# Evolutionary Algorithms on Volunteer Computing Platforms: The MilkyWay@Home Project

Nate Cole, Travis Desell, Daniel Lombraña González, Francisco Fernández de Vega, Malik Magdon-Ismail, Heidi Newberg, Boleslaw Szymanski, and Carlos Varela

## 1 Introduction

Evolutionary algorithms (EAs) require large scale computing resources when tackling real world problems. Such computational requirement is derived from inherently complex fitness evaluation functions, large numbers of individuals per generation, and the number of iterations required by EAs to converge to a satisfactory solution. Therefore, any source of computing power can significantly benefit researchers using evolutionary algorithms. We present the use of volunteer computing (VC) as a platform for harnessing the computing resources of commodity machines that are nowadays present at homes, companies and institutions. Taking into account that currently desktop machines feature significant computing resources (dual cores, gigabytes of memory, gigabit network connections, etc.), VC has become a cost-effective platform for running time consuming evolutionary algorithms in order to solve complex problems, such as finding substructure in the Milky Way Galaxy, the problem we address in detail in this chapter.

In order to tackle the complexity of evolutionary algorithms when applied to real world problems, different parallel models and computer architectures have been used in the past, for instance the parallel transputer network architecture [5] or a 10 nodes Beowulf style cluster [7] improved later to 1000

Nate Cole · Travis Desell · Malik Magdon-Ismail · Heidi Newberg · Boleslaw Szymanski · Carlos Varela
Rensselaer Polytechnic Institute. 110 8th St. Troy, NY 12180, USA
e-mail: astro@cs.rpi.edu

D. Lombraña González · F. Fernández de Vega
Centro Universitario de Mérida, Universidad de Extremadura.
Sta. Teresa Jornet, 38. 06800 Mérida (Badajoz), Spain.
e-mail: fcofdez@unex.es, daniellg@unex.es

Pentiums nodes[1]. Nowadays, large efforts are still carried out to improve results while reducing computing time, by embodying parallel techniques within EAs (see e.g., [21, 18]).

One of the most promising technologies capable of circumventing the high computational requirements of EAs, and thus reducing the solution time of many applications is the *grid* computing paradigm [32]. *Grid computing* generally refers to the sharing of computing resources within and between organizations by harnessing the power of super computers, clusters and desktop PCs, which are geographically distributed and connected by networks. Grid nodes use a special software, called *middleware*, to coordinate distributed computations.

Two of the most used middleware frameworks in the world are Globus [22] and gLite [33]. These middleware frameworks are normally complex and focused on upmarket hardware and facilities. For this reason, other grid middleware employs commodity hardware to reduce economic investment and to handle the complexity of deployment to idle desktops, thus giving rise to *desktop grids* (DGC). Examples of DGC middleware are Condor [37], Xtremweb [19] and BOINC [2].

Two main kinds of DGC are available, *enterprise grids* and *volunteer computing grids*. Enterprise grids are typically more homogeneous and usually entail processors connected by a Local Area Network (LAN) under a single root of administrative control, albeit with potentially different administrative units below. On the other hand, volunteer computing grids (e.g., as enabled by BOINC [2]) are composed of Internet-connected processors volunteered by users worldwide, resulting in larger but more heterogeneous grids.

Desktop grids have the potential to provide significant processing power since desktop computers have become an essential working tool in any market. Companies and institutions provide commodity machines to their employees to improve their efficiency when solving their everyday tasks. The hardware specifications of those desktop machines become more powerful everyday: for example, quad cores, 4 gigas of RAM memory and up to 1 Terabyte hard disks, are not uncommon. Thus, desktops are really good candidates for running complex and time consuming computational experiments. Furthermore, if we take into account that most of those desktops are underutilized by their owners, there is a potential for a large processing and storage capability within current energy usage [4].

At the same time, researchers commonly face the opposite problem: real world problems approached with EAs require more computing resources than what researchers have at their disposal. Thus, a question naturally arises: why not to exploit those unused available desktop resources to help scientific applications? Desktop grid computing, and in particular, *volunteer computing*, provides a plausible answer to this question (for the sake of simplicity, we will employ the term *volunteer computing* (VC) from now on as an umbrella for

[1] For further details see `http://www.genetic-programming.com/machine1000.html`

different but related terms: DGC, enterprise grids and volunteer computing, given the specific study described and the real-world application shown in this chapter).

VC is being successfully used for addressing problems related to climate prediction models [1], high energy physics [45], protein folding [42] and astronomy [3], [45], to name but a few. Frequently, projects rely completely on the computing power of volunteers, converting the users in a fundamental part of the project.

Yet, the number of projects using VC is still relatively narrow, and particularly unknown by Computational Intelligence community of researchers. In this chapter, we analyze the application of VC to real life problems addressed by means of Evolutionary Algorithms (EAs), and also possible extensions to other Computational Intelligence techniques. We show the VC technology, its cooperation with EAs for solving hard problems, and a real-life application: The Milky Way at home project.

The chapter is organized as follows: Section 2 presents related work on volunteer computing and evolutionary algorithms. Section 3 describes a specific astronomy problem to which we apply EAs on VC: finding sub-structure in our own Milky Way. Section 4 describes asynchronous versions of evolutionary algorithms, especially designed to be executed on heterogeneous, failure-prone volunteer computing environments. Section 5 discusses their implementation in the MilkyWay@Home project highlighting interesting results. Finally, we conclude with remarks and potential avenues for future work.

## 2 Related Work

The Berkeley Open Infrastructure for Network Computing (BOINC) [2] is the generalization of the well known SETI@Home project [3]. This volunteer computing middleware framework aggregates the computing power of idle desktop machines provided by volunteers worldwide (e.g., home computers or office workstations). One of the main advantages of using VC systems is that they provide large-scale parallel computing capabilities for specific classes of applications at a very low cost. Consequently, VC is a promising platform for running real world optimization problems solved by means of EAs.

The following sub-sections present the first VC projects, the VC technology more widely used and its relationships with Parallel EAs: possibilities of collaboration and first proposals.

## 2.1 Volunteer computing

The first successful VC project was "The Great Internet Mersenne Prime Search" (GIMPS) [50]. The aim of this project is to find new mersenne primes. The project has engaged 11,875 volunteers who provide 73,439 CPUs giving, as of April 2009, a processing power of 44.3 Teraflops.

Another successful VC project is Distributed.net [48]. The aim of this project is to win the *RSA Secret Key challenge* which consist of deciphering an RSA text with different difficulties. Distributed.net has won two of the challenges, the 56 and 64 RC5 bit encryption challenges. Additionally the Distributed.net team is also running a distributed project named OGR which tries to find the *Optimal Golomb Ruler* [49] for 24 or more marks. At the time of this writing, Distributed.net has found a solution for the OGR-24, 25 and 26. Currently they are trying to find a solution to the OGR of 27 marks.

Both projects, GIMPS and Distributed.net, are specialized VC projects for a given application. Thus, those systems cannot be used as a general tool for running any kind of research project. Nevertheless, GIMPS and Distributed.net are good examples of successful ad hoc VC projects.

Over the years, several general purpose VC middleware have been developed including Xtremweb [19], Condor [37] and BOINC [2] (as described above, we consider all the technology from the point of view of VC, and avoid subtle distinctions about DGC or VC technology).

From all the general purpose VC middleware frameworks, BOINC is the most used one. BOINC has the largest pool of volunteers/users around the world. As of April 2009, BOINC has $1,622,500$ users, providing $3,849,182$ processors which give $1,399.4 TeraFLOPS$ of aggregated computing power to VC applications. For this reason, BOINC is the best candidate for deploying an application using volunteer computing. BOINC volunteers are willing to collaborate with new projects of interest.

In summary, VC is a good candidate to run computationally intensive problems and could thus be employed to obtain "free" computing resources for running real world EA tackling hard problems. In reality, resources are not completely "free", since a community of volunteers needs to be continuously informed of the scientific goals and outcomes of the project. New volunteers need to be recruited, and existing volunteers need to be retained. However, the benefit of the keeping volunteers informed is the ensuing "open democratic science" where people can choose which research projects are worthy of their computing resources.

## 2.2 BOINC: a volunteer computing technology

As explained above, BOINC has the largest user and volunteer community. Currently, there are 28 official supported projects and 23 non-verified BOINC projects [2]. These projects belong to different research fields like: astronomy, physics, chemistry, biology, medicine, cognitive science, earth and environmental science, mathematicians, games, etc.

BOINC is a *multiplatform* and *open source* middleware that comes from the SETI@home project [3]. SETI@home aims at finding extraterrestrial intelligence by analyzing radio telescope data. SETI@home has engaged the largest community of BOINC users, when writing this chapter $944,691$ users are providing $2,239,695$ hosts which produces a computer power equal to 515.6 TeraFLOPS.

Due to the success of SETI@home, the developers decided to create BOINC, based on the SETI@home software. The goal was to provide a general tool for developing new DGC projects based on their technology. The main features of BOINC are:

- *Project autonomy.* Each BOINC project is independent, so each project has its own servers and databases. Additionally there is no central directory or approval process for the projects.
- *Volunteer flexibility.* Volunteers can decide in which and how many projects they will take part. Volunteers also decide how their resources will be shared between different projects.
- *Flexible application framework.* Applications coded in C, C++ or Fortran can be run within BOINC with little or no modification.
- *Security.* BOINC employs digital signatures to protect clients from distributing viruses or malware.
- *Server performance and scalability.* The BOINC server is extremely efficient, so that a single mid-range server can handle and dispatch millions of jobs per day. The server architecture is also highly scalable by adding more machines and distributing the load between them.
- *Open source.* The BOINC code is released under the Lesser GNU General Public License version 3 [23].

BOINC is suitable for applications that have one or both of the following requirements:

- large computation requirements,
- storage requirements.

The main requirement for running an application within BOINC is that it is divisible into multiple sub-tasks or jobs that can be run independently. As we may foresee, EAs are perfect candidates for running projects supported by BOINC: the standard parallel evaluation of individuals could be performed on different volunteer computers.

If the project is considering to employ basically volunteer resources, the project's web site must be compelling to attract volunteers and take into

[2] For further details see `http://boinc.berkeley.edu/projects.php` and `http://boinc.berkeley.edu/wiki/Project_list`

account the volunteer's bandwidth connections: lots of users do not have fast upload/download speeds.

BOINC is composed by two key elements: the server and the clients. BOINC employs a master-slave architecture. In order to facilitate the communications between the clients and the server, the HTTP protocol is used and the clients start always the communications. Thanks to this approach, the clients can collaborate with science even if they are behind a firewall or a proxy –general security set up for communications on institutions like companies or universities.

The BOINC server is in charge of:

- *Hosting the scientific project experiments.* A project is composed by a binary (the algorithm or application) and some input files.
- *Creation and distribution of jobs.* In BOINC's terminology a job is called a "work unit" (WU). A WU describes how the experiment must be run by the clients (the name of the binary, the input/output files and the command line arguments).

On the other hand, the BOINC client connects to the server and asks for work (WU). The client downloads the necessary files (WU) and starts the computations. Once the results are obtained, the client uploads them to the server.

BOINC measures the contributions of volunteers with *credit*. A *credit* is a numerical measure of the work done by a given volunteer and his computers. Volunteers care so much about the obtained credit when collaborating with a project, and it is one of the leitmotif of continuing collaborating with a given project. Thus, it is very important to handle correctly the granted credit to users, as BOINC projects can grant/handle credit differently.

Volunteer computing has a main drawback: resources are not reliable. For this reason, many types of attacks are possible in BOINC: hacking the server, abuse of participant hosts, etc. From all the possible attacks, there are two which are very important:

- *Result falsification.* Attackers return incorrect results.
- *Malicious executable distribution.* Attackers break into a BOINC server and, by modifying the database and files, attempt to distribute their own executable (e.g. a virus) disguised as a BOINC application.

In order to avoid these possible attacks, BOINC provides several mechanisms to reduce the likelihood of some of the above attacks:

- *Result falsification can be reduced using replication*: a result is sent to at least two different clients to check out that the obtained result has not been forged. BOINC provides different types of replication: fuzzy, homogeneous or adaptive replication.
- *Malicious executable distribution is avoided as BOINC uses digital signatures to distribute the binaries.* The server uses two signatures, one public and the other private to sign the applications. The private signature is

used to sign locally the binaries and the public signature is distributed to clients for checking the origin of the application by the clients. It is important to have the private key stored in safe storage not connected to Internet to avoid possible network break-ins.

To sum up, BOINC provides enough facilities to reduce as much as possible likelihood of attacks under a volunteer computing infrastructure. However, without the cooperation of administrators, the security could be risked if for example BOINC keys are stored in the server machine or protocols like Telnet are being used for accessing the machine.

## 2.3 Parallel Evolutionary Algorithms and VC

EA practitioners have found that the time to solution on a single computer is often prohibitively long. For instance, Trujillo et al. employed more than 24 hours to obtain a solution for a real world computer vision problem [52]. Times to solution can be much worse, taking up to weeks or even months. Consequently, several researchers have studied the application of parallel computing techniques and distributed computing platforms to shorten times to solution [20, 51].

Examples of these efforts are the old-fashioned Transputer platform [5], new modern frameworks such as Beagle [26], or grid based tools like Paradiseo [8]. However, there are not many real world problems that are using VC for running experiments using EAs.

Considering the VC technology, two main parallel approaches for running EAs are useful for profiting volunteer computing resources:

- *Parallel fitness evaluation*. Individuals can be distributed to be evaluated on different volunteer computers simultaneously. This is useful when the fitness evaluation time is the most time-consuming part of the algorithm.
- *Parallel execution of experiments*. When a number of runs are required for obtaining statistically significant results, different runs can be distributed on a number of computers. This model is also useful for high-throughput parameter sweep experiments.

The latest case is particularly useful when running experiments in conjunction with VC technology: no changes are required in the main EA algorithm. The algorithm is simply sent to a number of computers, with different input parameters if required.

If we focus on some of the techniques comprised within EAs, only Chávez et al. [40] presented an extension to the well-known genetic programming framework LilGP [14], that runs within a BOINC infrastructure. In this work, the experiments were a proof of concept using a couple of well-known GP problems: the ant on the Santa fe trail and the even parity 5 problem.

Moreover, the experiments only used a controlled environment without the collaboration of external volunteers.

A more generic approach was presented by Lombraña et. al. [27, 35]. The idea was to run a modern and widely used framework for EAs, ECJ [36], within a BOINC infrastructure. In this case, the experiments used a real environment but only a reduced number of volunteers where engaged. Furthermore, the experiments were again complex GP benchmarks but no real world optimization problems were addressed.

Another approach was presented by Samples et al. [43]. Samples et. al. showed the feasibility of using DGC for a typical genetic programming parameter sweep application using a pool of desktop PCs. Nevertheless, the lack of a standard middleware and a genetic programming tool has kept this approach from being commonly adopted by researchers.

All the previous approaches were simple proof-of-concepts, thus, to the best of the author's knowledge, the only real-world problem that is already using VC and Evolutionary Algorithms is the MilkyWay@home project; which is described in depth within the next sections.

## 3 Finding Milky Way Galaxy Substructure

The Milky Way spheroid is one of the major components of the Galaxy. It occupies a roughly spherical volume with the other components (disks, bulge, bar, etc.) embedded in it. Despite its volume it produces only a small fraction of the starlight emitted by the Galaxy. The stellar spheroid is composed of primarily older and more metal poor stars that produce little light compared to the brighter, more massive, stars that form in the gas-rich components of the disk. The majority of the mass of the Milky Way exists within the spheroid as dark matter; however, the nature, distribution, and structure of this mass is unknown.

With the construction and operation of large scale surveys such as the Sloan Digital Sky Survey (SDSS), the Two Micron All Sky Survey (2MASS), and many other current and upcoming projects there is an "astronomical" amount of data to be sorted through. This huge amount of data is not only composed of photometric studies, but large numbers of stellar spectra are being taken with many surveys being operated or completed solely focused on taking a large number of incredibly accurate spectra, the Sloan Extension for Galactic Understanding and Exploration (SEGUE) for example. This increase in the amount of data, as well as the increase in the accuracy of this data has led to many discoveries involving the spheroid and substructure in the Galaxy. The Sagittarius dwarf spheroidal galaxy (Sgr dSph) and its associated tidal stream, were the first known example of a current merger event to be discovered [30, 28, 29, 56]. Since its discovery, the study of substructure has dominated the research efforts towards the spheroid. This has resulted in the discovery of several additional dwarf galaxies, tidal streams and globular clusters of the Milky Way as can be found in [39, 55, 53, 15, 38, 6, 54], and [57] among others.

For many years, the spheroid was imagined to have evolved from outward flows from the disk of the Galaxy [17], or to have formed in conjunction with the rest of the Galaxy and gradually evolved to its current state [44]. It has also long been imagined to have a smooth and continuous power law density distribution [25]. However, the advancement in technology and analysis techniques have discovered the large amount of substructure, discussed above, and has shown that at least some of the spheroid was constructed via merger events and that the spheroid was composed of debris from hierarchical structure formation [24]. A shift in the thinking of the spheroid has therefore come about, and the more substructure that is discovered the stronger the case that the spheroid is non-smooth and was constructed primarily in this manner.

## 3.1 Why substructure?

Dwarf galaxies and star clusters are gravitationally bound systems, which can themselves be bound in the gravitationally potential of the Milky Way. As dwarf galaxies approach the center of the Milky Way, the differential gravitational forces can pull stars out of their bound orbits in the dwarf galaxy, and cause them to orbit the Milky Way instead. Stars with lower energy are pulled ahead of the dwarf galaxy core while stars with higher energy fall behind, resulting in the creation of long streams of stars, also known as tidal streams, that extend the longer the dwarf is bound by the Milky Way. As this disruption continues, the streams will become longer and cover more of the sky as more and more of the stars are stripped from the smaller body. Over time, all traces of the merging body will fade as the stars become more and more dispersed and become assimilated into the Milky Way spheroid. These long tidal streams of stars provide a unique opportunity for these are the only stars in which it is possible to know not only where they are and where they are going, but also where they have been. They trace out the path the merging body took as it traversed the gravitational potential of the Milky Way. In this way substructure in the spheroid can be seen as something of a cosmic fingerprint powder bringing all of the details of the spheroid itself into focus.

By studying the substructure of the spheroid, specifically that of tidal streams, it is possible to study the galactic potential. This is not done via direct measurement, but is primarily studied via simulations, which are adjusted to replicate the observational data. The more observational points with which to compare and the more accurate those points are, the more precisely the simulation can constrain the models of the Galactic potential.

As previously stated, the majority of the mass of the Milky Way is within the spheroid and this is composed of dark matter. Therefore, the dominant component of the Galactic potential is provided by the spheroid and specifically the dark matter of the spheroid. Thus, by constraining the models of the Galactic potential it is possible to determine the distribution of dark matter within the Galaxy.

It is important to develop techniques that are accurate and efficient means of studying substructure, so the results may then be used to compare against the simulations. There are primarily two methods used for discovery and study of substructure: *kinematical* and *spatial*. The kinematical approach attempts to find co-moving groups of stars that can be identified by groups of stars in a similar location with common velocity. These are indicators that the stars might have come from a common structure instead of simply being part of the smooth spheroid. This approach, though potentially more powerful, is limited in that it requires a spectroscopic analysis of all stars studied in order to determine the velocities. Accurate stellar spectroscopy is significantly harder to obtain than photometry. The second approach for substructure discovery and analysis is to simply search for overdensities in the star counts within the data. This is done by looking for statistically relevant deviations from the assumed background distribution of the spheroid. This technique benefits from the fact that only a photometric analysis of the data need be accomplished, thus the amount of data available for a study of this kind is much greater. The majority of the substructure discoveries in the Milky Way spheroid have been made through analysis of the photometric data.

## 3.2 The Maximum Likelihood Approach to Substructure

We have developed a maximum likelihood method for the discovery and analysis of substructure within the stellar spheroid. This method seeks to provide an efficient, automated, accurate, and mathematically rigorous means to study substructure. The method has been designed to determine the spatial characteristics of tidal debris and the stellar spheroid through the use of photometric data.

The SDSS is a large, international collaboration that has generated an enormous amount of data over 10,000 square degrees of the sky. The SDSS data was taken with a 2.5m dedicated telescope at Apache Point Observatory in New Mexico. Due to its placement in the northern hemisphere, the data covers primarily the north Galactic cap with some minimal coverage in the south. The SDSS survey area is composed of 2.5° wide stripes taken on great circles across the sky (the entire sky is comprised of 144 such stripes) with data taken at high Galactic latitude. Since the SDSS imaging survey is well calibrated and is comprised of mostly contiguous data, it is a good candidate for studying substructure as it traces across the sky.

Specifically, we extract stars of the color of blue F turnoff stars in SDSS data to study substructure. The relatively large number of F turnoff stars make them a good candidate for a statistical study of this nature. Also, the color of F turnoff stars within the spheroid is bluer than that of the Milky Way disk, therefore contamination of stars from non-spheroid components can be minimized necessitating a model for only one Galaxy component, the spheroid. Finally, F turnoff stars were chosen for this study for it is possible to reasonably estimate their distance although not as well as other less numerous "standard candles." By modeling the distribution of absolute magnitudes that F turnoff stars can take, it is possible to utilize stars of this type quite effectively.

Tidal streams generally follow very complex paths across the sky and are therefore difficult to model. However, over small volume, such as a $2.5°$ wide stripe of data taken by the SDSS, the tidal stream may be approximated as linear. In this manner, the tidal stream is estimated in a piecewise fashion across the sky with each stripe of data maintaining its own set of stream parameters. In this way, the tidal stream density distribution is modeled as a cylinder with Gaussian fall off from its axis. The tidal stream model is thus parameterized by the cylinder position, its orientation, and the standard deviation of the Gaussian fall off. The smooth spheroid component is modeled as a standard Hernquist profile and is therefore parameterized with a scaling factor in the Galactic Z direction, and a core radius of the power law component. Finally, the absolute magnitude distribution in F turnoff stars is modeled as a Gaussian distribution with fixed variance.

Utilizing the above models we developed a likelihood function for the probability of the data given the model and the parameters. The input data set is thus composed of a three column file in which each row represents the spatial position of an F turnoff star as observed via the SDSS telescope. An optimization method is then applied to the likelihood function and the best-fit parameters are determined for the given dataset and the models. A detailed description of the model descriptions and likelihood function can be found in [11]. The approach of this algorithm offers a unique advantage in that the spheroid and tidal debris are fit simultaneously. Because of this, it becomes possible to probabilistically extract the tidal debris distribution from that of the spheroid. In this manner, the structures can then be separately analyzed via additional means. This provides a method of independent study previously unavailable to researchers.

After successfully testing the algorithm upon a simulated data set, the initial goal has been to analyze the Sgr tidal stream across all of the SDSS data. The algorithm has successfully been used to analyze the stream in the three stripes of data in the southern Galactic cap. The results presented in [11] for the analysis of stripe 82 were a success, for the well established values determined via other methods were recovered, and the error bars upon the results generated via this method are much smaller than those previously achieved. However, the true potential of this method can be seen in the analysis of the two surrounding stripes [10]. These include a new detection of the Sgr tidal stream in the south as well as the great improvement in the position of the previous detection in the opposing stripe. Figure 1 depicts the results of analyzing these three stripes and how these results compare to a model of the Sgr dSph disruption. Ongoing research seeks to create a complete map of the leading Sgr tidal stream throughout the north Galactic cap (and fill in the northern part of the figure). A preliminary analysis of part of this data has already shown significant discrepancy between the model of the Sgr dSph disruption and the spatial properties being determined via this method. However, the breadth of this discrepancy will not be known until a complete analysis of the data has been performed. It is also quite interesting to note that in order to accurately fit the data in the northern stripes, it is necessary to fit multiple tidal streams within the same dataset, for there is substructure beyond the Sgr tidal stream in the data collected from the north. Therefore, even though the analysis of the Sgr tidal stream is the primary concern of the project at this stage, an analysis of other substructure will be occurring simultaneously.

Following the completion of the Sgr tidal stream study, we would like to use the information obtained to attempt to constrain the models for the Galactic potential by fitting a simulation of the Sgr dSph to the values obtained in this study. This will be difficult, for an appropriate measure of the “goodness of fit” must be determined with which to compare the simulation and the analysis results. Also, the computational complexity of an undertaking of this nature is far beyond that of even the current algorithm. This is due to not only having to analyze the data, but having to actually create the data via N-body simulation. Another topic of great interest that will provide a significant challenge, is an effort to determine the true distribution of the smooth spheroid component. To do this, all substructure must be removed from the data via this algorithm and then all remaining data (representing the smooth spheroid) fit at once. This will demand significant changes to the current algorithm to fit over multiple datasets at once, plus a method for determining the correct model must be determined as well.

## 4 Asynchronous Search

Maximum likelihood estimation attempts to find the set of parameters of a mathematical function that best fits a given data set. The mathematical function represents a scientific model, for example, the geometrical substructure of the stars in a galaxy wedge, given certain parameters such as a star stream position, width, intersection angle, etc. The data set represents the

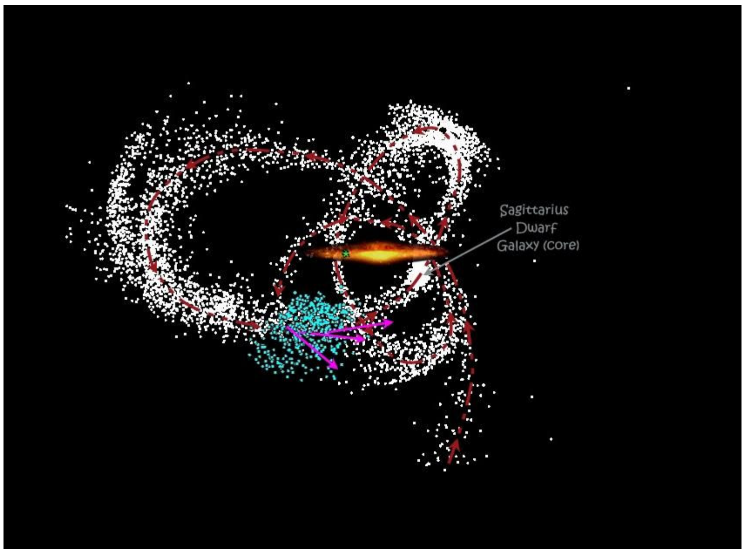


**Fig. 1** Sagittarius dwarf disruption with fits to SDSS southern stripes. Plotted here is the Sgr dSph face on, which is nearly orthogonal to the plane of the Milky Way. A galaxy consistent with the Milky Way has been overlayed and the Solar position is marked with a star. The dotted arrow shows the orbital path of the dwarf and the current position of the Sgr dwarf core is clearly labeled. The points following this orbital path are a subsampling of an N-body simulation consistent with the result from [34]. The complete arrows depict the position and spatial direction of the Sgr tidal stream within the three southern SDSS stripes (from left:79, 82, and 86). The remaining points represent a subsampling of those stars found to fit the density profile of the stream within the stripe using the separation technique to extract the stream from the stellar spheroid.

"ground truth", for example, the observed photometric galaxy data, or the result of an N-body simulation.

Population based search methods such as differential evolution (DE) [46], genetic algorithms (GA) [9], and particle swarm optimization (PSO) [31, 16] use the notion of *generations* of individuals in a population that evolves over time. Analogous to their biological counterparts, the most *fit* individuals have better probability of survival and reproduction in newer generations. We call *genetic search* the process of using genetic algorithms to search in $n$-dimensional space for the best set of parameters that optimizes a given $n$-arity function. In genetic search, individuals are sets of parameters to a function, and their fitness value is computed by applying the function to the parameter set. For this work, individuals are sets of real valued numbers, which are the input parameters to the maximum likelihood calculation.

Traditional population based search methods, such as differential evolution, genetic search and particle swarm optimization are typically iterative in nature, which limits their scalability by the size of the population. This section introduces an asynchronous search (AS) strategy, which while being similar to traditional population based search methods in that it keeps a population of parameters (or individuals) and uses combinations and modifications of those individuals to evolve the population. The main difference is that AS uses a master-worker approach instead of a parallel model of concurrency. Rather than iteratively generating new populations, new members of the population are generated in response to requests for work and the population is updated whenever work is reported to the master.

Asynchronous search consists of two phases and uses two asynchronous message handlers. The server can either be processing a *request work* or a *report work* message and cannot process multiple messages at the same time. Workers repeatedly request work then report work. In some ways this approach is very similar to steady-state genetic search, where $n$ members of the population are replaced at a time by newly generated members.

In the first phase of the algorithm (while the population size is less than the maximum population size) the server is being initialized and a random population is generated. When a *request work* message is processed, a random parameter set is generated, and when a *report work* message is processed, the parameters and the fitness of that evaluation are added to the server's population. When enough *report work* messages have been processed, the algorithm proceeds into the second phase which performs the actual genetic search.

In the second phase, *report work* will insert the new parameters and their fitness into the population but only if they are better than the worst current member and remove the worst member if required to keep the population size the same. Otherwise the parameters and the result are discarded. Processing a *request work* message will either return a mutation or reproduction (combination) from the population. Section 4.1 describes different methods for this in detail.

This algorithm has significant benefits in heterogeneous environments because the calculation of fitness can be done by each worker concurrently and independently of each other. The algorithm progresses as fast as work is received, and faster workers can processes multiple *request work* messages without waiting on slow workers. This approach is also highly scalable, as the only limiting factor is how fast results can be inserted into the population and how fast *request work* messages can be processed. It is also possible to have multiple masters using an island approach for even greater scalability. This approach is also highly resilient to client side faults, because unreturned work does not effect the server side genetic search.

## *4.1 Combination Operators*

The previous section gave a generic outline for asynchronous search which allows for various combination and mutation operators to be used in generating new parameter sets to be evaluated. This makes the algorithm easily adaptable, which is a desirable quality because no particular search method is ever optimal for all optimization problems. This section describes using the asynchronous search strategy to implement both genetic search (AGS) and particle swarm optimization (APSO).

### 4.1.1 Asynchronous Genetic Search (AGS)

Asynchronous genetic search generates new individuals using either mutation or a combination operator, as follows.

#### Mutation Operators

The standard mutation operator for an optimization problem with a continuous domain is to take an individual from the population, and randomly regenerate one of its parameters within the bounds of that parameters. This is often modified so that the range in which the parameter can be modified is gradually reduced as the search progresses, which can result in improved convergence rates.

#### Average Operator

The standard and most simple combination operator for real variables over a continuous domain is to take the average of two parent individuals and use that as the child.

### Double Shot Operator

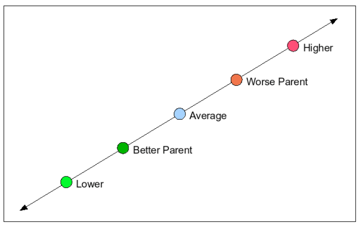


**Fig. 2** The double shot operator generates three children: the average, a point outside the worse parent (higher), and a point outside the better parent (lower), the latter two points are a distance from the average equal to the distance between their parents.

Desell et al. [12] show that using a double shot operator as opposed to a standard average operator can significantly improve convergence rates for the astronomical modeling application. The double shot operator produces three children instead of one. The first is the average of the two parents, and the other two are located outside the parents, equidistant from the average (see Figure 2). This approach is loosely based on line search, the point outside the more fit parent is in a sense moving down the gradient, while the point outside the less fit parent is moving up the gradient created by the two parents. The motivation for the latter point is to escape local minima.

### Probabilistic Simplex Operator

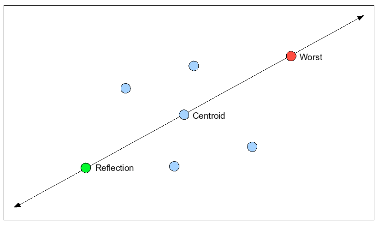


**Fig. 3** The simplex method takes the worst point and reflects it through the centroid of the remaining points. The probabilistic simplex operator randomly generates a point on some section of the line connecting the worst point and its reflection.

Unfortunately, all of these approaches require synchrony by creating dependence between fitness calculations. While it is not possible to effectively perform the traditional Nelder-Mead simplex search in a highly heterogeneous and volatile environment like BOINC, a probabilistic operator can mimic its behavior. The Nelder-Mead simplex search takes $N + 1$ sets of parameters, and performs *reflection*, *contraction* and *expansion* operators between the worst set of parameters and the centroid of the remaining $N$ (see Figure 3). After calculating the centroid, a line search is performed by expanding or contracting the simplex along this line. Because in our asynchronous model it is not possible to iteratively perform expansions and contractions, a random point is selected on the line joining the worst point and its reflection. There are three parameters involved in this operator, $N$, the number of points used to form the simplex (chosen randomly from the population), and two limits $l_1$ and $l_2$ which specify where on the line the point can be generated. For example, $l_1 = -1$ would set one limit to the reflection and $l_2 = 1$ would set the other limit to the worst point. For the purposes of this study, we use $l_1 = -1.5$ and $l_2 = 1.5$ and examine how children generated from different parts of this line effect the

#### 4.1.2 Asynchronous Particle Swarm Optimization (APSO)

Particle swarm optimization was initially introduced by Kennedy and Eberhart [31, 16] and is a population based global optimization method based on biological swarm intelligence, such as bird flocking, fish schooling, etc. This approach consists of a population of particles, which ”fly” through the search space based on their previous velocity, their individual best found position (cognitive intelligence) and the global best found position (social intelligence). The population of particles is updated iteratively as follows, where $x$ is the position of the particle at iteration $t$, $v$ is it’s velocity, $p$ is the individual best for that particle, and $g$ is the global best position. Two user defined constants, $c_1$ and $c_2$, allow modification of the balance between local (cognitive) and global (social) search, while another constant the *inertia weight*, $w$, scales the particle’s previous velocity. The search updates the positions and velocities of the particles as follows:

$$v_i(t+1) = w * v_i(t) + c_1 * rand() * (p_i - x_i(t)) + c_2 * rand() * (g_i - x_i(t)) \quad (1)$$

$$x_i(t+1) = x_i(t) + v_i(t+1) \quad (2)$$

In a parallel computing scenario, the search typically progresses iteratively. The fitness of each particle is computed in parallel, then the local and global best points are updated. Following this a new positions for each particle are computed and the process repeats.

PSO can be made asynchronous by noting that the method in which the particles move around the search space is not completely dependent on the fitness computed in the previous iteration. A particle will continue to move using its previous velocity and the current local and global best positions found until a new local or global best position is found. By relaxing this restriction slightly by allowing to a particle to continue to move in absence of knowledge of the fitness of previous states, we can utilize the asynchronous search strategy and remove the scalability limitations of traditional PSO.

APSO works as follows. New positions for particles are generated in a round robin fashion in response to *request work* messages. Instead of waiting for previously sent out particles to be evaluated, new particles are generated using the current *known global best* and *known local best*. This allows the search to progress and generate new particles asynchronously. When a particle is evaluated, its fitness, position, velocity are reported and the search is updated if the particle is a new local or global best. In this way the APSO performs nearly identically to PSO, without scalability limitations. As more processors request work, particles are generated further ahead increasing the exploratory component of the search.

## 5 MilkyWay@Home: Finding Galactic Substructure using Genetic Search on Volunteered Computers

### *5.1 Convergence*

#### 5.1.1 Asynchronous Genetic Search

The hybrid simplex method was evaluated using the astronomical modeling problem detailed by Purnell et al [41]. Performing the evaluation of a single model to a small wedge of the sky consisting of approximately 200,000 stars can take between 15 minutes to an hour on a single high end processor. Because of this, to be able to determine the globally optimal model for that wedge in any tractable amount of time requires extremely high powered computing environments. To measure the effect of asynchronicity on the hybrid genetic search, both synchronous and asynchronous computing environments are used, 1024 processors of an IBM BlueGene/L and a BOINC volunteer computing project with over 1,000 volunteered computers.

Figure 4 shows the performance of the double shot approach, and the simplex approach with varying numbers of parents $N$ being used to calculate the centroid on both environments. Previous work has shown that the double shot approach significantly outperforms iterative genetic search and asynchronous genetic search using only the average and mutation operators [47]. All approaches converged to the known global optimum of the data. For both computing environments, a population of size 300 was used, and the mutation operator was applied 20% of the time, all other members were generated with the corresponding operator. The range of the probabilistic line search for the simplex hybrid was defined by the limits $l_1 = -1.5$ and $l_2 = 1.5$. For the syn-

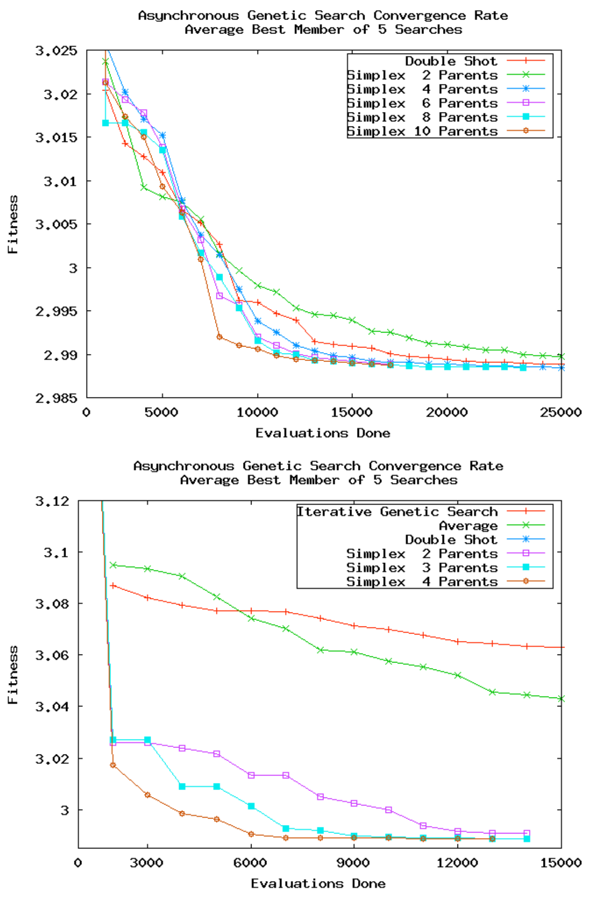


**Fig. 4** Fitness of the best member found averaged over five searches for the double shot approach and the simplex hybrid with $N = 2..5$, using the BOINC volunteered computers and the BlueGene supercomputer.

chronous execution on the BlueGene, each model evaluation was performed by dividing the work over the 1024 processors, and immediately attempting to insert the member into the population - in this way only the most evolved population was used to generate new members and the population was continuously updated. The asynchronous execution on BOINC generates new members from the current population whenever users request more work. After a user has completed the evaluation of a member, it's sent to the server and inserted into the population. There is no guarantee of when the fitness of a generated member will be returned, or even if it will be returned at all.

On the BlueGene, the hybrid simplex method shows dramatic improvement over the double shot approach, with the difference increasing as more

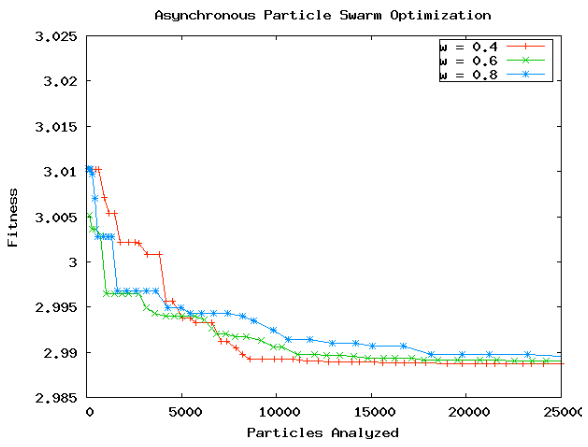


**Fig. 5** Fitness of the best particle found averaged over five searches for asynchronous particle swarm optimization on the BOINC volunteered computers, using constants $c1 = c2 = 2.0$ and inertia weight $w = 0.4$, 0.6 and 0.8.

parents are used to calculate the centroid. While the double shot method typically converges in around 18,000 iterations, the simplex hybrid with $N = 4$ converges in approximately 8,000. Compared to the 50,000 iterations reported for traditional iterative genetic search [12], the convergence rate is excellent. Using BOINC shows similar results, however the convergence rates are not as fast on the BlueGene, which is to be expected. Generally, increasing the number of points used to calculate the centroid results in better searches, however on BOINC the simplex with $N = 2$ and double shot operators initially seem to converge more quickly than the more informed simplex with $N = 4..10$, which was not the case on the BlueGene. The asynchronous approach on BOINC may take more iterations, but BOINC is much more accessible as it is dedicated to the project at hand, while use of the BlueGene is shared among many researchers. Because of this, even though the quantity of fitness evaluations done per second is similar for both computing environments, the BOINC framework can perform more searches and does so at a fraction of the cost. These results are very promising for the use of asynchronous search and volunteer computing for computationally intensive scientific modeling.

### 5.1.2 Asynchronous Particle Swarm Optimization

Asynchronous particle swarm optimization (APSO) was also tested using the BOINC computing platform, with results competitive to asynchronous genetic search. Figure 5 shows how the search progressed using different values of the inertia weight $w$. Typically, regular particle swarm optimization uses

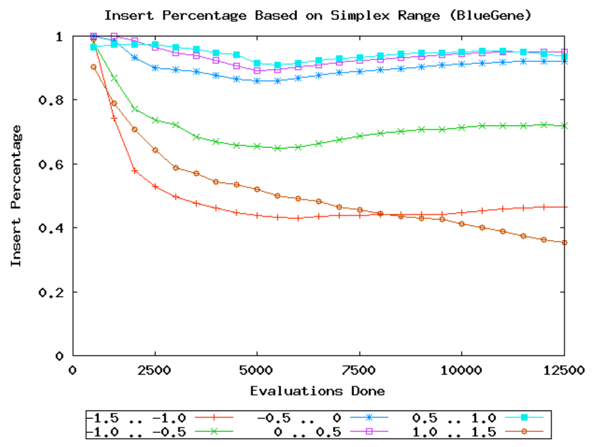
$w = 1$, $c1 = 2$, and $c2 = 2$ as standard values for constants, however we found that in an asynchronous setting, lower values of $w$ performed significantly better. While all values did eventually reach the global optimum, a value of 0.4 tended to find the optimum the fastest. It is however interesting to note that while values 0.6 and 0.8 initially converge faster, they then spend much longer zeroing in on the correct value. It could be said that this is evidence of a higher inertia weight being more exploratory, finding good possible areas to search, but these higher values lack the required exploitation of those good areas needed to ultimately find a correct value. It also supports results found by Dingxue et al. which find that using an adaptive value for $w$ improves convergence [13].

## *5.2 Operator Analysis*

To better understand the effect of the operators in evolving the population, as well as the effect of asynchronicity and of a highly heterogeneous computing environment on the fitness returned, the number of members processed between the generation and reporting of a members fitness was tracked, as well as information about how it was generated. For both environments, the best $N$ was used. Figure 6 shows the percentage of members inserted into the population and at what position in the population they were inserted based on what part of the line they were generated with using the simplex hybrid with $N = 4$ on the BlueGene. The population is sorted from the best fit to the least, so the lower the position at which a member is inserted, the better its fitness with respect to the rest of the population. Figures 7 and 8 show the same information for BOINC and $N = 4$. To provide a measure of how far the population evolved while a member was being evaluated, these results are partitioned by how many other members were reported before the fitness of the current member was reported. The range of the probabilistic line search for the simplex was defined by limits $l_1 = -1.5$ and $l_2 = 1.5$ and the statistics are taken from five separate searches.

On the BlueGene, the best insert rate and quality was from points around the centroid (generated between limits of 0.5 and -0.5). While inside of the worst point (1.0 to 0.5) had the highest insert rate, the quality of inserted members was rather low. Points near the reflection of the worst point through the centroid (-1.5 to -0.5) tended to have low insert rates, however when they were inserted they tended to be very fit. Points outside of the worst member (1.0 to 1.5) had the worst insert rate and the least fit. These results suggest that the probabilistic simplex search could be further optimized by restricting the range to limits $l_1 = -1.5$ and $l_2 = 0.5$, by eliminating the poorest performing range of 0.5 to 1.5.

BOINC showed similar results for quickly reported results (less than 200 members reported while the member was being evaluated) with points gen-



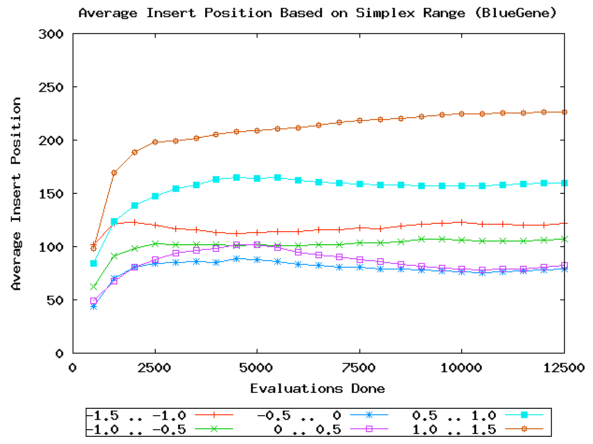


**Fig. 6** Average insert rate and insert position of members based on what part of the line calculated by the simplex hybrid they were generated on, for $N = 4$ using the BlueGene supercomputer. A lower insert position means the member is more fit than the rest of the population.

erated near the centroid (-0.5 to 0.5) having the best fitness and insert rate (see Figures 7 and 8). One notable exception was that points generated on the inside of the worst point (0.5 to 1.0) had a notably lower insert rate and that points generated near the worst point (0.5 to 1.5) quickly degraded in terms of insert rate compared to other points. With over 1600 evaluations being reported during a members round trip time, not a single point generated past the worst point was inserted. Another point of interest is that while points generated near the reflection (-1.5 to -0.5) had lower insertion rates than those near the centroid (-0.5 to 0.5), as the report time increased, their average insert position stayed the same and eventually had better fitness than

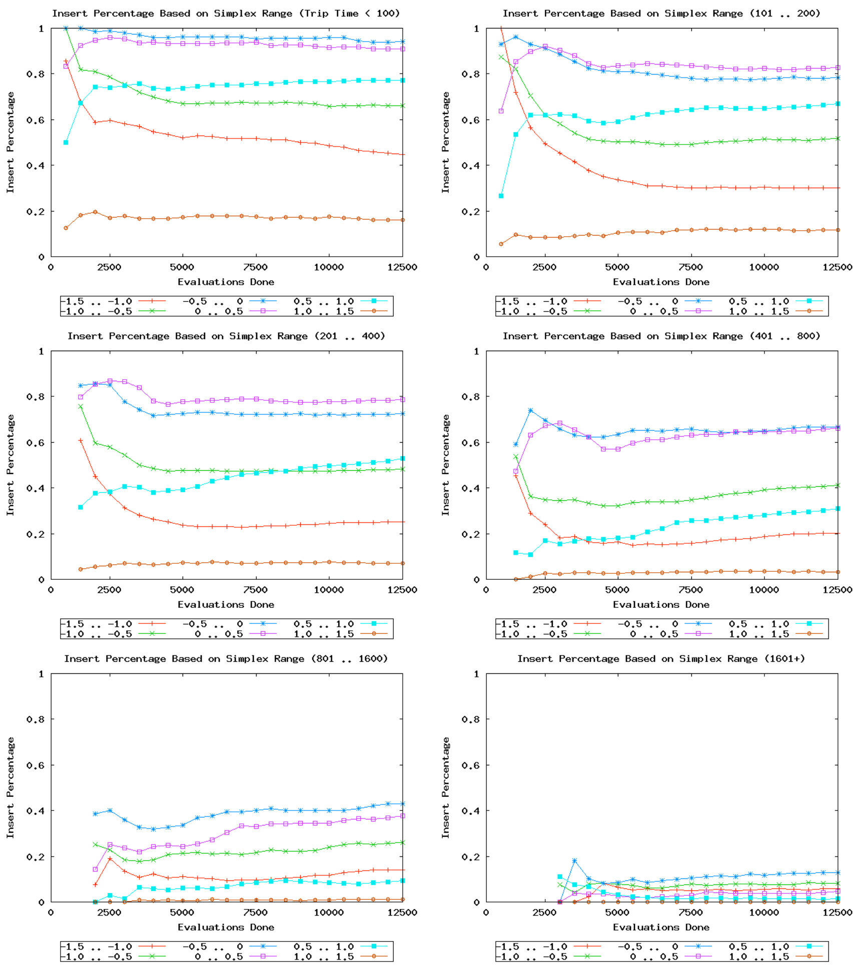


**Fig. 7** Average insert rate of members based on what part of the line calculated by the simplex hybrid they were generated on, for $N = 5$ using the BOINC framework. The results are partitioned by how many other members were reported while the used members were being generated (0..100 to 1601+) to show the effects of asynchronicity and of a heterogeneous computing environment.

points generated near the centroid. As with the BlueGene, the results suggest that refining the limit on the probabilistic simplex operator to $l_1 = -1.5$ and $l_2 = 0.5$ would improve the convergence rates. Additionally, it appears that the result report time does have an effect on which part of the line used by the probabilistic simplex operator is better to draw new members from. An intelligent work scheduling mechanism could assign members generated near the reflection to processors with slower reporting times, and those gen-

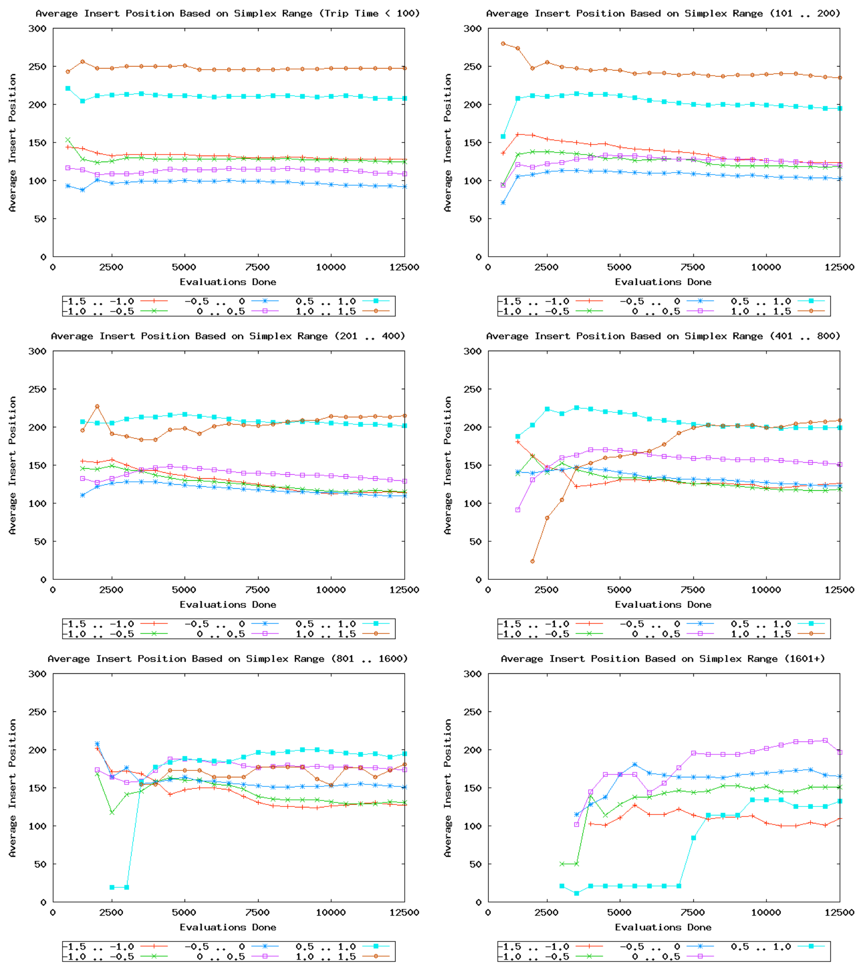


**Fig. 8** Average insert position of members based on what part of the line calculated by the simplex hybrid they were generated on, for $N = 4$ using the BOINC framework. A lower insert position means the member is more fit than the rest of the population. The results are partitioned by how many other members were reported while the used members were being generated (0..100 to 1601+) to show the effects of asynchronicity and of a heterogeneous computing environment.

erated near the centroid to processors with faster reporting times. Also, as the search progresses, there are fluctuations as to where the best points are generated from. An adaptive search could refine the limits to improve convergence rates. It is important to note that even the slowest processors retain their ability to evaluate members that are of benefit to the search, which is an important attribute for any algorithm running on massively distributed and heterogeneous environments.

## 6 Conclusions

Volunteer computing platforms can significantly enable scientists using evolutionary algorithms by providing them access to thousands of processors worldwide. The heterogeneity inherent in this worldwide computing infrastructure can be tackled by using asynchronous versions of evolutionary algorithms, which are better suited to deal with the wide variety of processing speeds and failure characteristics found in volunteer computing environments.

We have shown a specific application in astronomy, MilkyWay@Home, that uses asynchronous genetic search on BOINC to discover substructure in the Milky Way Galaxy from Sloan Digital Sky Survey data. The availability of more processing power for scientists has the potential to enable better science: more complex models can be tested on larger data sets, streamlining the scientific process.

Additional research directions include adapting additional evolutionary algorithms to the heterogeneous and failure-prone nature of volunteer computing environments, creating generalized scientific computing frameworks that lower the barrier of entry to new scientific domains, and developing hybrid mechanisms that can make efficient use of diverse distributed computing environments including supercomputers, grids, clusters, clouds and the Internet.

## 7 Acknowledgments

The U.S.-based authors were partially supported by the National Science Foundation under Grants No. 0612213, 0607618 and 0448407. The Spain-based authors were partially supported by Regional Gridex project PRI06A223 Junta de Extremadura and National Nohnes project TIN2007-68083-C02-01 Spanish Ministry of Science and Education.

## References

1. Allen, M.: Do it yourself climate prediction. Nature (1999)
2. Anderson, D.: Boinc: a system for public-resource computing and storage. In: Grid Computing, 2004. Proceedings. Fifth IEEE/ACM International Workshop on, pp. 4–10 (2004)
3. Anderson, D.P., Cobb, J., Korpela, E., Lebofsky, M., Werthimer, D.: Seti@home: an experiment in public-resource computing. Commun. ACM **45**(11), 56–61 (2002). DOI http://doi.acm.org/10.1145/581571.581573
4. Anderson, D.P., Fedak, G.: The computational and storage potential of volunteer computing. In: Proceedings of CCGRID, pp. 73–80 (2006)
5. Andre, D., Koza, J.R.: Parallel genetic programming: a scalable implementation using the transputer network architecture pp. 317–337 (1996)
6. Belokurov, V., Zucker, D.B., Evans, N.W., Gilmore, G., Vidrih, S., Bramich, D.M., Newberg, H.J., Wyse, R.F.G., Irwin, M.J., Fellhauer, M., Hewett, P.C., Walton, N.A., Wilkinson, M.I., Cole, N., Yanny, B., Rockosi, C.M., Beers, T.C., Bell, E.F., Brinkmann, J., Ivezić, Ž., Lupton, R.: The Field of Streams: Sagittarius and Its Siblings. The Astrophysical Journal Letters**642**, L137–L140 (2006). DOI 10.1086/504797
7. Bennet, F.H.I., John R. Koza, J.S., Stiffelman, O.: Building a parallel computer system for $18,000 that performs a half peta-flop per day. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 1484–1490. Orlando, Florida, USA (1999)
8. Cahon, S., Melab, N., Talbi, E.: ParadisEO: A Framework for the Reusable Design of Parallel and Distributed Metaheuristics. Journal of Heuristics **10**(3), 357–380 (2004)
9. Cantu-Paz, E.: A survey of parallel genetic algorithms. Calculateurs Paralleles, Reseaux et Systems Repartis **10**(2), 141–171 (1998)
10. Cole, N., Jo Newberg, H., Magdon-Ismail, M., Desell, T., Szymanski, B., Varela, C.: Tracing the Sagittarius Tidal Stream with Maximum Likelihood. In: American Institute of Physics Conference Series, *American Institute of Physics Conference Series*, vol. 1082, pp. 216–220 (2008). DOI 10.1063/1.3059049
11. Cole, N., Newberg, H.J., Magdon-Ismail, M., Desell, T., Dawsey, K., Hayashi, W., Liu, X.F., Purnell, J., Szymanski, B., Varela, C., Willett, B., Wisniewski, J.: Maximum Likelihood Fitting of Tidal Streams with Application to the Sagittarius Dwarf Tidal Tails. The Astrophysical Journal**683**, 750–766 (2008). DOI 10.1086/589681
12. Desell, T., Cole, N., Magdon-Ismail, M., Newberg, H., Szymanski, B., Varela, C.: Distributed and generic maximum likelihood evaluation. In: 3rd IEEE International Conference on e-Science and Grid Computing (eScience2007), pp. 337–344. Bangalore, India (2007)
13. Dingxue, Z., Zhihong, G., Xinzhi, L.: An adaptive particle swarm optimization algorithm and simulation. In: IEEE International Conference on Automation and Logistics, pp. 2399–2042 (2007)
14. Dr. Bill Punch, D.Z.: Lil-gp. `http://garage.cse.msu.edu/software/lil-gp/index.html`
15. Duffau, S., Zinn, R., Vivas, A.K., Carraro, G., Méndez, R.A., Winnick, R., Gallart, C.: Spectroscopy of QUEST RR Lyrae Variables: The New Virgo Stellar Stream. The Astrophysical Journal Letters**636**, L97–L100 (2006). DOI 10.1086/500130
16. Eberhart, R.C., Kennedy, J.: A new optimizer using particle swarm theory. In: Sixth International Symposium on Micromachine and Human Science, pp. 33–43 (1995)

17. Eggen, O.J., Lynden-Bell, D., Sandage, A.R.: Evidence from the motions of old stars that the Galaxy collapsed. The Astrophysical Journal**136**, 748–+ (1962). DOI 10.1086/147433
18. F. Fernández M. Tomassini, L.V.: An empirical study of multipopulation genetic programming. Genetic Programming and Evolvable Machines (2003)
19. Fedak, G., Germain, C., Neri, V., Cappello, F.: XtremWeb: A Generic Global Computing System. Proceedings of the IEEE International Symposium on Cluster Computing and the Grid (CCGRID'01) (2001)
20. Fernandez, F., Spezzano, G., Tomassini, M., Vanneschi, L.: Parallel genetic programming. In: E. Alba (ed.) Parallel Metaheuristics, Parallel and Distributed Computing, chap. 6, pp. 127–153. Wiley-Interscience, Hoboken, New Jersey, USA (2005)
21. Fernández, F., Cantú-Paz, E.: Special issue parallel bioinspired algorithms. Journal of Parallel and Distributed Computing **66**(8) (2006)
22. Foster, I., Kesselman, C.: Globus: A metacomputing infrastructure toolkit. The International Journal of Supercomputer Applications and High Performance Computing **11**(2), 115–128 (1997). URL `citeseer.ist.psu.edu/article/foster96globus.html`
23. Foundation, F.S.: Gnu lesser general public license, version 3. `http://www.gnu.org/licenses/lgpl-3.0.html`
24. Freeman, K., Bland-Hawthorn, J.: The New Galaxy: Signatures of Its Formation. Annual Review of Astronomy & Astrophysics**40**, 487–537 (2002). DOI 10.1146/annurev.astro.40.060401.093840
25. Freeman, K.C.: The Galactic spheroid and old disk. Annual Review of Astronomy & Astrophysics**25**, 603–632 (1987). DOI 10.1146/annurev.aa.25.090187.003131
26. Gagné, C., Parizeau, M., Dubreuil, M.: Distributed beagle: An environment for parallel and distributed evolutionary computations. In: Proc. of the 17th Annual International Symposium on High Performance Computing Systems and Applications (HPCS) 2003, pp. 201–208 (2003)
27. González, D.L., de Vega, F.F., Trujillo, L., Olague, G., Araujo, L., Castillo, P., Merelo, J.J., Sharman, K.: Increasing gp computing power for free via desktop grid computing and virtualization. In: Proceedings of the 17th Euromicro Conference on Parallel, Distributed and Network-based Processing, pp. 419–423. Weimar, Germany (2009)
28. Ibata, R., Irwin, M., Lewis, G.F., Stolte, A.: Galactic Halo Substructure in the Sloan Digital Sky Survey: The Ancient Tidal Stream from the Sagittarius Dwarf Galaxy. The Astrophysical Journal Letters**547**, L133–L136 (2001). DOI 10.1086/318894
29. Ibata, R., Lewis, G.F., Irwin, M., Totten, E., Quinn, T.: Great Circle Tidal Streams: Evidence for a Nearly Spherical Massive Dark Halo around the Milky Way. The Astrophysical Journal**551**, 294–311 (2001). DOI 10.1086/320060
30. Ibata, R.A., Gilmore, G., Irwin, M.J.: A Dwarf Satellite Galaxy in Sagittarius. Nature**370**, 194–+ (1994). DOI 10.1038/370194a0
31. Kennedy, J., Eberhart, R.C.: Particle swarm optimization. In: IEEE International Conference on Neural Networks, vol. 4, pp. 1942–1948 (1995)
32. Kesselman, C., Foster, I.: The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann (1999)
33. Laure, E., Fisher, S., Frohner, A., Grandi, C., Kunszt, P., Krenek, A., Mulmo, O., Pacini, F., Prelz, F., White, J., et al.: Programming the Grid with gLite. Computational Methods in Science and Technology **12**(1), 33–45 (2006)
34. Law, D.R., Johnston, K.V., Majewski, S.R.: A Two Micron All-Sky Survey View of the Sagittarius Dwarf Galaxy. IV. Modeling the Sagittarius Tidal Tails. The Astrophysical Journal**619**, 807–823 (2005). DOI 10.1086/426779

35. Lombraña, D., Fernández, F., Trujillo, L., Olague, G., Cárdenas, M., Araujo, L., Castillo, P., Sharman, K., Silva, A.: Interpreted applications within boinc infrastructure. In: Ibergrid 2008, pp. 261–272. Porto, Portugal (2008)
36. Luke, S., Panait, L., Balan, G., Paus, S., Skolicki, Z., Popovici, E., Harrison, J., Bassett, J., Hubley, R., Chircop, A.: Ecj a java-based evolutionary computation research system (2007). Http://cs.gmu.edu/ eclab/projects/ecj/
37. M. Litzkow T. Tannenbaum, J.B., Livny, M.: Checkpoint and migration of unix processes in the condor distributed processing system. Tech. rep., University of Wisconsin (1997)
38. Newberg, H.J., Yanny, B., Cole, N., Beers, T.C., Re Fiorentin, P., Schneider, D.P., Wilhelm, R.: The Overdensity in Virgo, Sagittarius Debris, and the Asymmetric Spheroid. The Astrophysical Journal**668**, 221–235 (2007). DOI 10.1086/521068
39. Newberg, H.J., Yanny, B., Rockosi, C., Grebel, E.K., Rix, H.W., Brinkmann, J., Csabai, I., Hennessy, G., Hindsley, R.B., Ibata, R., Ivezić, Z., Lamb, D., Nash, E.T., Odenkirchen, M., Rave, H.A., Schneider, D.P., Smith, J.A., Stolte, A., York, D.G.: The Ghost of Sagittarius and Lumps in the Halo of the Milky Way. The Astrophysical Journal**569**, 245–274 (2002). DOI 10.1086/338983
40. de la O, F.C., Guisado, J.L., Lombraña, D., Fernández, F.: Una herramienta de programación genética paralela que aprovecha recursos públicos de computación. In: V Congreso Español sobre Metaheurísticas, Algoritmos Evolutivos y Bioinspirados, vol. 1, pp. 167–173. Tenerife, Spain (2007)
41. Purnell, J., Magdon-Ismail, M., Newberg, H.: A probabilistic approach to finding geometric objects in spatial datasets of the Milky Way. In: Proceedings of the 15th International Symposium on Methodoligies for Intelligent Systems (ISMIS 2005), pp. 475–484. Springer, Saratoga Springs, NY, USA (2005)
42. S., P.V., Ian, B., Jarrod, C., P., E.S., Siraj, K., M., L.S., Rhee, Y.M., R., S.M., D., S.C., J., S.E., Bojan, Z.: Atomistic protein folding simulations on the submillisecond time scale using worldwide distributed computing. Biopolymers **68**, 91–109 (2003)
43. Samples, M., Daida, J., Byom, M., Pizzimenti, M.: Parameter sweeps for exploring GP parameters. GECCO 2005: Proceedings of the 2005 workshops on Genetic and evolutionary computation pp. 212–219 (2005)
44. Searle, L., Zinn, R.: Compositions of halo clusters and the formation of the galactic halo. The Astrophysical Journal**225**, 357–379 (1978). DOI 10.1086/156499
45. Sintes, A.M.: Recent results on the search for continuous sources with ligo and geo600. arXiv.org (2005)
46. Storn, R., Price, K.: Minimizing the real functions of the icec'96 contest by differential evolution. In: Proceedings of the IEEE International Conference on Evolutionary Computation, pp. 842–844. Nagoya, Japan (1996)
47. Szymanski, B., Desell, T., Varela, C.: The Effects of Heterogeneity on Asynchronous Panmictic Genetic Search. In: Parallel Processing and Applied Mathematics: 7th International Conference, Ppam 2007, LNCS, p. 457. Springer (2008)
48. Team, D.: distributed.net. `http://www.distributed.net`
49. Team, D.: Optimal golomb ruler. `http://www.distributed.net/ogr/`
50. Team, G.: Greate internet mersenne prime search. `http://www.mersenne.org`
51. Tomassini, M.: Spatially Structured Evolutionary Algorithms. Springer (2005)
52. Trujillo, L., Olague, G.: Automated Design of Image Operators that Detect Interest Points. pp. 483–507. MIT Press (2008)
53. Vivas, A.K., Zinn, R., Andrews, P., Bailyn, C., Baltay, C., Coppi, P., Ellman, N., Girard, T., Rabinowitz, D., Schaefer, B., Shin, J., Snyder, J., Sofia, S., van Altena, W., Abad, C., Bongiovanni, A., Briceño, C., Bruzual, G., Della Prugna, F., Herrera, D., Magris, G., Mateu, J., Pacheco, R., Sánchez, G., Sánchez, G., Schenner, H., Stock, J., Vicente, B., Vieira, K., Ferrín, I., Hernandez, J., Gebhard, M., Honeycutt, R., Mufson, S., Musser, J., Rengstorf, A.: The QUEST RR Lyrae

Survey: Confirmation of the Clump at 50 Kiloparsecs and Other Overdensities in the Outer Halo. The Astrophysical Journal Letters**554**, L33–L36 (2001). DOI 10.1086/320915

54. Willman, B., Dalcanton, J.J., Martinez-Delgado, D., West, A.A., Blanton, M.R., Hogg, D.W., Barentine, J.C., Brewington, H.J., Harvanek, M., Kleinman, S.J., Krzesinski, J., Long, D., Neilsen Jr., E.H., Nitta, A., Snedden, S.A.: A New Milky Way Dwarf Galaxy in Ursa Major. The Astrophysical Journal Letters**626**, L85–L88 (2005). DOI 10.1086/431760
55. Yanny, B., Newberg, H.J., Grebel, E.K., Kent, S., Odenkirchen, M., Rockosi, C.M., Schlegel, D., Subbarao, M., Brinkmann, J., Fukugita, M., Ivezic, Ž., Lamb, D.Q., Schneider, D.P., York, D.G.: A Low-Latitude Halo Stream around the Milky Way. The Astrophysical Journal**588**, 824–841 (2003). DOI 10.1086/374220
56. Yanny, B., Newberg, H.J., Kent, S., Laurent-Muehleisen, S.A., Pier, J.R., Richards, G.T., Stoughton, C., Anderson Jr., J.E., Annis, J., Brinkmann, J., Chen, B., Csabai, I., Doi, M., Fukugita, M., Hennessy, G.S., Ivezić, Ž., Knapp, G.R., Lupton, R., Munn, J.A., Nash, T., Rockosi, C.M., Schneider, D.P., Smith, J.A., York, D.G.: Identification of A-colored Stars and Structure in the Halo of the Milky Way from Sloan Digital Sky Survey Commissioning Data. The Astrophysical Journal**540**, 825–841 (2000). DOI 10.1086/309386
57. Zucker, D.B., Belokurov, V., Evans, N.W., Wilkinson, M.I., Irwin, M.J., Sivarani, T., Hodgkin, S., Bramich, D.M., Irwin, J.M., Gilmore, G., Willman, B., Vidrih, S., Fellhauer, M., Hewett, P.C., Beers, T.C., Bell, E.F., Grebel, E.K., Schneider, D.P., Newberg, H.J., Wyse, R.F.G., Rockosi, C.M., Yanny, B., Lupton, R., Smith, J.A., Barentine, J.C., Brewington, H., Brinkmann, J., Harvanek, M., Kleinman, S.J., Krzesinski, J., Long, D., Nitta, A., Snedden, S.A.: A New Milky Way Dwarf Satellite in Canes Venatici. The Astrophysical Journal Letters**643**, L103–L106 (2006). DOI 10.1086/505216