Peer-to-peer Evolutionary Algorithms with Adaptive Autonomous Selection

W. R. M. U. K Wickramasinghe
Vrije Universiteit, Amsterdam
The Netherlands
rmwickra@few.vu.nl

M. van Steen
Vrije Universiteit, Amsterdam
The Netherlands
steen@cs.vu.nl

A. E. Eiben
Vrije Universiteit, Amsterdam
The Netherlands
gusz@cs.vu.nl

ABSTRACT

In this paper we describe and evaluate a fully distributed P2P evolutionary algorithm (EA) with adaptive autonomous selection. Autonomous selection means that decisions regarding survival and reproduction are taken by the individuals themselves independently, without any central control. This allows for a fully distributed EA, 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 resolved 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. The algorithm is experimentally evaluated on a test problem to show the viability of the idea and to gain insight into the run-time dynamics of such an algorithm. The results convincingly demonstrate the feasibility of a fully decentralized EA in which the population size can be kept stable.

Categories and Subject Descriptors

D.1.3 [**Software**]: Concurrent Programming – Distributed programming; I.2.8 [**Artificial Intelligence**]: Problem Solving, Search

General Terms

Algorithms, Experimentation

Keywords

distributed EA, autonomous selection, parameter adaptation, gossiping, newscast protocol

1. INTRODUCTION

Evolutionary algorithms (EAs) have gained a long-standing history of successfully solving computationally hard problems. Their popularity can be partly attributed to the principal simplicity of the structure of evolutionary algorithms, the transferability of code (representations and operators) between application areas, and of course their good performance [8]. A characteristic element of the algorithmic structure is that data and computations are clearly separated: a common data space is used to store a population of candidate solutions (individuals, genotypes), whereas potentially a large number of threads of computation (selection and variation operators) act concurrently on this data space.

From the beginning, efforts have been taken to distribute the population in such a way that concurrent threads could operate as independently as possible, effectively aiming at maximizing the attainable degree of parallelism in the evolutionary algorithm as a whole [2, 3, 22]. Key to this approach is constructing groups of individuals such that local decisions, i.e., intra-group decisions, can be taken in a fully autonomous fashion, with at the very extreme having only one individual per group. To this end, it is important that variation operators are local by nature, but selection operators are not. As for variation, mutation and crossover operators involve only one (two) individual(s), and mutation (crossover) can be executed on many individuals independently from each other. In contrast, selection in an EA (parent selection and survivor selection) typically involves a comparison of an individual with all others in the population, as in fitness proportional and ranked-based selection. Tournament selection involves only few other individuals, but the pool for the tournament is composed by a central oracle applying a (random) selection mechanism to the whole population.

For designing fully distributed evolutionary algorithms selection mechanisms are required that can work in a fully decentralized way such that threads can operate asynchronously and independently from each other. This type of evolutionary algorithm is becoming increasingly important with the rising need for decentralized decision-making in fields where an evolutionary approach has proven to be successful. Examples include data allocation in large-scale (collaborative) content distribution networks [20], as well as optimal distributed scheduling in BitTorrent systems (see, e.g., [10]).

In this paper we describe and experimentally evaluate a selection method, *autonomous selection*, that meets this requirement. Our mechanism is based on:

1. Locally available global information. In particular, statistical information about the population's fitness, available at each individual (e.g. average fitness).
2. A locally executable function that determines selection probabilities for each individual based on its own fitness and the available global information.

Additionally, we have:

3. An adaptation method regulating the parameters of the selection mechanism on-the-fly, depending on the course of the search.

Technically, we work with a P2P system, with no centralized components, where each individual corresponds to a node and each node has a collection of neighbors, cf. Figure 4.

There are two main research challenges we address. First, we need to ensure that the population size remains within bounds. By the decentralized reproduction the population can explode of implode (die out) and controlling this has proven to be a difficult problem [18]. Second, we investigate whether a gossip-based dissemination protocol can be effectively used to allow for local-only decision making by nodes, i.e., individuals in the population. In this paper we show a solution for both problems.

Our key contribution is that we demonstrate how fully autonomous decision making in evolutionary computing can be achieved by combining standard evolutionary algorithms with decentralized aggregation of global statistics through gossiping. In principle, our approach will allow the development of solutions that can operate on decentralized networks, notably peer-to-peer overlays, thus opening the road to massively, large-scale evolutionary computations in completely asynchronous environments.

2. RELATED WORK

The research behind this paper is a new idea in both fields of evolutionary computing and peer-to-peer (P2P) computing. It is important to note that most of the related research work referenced here is based on centralized algorithms.

2.1 Related work in evolutionary computing

Evolutionary computing is mainly used in the form of centralized algorithms running on single-processor computers. In decentralized algorithms, we generally see the members of populations being spatially organized in the form of graphs [22]. However, even in these cases the execution of the actual algorithm takes place in a centralized fashion. Parallel versions of evolutionary algorithms are described in [3, 4, 1, 19]. The main goal in these cases is to simply improve efficiency by exploiting parallelism in the evolutionary computations. Due to the fact that virtually all algorithms make use of a shared data space, success has been mainly limited to shared-memory parallel processors, although combining spatial structures and parallelism has also proven to be a promising approach [1]. The work described in this paper essentially follows this last approach as well.

Local selection algorithms for distributed models of evolutionary algorithms have been given some attention in the last decade [5, 11, 21, 22, 19, 9]. Recent work [7] concerns devising a locally executable function to determine selection probabilities for each individual. Parent selection and survivor selection are separated and handled independently, but selection probabilities in both cases are determined by a sigmoid function. This function has two parameters, m and s, that determine the properties of the selection mechanism. Using this function an individual first determines if it should live or die and if it survives it also checks if it is good enough to create offspring. If it proves good enough it will mate a randomly chosen other individual and the new offspring can be added to the population without replacing any old individual. Note that by this latter property the population can shrink or grow. This poses a new challenge to the EA designer, because population explosion and implosion should be prevented by calibrating the parameters m and s. Previous work considered a system with perfectly informed individuals that received the exact population statistics from an "oracle" and provided proof-of-principle evidence that the autonomous selection idea is viable [7]. However, tuning m and s required substantial efforts. The work described in this paper exceeds previous research in two aspects.

- We remove the central oracle and use a gossiping protocol to acquire global statistics locally.
- We introduce an adaptation mechanism to calibrate m and s on-the-fly, depending on how the search proceeds.

In this system an individual can exchange information only with its neighbors. The individual can ask a neighbor for its estimations of the population's average fitness and the population size. With this information it can make (adjust) its own estimations for the population size and average fitness. Then according to this information an individual would make decisions for the selection process. This method is the main feature of our proposed decentralized algorithm.

Another angle to position the present work is that of parameter control in evolutionary algorithms. Over the last decades significant efforts have been devoted to regulate EA parameters either deterministically, adaptively, or self-adaptively [6]. The traditional efforts mainly concern variation operators. In this paper we introduce a novel way to control selection by an adaptive mechanism.

2.2 Related research in P2P computing

Our research is based on results from gossip algorithms [15, 12, 13, 14, 17] and decentralized peer sampling [12, 14].

Gossip-based (or epidemic based) algorithms have the inherent ability to reliably pass information among a large set of interconnected nodes. They are robust even if the nodes regularly join and leave the system (either purposefully or on account of failures), or the underlying network suffers from broken or slow links. In a gossip-based protocol, each node in the system periodically exchanges information with a subset of its peers. The choice of this subset is crucial to the wide dissemination of the gossip. This exchange of information is in the form of either push or pull. In a push methodology a node will inspect to see if one of its peers (neighbors) has data and if not send (or push) the current data to the peer. In a pull-based system it is the reverse, where a node will get information from its peers. A combination of push and pull has proved to operate best [17].

The gossiping protocol that was used in our research was the Newscast protocol [16]. Although this is a simple protocol it is very efficient in networks where nodes join and leave the network continuously (churn). Using the Newscast protocol the peer-sampling service has been defined. This fully decentralized service provides a node a uniform randomly selected set of peers, which can be used to exchange information with.

In this setting once we have our individuals of the evolutionary algorithm running on nodes of a P2P system, they exchange information by gossiping. This gossiping is crucial for the execution of our algorithm, because it defines a methodology to make estimations of the population size and average fitness. An individual can also use the peer-sampling service to locate a mate.

3. SYSTEM DESCRIPTION

The system we propose resembles a staged, layered protocol architecture. This allows us to divide the functionalities of the evolutionary algorithm and the P2P networking components. We used PeerSim [16] as the P2P computing simulator for our experiments.

3.1 Layered algorithm approach

Each node of the P2P system runs our algorithm, which is divided into three stages organized in a layered fashion. The core is formed by the evolutionary stage during which selection and variation operators (mutation and crossover) are executed locally. The execution of this algorithm is interrupted to allow for the adaptation of selection mechanism (that decides on the probabilities for selecting parents and survivors on this node). This adaptation concerns properly setting the values s and m of Equation 1. To this end, the adaptation stage itself is interrupted for a gossiping stage during which (1) estimates of the network size and global average fitness are computed, and (2) the neighbor set of each node is randomized. When the gossiping stage finishes, the adaptation stage can complete its work, in turn allowing the completion of the evolutionary stage. Execution of these three stages is iterated until a solution is found.

3.2 Algorithmic details

Our algorithm runs on each node of the P2P system, where each node represents a candidate solution to the given problem. There are two main algorithmic parts: the evolutionary stage and the adaptation stage, see Algorithm 1. The termination of the algorithm is managed in a distributed fashion too: each node decides to terminate when it reaches the optimal fitness value or it "hears" that other nodes have done so.

Algorithm 1 Outline of the adaptive distributed evolutionary algorithm

```
initialize
repeat
  if a solution is found then
    inform the neighbors (by gossiping)
  end if
  if adaptation stage then
    exchange information by gossiping
    estimate the population size and average fitness
    update selection parameters by adaptation
  end if
  if evolutionary stage then
    if not able to survive then
      die
    end if
    if fertile then
      get a neighbor and mate and create a new offspring
    end if
  end if
until stop criteria
```

3.2.1 Evolutionary algorithm

The core of the evolutionary mechanism is the selection-variation cycle. Variation operators, mutation and crossover, must always match the problem at hand, that is, the data structure representing an individual. Therefore, we do not specify them in this generic description. The autonomous selection mechanism is, however, generic. As mentioned before, it determines selection probabilities by a sigmoid function.

$$sig_{m,s}(x) = \frac{1}{1+e^{-m\cdot(x-s)}} \tag{1}$$

This sigmoid yields large probabilities when $x > 0$ and small probabilities when $x < 0$ as illustrated in Figure 1. If $\bar{f}$ is the average fitness of the population and we fill in the fitness deviation $\Delta f(x) = f(x) - \bar{f}$ of an individual x in sig, we get:

$$P(x) = sig_{m,s}(\Delta f(x)) = \frac{1}{1+e^{-m\cdot(\Delta f(x)-s)}} \tag{2}$$

Note that the sigmoid function depends on two parameters s and

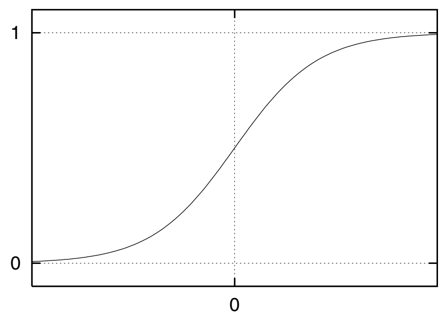

Figure 1: Sigmoid curve

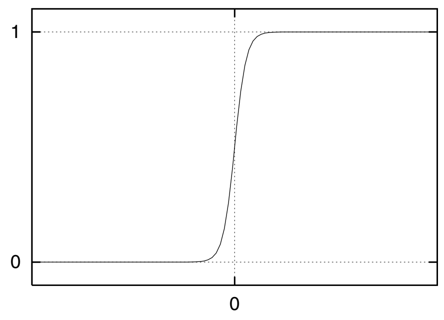

Figure 2: For $m > 1$ the curve becomes step-like

m. The *shift* s determines where the transition from low probabilities to high probabilities takes place. Using $s = 0$ centers the transition interval in the middle of the whole region, increasing s will shift it to the right, thus decreasing the number of individuals selected. The *multiplier* m determines how sharp the transition is. Low values imply a smooth curve with a broad slope, while increasing m will make the transition sharper. This effect is illustrated in Figure 2 ($m > 1$, probabilities are more discrete) and Figure 3 ($0 < m < 1$, a stretched sigmoid curve making the differences in probability smaller). Obviously, the choice of these parameters greatly influences the response of the sigmoid and hereby the properties of the selection mechanisms.

This selection mechanism is used in a fully decentralized manner. Each node applies survivor selection and parent selection on itself with possibly different m and s values. First a node applies the sigmoid formula for survivor selection to see whether it is fit to survive. If not, it simply puts itself in a *dead* state. If the node survives it applies the sigmoid formula for parent selection to determine if it is fertile (i.e., good enough to mate). When a surviving node is fertile, it selects a random neighbor to mate with. Random mate selection is possible as a by-product of gossiping that not only exchanges information, but also imposes an overlay network of neighbors on the population. This network is updated by exchanges of neighbors among nodes. This exchange of neighbors is crucial to operate as a peer-sampling service [12].

If two individuals are coupled for mating, they reproduce by executing crossover followed by mutation. The resulting child is added to the system as a new node *without removing an existing one*. This

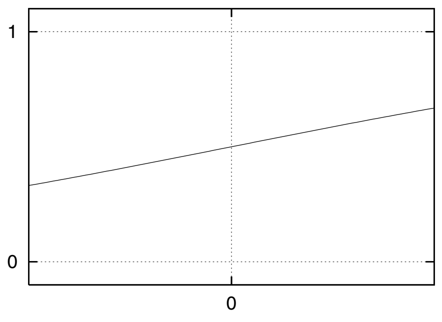

Figure 3: For $0 < m < 1$ the curve resembles a line

latter feature implies that the population size becomes an observable, rather than a user parameter. It is important to note that a node/individual will never change through the evolutionary operators. More precisely, the candidate solution it represents will remain the same, only the status of the node can change from alive to dead by survivor selection. The gossiping algorithm, however, can change a node: if a node hits upon a solution it will infect other nodes through gossiping, thus proliferating a solution over the network. As mentioned above, the evolutionary algorithm terminates if all nodes become "aware" of a solution, either by being one, or by being told about one.

3.2.2 Adaptation algorithm

The adaptation algorithm as part of our layered architecture forms the interface stage between the gossiping stage and the evolutionary stage. The adaptation process is mainly deployed to prevent explosion and implosion of the population by adjusting the parameters of the sigmoid. We do this in each node independently, which requires that nodes know whether the population as a whole is growing or shrinking. However, in our situation there is no meta observer to provide this information. A node only has its locally computed estimations. Therefore, within the adaptation stage the counting algorithm is called that makes estimations about the size of the P2P system. Based on that information the parameters for the selection functions (m and s) are adjusted. We use simple heuristic rules to achieve this. The rules are based on observing how the sigmoid responses to various value, e.g., the population grows if parent selection uses $s < 0$ and shrinks when $s > 0$.

Recall that the autonomous selection mechanism is used for parent selection and survivor selection as well, so all together there are four parameters: mS and mF denoting the m values for survival selection and parent selection (F as in fertile) and sS and sF that stand for the s values for survival and parent selection. The adaptation heuristic meant to prevent implosion of the population is based on an absolute boundary rule: If the estimated population size is smaller than a user defined implosion threshold T_I, then the selection parameters are increased/decreased by adding a positive/negative δ. The adaptation heuristic meant to prevent explosion of the population is based on a relative increase rule: If the present estimation of the population size exceeds the previous one by more than a user defined explosion threshold T_E, then the selection parameters are increased/decreased by adding a positive/negative δ. Specific δ values we use in this study are shown in Table 3 later on.

3.2.3 Counting algorithm

The counting algorithm described in Algorithm 2 is part of the gossiping layer. It deploys an aggregation protocol described in [13]. We have modified the original protocol so that it also provides each node with an estimate of the global average fitness. It is only after these estimations have been obtained that the the adaptation stage can complete.

Algorithm 2 Counting algorithm

```
initially
msg_tag := id /* all nodes have unique identifier */
size_est := 1 /* initially a node knows that only it exists */
avg_est := fitness_value /* a node knows its own fitness value */
compute estimates(size_est, avg_est, msg_tag)
repeat
  pull estimates(size_est_p, avg_est_p, msg_tag_p) from neighbor p
  if (msg_tag < msg_tag_p) then
    /* abort own counting process */
    msg_tag := msg_tag_p
    size_est := 0
  else if (msg_tag > msg_tag_p) then
    /* abort other counting process */
    size_est_p := 0
  end if
  size_est := (size_est + size_est_p) / 2
  avg_est := (avg_est + avg_est_p) / 2
  push estimates(size_est, avg_est, msg_tag) to neighbor p
until desired number of gossiping rounds
```

Note that although each node starts an estimation process for the size and fitness of the system, at the end only one estimation process survives, namely the one started by the node with the highest identifier.

The counting algorithm needs to be executed in several time steps, or rounds [13]. The required number of rounds grows logarithmically in the size of the P2P system. We executed the counting algorithm alone first on the P2P system with the Newscast protocol to find out how many rounds are needed to get good (almost perfect) estimations of the size of the system.

Using Table 1 we computed Equation 3, which gives the required number of gossiping rounds needed to get a good estimation of the size of the network. The variable γ is the current population size, α is the initial population size and β is the number of rounds needed for the initial population size.

$$\#rounds = (log_{10}(\gamma) - log_{10}(\alpha)) * 5 + \beta \qquad (3)$$

During our experiments we found that for our initial population size of 200 we required about 13 rounds to accurately compute the network size. This finally led to the following simpler approximation:

$$\#rounds \approx (log_{10}(\gamma) - 2) * 5 + 15 \qquad (4)$$

This equation gives more than the minimum number of rounds needed to estimate the network size. It is robust enough for the growth of the network also since the network can never grow more than a factor 2 during the execution of the evolutionary stage, in which every node can at best create one offspring.

3.3 Workflow of the algorithm

Our solution is deployed on a P2P system, with no centralized components, in contrast to a traditional evolutionary computing

Table 1: Minimum number of rounds (n) for a good estimate

Population size	Minimum rounds	Average estimation	Standard deviation
10	6	9.6	0.5163977794943321
100	12	99.48	0.9043106644159661
1,000	18	999.477	0.7266353961777479
10,000	23	9,999.4976	0.7790089272147155
100,000	29	99,999.50162	0.6690605610882638

setup. Each individual corresponds to a node in our P2P system, and like cellular evolutionary algorithms, will have a collection of neighbors. An individual can only mate with one of its neighbors. As we mentioned, the set of neighbors of a given node changes over time.

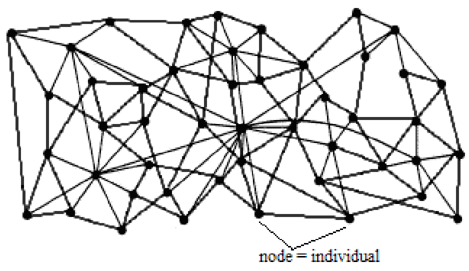

Figure 4: A simple 1-1 mapping of a cellular EA to a P2P network

In our setup the nodes can be in one of two states: alive or dead. A live node participates in the algorithm, while a dead node does not. A node is set to *dead* if the evolutionary algorithm finds that it is not fit for survival. Initially, we start with a number of live nodes, while all others are declared dead.

Live nodes are initialized with candidate solutions as is usually done in any evolutionary algorithm (here: randomly). After initialization, each live node gossips with its neighbors (some of which may be dead) in order to estimate the current network (population) size and the global average fitness. With these estimates, a node can then adapt the parameter values of the sigmoid functions, and take subsequent decisions on its viability and fertility.

Newly created offspring (i.e., a node that becomes alive) is positioned close to its parents, which is done by assigning neighbors from those of its parents. This proximal placement turned out to be important for the efficient dissemination of information.

Every (live) node in the system is able to choose a mate if it is fertile. Thus, there are two ways to become a parent: by selecting a mate, or by being selected as mate by another node. After mating, a node resets its estimation of network size and fitness value to the default values. However, the previously known estimated size of the network is used to decide on the next number of gossiping rounds. The number of rounds chosen is good enough even if the population has changed during the evolutionary stage.

If a node finds a solution it will piggyback it while gossiping for estimating the population size and overall fitness. The net effect is that the solution will spread across all nodes, allowing each of them to take the correct termination decision.

This brings us to a complete single iteration of our algorithm. Each iteration consists of $n+3$ steps, as shown in Algorithm 3. Note that n may change as the algorithm iterates. The bulk of an iteration consists of n gossiping rounds, normally in the order of a few tens. Gossiping is done for disseminating information and computing estimates. Step $n+1$ consists of adapting the selection function, while step $n+2$ constitutes an evolutionary step in which the population is adjusted. Finally, step $n+3$ consists of resetting estimates and determining the next number of steps for the following iteration.

Algorithm 3 Outline of an iteration of the distributed algorithm

```
1 to $n$ time steps: Gossiping rounds
    Exchange information with neighbors
    If there is a solution inform the neighbors
    Perform the counting algorithm
$n+1$ time step: Adaptation
    Call the adaptation process
    Update the parameters for selection
$n+2$ time step: Evolution
    Call selection process
    Either die or mate or do nothing accordingly
$n+3$ time step: Resetting
    Reset the values for the counting algorithm
    Calculate steps needed ($n$) for next iteration
```

4. EXPERIMENTAL SETUP

For an empirical study we use the N Queens problem with different problem sizes (values of N). We represent a board setting as an array of integers that form a permutation. The value $x_i \in \{1,\ldots,N\}$ on the i-th position denotes the row of the queen standing in the i-th column. This representation ensures that each column and each row contains only one queen, thus we only need to resolve the constraints on diagonals. Such permutations form then the individuals (nodes). The fitness value of an individual is the number of queens who attack each other in the board setting it represents. The optimal fitness value is 0 meaning that no queens are attacking each other. As for the variation operators, we used the Partially Mapped Crossover (PMX) and swap mutation to ensure that all children are permutations if the parents are [8]. The numerical parameters are shown in Table 2 and Table 3. Note that although the Newscast gossip protocol is mentioned as a static value in this table, Jelasity *et al.* [12] have shown that this choice is generally not crucial for the correct execution of higher layer protocols.

5. EXPERIMENTAL RESULTS

We monitored all experiments from an external observer's view point and collected data from this perspective. The data plots in the following subsections show exact values on the average fitness, the size of the network (population), etc. during a typical run.

5.1 Fitness of the population

We illustrate how the fitness of the population evolves over time by plotting the average and best fitness values. Each evaluation time corresponds to a completed iteration from Algorithm 3. Figure 5 depicts how the fitness of the population increases as the algo-

Table 2: Parameter settings for the algorithm setup

Evolutionary Algorithm	
Initial population size	200
Initial multipliers (*mS* and *mF*)	10
Initial shifts (*sS* and *sF*)	0.1
Crossover probability (p_c)	0.2
Mutation probability (p_m)	0.05
Adaptation Algorithm	
Implosion threshold T_I	200
Explosion threshold T_E	500
Gossip Algorithm	
Number of neighbors (*elements in a view*)	20
Gossip protocol	Newscast

Table 3: values for δ used in the adaptation heuristics

	heuristic to prevent	
	implosion	explosion
δ for mS	100	- 10
δ for sS	- 1	1
δ for mF	20	-10
δ for sF	0.1	- 0.1

rithm progresses. From an evolutionary computing point of view, this shows that our algorithm is capable of finding solutions to the given problem. Quite naturally, the best fitness curve shows a step-wise progress with plateaus indicating periods where no improvement of the best known fitness value takes place. The curve exhibiting the development of the average fitness shows more gradual improvements. It is especially interesting to observe the very last step before termination. The last plateau is short, which indicates that once a node (individual) has found a solution (fitness 0), it spreads quickly within the population. As we have mentioned before this is done by the use of gossiping. Note that gossiping about a solution may not reach the whole population at once, making other nodes continue to work towards becoming a solution themselves or until they hear the solution from a neighbor.

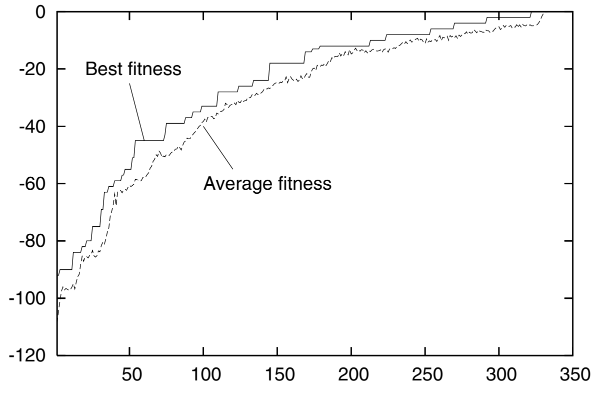

Figure 5: Fitness curves for the 96 Queens problem.

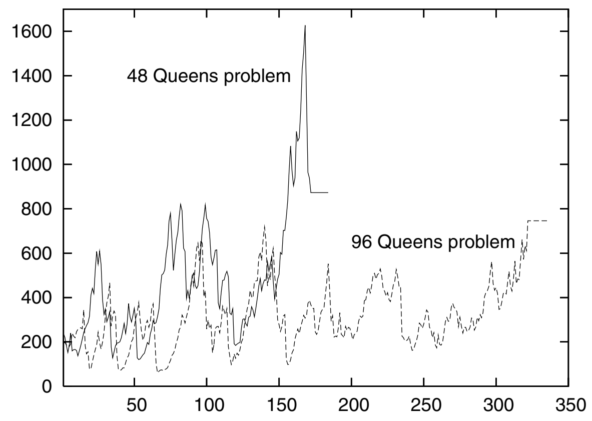

Figure 6: Changes of the population size.

5.2 Population size and selection parameters

Figure 6 displays the population sizes during a run. There are several key observations to be made. First, we have started the experiments with an initial population size of 200 live nodes. During the algorithm run the population shrinks and grows, but it never explodes or implodes to extinction. The plots show that there were periods when the population plunged down due to more deaths of individuals than births, but implosion of the population has been automatically avoided through controlling the selection pressure on-the-fly.

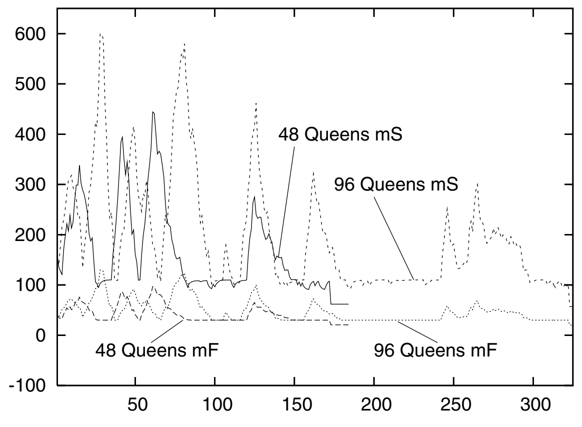

Figure 7: Average multiplier values for parent selection (mF) and survivor selection (mS).

Figures 7 and 8 show how the parameter values of the autonomous selection mechanism are adapted during a run. Recall, that the autonomous selection mechanism has two parameters (*m* and *s*) and that it is used for parent selection and survivor selection as well, all together amounting to four parameters for selection: *mS*, *mF*, *sS*, and *sF*. Furthermore, each individual has its own parameter values and it changes these values independently from the other individuals (using the same adaptation mechanism). In the figures we show the average values for the whole population.

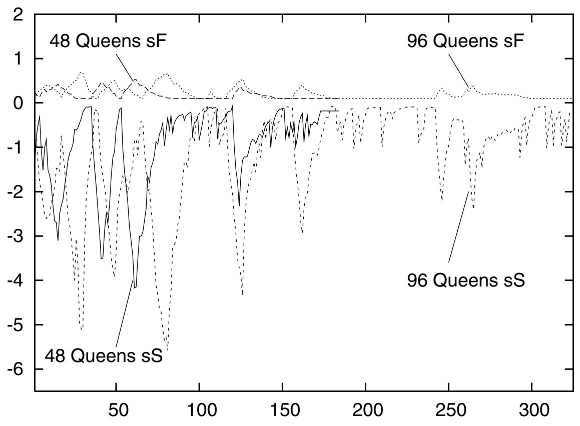

Figure 8: Average shift values for parent selection (sF) and survivor selection (sS).

These plots can be related to those showing the population variation as depicted in Figure 6. Aligning these we can see how the multiplier and shift adapt to keep the population size more or less stable. When the population size is stable only minute changes occur for the multiplier values, while in need of avoiding an implosion or explosion of the population larger variances are seen. Figure 8 shows clearly that the shift value for parent selection is larger than zero, while for survival selection the value is negative.

In our experiments the parameter *s* was found to be more sensitive to the adaptation mechanism than *m*. We also noted that *m* should be a large positive integer for the mechanism to work properly.

All in all, these figures illustrate how locally made decisions and adaptation at local level are able to keep the population alive and stable on global level.

5.3 Accuracy of population estimates

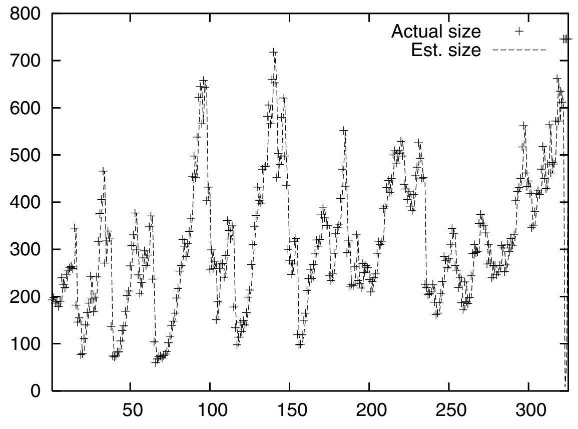

Figure 9: Population estimates for the 96 Queen puzzle

Figure 9 exhibits the the actual population size and the average of the population size estimates in the individuals over a run. The averaged estimates and the real values are very close to each other. The standard deviation of the average was always about 0.5 indicating that almost every individual was able to correctly estimate the population size.

5.4 Gossiping a solution

Table 4 shows how a node is able to proliferate a solution to the rest of the population by gossiping. We can see how the population's average fitness increases during each time step.

Table 4: Infection process for the 96 Queens problem

Time Step	Pop. size	Avg. Fitness	Std. Dev
1	746	-4.9571	2.829525
2	746	-4.94102	2.846263
3	746	-4.87668	2.842311
4	746	-4.77748	2.907307
5	746	-4.50134	3.048269
6	746	-3.88606	3.21392
7	746	-2.97587	3.269868
8	746	-1.72118	2.880624
9	746	-0.80027	2.089681
10	746	-0.2748	1.154313
11	746	-0.10456	0.693057
12	746	-0.03083	0.351966
13	746	-0.01609	0.273695
14	746	-0.00268	0.073225
15	746	0	0

The population size has not changed during the gossiping rounds of the solution. This is because it has been possible to disseminate the solution to the others within the required number of time steps for the counting algorithm. Therefore no evolutionary algorithm step gets executed within that period. Also, since the termination criterion is satisfied within the amount of time steps required for the counting algorithm the distributed algorithm stops with the population having a solution to the given problem instance. In effect, what we see is that during the gossiping rounds when multiple tasks are being carried out, the dissemination of a solution is quicker than any other task can complete.

6. CONCLUSIONS

Our experiments demonstrate the feasibility of a fully decentralized evolutionary algorithm in which the population size can be kept stable. What makes our solution unique, is that parent and survivor selection can be done completely autonomously and asynchronously, without central control, yet avoiding the risk of population explosion or implosion.

The gossip protocols play a major role with respect to node selection and the dissemination of information. Note that gossiping does require that nodes cooperate. In other words, it requires (coarse) synchronization between nodes. Each node starts the gossiping stage for estimations, followed by the adaptation and evolutionary stages, and finally resets before executing the next iteration. In practice, enforcing such a synchronization can be simply established by having a node defer responding to gossiping messages until it has entered its gossiping stage again.

The effect of estimations to our algorithm compared to the actual values was also an important finding. Since the estimates were as good as the actual value, the distributed algorithm was able to perform properly. The estimates did however play a crucial part in the whole algorithm setup, since it was the main building block of the adaptive process. This adaptive process made sure that the population size stayed within bounds. It is worthwhile to note that because of the gossip algorithm's ability to disseminate information reliably each of the nodes could obtain proper estimates about the network. Also since the new offspring stay close to the parents (in the sense of sharing neighbors), there is a much better chance of proper delivery of messages and information between the nodes from gossiping. These factors also made possible for nodes to communicate better with each other and make correct estimates about the population.

7. FUTURE WORK

An important area for improvement is making the individual nodes more independent by loosening or removing the implied synchronization between them. In particular, we are interested in replacing the gossiping stage with a solution that will allow a node to take a local decision whenever needed. For example, it may be possible to let nodes gossip continuously and in this way also continuously collect information about the behavior of the population.

From the EA perspective, further study is required to investigate the algorithm's sensitivity to its parameters, e.g., the initial values for population size, m, s, and the adaptation heuristics. It would be also interesting to try whether this, or a similar, adaptation mechanism could be used in a traditional EA for on-the-fly population size control.

Last, but not least, the algorithm should be evaluated on more problems. A specific area of interest is to apply decentralized evolutionary algorithms for hard scheduling problems in P2P networks. One such example is formed by content distribution networks in which placement of data is crucial for optimizing client-perceived performance [20].

8. REFERENCES

[1] E. Alba and B. Dorronsoro. The exploration/exploitation tradeoff in dynamic cellular genetic algorithms. *IEEE Transactions on Evolutionary Computation*, 9(2):126–142, 2005.

[2] E. Alba, B. Dorronsoro, M. Giacobini, and M. Tomassini. Decentralized cellular evolutionary algorithms. In S. Olariu and A. Y. Zomaya, editors, *Handbook of Bioinspired Algorithms and Applications*, volume 7 of *Chapman and HallCRC Computer and Information Science Series*, pages 103–120. 2005.

[3] E. Alba and M. Tomassini. Parallelism and evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 6(5):443– 462, 2002.

[4] E. Cantú-Paz and D. Goldberg. Efficient Parallel Genetic Algorithms: Theory and Practice. *Computer Methods in Applied Mechanics and Engineering*, 186:221–238, 2000.

[5] K. A. DeJong and J. Sarma. On decentralizing selection algorithms. In L. Eshelman, editor, *Proc. of the Sixth Int'l Conf. on Genetic Algorithms*, pages 17–23. Morgan Kaufman, 1995.

[6] A. Eiben, R. Hinterding, and Z. Michalewicz. Parameter control in evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 3(2):124–141, 1999.

[7] A. E. Eiben, M. Schoenauer, D. W. F. van Krevelen, M. C. Hobbelman, M. A. ten Hagen, and R. C. van het Schip. Autonomous selection in evolutionary algorithms. In D. Thierens, editor, *Procceedings of GECCO 2007*. ACM Press, 2007. to be published.

[8] A. E. Eiben and J. E. Smith. *Introduction to Evolutionary Computing*. Natural Computing Series. Springer, Berlin, Heidelberg, New York, 2003.

[9] L. J. Eshelman. The chc adaptive search algorithm: How to have safe search when engaging in nontraditional genetic recombination. In G. J. E. Rawlins, editor, *Proceedings of the First Workshop on Foundations of Genetic Algorithms*, pages 265–283. Morgan Kaufmann, 1991.

[10] C. Fry and M. Reiter. Really truly trackerless bittorrent. Technical Report CMU-CS-06-148, Carnegie Mellon University, Aug. 2006.

[11] M. Gorges-Schleuter. A comparative study of global and local selection in evolution strategies. In T. Bäck, A. Eiben, M. Schoenauer, and H.-P. Schwefel, editors, *Proceedings of the 5th Conference on Parallel Problems Solving from Nature*, pages 367–377. Springer, 1998.

[12] M. Jelasity, R. Guerraoui, A.-M. Kermarrec, and M. van Steen. The peer sampling service: experimental evaluation of unstructured gossip-based implementations. In *Middleware '04: Proceedings of the 5th ACM/IFIP/USENIX international conference on Middleware*, pages 79–98, New York, NY, USA, 2004. Springer-Verlag New York, Inc.

[13] M. Jelasity, A. Montresor, and O. Babaoglu. Gossip-based aggregation in large dynamic networks. *ACM Trans. Comput. Syst.*, 23(3):219–252, 2005.

[14] M. Jelasity, S. Volgaris, R. Guerraoui, A.-M. Kermarrec, and M. van Steen. Gossip based peer sampling. Technical report, Vrije Universiteit, 2004.

[15] K. Jenkins, K. Hopkinson, and K. Birman. A gossip protocol for subgroup multicast. In *Proceedings of the 21st International Conference on Distributed Computing Systems Workshops*, Washington, DC, USA, 2001. IEEE Computer Society.

[16] G. P. Jesi. Peersim, a peer-to-peer simulator. http://peersim.sourceforge.net/.

[17] D. Kempe, A. Dobra, and J. Gehrke. Gossip-based computation of aggregate information. In *FOCS '03: Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*, page 482, Washington, DC, USA, 2003. IEEE Computer Society.

[18] V. K. Koumousis and C. P. Katsaras. A saw-tooth genetic algorithm combining the effects of variable population size and reinitialization to enhance performance. *IEEE Trans. Evolutionary Computation*, 10(1):19–28, 2006.

[19] N. Melab, M. Mezmaz, and E.-G. Talbi. Parallel hybrid multi-objective island model in peer-to-peer environment. In *IPDPS '05: Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05) - Workshop 6*, page 190.2, Washington, DC, USA, 2005. IEEE Computer Society.

[20] G. Pierre and M. van Steen. Globule: A Collaborative Content Delivery Network. *IEEE Communications Magazine*, 44(8):127–133, Aug. 2006.

[21] G. Rudolph. On risky methods for local selection under noise. In T. Bäck, A. Eiben, M. Schoenauer, and H.-P. Schwefel, editors, *Proceedings of the 5th Conference on Parallel Problems Solving from Nature*, pages 169–177. Springer, 1998.

[22] M. Tomassini. *Spatially Structured Evolutionary Algorithms*. Natural Computing Series. Springer, Berlin, Heidelberg, New York, 2005.