Evolving N-Body Simulations to Determine the Origin and Structure of the Milky Way Galaxy's Halo using Volunteer Computing

Travis Desell, Malik Magdon-Ismail, Boleslaw Szymanski, Carlos A. Varela
Department of Computer Science
Rensselaer Polytechnic Institute
Troy, NY, USA
deselt,magdon,szymansk,cvarela@cs.rpi.edu

Benjamin A. Willett, Matthew Arsenault, Heidi Newberg
Department of Physics, Applied Physics and Astronomy
Rensselaer Polytechnic Institute
Troy, NY, USA
willeb,arsenm2,heidi@rpi.edu

***Abstract*—This work describes research done by the Milky-Way@Home project to use N-Body simulations to model the formation of the Milky Way Galaxy's halo. While there have been previous efforts to use N-Body simulations to perform astronomical modeling, to our knowledge this is the first to use evolutionary algorithms to discover the initial parameters to the N-Body simulations so that they accurately model astronomical data. Performing a single 32,000 body simulation can take up to 200 hours on a typical processor, with an average of 15 hours. As optimizing the input parameters to these N-Body simulations typically takes at least 30,000 or more simulations, this work is made possible by utilizing the computing power of the 35,000 volunteered hosts at the MilkyWay@Home project, which are currently providing around 800 teraFLOPS. An open-source framework for generic distributed optimization (FGDO) is used to perform the evolutionary algorithms in conjunction the Berkeley Open Infrastructure for Network Computing (BOINC) which is used by many other volunteer computing projects, such as SETI@Home. The architecture of FGDO is described and results are presented which show that the evolutionary algorithms used correctly discover the initial parameters to N-Body simulations which match test data and find initial parameters which produce N-body simulations that accurately model tidal debris in the Milky Way Galaxy's halo.**

***Keywords*-Volunteer Computing, Evolutionary Algorithms, N-Body Simulation, Astroinformatics**

I. Introduction

MilkyWay@Home has used a probabilistic sampling method to measure the shape of stellar substructure in the Milky Way, primarily from *tidal streams*, stars that have been tidally stripped from dwarf galaxies as they are pulled apart by the Milky Way's gravity [1], [2]. This method simultaneously fits a smooth component of the Milky Way's stellar halo that is presumably the result of galaxy mergers that occurred early in the formation of the Milky Way, along with these disrupted dwarf galaxy stars around the entire galaxy. Since the Milky Way galaxy is the only galaxy for which it is possible to measure the positions and velocities of stars in three dimensions, our galaxy provides important clues to the mechanisms through which galaxies form and the nature of dark matter.

However, this approach has some limitations. As an accurate model of the smooth component (or the *background model*) is unknown, models of the disrupted dwarf galaxy stars can end up fitting errors in the background model. Additionally, the models generated from this approach only provide information about *where* these tidal streams are, not *why* they are there. This work addresses these deficiencies by using N-body simulations to model this tidal disruption of dwarf galaxies and their interaction with the Milky Way. The disruption depends on initial properties of the dwarf galaxy and on the gravitational potential of the Milky Way, which is primarily due to dark matter. Asynchronous evolutionary algorithms (AEAs) have been used successfully to find the optimal input parameters for the probabilistic sampling method [3], [4], [3], [5], [6], and this work expands these AEAs to find the optimal input parameters for the N-body simulations, which will in turn provide new understanding of the origin and structure of the Milky Way.

Successfully evolving these N-body simulations using a volunteer computing system such as MilkyWay@Home involved three main challenges. First, as volunteer computing systems consist of highly heterogeneous hosts which have unreliable availability and are potentially malicious, existing N-body simulation code had to be modified to enable checkpointing so applications can stop and restart, as well as deterministic cross-platform execution. This allows work done by hosts to be validated by comparing the results of multiple heterogeneous hosts. Second, the framework for generic distributed optimization (FGDO) used to optimize the probabilistic sampling models was improved and extended to also work with the N-body simulations. As the N-body simulation code is the second real scientific application being optimized using FGDO, this work has made steps to improve the generality and usability of that framework. Further, improvements were made to the validation strategy used by FGDO, improving its robustness while reducing validation overhead. Lastly, a method for determining how accurately an N-body simulation represents astronomical data gathered by various sky surveys, such as the Two

Micron All Sky Survey (2MASS) [7] and the Sloan Digital Sky Survey (SDSS) [8], was required so that their fitness could be optimized by the AEAs.

These N-body simulations are extremely computationally expensive. A single 32,000 particle simulation can take up to 200 hours on a standard processor, and for sufficient resolution at least 100,000 particles are required. The massive amounts of computing power produced by the volunteer computing hosts at MilkyWay@Home provide one of the few computing systems where performing such optimization in a realistic amount of time is possible. Preliminary results show that for a test data set with known optimal parameters, the 35,000 volunteered computing hosts at MilkyWay@Home can be successfully harnessed to evolve 4,096 particle N-body simulations to accurately model the Milky Way Galaxy and the formation of structure in its halo. Further, the FGDO framework is now successfully being used to optimize 32,768 particle N-body simulations to real observed data from the SDSS. The authors feel that these results show the potential for this approach of using volunteer computing grids and an asynchronous optimization framework, and that this will allow us to evolve 100,000 or more particle N-body simulations successfully.

This paper proceeds as follows. Section II presents related work. Section III describes the N-body simulations being performed, what parameters are being optimized and how the fitness of a simulation is determined. Section IV describes the improvements made to the FGDO framework used to optimize and validate the N-body simulations using the Berkeley Open Infrastructure for Network Computing (BOINC) [9]. Preliminary results are presented in Section V. Concluding remarks and directions for future work are given in Section VI.

II. Related Work

A. Distributed Evolutionary Algorithms

Evolutionary algorithms (EAs) are a popular approach for parameter optimization where the search space contains many local optima that traditional search methods such as gradient descent and simplex get trapped in. As the search space for these N-body simulations is highly complex, EAs are an ideal candidate for performing the parameter optimization.

Common EAs for continuous search spaces include differential evolution (DE) [10], particle swarm optimization (PSO) [11], [12], [13] and genetic search (GS). In general, an EA keeps track of an *population* of potential solutions, where each *individual* in the population represents a set of parameters in the search space and has a *fitness* that represents how good of a solution that individual is. As the EA progresses, new individuals are generated by recombining individuals in the current population, and those with higher fitnesses are kept while those with lower fitnesses are discarded. As the generation of new individuals involves random elements, newly generated individuals have the potential to both *explore* new regions of the search space, and *exploit* areas of the search space that are known to have good fitness. This results in the population of solutions evolving towards an optimal solution.

There have been many different approaches to making EAs work on different distributed computing systems. In general these approaches are either sequential, with distinct synchronization points; asynchronous, without distinct synchronization points; or some combination of the two.

Sequential approaches to distributed EAs typically use a single-population strategy, where new populations are generated repeatedly by evaluating each individual in parallel; repeating this process until the population has reached convergence criteria [14], [15]. It is possible to increase scalability past the population size by additionally evaluating the objective function in parallel. This type of approach is best suited to highly reliable and homogeneous computing nodes, as found in clusters and supercomputers.

Hybrid approaches involve *islands* of populations [16]. The different populations are evaluated sequentially and then asynchronously migrate selected individuals to neighboring islands when certain criteria are met [17], [18], [19], [20]. Tasoulis *et al.* have shown that having moderate values for the frequency of migration result in the best convergence for differential evolution across a variety of benchmarks [21]. It has been shown that super-linear speedup can be attained using this method, as smaller populations can converge to minima quicker than larger populations [22], [23]. However, having populations of different sizes and/or populations running on clusters of different speeds can have varying negative effects on the performance of the search. As each island can be parallelized in the same manner as a single population EA, this approach is well suited to grid computing systems, where islands can be assigned to individual clusters within the grid. Island EAs have also been shown to be effective in peer-to-peer computing systems [24], [25].

Asynchronous approaches to distributed EAs typically use a single-population and a master-worker model. One approach has been to generate the population and have workers request individuals to evaluate, using a work-stealing model [26], [27], however these approaches wait for the evaluation of one population to complete before continuing to the next. This synchronization limits their scalability to a number of processors equal to the size of the population. Asynchronous approaches to particle swarm optimization have used a method where individual particles are calculated in parallel and newly found global best positions are then broadcast asynchronously [28], [29]. However, similarly to single-population sequential approaches, these strategies are limited by the population size used by the EA. Additionally, broadcasts are not feasible on very large computing environments and very large population sizes can significantly increase the time taken to find a solution [4].

FGDO supports asynchronous versions of differential evolution, particle swarm optimization and genetic search. However, the asynchronous approach used by FGDO differs from the related approaches in that it has no synchronization points at all, allowing these optimization methods to scale to hundreds of thousands or more computing hosts, as shown by Desell [4]. The asynchronous optimization approach used by FGDO generates new individuals based on the current state of a population in response to requests for work and later inserts the results to the population if and when they are reported. As the heuristics for generating new individuals are randomized, this allows the AEAs to generate as many unique individuals as are required to satisfy all potential workers. In addition, there are no dependencies between generated individuals, so if a worker fails and does not report the fitness of its individual, the search does not need to wait for that individual to be recalculated. Section IV describes FGDO in more detail.

B. Volunteer Computing

Volunteer computing enables people across the world to volunteer their computing resources, such as processors, graphics processing units (GPUs) and hard drive space. Popular projects include SETI@Home [30], the infrastructure of which was generalized to become the Berkeley Open Infrastructure for Network Computing (BOINC) [9], IBM's World Community Grid[1] and Stanford's Folding@Home [31]. Users can even volunteer non-standard computing units such as gaming consoles like the XBOX 360 and Playstation 3. These computing frameworks provide very powerful distributed computing environments consisting of thousands (or even millions) of personal computers for the low price of running a server. In addition, they provide an effective means of generating public interest in different scientific computing projects.

However, these benefits do come at some cost. As these projects are open to the public, individual computing hosts are highly unreliable and potentially malicious. As any volunteer has the right to stop participating at their own discretion, in order to prevent lost work applications running on volunteered hosts must be able to save their state and restart to prevent lost work. Additionally, there is the potential for users to return invalid or incorrect results to unfairly gain *credit*, a measure of a user's participation in a project. The common solution to this issue is to use validation of results, where the same piece of work is issued to multiple clients and then the results are compared. However, as the volunteered hosts are highly heterogeneous, ensuring that different hosts return the same results for the same work can be problematic. Validation is also highly expensive, as it requires multiple hosts to do the same work.

[1]http://www.worldcommunitygrid.org

Section III-B describes how checkpointing was implemented in the N-body simulation code, as well has how checkpointing and execution were done in order to ensure that the results were deterministic across heterogeneous architectures. The issue of expensive validation is addressed by using FGDO, which provides methods to dramatically reduce the amount of validation required for volunteer computing projects performing evolutionary algorithms and this is described in detail in Section IV-A.

C. N-Body Simulations in Astroinformatics

N-body simulations are a well established tool for modeling tidal disruption in the Milky Way. The Sagittarius Dwarf Tidal Stream was initially modeled by Johnston *et al.* [32], and followed up by Law *et al.* [7]. While this placed some constraints on the kinematics of the Sagittarius dwarf, Law *et al.* were unable to simultaneously fit the kinematics and sky positions of the Sagittarius stream within an axisymmetric dark matter halo. Only by expanding to a triaxial halo did Law & Majewski [33] satisfy all constraints. Predating this work, Dehnen *et al.* [34] modeled the Palomar 5 globular cluster tidal stream via N-body simulations and showed that the majority of its properties were results of its orbital kinematics. Similar studies of the GD-1 (Grillmair & Dionatos [35] stellar stream by Willett *et al.* [36] and Koposov *et al.* [37] were able to determine orbital kinematics, but did not perform N-body simulations.

While these studies were groundbreaking in their ability to constrain tidal streams, they did not address the interesting research question: can N-body simulations be used to rigorously fit the stellar density along a tidal stream? Newberg *et al.* [38] published a re-analysis of the Orphan Stream (Belokurov *et al.* [39], Grillmair [40]), and extracted the density of Orphan Stream F-turnoff stars as a function of Orphan Stream longitude $\Lambda_{\mathrm{Orphan}}$, which is shown in Figure 1. Newberg *et al.* [38] were able to reproduce the overall form of the Orphan density using a Plummer model with mass $M_P = 2 \times 10^6\ M_{\mathrm{Sun}}$ (where M_{Sun} is the mass of the sun), scale length $r_s = 0.2$ kpc (kiloparsecs), orbit time $t_{\mathrm{back}} = 4$ Gyr (gigayears) and evolution time $t_{\mathrm{back}} = 3.945$ Gyr, evolved along the best fit orbit within a Galactic potential. While these parameters produce a model that broadly reproduces the Orphan Stream density, an interesting research question emerges: can an N-body model of the Orphan Stream progenitor actually be fit to the Orphan stream density?

Section III-A provides a proof of concept that it is possible for an N-body simulation to produce a body of particles which matches observed data. Section V-A shows that using FGDO on MilkyWay@Home can discover N-body simulations that match the results of an N-body simulation with known parameters and Section V-B provides preliminary results that evolving N-body simulations to match actual data

provides results with accuracy similar to that achieved with the test data.

III. Modeling the Milky Way Galaxy using N-Body Simulations

The scientific purpose of this work is to utilize the BOINC volunteer computing environment to perform distributed gravitational N-body simulations of dwarf galaxies orbiting the Milky Way. A dwarf galaxy is a small spherical galaxy that typically possesses millions of stars and has a mass on the order of one ten-thousandth of the Milky Way's mass. As it orbits our Galaxy, it becomes disrupted by gravity and forms tidal streams: long arms of stars that can span the entire sky. Utilizing massive and well calibrated photometric surveys such as the Two Micron All Sky Survey (2MASS) [7] and the Sloan Digital Sky Survey (SDSS) [8], astronomers have identified tens of streams orbiting the Milky Way. Figure 1 shows a stellar density map of SDSS F-turnoff stars in the Milky Way halo from [38]. Darker areas indicate higher stellar density. There are two tidal streams in this Figure. The first runs nearly vertically from $l = 200°$ to $l = 240°$. This is the Sagittarius Dwarf Tidal Stream. The other, running horizontally at $b \approx 50°$, is the Orphan Stream.

The physical problem in understanding tidal streams is that they represent the disordered state of the original dwarf galaxy: they have already been disrupted. How can we determine the properties of the original dwarf galaxy that created the stream? The simplest way to resolve this difficulty is to understand the kinematics of the stream, propagate an orbit back in time to a previously ordered state, and propagate a collection of particles forward in time to the present day. We can understand the kinematic properties of the stream by determining is velocity and distance at various points along the sky. For most purposes, the radial velocity (velocity along the line of sight) is the only knowable velocity component. Some stars have known proper motions, which would allow other velocity components to be determined, but their errors are often so large as to preclude their use. Knowing the line of sight kinematics of and distance to the stream, we can use search algorithms to find the best fit three dimensional kinematics and background Galactic model [36].

With the orbit of the stream understood, we can now create a group of particles at some time in the past, place it on this orbit, and propagate it forward to the present day. The model from the group of particles is a Plummer Sphere, which is an energetically stable three dimensional spherical distribution [41]. This model has two parameters: its total mass M and scale length a. In addition, we will consider two more parameters: t_{orbit}, the amount of time the orbit is evolved back in time, and t_{dwarf}, the amount of time the dwarf is evolved forward in time. In order to determine the parameters of the best fit model dwarf galaxy, we need some

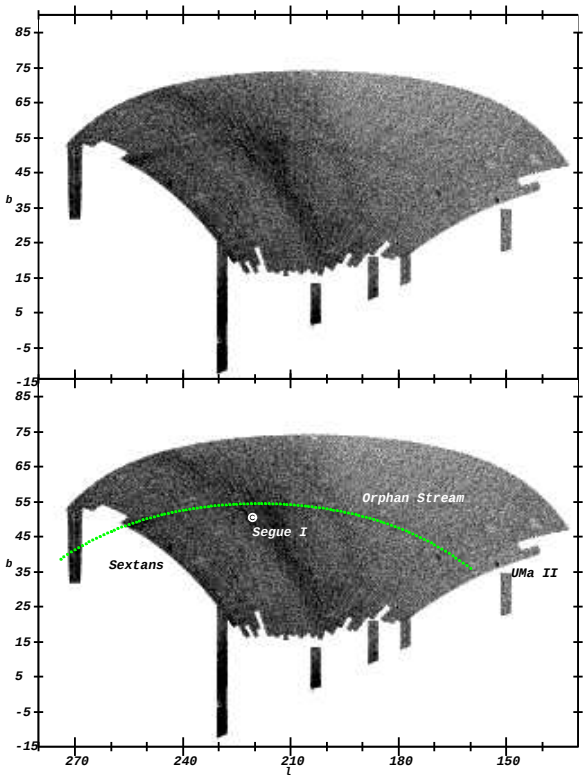

Figure 1. Shown is the sky position of the Orphan Stream as traced by F turnoff stars from the Sloan Digital Sky Survey [38]. Darker areas indicate higher stellar density. There are two tidal streams in this Figure. The first runs nearly vertically from $l = 200°$ to $l = 240°$. This is the Sagittarius Dwarf Tidal Stream. The other, running horizontally at $b \approx 50°$, is the Orphan Stream. The Sextans and Ursa Major II dwarf galaxies are labeled in the lower panel.

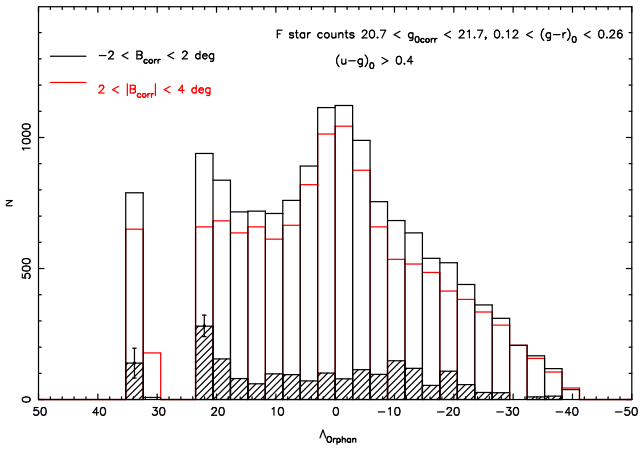

Figure 2. Number counts of F turnoff stars within $\pm 2°$ of the stream are plotted as an open black histogram [38]. The number of background turnoff stars off-the-stream on either side are plotted in red. The difference is plotted as a hashed histogram. Note the significant excess of turnoff stars over background near $\Lambda_{\rm Orphan} = +23°$, corresponding to $(l, b) = (255°, 49°)$.

measure of how the stars are distributed in the stream. The density of stars along the stream as a function of angle on the sky provides this very measure.

The hashed histogram in Figure 2 shows the Orphan stream stellar density as a function of Orphan stream longitude ($\Lambda_{\rm Orphan}$, a spherical coordinate system whose equator is along the stream). Note that the gap in the histogram around $\Lambda_{\rm Orphan} = 25^\circ$ is the same gap in Figure 1 at $l = 260^\circ$, and thus is not a true absence of stars. We also see that the high density of stars near $(l, b) = (255^\circ, 49^\circ)$ corresponds with a peak in stellar density at $\Lambda_{\rm Orphan} \approx +23^\circ$. We wish to determine the four parameters of the model dwarf galaxy that best fits the density profile given in Figure 2. Our metric for determining the goodness of fit to the density profile is given in Equation 1:

$$\chi^2 = \sum_i \left(\frac{\eta_{i,model} - \eta_{i,data}}{\sigma_i} \right)^2 \qquad (1)$$

where $\eta_i = N_i / N_{\rm total}$ is the normalized bin height, $\sigma_i = \sqrt{N_i}/N_{\rm total}$ is the normalized error of a data bin, and N_i is the number of stars in bin i, and $N_{\rm total}$ is the total number of stars in the histogram.

A. Proof of Concept

A proof of concept simulation can be generated by selecting a dwarf mass of $M = 1 \times 10^6 * M_{Sun}$ (where M_{Sun} is the mass of the sun), a scale radius of $a = 0.2$ kpc (kiloparsecs), and evolution times $t_{orbit} = 4.0$ Gyr (gigayears) and $t_{dwarf} = 3.945$ Gyr. A simulation with these parameters within the low halo mass model described in [38] produces the density profile given in Figure 3. As can be seen, this density has the same overall shape as the stream density. A density plot in galactic coordinates is shown in Figure 4. Comparison between this and Figure 1 shows a stream with an overdensity in the same area in the sky, as well as a stream of approximately the same length. This simple test shows that physically intuitive parameters lead to a satisfactory result. However, the histograms need to be directly and objectively compared using the goodness of fit metric in order to find the optimal parameters.

B. N-Body Simulation Code

To perform N-Body simulations on a BOINC volunteer computing grid, Barnes and Hut [42] treecode has been modified. This treecode uses a hierarchical method of N-body simulation, which results in a faster O($N\ log\ N$) runtime, where N is the number of bodies[2].

This treecode, further described and parallelized by Dubinski [43], operates by grouping particles into cell, each with eight siblings. At the beginning of the simulation, all particles are enclosed by a cell, which is then subdivided into eight subcells. A tree of subcells is created until each

[2]http://www.ifa.hawaii.edu/ barnes/treecode/treeguide.html

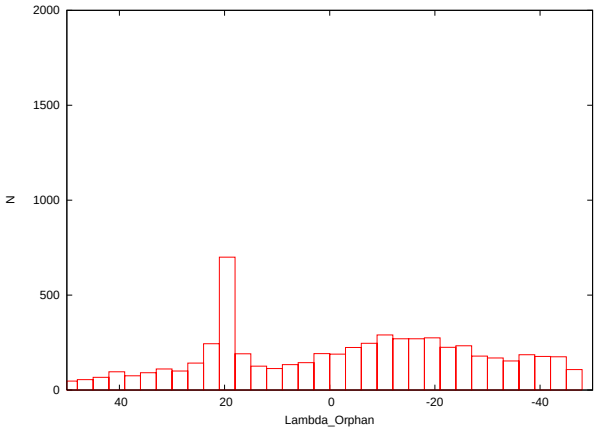

Figure 3. Number counts of simulated stream stars on the same axis as the data histogram. The evolution time of the dwarf directly determines the placement of the peak near $\Lambda_{\rm Orphan} = +23^\circ$ and the length of the stream while the mass and scale length determine the ratio between this peak and the number of stars in the tail.

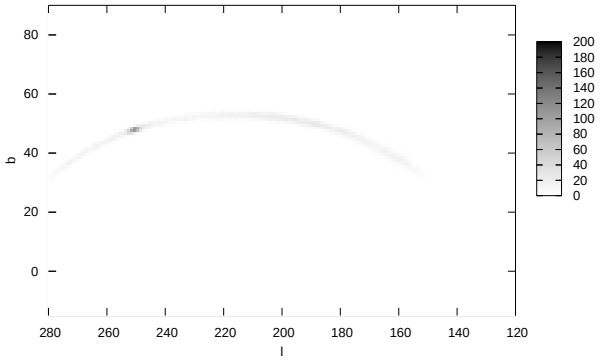

Figure 4. Shown is an N-body particle density plot in sky coordinates. Darker colors indicate higher particle density. Comparison with Figure 1 shows an overdensity near $l = 250^\circ$ as well as a stream of comparable length. This simple test shows that physically intuitive parameters lead to a satisfactory result.

cell contains only one particle. The force on each particle is evaluated by "walking" down the tree. If a particular cell is "too distant", it contributes en-masse to the force. However, if it is "too close", the cell is "opened" and the force is evaluated for the subcells.

The opening angle parameter θ determines if a cell is "too distant" and "too close". If the size of a cell is l and the distance of the particle to the cell's center of mass is d, the cell is accepted for force evaluation if:

$$d > \frac{l}{\theta}. \qquad (2)$$

Smaller values of θ therefore give rise to more precise

force evaluations. A θ value of 1 typically results in acceleration errors of one percent compared to the full N^2 force algorithm [44].

Checkpointing has been implemented which allows the N-body simulations to be restarted when clients stop or pause the BOINC client that runs the N-body simulation. As the hosts at MilkyWay@Home are volunteered, this checkpointing minimizes the amount of work lost by volunteers using their computers. Checkpointing can be done after each timestep. Typically the BOINC client determines checkpointing is required every few minutes. The positions and velocities of the particles, as well as the simulation time are saved in a binary format. This information is sufficient to resume the simulation. The binary format ensures this is a lossless process, avoiding inconsistencies in string to floating point conversions present in nearly every compiler.

In addition, the treecode has been adapted to make it easier to add and use different initial distributions of particles for the dwarf model, such as Plummer models [41] as well as a selection of different components for an external acceleration due to the Milky Way, such as spherical bulges [45], exponential disks [46], and dark matter halos [7], [47].

The last obstacle for viable use of N-body simulation code on BOINC was to ensure the consistency of floating point results from different systems. The volunteered hosts at MilkyWay@Home are highly diverse, consisting of practically every common type of architecture and operating system available. Other BOINC projects, in particular LHC@home, have had similar problems in the past [48], [49]. Some of these projects have used *homogeneous redundancy*, where duplicate work for validation is only sent to hosts with similar architectures so that their results are comparable [50]. However, this approach is not particularly useful for optimization, as in order to evolve a population of simulations, all their fitnesses need to be comparable, regardless of the host that performed the N-body simulation and produced the fitness. With homogeneous redundancy, potentially valid fitnesses could be discarded because the architecture they were calculated on resulted in a lower fitness than a similar simulation on a different architecture.

Because of this, the code was modified so that it returned fitness values within an acceptable tolerance of $10e^{-13}$ across all architectures being used. The lack of standardization in precision of most of the standard C math library functions needed to be addressed to achieve this. Over the long life of the simulation, the small differences accumulate into rather different results from the same initial conditions. This lack of standardization was solved by using the CRlibm library[3], which provides efficient implementations of the double-precision C99 standard elementary functions with rounding correct to the IEEE-754 rounding modes. The result of this is N-body simulation code which produces verifiable results across heterogeneous computing platforms. Given results generated by hosts at MilkyWay@Home sampled over a week, only 0.6% of the results failed validation, which we feel is low enough to attribute to poorly configured clients and hardware issues. This modified code has also been made freely available as a public repository on GitHub[4].

[3]http://lipforge.ens-lyon.fr/www/crlibm/

IV. A Framework for Generic Distributed Optimization (FGDO)

FGDO has made a series of improvements enabling its use for the optimization of N-body simulations in addition to the probabilistic sampling method, making it more generic and easier to use with other computing projects. It is also available as public repository on GitHub for public use[5]. The new implementation has been done in Java, which allows for easier extension of the search methods being used because of its Object oriented nature. In addition, Java has made it much simpler to plug in different credit and validation implementations for the different applications being used.

A. Reducing Validation Overhead

The previous implementation of FGDO could use either *optimistic* or *pessimistic* validation to reduce the amount of duplicate work done by volunteered hosts [51]. These approaches involve keeping track of two populations, one of unvalidated results and the other of validated results. The pessimistic approach either generates duplicate individuals from the unvalidated population for verification, or new individuals from recombinations of the validated population. A user defined verification rate is used to determine how frequently duplicate work for validation is generated. The optimistic approach still uses the unvalidated population to generate duplicate work for validation, but it also uses it for recombination of new individuals, with the assumption that most unvalidated results are correct. When an individual in the unvalidated population has been found incorrect, then it is replaced with an individual known to be correct from the validated population. As with pessimistic validation, a user defined verification rate is used to determine how frequently duplicate results are generated for validation.

This previous implementation suffers from some drawbacks. First, the verification rate has a significant impact on the speed that the search progresses, and a fixed verification rate is not an optimal solution. This is because as a search progresses, it becomes harder to find individuals that potentially improve the populations. If the verification rate is low, then it can be a long time before a new good solution is actually used to generate new individuals in the case of pessimistic validation. In the case of optimistic validation, if the verification rate is too low then erroneous results last longer in the populations generating more potentially

[4]http://github.com/Milkyway-at-home/milkywayathome_client

[5]http://github.com/Milkyway-at-home/fgdo_java

poor new individuals. Another issue is that many results are simply not validated, as they will not potentially improve the population. This makes it easier for malicious hosts to cheat by reporting bad results as there is a decent chance they will still receive credit for them when they do not need to be validated.

The new implementation solves these problems by combining optimistic and pessimistic validation with BOINCs quorum and adaptive replication schemes (which are described in detail in [52]). This is done as follows:

- When the queue of available work is low, new individuals are generated through recombination from the unvalidated population if the optimistic approach is used, otherwise they are generated from the validated population. Newly generated work has an initial quorum of one.
- When a result is reported for work with a quorum of size one and it cannot improve the validated population, it is validated with a chance equal to a host's error rate. A host's error rate is initialized to 0.1 (meaning 10% of its results will be validated). When a host returns a correct result (one that is validated successfully against another result), the error rate is multiplied by 0.95. When a host returns an incorrect result (one that is validated unsuccessfully against other results that match each other), its error rate is increased by 0.1. The error rate also cannot fall below 0.1 or go above 1.0. In this way, hosts which frequently return bad results quickly reach the point where all their results are validated, thus not earning credit for bad results. On the other hand, hosts which frequently return good results only have a few of their results validated when it is not needed.
- When a result is reported for work with a quorum of size one and it can improve the population, FGDO will try to insert the individual into its unvalidated population. In addition, the quorum for this piece of work is increased to the amount specified by the project, which will cause the BOINC scheduler to send out copies of this work for verification.
- When results have been reported that complete a quorum and enough results match to successfully determine a canonical result, the canonical result is inserted into the validation population. Any of these results that do not match the canonical result (and thus are invalid) are removed from the unvalidated population.
- When results have been reported that complete a quorum but there are not enough that match to successfully determine a canonical result, the quorum size is again increased to allow the BOINC scheduler to generate more copies of this work for validation.

In this way, the user defined verification rate no longer required, as the BOINC scheduler will take care of the frequency in which duplicate work is sent out to hosts for verification and will try and generate it in a manner that verifies the most important results first (the ones with the shortest deadline). This allows the BOINC computing project to spend more time on verification when it is needed, and more time on exploration when not many results require verification. Additionally, it significantly reduces the amount of credits a malicious or broken host can gain by returning bad results by adaptively modifying how frequently a host's results are verified based on their previous performance.

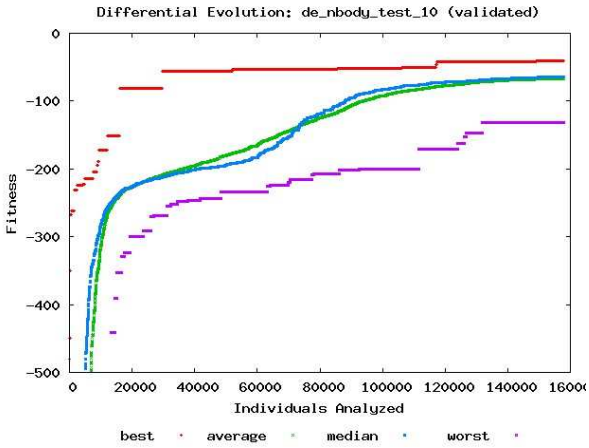

Figure 5. Progress of the best, average, median and worst validated individuals for an asynchronous differential evolution search over the search space $M_P = 0.22 \ldots 11.11 \times 10^6\ M_{\mathrm{Sun}}$, $r_s = 0.05 \ldots 1.0$ kpc, $t_{orbit} = 1 \ldots 5$ Gyr and $t_{dwarf} = 1 \ldots ; 5$ Gyr, with a population of 300 individuals. The fitness of these 4096 particle N-body simulations is calculated by comparing their stellar density histogram to the histogram of an N-body simulation with dwarf parameters $(M_P, r_s, t_{orbit}, t_{back}) = (3.6 \times 10^6\ M_{\mathrm{Sun}}, 0.2\ \mathrm{kpc}, 4\ \mathrm{Gyr}, 3.945\ \mathrm{Gyr})$.

B. Search Progress and Error Logging

The new implementation of FGDO also has improved result and error logging. The progress of all searches is exported to logs and scripts are provided which generate graphical representations of their progress. For an example, MilkyWay@Home displays these graphs showing the progress of current searches at http://milkyway.cs.rpi.edu/milkyway/plots.php, which not only provides an easy way for the project's scientists to see how the searches are progressing, but also gives the project's users an indication of how things are performing. Error logging was improved by having search specific error logs, which record the user, host and error information for every invalidated result. This allows for easier debugging of the distributed system on a search by search basis.

V. Results

A. Comparison to Known Test Data

As a test to see the potential for using AEAs to optimize the parameters to these N-body

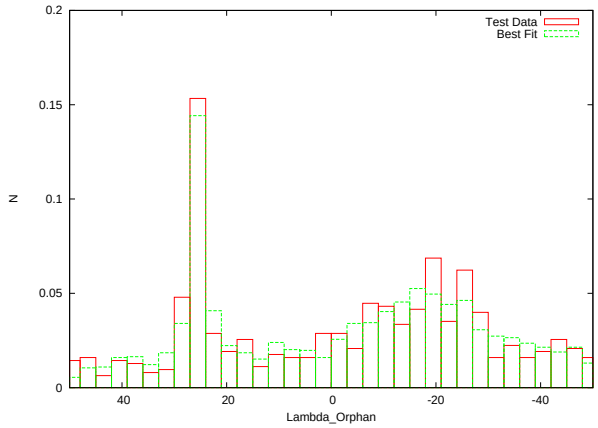

Figure 6. Simulated Orphan Stream stellar density modeled via Barnes & Hut treecode. The solid red histogram is the n-body simulation of the Orphan Stream orbit was performed using the parameters from Newberg *et al.* [38], dwarf parameters $(M_P, r_s, t_{orbit}, t_{back}) = (3.6 \times 10^6 \ M_{\rm Sun}, 0.2 \ \text{kpc}, 4 \ \text{Gyr}, 3.945 \ \text{Gyr})$. The dotted green histogram is the best fit found to this histogram using FGDO on MilkyWay@Home, $(M_P, r_s, t_{orbit}, t_{dwarf}) = (3.591 \times 10^6 \ M_{\rm Sun}, 0.22 \ \text{kpc}, 3.97 \ \text{Gyr}, 3.91 \ \text{Gyr})$.

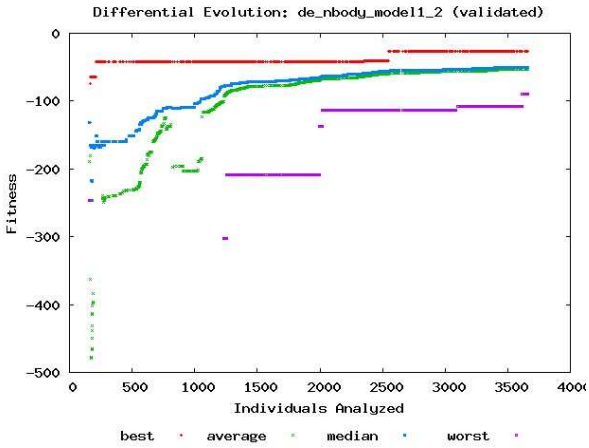

Figure 7. Progress of the best, average, median and worst validated individuals for an asynchronous differential evolution search over the search space $M_P = 0.22 \ ... \ 11.11 \times 10^6 \ M_{\rm Sun}$, $r_s = 0.05 \ ... \ 1.0$ kpc, $t_{orbit} = 1 \ ... \ 5$ Gyr and $t_{dwarf} = 1 \ ... \ ; 5$ Gyr, with a population of 300 individuals. The N-body simulations consist of 32,768 particles. Model 1 utilizes an exponential disk and NFW halo profile with an enclosed mass of $M_{60} = 40 \times 10^{10} M_{Sun}$.

simulations, a 4096 particle N-body simulation was performed using the parameters from Newberg *et al.* [38], dwarf parameters $(M_P, r_s, t_{orbit}, t_{back}) = (3.6 \times 10^6 \ M_{\rm Sun}, 0.2 \ \text{kpc}, 4 \ \text{Gyr}, 3.945 \ \text{Gyr})$. Asynchronous differential evolution was then used with a population of size 300, best parent selection, and binary recombination with a crossover rate of 0.5, a pair weight of 0.5, or DE/best/1/bin (for more detail on differential evolution variants, see Mezura-Montes *et al.* [53]). The search space given was $M_P = 0.22...11.11 \times 10^6 \ M_{\rm Sun}$, $r_s = 0.05...1.0$ kpc, $t_{orbit} = 1...5$ Gyr and $t_{dwarf} = 1...5$ Gyr. Figure 5 shows the progress of the fitness of the best, average, median and worst individuals in the validation population for this search and Figure 6 compares the stellar density histograms of the known test data to the parameters of the best fit individual found at the end of the search.

Some discrepancies arose because clients used a random seed to generate the initial particle distribution and with the N-body simulations using only 4096 particles the initial distribution played a large factor in the final stellar density model. As this initial distribution was due to randomly generated seeds, the search space ended up being highly noisy. For example, using the best fit parameters found by the search, different seeds resulted in fitness values from anywhere between -30 to -1200. However, in spite of this noisy search space, asynchronous differential evolution was able to find parameters quite similar to what the test data was generated from: $(M_P, r_s, t_{orbit}, t_{dwarf}) = (3.591 \times 10^6 \ M_{\rm Sun}, 0.22 \ \text{kpc}, 3.97 \ \text{Gyr}, 3.91 \ \text{Gyr})$, which also had very similar histogram to the test data (as shown in Figure 6). The authors feel that this shows that this approach is not

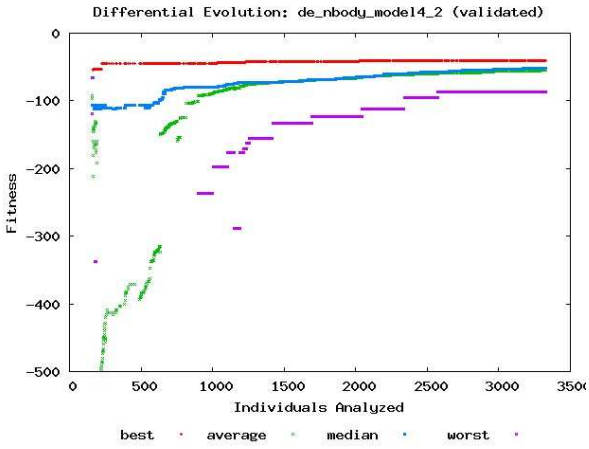

Figure 8. Progress of the best, average, median and worst validated individuals for an asynchronous differential evolution search over the search space $M_P = 0.22 \ ... \ 11.11 \times 10^6 \ M_{\rm Sun}$, $r_s = 0.05 \ ... \ 1.0$ kpc, $t_{orbit} = 1 \ ... \ 5$ Gyr and $t_{dwarf} = 1 \ ... \ ; 5$ Gyr, with a population of 300 individuals. The N-body simulations consist of 32,768 particles. Model 4 is a standard Galactic model, using a Miyamoto-Nagai disk and logarithmic halo and having an enclosed mass of $M_{60} = 47 \times 10^{10} M_{Sun}$.

only feasible, but highly robust. Further, using larger N-body simulations, which reduces the noise caused by variance in the initial distribution results in much faster convergence to good fitness values by the search population, as shown in Section V-B.

B. Comparison to Actual Data

With the test simulations producing results with consistent parameters, fits are currently being run on the true Orphan

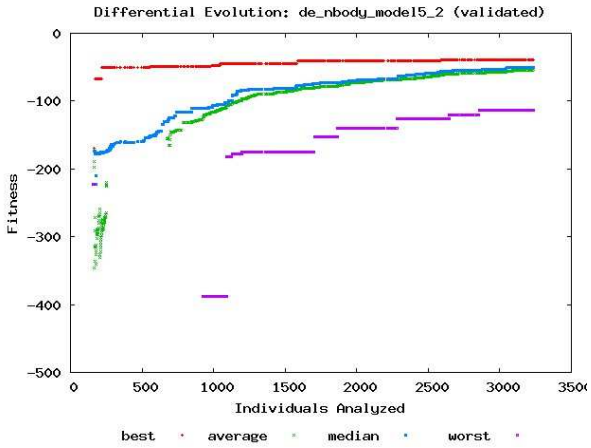

Figure 9. Progress of the best, average, median and worst validated individuals for an asynchronous differential evolution search over the search space $M_P = 0.22$ ... 11.11×10^6 $M_{\rm Sun}$, $r_s = 0.05$ ... 1.0 kpc, $t_{orbit} = 1$ ... 5 Gyr and $t_{dwarf} = 1$ ... ; 5 Gyr, with a population of 300 individuals. The N-body simulations consist of 32,768 particles. Model 5 is the same as Model 4 except with a lower mass of $M_{60} = 26.4 \times 10^{10} M_{Sun}$. This model was the best fit to the Orphan Stream distances.

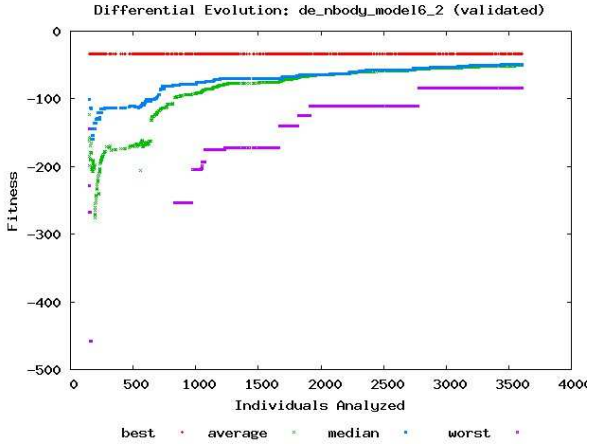

Figure 10. Progress of the best, average, median and worst validated individuals for an asynchronous differential evolution search over the search space $M_P = 0.22$ ... 11.11×10^6 $M_{\rm Sun}$, $r_s = 0.05$ ... 1.0 kpc, $t_{orbit} = 1$ ... 5 Gyr and $t_{dwarf} = 1$ ... ; 5 Gyr, with a population of 300 individuals. The N-body simulations consist of 32,768 particles. Model 6 is a hybrid model that combines a Miyamoto-Nagai disk and NFW halo, with a mass of $M_{60} = 43.5 \times 10^{10} M_{Sun}$.

Stream stellar density obtained from the SDSS sky survey. N-body simulations are being run within seven models of the Orphan Stream kinematics and Galactic potential from [38]. These models consist of the best fit orbits to the Orphan Stream kinematics in a variety of Galactic potentials. The aim of this work is to test the dependence of the Orphan Stream progenitor parameters (mass, scale length, and evolution time) on the various Galactic potential models. Dwarf parameters that depend on the Galactic potential could be a powerful probe into the structure of the Milky Way.

Figures 7, 8, 9 and 10 show the progress of a selection of these models after approximately a week being run on MilkyWay@Home. As the average N-body simulation time is around 15 hours on a typical computer, to achieve this amount of progress for single one of these searches would take over 5 years on a single computer. These results show that not only can MilkyWay@Home perform multiple N-Body simulation optimizations concurrently (while also computing it's other optimization problem), it can also provide results in a reasonable amount of time for scientific progress.

These searches are being run with the same search space and search parameters as the test data, except the population size has been lowered to 100, as 300 was a bit high for only 4 optimization parameters. Additionally, the size of the N-body simulations has been increased to 32,768 particles, which should provide more accurate results with less variance based on the seed of the initial distribution. The figures show the progress of the fitness of best, average, median and worst individuals in the evolutionary algorithm populations used to perform the optimization; where each individual of the population represents a different N-body simulation and its fitness is how well that simulation matched the stars observed by the SDSS.

The various models interchange disk and halo gravitational potentials to find the best fit combination. They are characterized by their mass enclosed within 60 kpc of the Galactic center, M_{60}. Model 1 utilizes an exponential disk and NFW halo profile with an enclosed mass of $M_{60} = 40 \times 10^{10} M_{Sun}$. Model 4 is a standard Galactic model, using a Miyamoto-Nagai disk and logarithmic halo and having an enclosed mass of $M_{60} = 47 \times 10^{10} M_{Sun}$. Model 5 is the same as Model 4 except with a lower mass of $M_{60} = 26.4 \times 10^{10} M_{Sun}$. This model was the best fit to the Orphan Stream distances. Model 6 is a hybrid model that combines a Miyamoto-Nagai disk and NFW halo, with a mass of $M_{60} = 43.5 \times 10^{10} M_{Sun}$.

The progress of these different searches shows that reducing the population size along with a large number of particles in the N-body simulation has a dramatic effect on the convergence rates of the searches. In less than 4,000 evaluations the populations have already reached similar or better fitnesses to the 4096 particle test data N-body simulations after 150,000 evaluations. This means that the histogram made by the final state of the best N-Body simulations found matches the histogram of observed stars as well as or better than the histograms in Figure 6. The authors feel that this provides evidence that larger N-body simulations will be able to provide even better models of the Milky Way galaxy's halo, and increasing the number of particles can potentially improve the number of evaluations required to reach a good fit as the noise due to the random

seeding of the initial particle distribution in the search space decreases. The current progress of these and the other searches being done by MilkyWay@Home are available at: http://milkyway.cs.rpi.edu/milkyway/plots.php.

VI. Conclusion

This work presents preliminary results showing that a large scale volunteer computing project such as MilkyWay@Home can successfully evolve N-body simulations to model the formation of debris in the Milky Way galaxy's halo. Good results are being achieved with 32,768 particle simulations, and 100,000 or more particle simulations are exepected to provide even better results. This was made succesfull by modifying existing N-body simulation code to allow for different galactic models and initial particle distributions, and using the CRlibm math library so that results are uniform across the over 35,000 heterogeneous volunteered computing hosts at MilkyWay@Home. In addition, this work has involved improvements to a framework for generic distributed optimization (FGDO), which can scalably run asynchronous evolutionary algorithms using BOINC with minimal validation overhead, and provides various tools for displaying the progress of these searches and finding errors in the applications run on volunteered hosts.

This work also opens up many future avenues for future research. For example, the parameters of the different models being tested against the actual data could be optimized as well. Further, the type of initial particle distribution and type of model of the Milky Way could also become optimization parameters. This could lead to even more accurate representations of the formation of the Milky Way galaxy.

Volunteered GPUs provide the majority of the results for the probabilistic sampling method done on MilkyWay@Home. Work is in progress to run the N-body simulations on GPUs using OpenCL. Various teams have attempted to run Barnes-Hut treecode derivatives on GPUs in the past, with mixed performance results. Using some lessons from these previous attempts, as well as from computer graphics techniques to take advantage of GPU architecture, there will hopefully be substantial performance increases over standard CPUs.

In addition to providing a solution to the immediate problem of tidal stream modeling, this method can also be extended to other disciplines. Some examples include models of electromagnetic phenomena (namely charges in external fields), as well as molecules which bond to create organic compounds. The external potentials currently used are Galaxy specific but others can be easily added. Ultimately, any process that models the interactions of particles in an attempt to minimize or maximize a quantity can benefit from this method.

References

[1] N. Cole, H. Newberg, M. Magdon-Ismail, T. Desell, K. Dawsey, W. Hayashi, J. Purnell, B. Szymanski, C. A. Varela, B. Willett, and J. Wisniewski, "Maximum likelihood fitting of tidal streams with application to the sagittarius dwarf tidal tails," *Astrophysical Journal*, vol. 683, pp. 750–766, 2008.

[2] N. Cole, "Maximum likelihood fitting of tidal streams with application to the sagittarius dwarf tidal tails," Ph.D. dissertation, Rensselaer Polytechnic Institute, 2009.

[3] T. Desell, B. Szymanski, and C. Varela, "Asynchronous genetic search for scientific modeling on large-scale heterogeneous environments," in *17th International Heterogeneity in Computing Workshop*, Miami, Florida, April 2008.

[4] T. Desell, "Asynchronous global optimization for massive scale computing," Ph.D. dissertation, Rensselaer Polytechnic Institute, 2009.

[5] T. Desell, B. Szymanski, and C. Varela, "An asynchronous hybrid genetic-simplex search for modeling the milky way galaxy using volunteer computing," in *Genetic and Evolutionary Computation Conference*, Atlanta, Georgia, July 2008.

[6] B. Szymanski, T. Desell, and C. Varela, "The effect of heterogeneity on asynchronous panmictic genetic search," in *Proc. of the Seventh International Conference on Parallel Processing and Applied Mathematics (PPAM'2007)*, ser. LNCS, Gdansk, Poland, September 2007.

[7] D. R. Law, K. V. Johnston, and S. R. Majewski, "A Two Micron All-Sky Survey View of the Sagittarius Dwarf Galaxy. IV. Modeling the Sagittarius Tidal Tails," *The Astrophysical Journal*, vol. 619, pp. 807–823, Feb. 2005.

[8] K. N. Abazajian, J. K. Adelman-McCarthy, M. A. Agüeros, S. S. Allam, C. Allende Prieto, D. An, K. S. J. Anderson, S. F. Anderson, J. Annis, N. A. Bahcall, and et al., "The Seventh Data Release of the Sloan Digital Sky Survey," *Astrophysical Journal Supplement*, vol. 182, pp. 543–558, Jun. 2009.

[9] D. P. Anderson, E. Korpela, and R. Walton, "High-performance task distribution for volunteer computing." in *e-Science*. IEEE Computer Society, 2005, pp. 196–203.

[10] R. Storn and K. Price, "Minimizing the real functions of the ICEC'96 contest by differential evolution," in *Proceedings of the IEEE International Conference on Evolutionary Computation*, Nagoya, Japan, 1996, pp. 842–844.

[11] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *IEEE International Conference on Neural Networks*, vol. 4, 1995, pp. 1942–1948.

[12] R. C. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Sixth International Symposium on Micromachine and Human Science*, 1995, pp. 33–43.

[13] Y. Shi and R. C. Eberhart, "A modified particle swarm optimizer," in *IEEE World Congress on Computational Intelligence*, May 1998, pp. 69–73.

[14] J. F. Schutte, J. A. Reinbolt, B. J. Fregly, R. T. Haftka, and A. D. George, "Parallel global optimization with the particle swarm algorithm," *International Journal for Numerical Methods in Engineering*, vol. 61, no. 13, pp. 2296–2315, December 2004.

[15] S. Baskar, A. Alphones, and P. N. Suganthan, "Concurrent PSO and FDR-PSO based reconfigurable phase-differentiated antenna array design," in *Congress on Evolutionary Computation*, vol. 2, June 2004, pp. 2173–2179.

[16] E. Cantu-Paz, "A survey of parallel genetic algorithms," *Calculateurs Paralleles, Reseaux et Systems Repartis*, vol. 10, no. 2, pp. 141–171, 1998.

[17] H. Imade, R. Morishita, I. Ono, N. Ono, and M. Okamoto, "A grid-oriented genetic algorithm framework for bioinformatics," *New Generation Computing: Grid Systems for Life Sciences*, vol. 22, pp. 177–186, January 2004.

[18] D. Lim, Y.-S. Ong, Y. Jin, B. Sendhoff, and B.-S. Lee, "Efficient hierarchical parallel genetic algorithms using grid computing," *Future Generation Computer Systems*, vol. 23, pp. 658–670, May 2007.

[19] T. Peachey, D. Abramson, and A. Lewis, "Model optimization and parameter estimation with Nimrod/o," in *International Conference on Computational Science*, University of Reading, UK, May 2006.

[20] A. Lewis and D. Abramson, "An evolutionary programming algorithm for multi-objective optimisation," in *IEEE Congress on Evolutionary Computation (CEC2003)*, vol. 3, December 2003, pp. 1926–1932.

[21] D. K. Tasoulis, N. G. Pavlidis, V. P. Plagianakos, and M. N. Vrahatis, "Parallel differential evolution," in *Congress on Evolutionary Computation 2004 (CEC2004)*, vol. 2, June 2004, pp. 2023–2029.

[22] E. Alba and J. M. Troya, "Analyzing synchronous and asynchronous parallel distributed genetic algorithms," *Future Generation Computer Systems*, vol. 17, pp. 451–465, January 2001.

[23] J. Berntsson and M. Tang, "A convergence model for asynchronous parallel genetic algorithms," in *IEEE Congress on Evolutionary Computation (CEC2003)*, vol. 4, December 2003, pp. 2627–2634.

[24] G. Folino, A. Forestiero, and G. Spezzano, "A JXTA based asynchronous peer-to-peer implementation of genetic programming," *Journal of Software*, vol. 1, pp. 12–23, August 2006.

[25] B. Bánhelyi, M. Biazzini, A. Montresor, and M. Jelasity, "Peer-to-peer optimization in large unreliable networks with branch-and-bound and particle swarms," in *EvoWorkshops '09: Proceedings of the EvoWorkshops 2009 on Applications of Evolutionary Computing*. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 87–92.

[26] B.-I. Koh, A. D. George, and R. T. Haftka, "Parallel asynchronous particle swarm optimization," *International Journal of Numerical Methods in Engineering*, vol. 67, no. 4, pp. 578–595, July 2006.

[27] G. Venter and J. Sobieszczanski-Sobieski, "A parallel particle swarm optimization algorithm accelerated by asynchronous evaluations," in *Sixth World Congresses of Structural and Multidisciplinary Optimization*, May 2005, pp. 1–10.

[28] J. R. Prez and J. Basterrechea, "Particle swarms applied to array synthesis and planar near-field antenna measurements," *Microwave and Optical Technology Letters*, vol. 50, no. 2, pp. 544–548, February 2008.

[29] L. Xu and F. Zhang, "Parallel particle swarm optimization for attribute reduction," in *Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing*, vol. 1, July 2007, pp. 770–775.

[30] D. P. Anderson, J. Cobb, E. Korpela, M. Lebofsky, and D. Werthimer, "SETI@home: an experiment in public-resource computing," *Commun. ACM*, vol. 45, no. 11, pp. 56–61, 2002.

[31] V. Pande *et al.*, "Atomistic protein folding simulations on the submillisecond timescale using worldwide distributed computing," *Biopolymers*, vol. 68, no. 1, pp. 91–109, 2002, peter Kollman Memorial Issue.

[32] K. V. Johnston, D. N. Spergel, and L. Hernquist, "The Disruption of the Sagittarius Dwarf Galaxy," *The Astrophysical Journal*, vol. 451, pp. 598–+, Oct. 1995.

[33] D. R. Law and S. R. Majewski, "The Sagittarius Dwarf Galaxy: A Model for Evolution in a Triaxial Milky Way Halo," *The Astrophysical Journal*, vol. 714, pp. 229–254, May 2010.

[34] W. Dehnen, M. Odenkirchen, E. K. Grebel, and H. Rix, "Modeling the Disruption of the Globular Cluster Palomar 5 by Galactic Tides," *The Astronomical Journal*, vol. 127, pp. 2753–2770, May 2004.

[35] C. J. Grillmair and O. Dionatos, "Detection of a 63 degree Cold Stellar Stream in the Sloan Digital Sky Survey," *Astrophysical Journal Letters*, vol. 643, pp. L17–L20, May 2006.

[36] B. A. Willett, H. J. Newberg, H. Zhang, B. Yanny, and T. C. Beers, "An Orbit Fit for the Grillmair Dionatos Cold Stellar Stream," *The Astrophysical Journal*, vol. 697, pp. 207–223, May 2009.

[37] S. E. Koposov, H. Rix, and D. W. Hogg, "Constraining the Milky Way Potential with a Six-Dimensional Phase-Space Map of the GD-1 Stellar Stream," *The Astrophysical Journal*, vol. 712, pp. 260–273, Mar. 2010.

[38] H. J. Newberg, B. A. Willett, B. Yanny, and Y. Xu, "The Orbit of the Orphan Stream," *The Astrophysical Journal*, vol. 711, pp. 32–49, Mar. 2010.

[39] V. Belokurov, D. B. Zucker, N. W. Evans, G. Gilmore, S. Vidrih, D. M. Bramich, H. J. Newberg, R. F. G. Wyse, M. J. Irwin, M. Fellhauer, P. C. Hewett, N. A. Walton, M. I. Wilkinson, N. Cole, B. Yanny, C. M. Rockosi, T. C. Beers, E. F. Bell, J. Brinkmann, Ž. Ivezić, and R. Lupton, "The Field of Streams: Sagittarius and Its Siblings," *Astrophysical Journal Letters*, vol. 642, pp. L137–L140, May 2006.

[40] C. J. Grillmair, "Detection of a 60 degree-long Dwarf Galaxy Debris Stream," *Astrophysical Journal Letters*, vol. 645, pp. L37–L40, Jul. 2006.

[41] H. C. Plummer, "On the problem of distribution in globular star clusters," *Monthly Notices of the Royal Astronomical Survey*, vol. 71, pp. 460–470, Mar. 1911.

[42] J. Barnes and P. Hut, "A hierarchical O(N log N) force-calculation algorithm," *Nature*, vol. 324, pp. 446–449, Dec. 1986.

[43] J. Dubinski, "A parallel tree code," *New Astronomy*, vol. 1, pp. 133–147, Oct. 1996.

[44] L. Hernquist, "Performance characteristics of tree codes," *Astrophysical Journal Supplement*, vol. 64, pp. 715–734, Aug. 1987.

[45] M. Miyamoto and R. Nagai, "Three-dimensional models for the distribution of mass in galaxies," *Publications of the Astronomical Society of Japan*, vol. 27, pp. 533–543, 1975.

[46] X. X. Xue, H. W. Rix, G. Zhao, P. Re Fiorentin, T. Naab, M. Steinmetz, F. C. van den Bosch, T. C. Beers, Y. S. Lee, E. F. Bell, C. Rockosi, B. Yanny, H. Newberg, R. Wilhelm, X. Kang, M. C. Smith, and D. P. Schneider, "The Milky Way's Circular Velocity Curve to 60 kpc and an Estimate of the Dark Matter Halo Mass from the Kinematics of ˜2400 SDSS Blue Horizontal-Branch Stars," *The Astrophysical Journal*, vol. 684, pp. 1143–1158, Sep. 2008.

[47] J. F. Navarro, C. S. Frenk, and S. D. M. White, "A Universal Density Profile from Hierarchical Clustering," *The Astrophysical Journal*, vol. 490, pp. 493–+, Dec. 1997.

[48] E. McIntosh, F. Schmidt, and F. de Dinechin, "Massive tracking on heterogeneous platforms," in *9th International Computational Accelerator Physics Conference (ICAP)*, October 2006.

[49] W. Herr, D. I. Kaltchev, E. McIntosh, and F. Schmidt, "Large scale beam-beam simulations for the cern lhc using distributed computing," in *10th European Particle Accelerator Conference*, July 2006, pp. 526–528.

[50] M. Taufer, D. Anderson, P. Cicotti, and C. L. B. Iii, "Homogeneous redundancy: a technique to ensure integrity of molecular simulation results using public computing," in *Proceedings of the 14th Heterogeneous Computing Workshop HCW (2005), in conjunction with IPDPS*, 2005, p. 119.

[51] T. Desell, M. Magdon-Ismail, B. Szymanski, C. Varela, H. Newberg, and D. Anderson, "Validating evolutionary algorithms on volunteer computing grids," in *The 10th IFIP international conference on distributed applications and interoperable systems (DAIS)*. Amsterdam, Netherlands: Springer-Verlag, June 2010.

[52] D. P. Anderson, "Volunteer computing: the ultimate cloud," *Crossroads*, vol. 16, no. 3, pp. 7–10, 2010.

[53] E. Mezura-Montes, J. Velzquez-Reyes, and C. A. C. Coello, "A comparative study of differential evolution variants for global optimization," in *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*, 2006, pp. 485–492.