

Memetic Networks: analyzing the effects of network properties in multi-agent performance

Ricardo M. Araujo and Luis C. Lamb

Institute of Informatics, Federal University of Rio Grande do Sul
Porto Alegre, 91501-970 - Brazil
rmaraujo@inf.ufrgs.br, LuisLamb@acm.org

Abstract

We explore the relationship between properties of the network defined by connected agents and the global system performance. This is achieved by means of a novel class of optimization algorithms. This new class makes explicit use of an underlying network that structures the information flow between multiple agents performing local searches. We show that this new class of algorithms is competitive with respect to other population-based optimization techniques. Finally, we demonstrate by numerical simulations that changes in the way the network is built leads to variations in the system's performance. In particular, we show how constrained hubs - highly connected agents - can be beneficial in particular optimization problems.

Introduction

Investigations on the connections between agents are increasingly gaining attention as networks become a central issue to understand large scale Multi-agent Systems (MAS). Such large MAS are of significant interest and are often used to model social and economic scenarios, including markets and business organizations (Gilbert & Conte 1995; Araújo & Lamb 2007; van Heck & Vervest 2007). These systems are typically composed of agents with very simple behavior. Due to their descriptive role, several research efforts have been directed towards relating agent's individual behavior to network properties that emerge from local interactions (Axtell 2000; Jin & Liu 2003; Shinoda, Matsuo, & Nakashima 2007).

It is well-known that MAS have been used to execute tasks, plans and solve problems in addition to modeling existing systems. However, far fewer studies have analyzed the relationship between MAS's network properties and system performance, e.g. (Dilkina, Gomes, & Sabharwal 2007). This is, *per se*, an important issue as such relation allows for the design of more efficient and effective MAS. One may then specify agents' behaviors that induce the desired network property and take advantage of results from graph theory and the so-called "new science of networks" (Newman, Barabási, & Watts 2006) to improve MAS effectiveness.

To a certain extent, the paucity of studies following these lines can be explained for there are few MAS that make an explicit, controllable use of networks and that have an objective way of measuring system's performance. We shall introduce a new class of optimization algorithms composed of a population of autonomous agents that exchange information by means of an underlying network. We call this new approach *Memetic Networks* and we use an instance of such class to show how network properties can influence the algorithm's performance.

Our results contribute to answering the question "to whom should agents connect to?". In particular, we shall show how limiting the existence of *hubs* in the network can be beneficial for certain classes of problems (but not for others). The results presented here are relevant in the design of MAS where connections between agents are a central design issue, such as peer-to-peer networks (Udupi, Yolum, & Singh 2004), autonomous systems, and sensor networks (Akyildiz *et al.* 2002).

The paper is structured as follows. Firstly, we introduce the basics of Memetic Networks. We then propose an implementation for a Memetic Network Algorithm that borrows the "preferential attachment" mechanism (Barabasi 2003) to construct scale-free networks and compare its performance to standard optimization techniques. We use this implementation to show how changes in network properties can have an impact in system performance. Finally, we conclude and point out directions for future research.

On Memetic Networks

In order to study the relationship between networks and MAS performance, we introduce a class of population-based optimization algorithms in which networks are part of the problem-solving process. Optimization tasks are suitable for our purposes since they present a well-defined performance measure and many problems can be modeled as optimization problems (Russell & Norvig 2002). An algorithm in this class makes use of independent searches (each represented by an agent) that communicate with each other through connections; the set of all connections and agents compose a network. We call this class *Memetic Networks* inspired in the concept of *memes*: information that is propagated through copies in a cultural setting (Dawkins 1976).

A *Memetic Network Algorithm* (MNA) is a population-

based stochastic optimization algorithm. It is composed of an ordered set of N agents (n_1, n_2, \dots, n_N), each encoding a complete solution for the optimization problem, a binary $N \times N$ matrix representing possible connections between individuals and a set of rules specifying how the matrix is formed and how interaction between agents take place. Thus an MNA's structure is a directed unlabeled graph whose nodes represent solutions to an optimization problem (agents) and edges represent connections between these solutions. The rules are described as follows:

- **Connection rule.** Specify how agents will connect to and disconnect from each other. This rule guides the construction of the network structure. For example, an instance of such rule could be “a connection between nodes n_1 and n_2 exists if and only if n_1 is better evaluated than n_2 ” or “connect randomly to individuals”. The connection rule is executed at every step of the algorithm, thus the network is dynamic and connections may change at each step. It defines the dynamics of the network.
- **Aggregation rule.** Given a connection, this rule specifies how information is to flow through it. It guides how the solution contained in each node is to be modified as a function of the connected nodes. For example, if each agent encodes a real number, the average of all numbers could be used to aggregate information. It defines the dynamics on the network.
- **Appropriation rule.** After information has been transmitted, this rule specifies any local changes to the information contained in a node. This could be the application of a hill-climbing search, for instance.

Therefore, the proposed algorithm makes use of a network during the optimization process to explicitly represent the exchange of information between several parallel searches. By defining these rules, several types of networks can be created. For example, by setting the connection rule to connect to random individuals, a random graph emerges. It is interesting to observe that this setup allows for a single node to be connected to several other nodes, thus a possible solution can influence and be influenced by several other solutions during a search. In an MNA, an unspecified and dynamic number of solutions may contribute to create a new solution, as defined by the number of connections (out-degree) of a node. By doing so, a more explicit and wider use of the multiple parallel searches is carried out.

Population-based parallel optimization techniques are commonly used to avoid being trapped in local optima during a search (Torn & Zilinskas 1989). When multiple searches interact, this process often improves the global performance by allowing efforts to focus on promising areas of the search space (Torn & Zilinskas 1989; Russell & Norvig 2002). Genetic Algorithms, for instance, use genetic crossovers to combine independent solutions from pairs of individuals and although a network is implicit (Oner, Garibay, & Wu 2006), it is not easily modifiable. Memetic Networks generalize this concept by allowing multiple combinations between multiple individuals within a network that is explicit and controllable. The key idea of such general

concept is to allow solutions (nodes) to receive information from several other solutions being evaluated in parallel in a structured way, so that the information flow is made explicit within a network. In this way, one can adjust the diverse rules to control how information is distributed through nodes in a general way, allowing for relating specific network properties and dynamics to specific problems to be solved. An instance of an MNA is specified by an algorithm implementing each of the above described rules.

In a certain way, Memetic Networks can be seen as a model of “cultural evolution”, in contrast to genetic evolution, the underlying inspiration of Evolutionary Computation. Dawkins (1976) has argued that cultural evolution proceeds at a much faster pace than genetic evolution: technology, social trends and habits all seem to develop very quickly and are guided by a very similar process to that of genetic evolution. To account for this similarity, Dawkins has coined the term *meme*, taken to be the cultural equivalent of a gene. A meme can be anything that can be passed from one human mind to another: business ideas, catch-phrases, ideologies, algorithms. Whenever an individual acquires a meme, he or she can pass the meme on to other individuals. Errors in transmission and deliberate changes made to a meme by any individual account for the possibility of new memes to emerge, in a similar fashion as mutations in genes allow for new variations to emerge (Dawkins 1976).

Despite these similarities, meme evolution differs in important ways from gene evolution and such differences could be accounted for the observed faster evolution rate. One such difference is that memes can pass from one individual to several others without having to “hop” from individual to individual. A teacher can pass a meme on to many students at once; a TV channel transmits memes to millions of viewers. The converse is also true and a single meme can be the result of the contribution of several individuals: a researcher, for example, reads several papers containing diverse ideas, each (partly) contributing to her ongoing research. These same aspects are present in an MNA. Note that the term *Memetic Algorithm* is commonly used to describe a type of genetic algorithms, in which individuals are able to make local optimizations (Moscato 1999). However, these algorithms have little relation to our proposal.

The Scale-free Memetic Network Algorithm

One central issue in MNAs is the connection rule, since it will ultimately define the network topology. The definition of such rule answers the question “with whom should an agent interact with”. Since we are concerned with optimization tasks, it is interesting to allow for agents to interact with peers that possess useful information that will allow for them to reach a better solution. However, in general the utility of any piece of information is not obvious or directly measurable and one may choose an indirect measure of such utility. We assume that it is possible to evaluate the solution encoded in any agent and a total order can be applied to such evaluations (they may be real numbers, for instance). A reasonable general rule could then be “connect to agents that are better than you”, since well-evaluated agents are likely to have good information that may be useful.

It is not difficult to observe that an algorithm based on this rule will often induce *hubs* - agents that have a higher than average number of connections. This is likely to happen because all agents will connect to a few best evaluated agents, while many badly evaluated ones will have a few (if any) incoming connections, but many outgoing connections. Indeed, this rule is a form of *preferential attachment* that characterize *scale-free networks* (Barabasi 2003; Newman, Barabási, & Watts 2006). A scale-free network is characterized by a fat-tailed power law distribution of node degrees, which means that there is a higher probability of having nodes with many connections than a normal distribution would predict. Hubs, being highly connected, are part of many paths in the network and act as distribution centers. For instance, the infection of hubs (by some disease) in a society is associated with the emergence of epidemics (Moore & Newman 2000).

In what follows we propose the “Scale-Free Memetic Network Algorithm” by defining the algorithm’s three rules, as well as a representation for the solutions. We borrow the binary representation usually used in genetic algorithms (Goldberg 1989; Fogel 2000), encoding each solution as a binary string of size k . We define $eval(x)$ as the evaluation of a solution x and we want to find an x^* such as $eval(x^*)$ is minimized (i.e. we want to minimize the function $eval$). We define the rules as follows:

- **Connection Rule:** node n_1 connects to node n_2 if and only if $eval(n_2) < eval(n_1)$; the connection is unidirectional and flows from n_2 to n_1 .
- **Aggregation Rule:** given the set $C(n_1)$ containing all nodes that n_1 connects to, the node n_1 is modified to represent the *majority vote* for each bit of the solutions in $C(n_1)$;
- **Appropriation Rule:** each bit of the encoded solution is flipped with probability p_n .

It is useful to detail the aggregation rule. For each bit of the solution encoded in n_1 , we take a majority vote among all nodes in $C(n_1)$, that is, all nodes that n_1 is connected to. Thus, each bit is taken to be the most common bit among all connected nodes (ties are decided by coin toss). By the connection rule, nodes connect to other nodes that are *better* than them, thus voting takes place among nodes that are better evaluated than the node in question.

The connection rule chosen induces a preferential attachment to network connections. Nodes with good performance will have many incoming connections and will have more influence over the network (i.e. they are able to reach and communicate with more nodes). On the other hand, bad performing nodes will have a higher number of outcome connections, since they will connect to every node that is better than them, thus making them susceptible to the influence of many (better) nodes. The appropriation rule is necessary in order to guarantee that the whole search space is available to the algorithm, since random initializations could cause an early stagnation of the search. This plays the role of mutation in evolutionary algorithms.

The algorithm is depicted in Algorithm 1 where: N is the number of nodes in the underlying graph and it defines the

maximum number of solutions being evaluated at one time; p_n is the probability of noise; k is the size in bits of the binary strings that encode solutions. The function $coin()$ returns 0 or 1 with equal probability.

The complexity of this algorithm depends on how dense the network is. For fully connected networks, it is $O(kN^2)$ if the evaluation of the function can be considered an elementary operation. Thus, the algorithm can add a considerable overhead to the search. However, for complex real-world problems the function evaluation can easily become the most dominant part of the algorithm, greatly reducing the relative overhead.

Algorithm 1 Scale-Free Memetic Network Algorithm

```

Initialize  $N$  random binary strings of size  $k$ 
while termination condition not met do
  evaluate solutions in nodes
  for  $i = 1$  to  $N$  do
    for  $j = 1$  to  $N$  do
      if  $eval(node(j)) < eval(node(i))$  then
        connect( $node(i)$ ,  $node(j)$ )
      else
        disconnect( $node(i)$ ,  $node(j)$ )
      end if
    end for
  end for
  for  $i = 1$  to  $N$  do
     $C =$  set of nodes  $node(i)$  is connected to
    for each bit  $b \in node(i)$  do
       $zeros =$  elements in  $C$  with bit  $k = 0$ 
       $ones =$  elements in  $C$  with bit  $k = 1$ 
      if  $|zeros| < |ones|$  then
         $b = 1$ 
      end if
      if  $|zeros| > |ones|$  then
         $b = 0$ 
      end if
      if  $|zeros| == |ones|$  then
         $b = coin()$ 
      end if
      if  $random(0,1) < p_n$  then
         $b = coin()$ 
      end if
    end for
  end for
end while

```

Performance Tests and Analysis

In order to validate the algorithm, we have implemented, analyzed and tested it over some functions commonly used as benchmarks, taken from (De Jong 1975; Gordon & Whitley 1993). These functions have well-known properties that can be used to better understand under which circumstances the algorithm is able to perform adequately. Figure 1 enumerates the testbed functions, which cover combinations of features regarding *modality* and *separability*. A function is *unimodal* if it has only one minimum and multimodal otherwise. When it is possible to find a global minimum by optimizing the input variables separately, then the function is said to be *separable*. Function $f1$ is known as the *sphere* and

is a simple unimodal and separable function. Function $f2$ is known as the *Rosenbrock Valley* and, while still unimodal, it is not separable. Function $f3$ (Rastrigin) is multimodal but separable and function $f4$ (Ackley) is both multimodal and non-separable.

For all cases, we use the same parameters for our algorithm: $N = 49$, $p_n = 0.06$, $k = 24$. The parameter p_n was adjusted manually and the chosen value was observed to show good performance across the test set. Variables are represented by binary fixed-point representation.

$$f1(x_1, x_2) = x_1^2 + x_2^2$$

$$f2(x_1, x_2) = 100(x_2 - x_1^2)^2 + (x_1 - 1)^2$$

$$f3(x_1, x_2) = 20 + x_1^2 + x_2^2 - 10\cos(2\pi x_1) - 10\cos(2\pi x_2)$$

$$f4(x_1, x_2) = 20 + e - 20e^{-0.2\sqrt{0.5(x_1^2+x_2^2)}} - e^{0.5(\cos(2\pi x_1)+\cos(2\pi x_2))}$$

Figure 1: Test set

We compare the proposed algorithm to three other algorithms over the same test set: random sampling, genetic algorithm (Goldberg 1989) and simulated annealing (Kirkpatrick, Gelatt, & Vecchi 1983). Random sampling works by randomly sampling solutions and keeping the best one and it is used only as a “straw man”, providing a lower bound in performance analysis. In order to allow for a fair comparison, we let the algorithm sample 50 solutions at each round. The genetic algorithm uses a single-point crossover, tournament elitist selection and commonly used parameters (probability of crossover of 0.7 and probability of mutation of 0.05) with 50 individuals. Simulated Annealing is also allowed to sample 50 independent solutions at each round and uses a proportional cooling schedule with $\alpha = 0.9$. The initial temperature was set *ad hoc* according to each function.

Figure 2 depicts convergence results for each algorithm. Each plot is the result of an average over 100 independent runs. The first result to be noticed in these figures is that our algorithm is able to optimize fairly well, being competitive with all tested algorithms in all but one test function ($f2$) where Simulated Annealing performed better.

Our algorithm presents a slower convergence rate when compared to the Genetic Algorithm, but more often it finds better solutions after a full run. It must be noted, however, that none of the algorithms were fully optimized and such results are used to show that our proposed approach is indeed able to optimize and be competitive with general optimization methods. It is also important to notice that while we have used a binary string representation in our implemented algorithm, this is not mandatory. For instance, one can use real numbers to (directly) represent the variables and use an aggregation rule that would calculate the average over the information contained in all connected nodes.

Hubs and Network Performance

We have argued that our implementation of a memetic network algorithm induces hubs by allowing nodes to connect

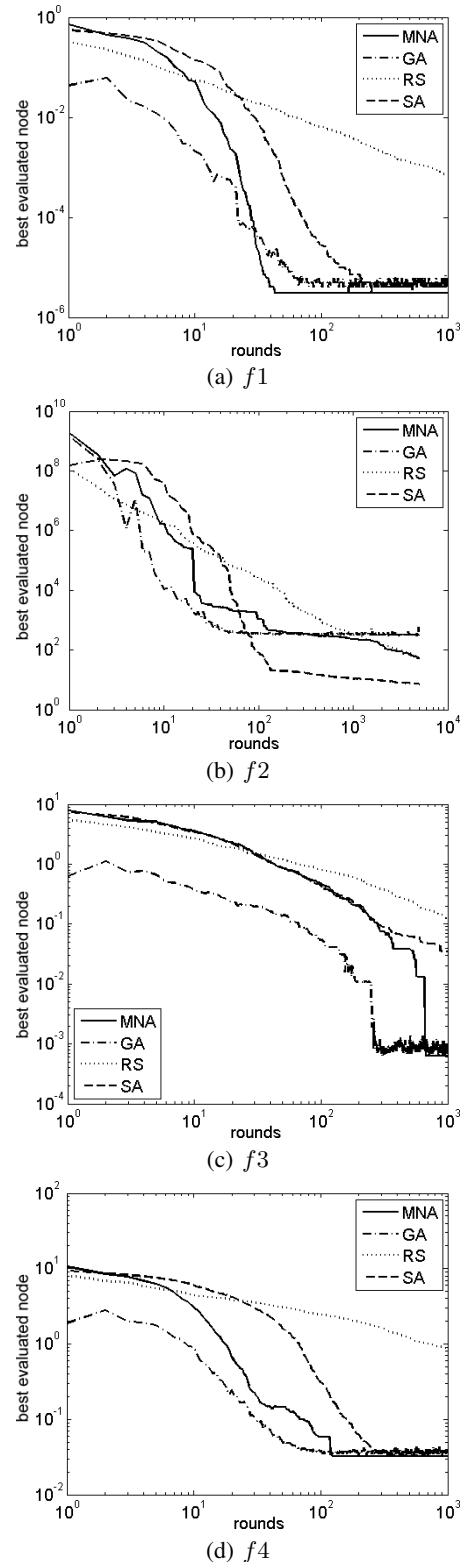


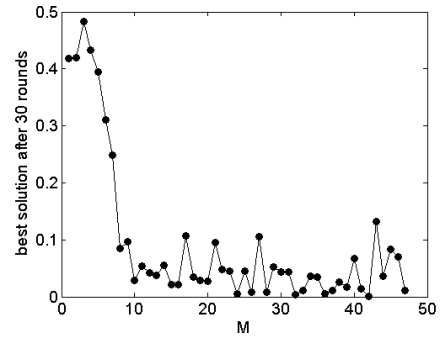
Figure 2: Convergence results for Memetic Network Algorithm (MNA), Simulated Annealing (SA), Genetic Algorithm (GA) and Random Sampling (RS).

to other better evaluated nodes. There are actually two types of hubs that emerge in such network. The first one, which we call *sources*, are the best evaluated nodes in the network and thus have many other nodes connecting to it. Sources play an important role in the network, since they distribute the same information to several nodes. The second one, named *sinks*, are the worst evaluated nodes that connect to many other better nodes. Sinks receive information from many other nodes and are more likely to explore the search space by combining parts that can lead to better solutions. Only if a better solution is found they will be able to provide information to other nodes. Sources, on one hand, are desirable - we want to have good information available to all nodes in the network, so they can exploit it. On the other hand, having the same information being distributed to too many nodes may cause all nodes to explore the same area of the search space, possibly trapping the algorithm in a local optima. The existence of sources in our algorithm represents the common exploration-exploitation problem (Fogel 2000) in a network context.

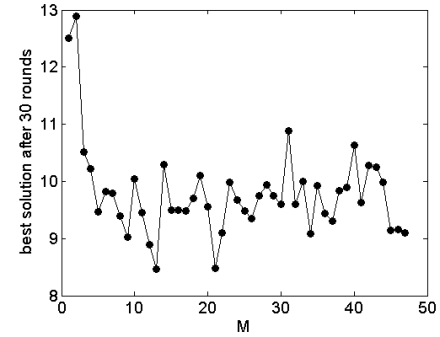
To better understand the role of sources in the algorithm performance, we establish a control mechanism to limit how large hubs can become. We do so by setting a maximum in-degree number to nodes - i.e. a maximum number of input connections. By varying this number, we can limit the influence sources have on the network. We have modified our algorithm in the following way. Let A be the set of nodes that wish to connect to node v ; from this set, select M nodes randomly and allow them to connect to v ; all other nodes are not allowed to connect. Previously connected nodes are allowed to keep the connection for as long as they want and whenever a connection becomes available (if the connected node becomes better than the source), a new random selection is performed. The value of M varies from 1 (nodes communicating with at most one other node) to $N - 1$ (the total number of other nodes in the network). We have run experiments in which this number varies along this range for all four test functions. Figure 3 shows the results.

We can observe that for the functions with only a single optimum value ($f1$ and $f2$), small hubs are detrimental to the system's performance. By increasing the size of the hubs we improve performance but such gains are limited - after some value of M the curve damps and no further gains are observed. On the other hand, for multimodal functions there is an optimum range for M itself - as for unimodal functions, very small hubs correlate with poor performance, but now having excessively large hubs is also detrimental to the system's performance. The best performance is found in between these extreme values. It must be noted that it is not clear that there is an optimal value for M , but rather a range of values perform equally well. In other words, the algorithm is reasonably robust to changes in M , but extreme cases cause a decrease in performance.

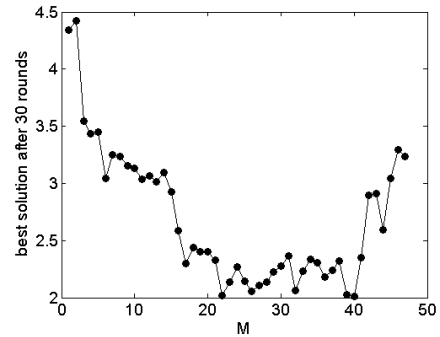
One possible explanation for such behavior is that, for unimodal functions, good local information are globally good and nodes always benefit from having access to them. Exploitation is more important than exploration in these cases, since exploiting good information provides guidance about where to further explore. For multimodal functions,



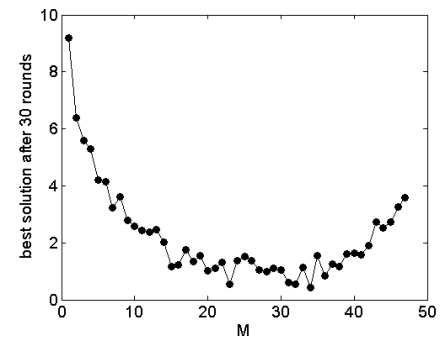
(a) $f1$



(b) $f2$



(c) $f3$



(d) $f4$

Figure 3: Average best solution after 30 rounds and different values of M .

however, locally good information may not translate into globally good information. Exploiting information about local optima may take attention away from the global optimum. In these cases, having hubs that are near local optima will stagnate the algorithm by having most nodes exploring such local optima. Thus having bigger hubs end up reducing the algorithm's performance. A trade-off between exploitation and exploration is necessary and this is reflected in the size of the hubs.

The precise value range for M is very likely to be problem-dependent and also dependent on other parameter of the algorithm. For instance, higher noise rates may allow for higher values of M to be acceptable. Nonetheless, our experiments clearly show how a network property (the number of in-degree connections) can influence the algorithm performance and how such properties may reflect characteristics of the problem being solved. We have also experimented with changing the out-degree limit of nodes, thus limiting how large sinks can become. By changing this limit, we are limiting the number of nodes sinks can connect to. For our four test functions, no differences could be observed in performance for all values of M . This is evidence that the way worst performing nodes combine information from other nodes plays a small role in the way how good solutions are constructed.

Conclusions and Future Work

We have investigated how a system of communicating agents that make use of a network to structure information flow is able to perform optimization. A class of optimization algorithms was introduced which models communicating agents searching for a single solution, namely Memetic Networks. Memetic Networks encode an underlying network that can be explicitly controlled or that is the result of an emergent decision-making process of multiple agents.

For a proposed instance of this class, based on scale-free networks, we have shown how it can be competitive with other commonly used optimization techniques. This encourages further research on the model as a general-purpose optimization tool. Nonetheless, we have focused our study on discovering some properties of one simple instance of a Memetic Network. It has become clear from the results presented that network properties can have an impact on system performance - we showed how the presence of hubs can be beneficial for certain problems but not for others. Not all network properties have such an impact. For instance, we have recently experimented with small-world properties and initial findings show that the algorithm performance is insensitive to changes in how many "shortcuts" are created in a regular grid connecting agents. We conclude that agents in network-oriented MAS may require attention during the design stage in order to induce networks that can improve overall system's performance.

Several paths for future research can be identified and are currently being pursued. While we have focused on scale-free networks, as they have become an important and almost ubiquitous network topology, other topologies are worth studying in depth such as small-worlds, hierarchical and sparsely-connected topologies. The application of the

algorithms to benchmark scenarios and real-world problems is also required in order to better understand how the algorithm scales up to more complex problems.

Acknowledgments

This work has been partly supported by CNPq and FAPERGS.

References

- Akyildiz, I.; Su, W.; Sankarasubramaniam, Y.; and Cayirci, E. 2002. A survey on sensor networks. *IEEE Comm. Mag.* 40(8):102–114.
- Araújo, R., and Lamb, L. 2007. An information-theoretic analysis of memory bounds in a distributed resource allocation mechanism. In *Proc. IJCAI-07*.
- Axtell, R. 2000. Effects of interaction topology and activation regime in several multi-agent systems. In *Proc. 2nd Intl. Workshop on Multi-Agent-Based Simulation*, 33–48.
- Barabasi, A.-L. 2003. *Linked: the new science of networks*. Plume / Penguin Books.
- Dawkins, R. 1976. *The Selfish Gene*. Oxford University Press.
- De Jong, K. 1975. *An analysis of the behavior of a class of genetic adaptive systems*. Ph.D. Dissertation, University of Michigan.
- Dilkina, B.; Gomes, C. P.; and Sabharwal, A. 2007. The impact of network topology on pure Nash equilibria in graphical games. In *Proc. AAAI-07*.
- Fogel, D. 2000. *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*. New York: IEEE.
- Gilbert, N., and Conte, R. 1995. *Artificial Societies: the computer simulation of social life*. London: UCL Press.
- Goldberg, D. E. 1989. *Genetic Algorithms in Search, Optimization and Machine Learning*. Boston: Addison-Wesley.
- Gordon, V. S., and Whitley, D. 1993. Serial and parallel genetic algorithms as function optimizers. In *Proc. of the 5th International Conference on Genetic Algorithms*, 177–183.
- Jin, X., and Liu, J. 2003. Agent network topology and complexity. In *Proc. AAMAS'03*.
- Kirkpatrick, S.; Gelatt, C. D.; and Vecchi, M. P. 1983. Optimization by simulated annealing. *Science* 220(4598):671–680.
- Moore, C., and Newman, M. 2000. Epidemics and percolation in small-world networks. *Phys. Rev. E* 61:5678–5682.
- Moscato, P. 1999. Memetic algorithms: A short introduction. In *New Ideas in Optimization*. McGraw-Hill.
- Newman, M.; Barabási, A.-L.; and Watts, D., eds. 2006. *The Structure and Dynamics of Networks*. Princeton University Press.
- Oner, M.; Garibay, I.; and Wu, A. 2006. Mating networks in steady state genetic algorithms are scale free. In *GECCO '06*, 1423–1424. ACM.
- Russell, S., and Norvig, P. 2002. *Artificial Intelligence: A Modern Approach*. Prentice Hall.
- Shinoda, K.; Matsuo, Y.; and Nakashima, H. 2007. Emergence of global network property based on multi-agent voting model. In *Proc. AAMAS'07*.
- Torn, A., and Zilinskas, A. 1989. *Global Optimization*. Lecture Notes in Computer Science. Springer-Verlag.
- Udupi, Y.; Yolum, P.; and Singh, M. 2004. Agent-based peer-to-peer service networks: A study of effectiveness and structure evolution. In *Proc. AAMAS'04*, 1336–1337.
- van Heck, E., and Vervest, P. 2007. Smart business networks: how the network wins. *Commun. ACM* 50(6):28–37.