algorithms, we applied it to a Grid Computing architecture where peer participants share their consumable resources. The results of our simulation experiments show that AEF enables large-scale autonomic distributed systems characterized by effective self-configuration and self-management mechanisms.

The next step of this research activity will be to define an AEF implementation that enables self-restructuring peers. Evolutionary programming could be the right strategy to achieve this objective. We envision self-restructuring peers to be the building blocks of highly innovative distributed systems, targeting different application fields ranging from high-performance computing to ambient intelligence.

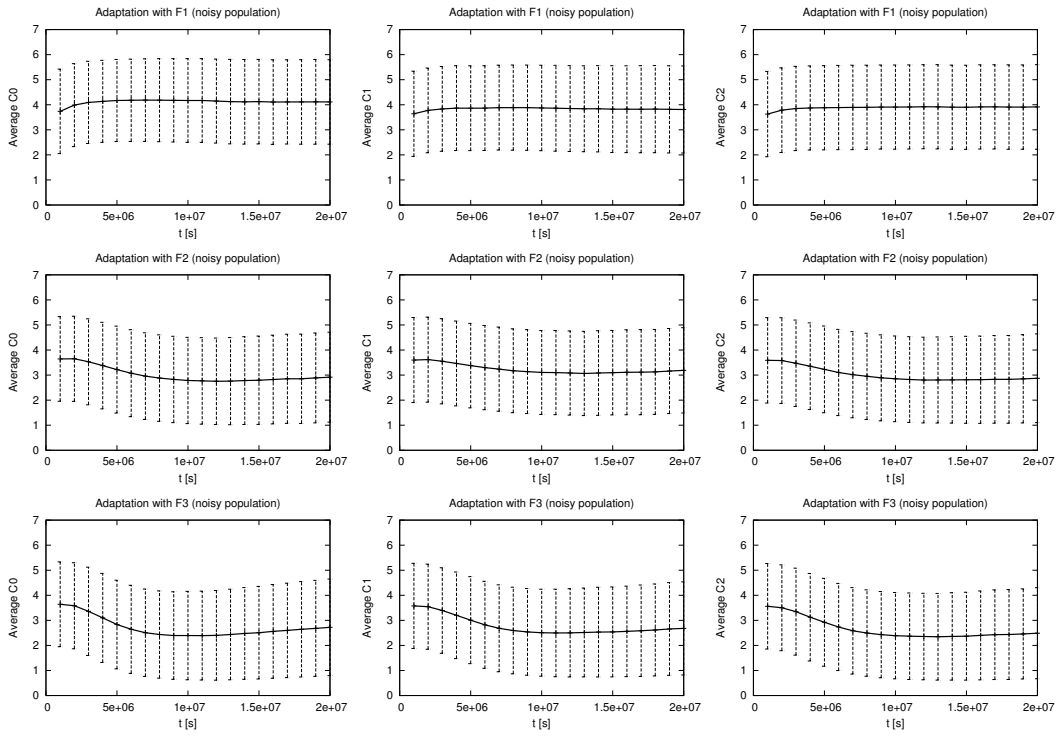

Figure 7. Evolution over Time of Genes C_0, C_1, C_2 for Networks with Noisy Population, Comparing the $F1$, $F2$, $F3$ Fitness Functions.

REFERENCES

[1] M. Amoretti, "A Framework for Evolutionary Peer-to-Peer Overlay Schemes", European Workshops on the Applications of Evolutionary Computation (EvoWorkshops 2009), Tubingen, Germany, April 2009.

[2] M. Amoretti, M. Agosti, F. Zanichelli, "DEUS: a Discrete Event Universal Simulator", Proc. of the 2nd ICST/ACM International Conference on Simulation Tools and Techniques (SIMUTools 2009), Roma, Italy, March 2009.

[3] A. E. Eiben, "Evolutionary Computing and Autonomic Computing: Shared Problems, Shared Solutions?", Self-Star Properties in Complex Information Systems, LNCS No. 3460, p.36-48, Springer, 2005.

[4] P.T. Eugster, R. Guerraoui, A.-M. Kermarrec, L. Massoulie, "From Epidemics to Distributed Computing", *IEEE Computer*, Vol. 37, No. 5, May 2004.

[5] The Gnutella Protocol Specification 0.4, *http://rfc-gnutella.sourceforge.net/developer/stable/index.html*

[6] J. Holland, ADAPTATION IN NATURAL AND ARTIFICIAL SYSTEMS, The MIT Press, 1992.

[7] A. Iamnitchi, I. Foster, "A Peer-to-Peer Approach to Resource Location in Grid Environments", In J. Weglarz, J. Nabrzyski, J. Schopf, and M. Stroinski (eds.), GRID RESOURCE MANAGEMENT, Kluwer Publishing, 2003.

[8] J. Kessler, K. Rasheed, I. Budak Arpinar, "Using genetic algorithms to reorganize superpeer structure in peer to peer networks", *Applied Intelligence*, Vol. 26, No. 1, pp. 35-52, 2007.

[9] P. Merz, K. Gorunova, "Fault-tolerant Resource Discovery in Peer-to-peer Grids", *Journal of Grid Computing*, Vol.5, No.3, September 2007.

[10] O. Ozkasap, M. Caglar, E. Cem, E. Ahi, E. Iskender, "Stepwise fair-share buffering for gossip-based peer-to-peer data dissemination", *Computer Networks*, Vol.53, No.133, 2009.

[11] J. Tang, W. Zhang, W. Xiao, D. Tang, and J. Song, "Self-Organizing Service-Oriented Peer Communities", International Conference on Internet and Web Applications and Services (ICIW 2006), Guadeloupe, French Caribbean, February 2006, pages 99-103.

[12] P. Trunfio, D. Talia, H. Papadakis, P. Fragopoulou, M. Mordacchini, M. Pennanen, K. Popov, V. Vlassov, and S. Haridi, "Peer-to-Peer resource discovery in Grids: Models and systems", *Future Generation Computer Systems*, vol. 23 (7), pp. 864-878 (2007)

[13] W. Wickramasinghe, M. van Steen, and A. E. Eiben, "Peer-to-peer evolutionary algorithms with adaptive autonomous selection", 9th Annual Conference on Genetic and Evolutionary Computation (GECCO 2007), pp. 1460-1467. ACMPress (2007)