Validating Evolutionary Algorithms on Volunteer Computing Grids

Travis Desell[1], Malik Magdon-Ismail[1], Boleslaw Szymanski[1], Carlos A. Varela[1], Heidi Newberg[2], and David P. Anderson[3]

[1] Department of Computer Science
Rensselaer Polytechnic Institute, Troy, NY, 12180, USA.
[2] Department of Physics, Applied Physics and Astronomy,
Rensselaer Polytechnic Institute, Troy, NY, 12180, USA.
[3] U.C. Berkeley Space Sciences Laboratory,
University of California, Berkeley, Berkeley, CA 94720, USA.

Abstract. Computational science is placing new demands on distributed computing systems as the rate of data acquisition is far outpacing the improvements in processor speed. Evolutionary algorithms provide efficient means of optimizing the increasingly complex models required by different scientific projects, which can have very complex search spaces with many local minima. This work describes different validation strategies used by MilkyWay@Home, a volunteer computing project created to address the extreme computational demands of 3-dimensionally modeling the Milky Way galaxy, which currently consists of over 27,000 highly heterogeneous and volatile computing hosts, which provide a combined computing power of over 1.55 petaflops. The validation strategies presented form a foundation for efficiently validating evolutionary algorithms on unreliable or even partially malicious computing systems, and have significantly reduced the time taken to obtain good fits of MilkyWay@Home's astronomical models.

Key words: Volunteer Computing, Evolutionary Algorithms, Validation

1 Introduction

The demands of computational science continue to increase as the rates of data acquisition and modeling complexity far outpace advances in processor speed. Because of this, highly distributed computing environments are very useful in achieving the amount of computational power required. A very effective way of accumulating a large amount of computational power is by utilizing volunteer computing grids with software such as BOINC [1], which provides software to let volunteers easily participate in many different computing projects based on their personal interests. The MilkyWay@Home project [4] has had particular success using this software, gathering a volunteer base of over 27,000 active computing

[4] http://milkyway.cs.rpi.edu

hosts with a combined computing power of over 1.55 petaflops in a little over two years.

However, utilizing a volunteer computing grid comes with its own set of unique challenges. As these projects are open to the public and often open source, as in the case of MilkyWay@Home, code run on clients can be executed on any type of hardware, resulting in a very heterogeneous environment. Users can also be geographically distributed, so latency can be highly heterogeneous (*e.g.*, MilkyWay@Home has active users in over 130 countries). Additionally, because users can configure their own hardware and compile the open source application with hardware specific optimizations, errors from improperly configured hardware and compiled applications must be handled without invalidating other work being done by the system.

Evolutionary algorithms (EAs) are an approach to performing optimization of challenging problems in computational science, as they can efficiently find global minima in challenging search spaces with many local minima. Traditional methods such as conjugate gradient descent and newton methods quickly get stuck in local minima and fail in these circumstances. This work examines how to efficiently perform validation for evolutionary algorithms on volunteer computing grids, using MilkyWay@Home as a test system. Typically many volunteer computing projects require every result returned by computing hosts to be validated in some manner, however in the case of evolutionary algorithms the number of results requiring validation can be significantly reduced. Two approaches are presented for performing validation. The first is pessimistic, assuming that results are invalid and waiting for their validation before they are used. The other is optimistic, using results as soon as they are reported and later reverting values that are found to be invalid. Using common EAs, differential evolution and particle swarm optimization, optimistic validation is shown to significantly improve the convergence rate of the EAs run by MilkyWay@Home.

The remainder of this paper is organised as follows. Section 2 presents the EAs used in this work. The validation strategies used are given in Section 3 and their performance on MilkyWay@Home is discussed in Section 4. The paper concludes with a discussion and future work in Section 5.

2 Evolutionary Algorithms for Continuous Search Spaces

Effective approaches to global optimization for continuous search spaces include differential evolution (DE) and particle swarm optimization (PSO). In general, individuals are sets of parameters to an objective function which is trying to be optimized. Applying the objective function to an individual provides the fitness of that individual, and the evolutionary algorithms evolve individuals through different heuristics to try and find the best possible fitness, which optimizes the objective function.

2.1 Differential Evolution

Differential evolution is an evolutionary algorithm used for continuous search spaces developed by Storn and Price over 1994–1995 [13]. Unlike other evolutionary algorithms, it does not use a binary encoding strategy or a probability density function to adapt its parameters, instead it performs mutations based on the distribution of its population [11]. For a wide range of benchmark functions, it has been shown to outperform or be competitive with other evolutionary algorithms and particle swarm optimization [15].

Differential evolution evolves individuals by selecting pairs of other individuals, calculating their differential, scaling it and then applying it to another parent individual. Some kind of recombination (*e.g.*, binary or exponential) is then performed between the current individual and the parent modified by the differentials. If the fitness of the generated individual is better than the current individual, the current individual is replaced with the new one. Differential evolution is often described with the following naming convention, "de/*parent*/*pairs*/*recombination*", where *parent* describes how the parent is selected (*e.g.*, best or random), *pairs* is the number of pairs used to calculate the differentials, and *recombination* is the type of recombination applied. Two common recombination strategies are:

- binomial recombination, $bin(p^1, p^2)$:

$$c_i = \begin{cases} p_i^1 \text{ if } r(0,1) < \sigma \text{ or } i = r(0,D) \\ p_i^2 \text{ otherwise} \end{cases} \tag{1}$$

- exponential recombination, $exp(p^1, p^2)$:

$$c_i = \begin{cases} p_i^1 \text{ from } r(0,1) < \sigma \text{ or } i = r(0,D) \\ p_i^2 \text{ otherwise} \end{cases} \tag{2}$$

Which take two parents, where the i^{th} parameters of parents p^1 and p^2, p_i^1 and p_i^2 respectively, are used to generate the i^{th} parameter of the child, c_i. Binomial recombination selects parameters randomly from p^1 at a recombination rate σ, and always selects at least 1 parameter of p^1 randomly, otherwise it uses the parameter from p^2. Exponential recombination selects all parameters from parent p^1 after a randomly chosen parameter, or a random number is generated lower than the recombination rate, whichever comes first.

In general, a new potential individual $n_i(l+1)$ for a new population $l+1$ is generated from the i^{th} individual $x_i(l)$ from the previous population l, and selected if its fitness, $f(x)$, is greater than the previous individual:

$$x_i(l+1) = \begin{cases} n_i(l+1) \text{ if } f(n_i(l+1)) > f(x_i(l)) \\ x_i(l) \text{ otherwise} \end{cases} \tag{3}$$

The j^{th} parameter is calculated given p pairs of random individuals from the population l, where $r(l)^0 \neq ... \neq r(l)^{2p}$. θ, ϕ and σ are the user defined *parent scaling factor*, *recombination scaling factor* and *crossover rate*, respectively. D

is the number of parameters in the objective function. $b(l)$ is the best individual in the population l. Two popular variants, used in this paper, are:

– *de/best/p/bin*:

$$n_i(l+1) = bin(x_i(l), \theta b(l)_j^0 + \phi \sum_{k=1}^{p} [r(l)_j^{1k} - r(l)_j^{2k}]) \tag{4}$$

– *de/rand/p/bin*:

$$n_i(l+1) = bin(x_i(l), \theta r(l)_j^0 + \phi \sum_{k=1}^{p} [r(l)_j^{1k} - r(l)_j^{2k}]) \tag{5}$$

For more detail, Mezura-Montes *et al.* have studied many different variants of differential evolution on a broad range of test functions [10].

2.2 Particle Swarm Optimization

Particle swarm optimization was initially introduced by Kennedy and Eberhart [9, 7] 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). Two user defined constants, c_1 and c_2, allow modification of the balance between local (cognitive) and global (social) search. Later, an inertia weight ω was added to the method by Shi and Eberhart to balance the local and global search capability of PSO [12] and is used in this work and by most modern PSO implementations. And recently, PSO has been shown to be effective in peer-to-peer computing environments by Bánhelyi et al [2]. 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:

$$\begin{aligned} v_i(t+1) &= \omega * v_i(t) \\ &\quad + c_1 * rand() * (p_i - x_i(t)) \\ &\quad + c_2 * rand() * (g_i - x_i(t)) \\ x_i(t+1) &= x_i(t) + v_i(t+1) \end{aligned} \tag{6}$$

3 Validation Strategies

As computing systems get larger, the potential for faulty or erroneous computing hosts increases. This can also be a concern with applications that may occasionally return incorrect results. In the case of volunteer computing systems such as MilkyWay@Home, which are open to the public, there is always the risk of

malicious users or bad or improperly configured hardware returning false results. Additionally, as volunteer computing platforms such as BOINC [1] typically encourage participation by awarding credit for completed work and tracking the best participants, there is some incentive for *cheating* to be awarded more credit than deserved.

Different approaches have been developed to perform validation on volunteer computing systems. The BOINC framework provides validation based on redundancy and reaching a quorum [1]. However, validation of every work unit in an asynchronous search setting leads to a large amount of wasted computation. BOINC also provides a strategy which uses a measure of *trust*. In this strategy, hosts become more trusted as they return results which validate, and lose trust as they return results that are invalid. Using this strategy, results from trusted hosts are assumed to be valid and only occasionally are their results validated to make sure they are still reporting correct results. This approach is unsuitable for EAs however, as a single invalid result that remains in the population can invalidate the entire search. Other work has examined strategies for dissuading participants in volunteer computing systems from cheating by detecting bad hosts [14, 8], however these do not address the issue of ensuring correct results.

The amount of validation required by EAs can be significantly reduced when compared to strategies which validate every result. This is because EAs only progress when new individuals are inserted into the populations. Individuals with lower fitness are simply discarded.

Search	0...25,000 Evaluations	25,001 ... 50,000 Evaluations
APSO	476	208
ADE/Best	551	221

Table 1. The average number of individuals inserted into the population during the given number of evaluations, averaged over 20 searches with different initial parameters.

Data from MilkyWay@Home's logs has shown that only a small number of results ever make it into the population. Table 1 shows the average number of inserts done over 20 different searches done on MilkyWay@Home using data from Sagittarius stripe 22. The number of inserts for both the first and second 25,000 reported results are shown for asynchronous particle swarm optimization and differential evolution using best parent selection. It becomes apparent from this information that only a small number of results ever make it into the population, less than 4% in the first half of the search and less than 2% in the second half. Additionally, as the searches progress it becomes more difficult to find new good search areas and the number of evaluated individuals inserted into the population decreases. Data from MilkyWay@Home's logs also show that for a random sample of 500,000 results, only 2,609, or 0.5% were errors that could have been inserted into the population.

By ignoring the erroneous results that would not be inserted into the population, even though they may have a correct result which could potentially be useful, the amount of computation dedicated to validation can be decreased dramatically. However, it is important to note that the search will not progress until better individuals are inserted into its population, so the longer it takes to verify good results, the slower the search will progress. So if too many resources are devoted to optimization the search may progress extremely slow.

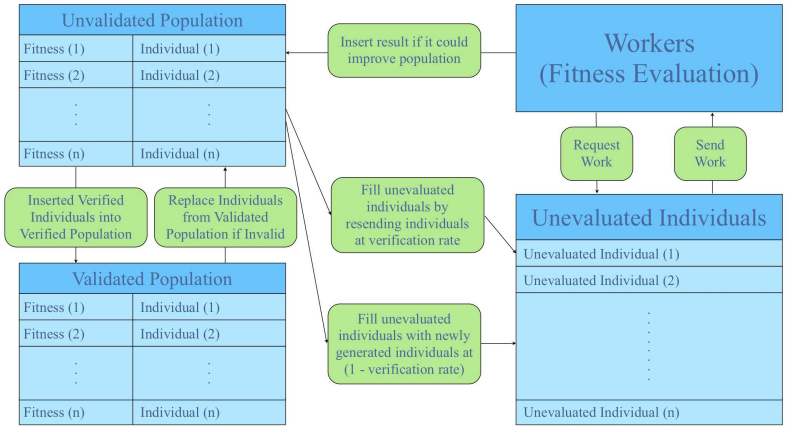

Fig. 1. The optimistic validation strategy. Results that improve the unvalidated population are used to generate new individuals as soon as they are received, and reverted to previously validated results if they are found to be invalid. When unvalidated results are validated, they are inserted into the validated population.

Two different validation strategies have been implemented and tested, *pessimistic* and *optimistic*. Pessimistic validation assumes results are faulty and only uses validated results to generate new work (see Figure 2). While this ensures that newly generated results are generated from valid individuals, progress in the search is delayed while waiting to validate reported individuals with good fitness. Optimistic validation assumes results are correct and uses the best known individuals to generate new work (see Figure 1). This allows the search to progress as fast as possible, however there is the chance that erroneous individuals will be used to generate new work until those erroneous results are found to be invalid.

These strategies were implemented by having two populations. One consisting of validated results and the other consisting of unvalidated results. For pessimistic validation, new individuals are generated by copying the individuals in the unvalidated population for validation at a specified rate, or by performing DE or PSO on the validated population otherwise. When an individual in the unvalidated population is validated, it is inserted into the validated population. For optimistic validation, new individuals are also generated from the unvali-

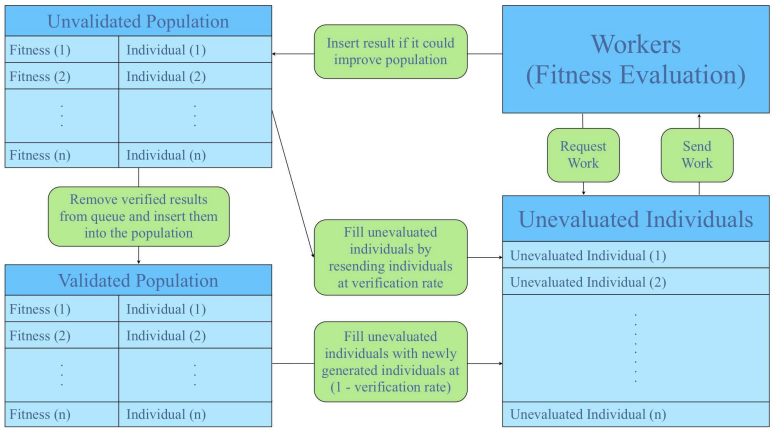

Fig. 2. The pessimistic validation strategy. Results are validated before they are used to generate new individuals.

dated population at the specified rate, however DE and PSO is also performed on the unvalidated population for the other generated individuals. When an individual in the unvalidated population is found to be valid, it is inserted into the validated population. If an individual is found to be invalid, it is reverted to the previously validated result in the validated population.

4 Results

The effects of optimistic and pessimistic validation were also tested using MilkyWay@Home while fitting a model of the Sagittarius dwarf galaxy on data acquired from Data Release 7 of the Sloan Digital Sky Survey. This problem involves calculating how well a model of three tidal streams of stars and a background function fit a 5 degree wedge of 100,789 observed stars collected such that the wedge is perpendicular to the direction of the tidal stream's motion (for more information about the astronomy and fitness function readers are referred to [4, 5]). In total there are 20 real valued parameters to be optimized in the objective function. This model is calculated by a wide variety of hosts. The fastest high end double precision GPUs can calculate the fitness in under two minutes. High end CPUs require around an hour, and the slowest CPUs can take days. At the time these results were gathered, MilkyWay@Home had approximately 27,000 volunteered hosts participating in the experiments and a combined computing power of 1.55 petaflops [5].

Both particle swarm optimization and differential evolution were tested with a fixed population of 200 individuals. Particle swarm optimization used an inertia

[5] Statistics taken from http://boincstats.com

weight of $\omega = 0.6$ and $c_1 = c_2 = 2.0$. Differential evolution used best parent selection and binomial recombination, *i.e.*, de/best/1/bin, which has been shown to be a very robust and fast version of differential evolution [10]. Differential evolution used a pair weight, recombination rate and recombination scales of $\theta = \phi = \sigma = 0.5$.

Figure 3 and 4 compares optimistic and pessimistic validation for DE/best and PSO, respectively. Five searches were run with for each validation rate, as it was increased from 10% to 40%. The best validated fitness presented is the average of those five searches. For both DE/best and PSO optimistic validation significantly outperformed pessimistic validation.

Figure 5 compares the best verification strategies and verification rates. For optimistic validation, DE/best found the best solutions with a validation rate of 20%, while PSO found the best solutions with a validation rate of 40%. For pessimistic validation, both PSO and DE/best found the best solutions on average with a 30% validation rate.

While using optimistic validation significantly improved the convergence rate of the optimization methods used, it also reduced the effect of the validation rate on the convergence of these methods. For pessimistic validation, changing the validation rate seemed to have a great effect at the search convergence rate, while this effect was significantly reduced for optimistic validation, almost to the point where the differences could be attributed to noise. However, it is still very interesting to note that the higher validation rates still outperformed lower validation rates.

5 Discussion

The requirement of high validation rates for pessimistic validation lends itself to the conclusion that it is very important to be able to quickly use newly found good individuals in the generation of new individuals to keep the search progressing quickly. However, the fact that optimistic validation also requires high validation rates, in spite of using results for the generation of new individuals as soon as they are reported, suggests that even a small amount of failures if not invalidated quickly can have very negative effects on the performance of the search. This negative effect could be for a few reasons. First, that the effect of invalid individuals in the search population is extremely negative, considering the validation rates required when only 0.5% of results reported are invalid. Another possibility is that with optimistic validation there could be a large delay between use of high fitness individuals and their insertion into the validated population. If such high fitness unvalidated individual is later found to be invalid, it could be reverted to a very old validated one which could significantly degrade the performance of the search. Yet another possibility involves examining the types of errors that occur in the system. For both optimization methods used, DE/best and PSO, the heuristic for generating new individuals always uses the best individual in the population. If the majority of errors have fitness that is better than the best individual, these will corrupt all newly generated until they

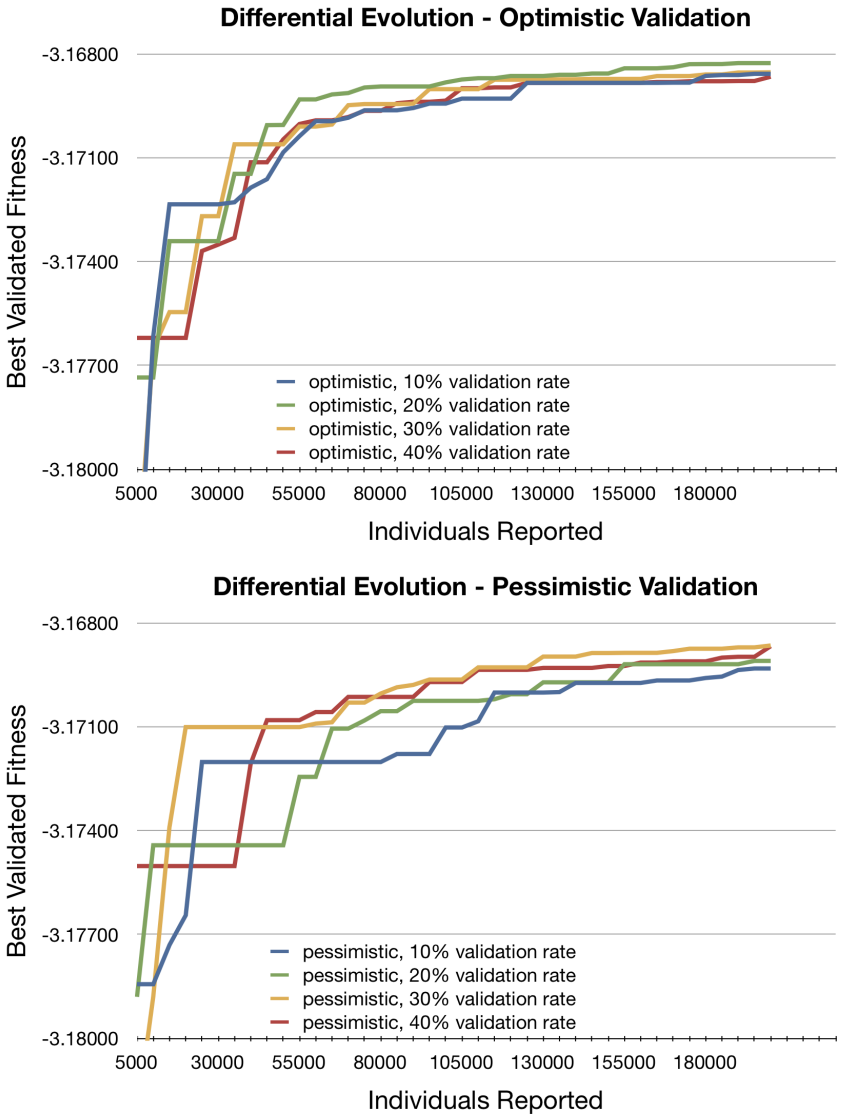

Fig. 3. Comparison of the progress of particle swarm optimization on MilkyWay@Home while optimizing the model for Saggitarius Stripe 22 with different validation rates for optimistic and pessimistic validation.

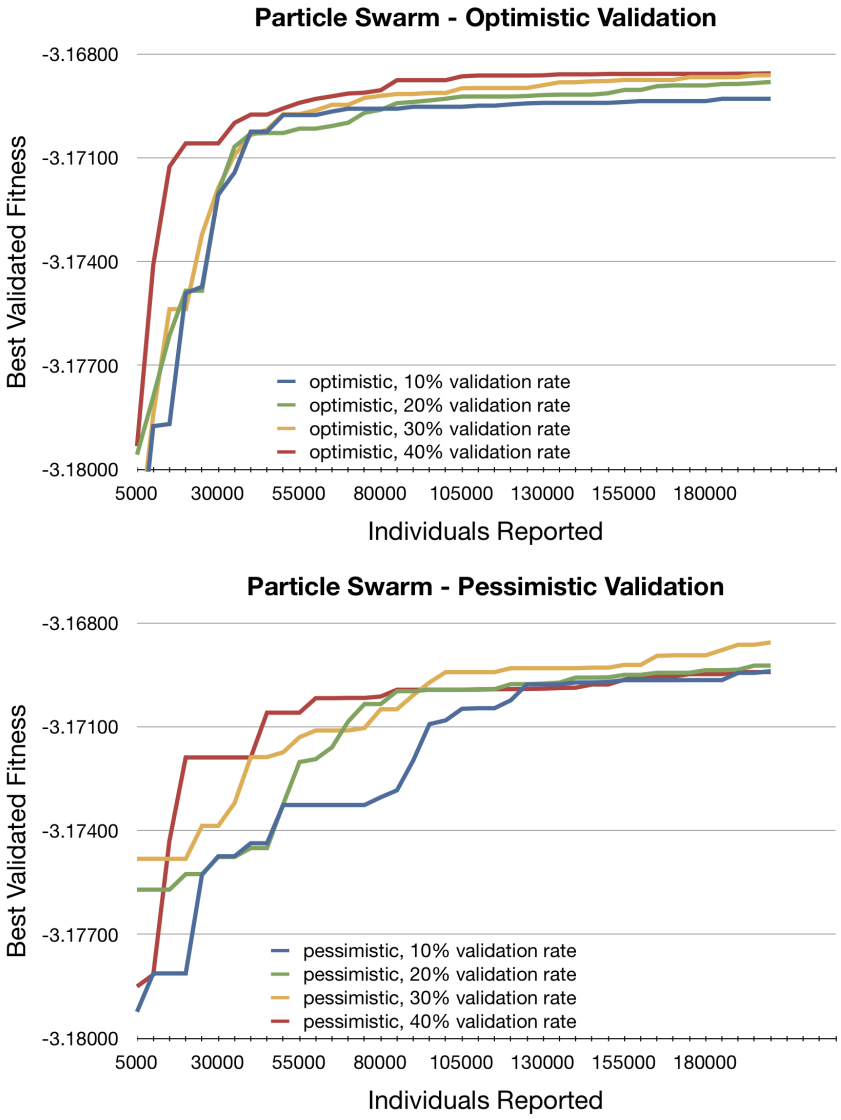

Fig. 4. Comparison of the progress of differential evolution with best parent selection on MilkyWay@Home while optimizing the model for Saggitarius Stripe 22 with different validation rates for optimistic and pessimistic validation.

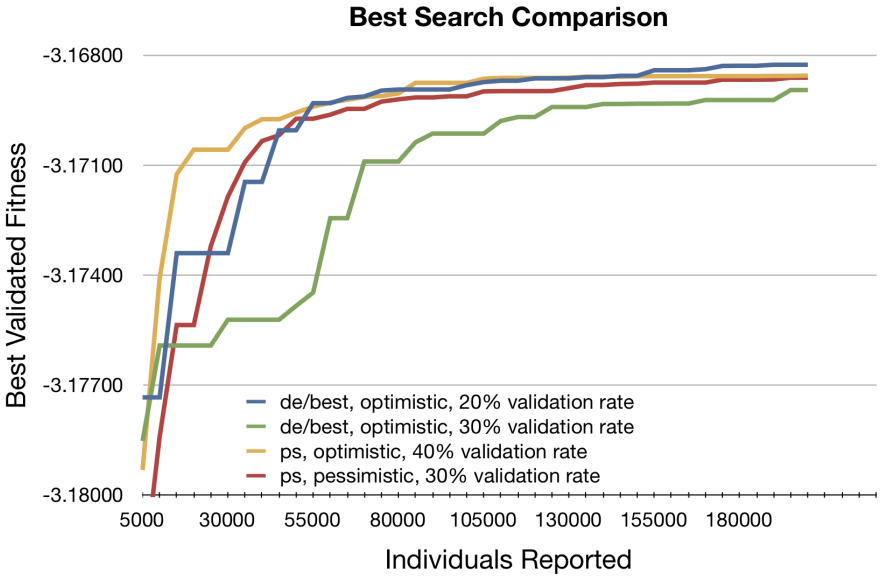

Fig. 5.

are found to be invalid which could also help explain the negative effects of not validating individuals quickly enough.

For future work, in order to further examine increasing the efficiency of validation of evolutionary algorithms, it would be interesting to measure the effect of errors in a controlled environment. Using the simulation framework developed in previous work [6], it would be possible to simulate errors (both those that are better than the best individual in the population, and those that only make a single individual invalid) and analyze these effects for different benchmark optimization problems. It would also be interesting to study what rate of errors causes pessimistic validation to outperform optimistic validation. Another area of potential research is to combine the validation strategies presented in this work with those available in BOINC. As opposed to keeping two populations of validated and unvalidated individuals and generating individuals for verification by copying them from the unvalidated population, the BOINC software could be used to handle the validation by quorum on individuals which could potentially be inserted into the validated population. This would ensure that any potentially improving result will be validated so that when an individual in the unvalidated population is found to be invalid, it will roll back only to the last best individual found. However, this approach can lead to slower validation of results and has the potential to utilize a significant amount of memory and disk space as the amount of results awaiting validation can grow unbounded; so those effects must be studied as well.

Another potential area of interest is that recently hyper-heuristics have been shown to be effective in distributed computing scenarios [3]. Using meta or hyper-heuristics to automatically tune not only the parameters involved in the optimization methods, but to tune the validation rate could further improve convergence times.

The results presented show that using optimistic validation can be a very effective strategy for improving the convergence rates of evolutionary algorithms on volunteer computing grids, without resorting to validating every result which can be required for other algorithms. These strategies have also been used on a live volunteer computing grid with over 27,000 active volunteered computing hosts and have been utilized to provide effective and efficient validation for the MilkyWay@Home project.

6 Acknowledgements

Special thanks go to the Marvin Clan, David Glogau, and the Dudley Observatory for their generous donations to the MilkyWay@Home project, as well as the thousands of volunteers that made this work a possibility. This work has been partially supported by the National Science Foundation under Grant Numbers 0612213, 0607618, 0448407 and 0947637. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

1. D. P. Anderson, E. Korpela, and R. Walton. High-performance task distribution for volunteer computing. In *e-Science*, pages 196–203. IEEE Computer Society, 2005.
2. 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*, pages 87–92, Berlin, Heidelberg, 2009. Springer-Verlag.
3. M. Biazzini, B. Banhelyi, A. Montresor, and M. Jelasity. Distributed hyper-heuristics for real parameter optimization. In *GECCO '09: Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pages 1339–1346, New York, NY, USA, 2009. ACM.
4. N. Cole. *Maximum Likelihood Fitting of Tidal Streams with Application to the Sagittarius Dwarf Tidal Tails*. PhD thesis, Rensselaer Polytechnic Institute, 2009.
5. 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*, 683:750–766, 2008.
6. T. Desell. *Asynchronous Global Optimization for Massive Scale Computing*. PhD thesis, Rensselaer Polytechnic Institute, 2009.
7. R. C. Eberhart and J. Kennedy. A new optimizer using particle swarm theory. In *Sixth International Symposium on Micromachine and Human Science*, pages 33–43, 1995.

8. P. Golle and I. Mironov. Uncheatable distributed computations. In *CT-RSA 2001: Proceedings of the 2001 Conference on Topics in Cryptology*, pages 425–440, London, UK, 2001. Springer-Verlag.
9. J. Kennedy and R. C. Eberhart. Particle swarm optimization. In *IEEE International Conference on Neural Networks*, volume 4, pages 1942–1948, 1995.
10. 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*, pages 485–492, 2006.
11. H.-P. Schwefel. *Evolution and Optimization Seeking*. John Wiley & Sons, New York, 1995.
12. Y. Shi and R. C. Eberhart. A modified particle swarm optimizer. In *IEEE World Congress on Computational Intelligence*, pages 69–73, May 1998.
13. 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*, pages 842–844, Nagoya, Japan, 1996.
14. D. Szajda, B. Lawson, and J. Owen. Hardening functions for large scale distributed computations. *Security and Privacy, IEEE Symposium on*, 0:216, 2003.
15. J. Vesterstrom and R. Thomsen. A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems. In *Congress on Evolutionary Computation 2004 (CEC2004)*, volume 2, pages 1980–1987, June 2004.