SAND2015-0461C

Asynchronous Parallel Evolutionary Algorithms: Leveraging Heterogeneous Fitness Evaluation Times for Scalability and Elitist Parsimony Pressure

Matthew A. Martin
Natural Computation Laboratory
Department of Computer Science
Missouri University of Science and Technology
Rolla, Missouri, U.S.A.
mam446@mst.edu

Alex R. Bertels
Natural Computation Laboratory
Department of Computer Science
Missouri University of Science and Technology
Rolla, Missouri, U.S.A.
arb9z4@mst.edu

Daniel R. Tauritz
Natural Computation Laboratory
Department of Computer Science
Missouri University of Science and Technology
Rolla, Missouri, U.S.A.
dtauritz@acm.org

ABSTRACT

Many important problem classes lead to large variations in fitness evaluation times, such as is often the case in Genetic Programming where the time complexity of executing one individual may differ greatly from that of another. Asynchronous parallel evolutionary algorithms (APEAs) omit the generational synchronization step of traditional evolutionary algorithms which work in well-defined cycles. They can provide scalability improvements proportional to the variation in fitness evaluation times of the evolved individuals, and therefore should be considered when faced with the aforementioned problem classes. This paper provides an empirical analysis of the scalability improvements obtained by applying APEAs to such problem classes. Furthermore, APEAs often suffer from bias towards individuals with shorter fitness evaluation times, simply because they propagate faster. This paper shows how to leverage this bias in order to provide a unique type of "elitist" parsimony pressure which rewards more efficient solutions with equal solution quality.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search; I.2.2 [**Artificial Intelligence**]: Automatic Programming–*program modification, program synthesis*

General Terms

Algorithms, Design

GECCO'15, July 11-15, 2015, Madrid, Spain.

Keywords

Black-Box Search Algorithms, Evolutionary Algorithms, Genetic Programming, Hyper-Heuristics

1. INTRODUCTION

The genotype representation of individuals used in evolutionary algorithms greatly influences the evaluation time. In various problem classes, the representation may result in evaluation times that fluctuate with the structure of each individual. Genetic programs (GPs) often attempt to control this time by employing limits or penalties. The time difference is found to be magnified in fields such as hyper-heuristics when fitness relies on a sample consisting of multiple test cases. While parallelization of the generational evolutionary algorithm provides a significant speed-up, many cycles in the cores are still wasted waiting for the larger individuals to be evaluated. The nature of asynchronous parallel evolutionary algorithms (APEAs) presents a solution to this inefficiency.

This research establishes the improvement of APEAs over synchronous PEAs on global populations. Additionally, this paper introduces and analyzes the "elitist" parsimony pressure (EPP) that is an effect of employing asynchronous EAs on heterogeneous populations. The rest of the article is organized as follows. Section 3.1 illustrates the structure of the APEA and the mechanics behind it. The design of the synchronous model and the inherent limitations are laid out in Section 3.2. This is followed by a comparison between the two models (Section 4). The comparison explores the relationship of the number of parallel work nodes to both the evaluation speed (Section 4.4) as well as the convergence time (Section 4.5). Section 5 reveals the effects of using the models on the heterogeneous populations. Lastly, the findings and future work will be discussed.

2. BACKGROUND

Employing evolutionary algorithms (EAs) in optimizing solutions to difficult problems is often limited by the size of the search space and the run time allotted for the search. The structure of these EAs must consider both scalability and efficiency when utilizing the machines at hand. This

results in EAs that must distribute individuals and/or populations across multiple machines and parallel processes that must attempt to reduce the number of wasted cycles within each available core.

Typically, the step in which the fitness, or quality, of each individual in a population is evaluated can be executed in parallel. When the fitness of every individual in a population can be determined in a constant time, the evaluation is said to be homogeneous across the population. Times that vary from one evaluation to the next are heterogeneous.

Populations within a parallel EA (PEA) can either be structured with one or more decentralized populations or as a single, centralized population [7]. Decentralized populations are easily scalable as a machine can host one or more sub-populations and these sub-populations can share individuals at independently determined time intervals. This allows for each sub-population to evolve at different paces, but still receive some genes from other populations. Alba has focused on the effectiveness of various behaviors, particularly distributed and cellular reproduction, in decentralized, parallel EAs [3, 1, 2].

Decentralized populations can be limited by size of the sub-populations and may take many generations to converge on an optimal solution. A single, global population would diminish that issue. In most traditional EAs, the population must be synchronized after each generation. In that situation, several slave processes will not be fully utilized with a heterogeneous population. An APEA would nearly eliminate the wasted cycles that come with differences in evaluation times.

Durillo et al. have empirical evidence supporting the significant improvement in terms of various quality metrics when employing APEAs rather than synchronous PEAs for NSGA-II [5]. The APEA master process creates and sends individuals to be evaluated as the slave processors become idle. In the generational version, the population is replaced when enough offspring have been generated. With the steady-state alternative, the offspring are considered as each is received. The researchers employed homogeneous populations as the test cases during experimentation. While these results still apply to heterogeneous populations, other quality metrics should be investigated to measure performance.

Those that have specifically addressed heterogeneous populations note that APEAs are biased toward individuals with shorter evaluation times [9, 8, 4]. This is a result of the master process receiving those individuals sooner and more often, flooding the population. This potentially reduces the search space that can be reached within the run time. Yagoubi and Schoenauer attempt to circumvent this with a duration-based selection on the received offspring [8]. This supposed defect can be taken advantage of in various situations, one of which is evolving genetic programs (GPs). GPs must use a mechanism (e.g., parsimony pressure) or must minimize a size-related objective value to prevent any individual from becoming too large. The bias provided by heterogeneous evaluation times can be used to produce an implicit time pressure.

3. EVOLUTION MODELS

Both the asynchronous and synchronous evolution models that are described here are designed to perform well when being parallelized across multiple computing nodes. Both models are designed around a master-slave architecture. In these models the master node stores the population and handles the population mechanics, such as adding new individuals into the population and survival selection, and genetic operations, such as mutation and crossover. The slave nodes solely evaluate the individuals. How the slave nodes acquire new individuals to be evaluated is described in Sections 3.1 and 3.2 for the asynchronous and synchronous algorithms respectively. In each of the models the initial population must be evaluated before the main iteration of the models, described below, is started. This is done by distributing the initial population to the nodes and treating them as a single iteration of the synchronous model.

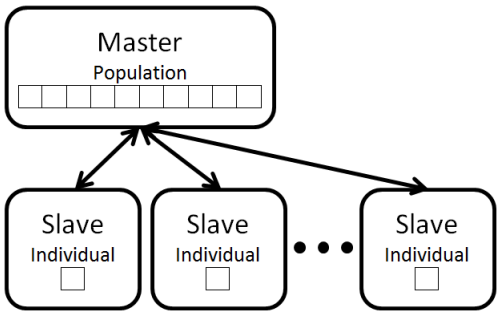

Figure 1: This diagram illustrates how both the asynchronous and synchronous models are structured.

3.1 Asynchronous Evolution Model

The asynchronous evolution model works by each slave node requesting an individual to evaluate. When the slave node is finished evaluating the individual, the individual is sent back to the master node to be added to the population, and the slave node receives a new individual to evaluate. This continues until the termination criteria is met. Each slave node is performing these operations simultaneously. Having each slave node act completely independently of any other slave node inherently provides work balancing and does not waste time by waiting on other individuals being evaluated. Thus, the only time wasted is in the time it takes for the individuals to be sent to and from the slave nodes which all PEAs would have.

There are many ways to handle population mechanics of asynchronous evolution models. The method chosen here is to treat the population like a (μ+1) Evolutionary Algorithm due to its simplicity. When a slave node requests a new individual the master node does parent selection, crossover, and mutation operations creating a new individual. This new individual is sent to the given slave node. When a slave node finishes the evaluation of an individual it returns it to the master node and is added to the population. Then the master node performs a survival selection to reduce the population size back down to size μ. The master node then creates a new solution and the cycle continues until convergence. Each slave node causes an update to the population independently which allows it to bypass certain limitations that the synchronous evolution model has. These limitations will be described in more depth in Section 3.2.

3.2 Synchronous Evolution Model

The synchronous evolution model works by creating batches of individuals and distributing them to slave nodes for evaluation. If λ is larger than the number of slave nodes and a slave node finishes evaluating an individual, that node requests another individual from the master node. This balances the work between all of the slave nodes as evenly as possible without knowing the evaluation time a priori. However, when λ is smaller than the number of work nodes, the λ new individuals are sent to λ slave nodes and are evaluated. The remaining slave nodes simply sit idle. Once all individuals are evaluated they are all added to the population, and the master node performs the survival selection and creates the next generation of individuals for distribution.

This model is not very scalable. The reason this model is not scalable is that it is based on batch evaluation. In each generation this model creates λ new individuals to be evaluated. If the number of slave nodes is greater than λ there will always be certain slave nodes that are not being used at all. Thus, this model cannot scale passed λ slave nodes. The only way around this limitation is to increase λ, but in doing the other parameters will also need to be increased (e.g., μ). If the number of slave nodes is less than λ not all of the slave nodes are evaluating individuals at all times. Once λ individuals have been sent to be evaluated, and a slave node finishes its evaluation, it has to wait until the last slave node is done being evaluated until it gets a new individual. If the evaluation time is homogeneous across the population then parameters can be chosen in such a way that no time is wasted, though if the evaluation time is heterogeneous this cannot be guaranteed.

4. COMPARISON

In this section, the two described models will be analyzed to see how the performance of the algorithms are affected by increasing the number of slave nodes they have access to. A heterogeneous evaluation time will be used in the experiments, meaning that the evaluation time is not the same for each individual. The primary metric for determining how well these algorithms scale will be the amount of time it takes for the model to converge on the solution. Instead of using a benchmark that inherently has a heterogeneous evaluation time, a standard benchmark problem will be selected and an artificial heterogeneous evaluation time will be added to the evaluation of the individual.

4.1 Discrete Event Simulated Evolution

A problem with running experiments on PEAs is that they require a large amount of resources to demonstrate how they scale. In addition to needing these resources, the algorithms can take a long time to complete when examining the algorithms run with a small number of slave nodes. In consideration of these two limitations, a simulator was developed such that the resource and time requirements could be bypassed. A Discrete Event Simulator was developed that would simulate the two algorithms in an environment that could be scaled very easily. This also removes the uncertainty in the runtime of these algorithms caused by other processes on the computer and network traffic. This means the only variation in convergence time is due to the stochastic nature of the algorithms.

To check the accuracy of the simulator, fully functional versions of the synchronous and asynchronous models were created. They were both tested by running the simulation and the fully functional models for a set number of evaluations and comparing the runtimes. The models were tested using 1 to 50 work nodes. The runtimes that the simulation predicted and the runtimes of the fully functional models were consistent. This verification gave evidence that the developed simulations are accurate and the results they give are meaningful.

4.2 Royal Road Function

The Royal Road function defines a simple fitness landscape designed to test the performance of an evolutionary algorithm [6]. This function is dependent upon a set of schemas. Schemas are patterns that consist of a set number of the following symbols: $\{0, 1, *\}$. A bit string matches a schema of the same length if the locations of the 0s and 1s in the schema match those in the bit string. A $*$ can be met with either a 0 or a 1. The order of a schema is the number of 0s and 1s it contains.

The Royal Road schema set is structured such that same-ordered schemas are combined to construct the higher order schemas. An example is shown in Figure 2. Mitchell et al. define the Royal Road function as follows [6]:

$$F(x) = \sum_{s \in S} c_s \sigma_s(x); \text{where } c_s = order(s)$$

$$\text{and } \sigma_s(x) = \begin{cases} 1 & \text{if } x \text{ is an instance of } s \\ 0 & \text{otherwise} \end{cases}$$

$$\begin{array}{rll} s_1 = & 11********** & c_1 = 2 \\ s_2 = & **11********** & c_2 = 2 \\ s_3 = & ****11********** & c_3 = 2 \\ s_4 = & ******11******** & c_4 = 2 \\ s_5 = & ********11****** & c_5 = 2 \\ s_6 = & **********11**** & c_6 = 2 \\ s_7 = & ************11** & c_7 = 2 \\ s_8 = & **************11 & c_8 = 2 \\ s_9 = & 1111********** & c_9 = 4 \\ s_{10} = & ****1111******** & c_{10} = 4 \\ s_{11} = & ********1111**** & c_{11} = 4 \\ s_{12} = & ************1111 & c_{12} = 4 \\ s_{13} = & 11111111******** & c_{13} = 8 \\ s_{14} = & ********11111111 & c_{14} = 8 \\ s_{15} = & 1111111111111111 & c_{15} = 16 \end{array}$$

Figure 2: Example of schema set for Royal Road function [6]

The optimal solution for this example would be a bit string of 16 1s. This would have a fitness of $(8 * 2) + (4 * 4) + (2 * 8) + (1 * 16) = 64$.

4.3 Experimental Setup

To ensure that the comparison between the two models is fair, the parameters and operations used in each of the algorithms is the same. The population size is 100 individuals, and the number of children created each generation is 50 for the synchronous algorithm. The parent selection operation used was k-tournament with replacement. Uniform crossover and bit-flip mutation were used as the variation operators. Survival selection was k-tournament without replacement which selected the individuals that would

Table 1:

Parameter	Value
μ	100
λ	50
Mutation Rate	0.007813
Parent Selection k	15
Survivor Selection k	50

be discarded. The parameters were chosen by hand. While there may exist a set of parameters that perform better than the ones selected, this paper is exploring the general performance of the asynchronous model and the comparison to the synchronous model under identical conditions. A summary of the parameters used can be found in Table **??**.

The initial experiment is to compare the performance of the asynchronous and synchronous evolution models. To do this, the Royal Road function was used as a benchmark function. The bitstring used in the Royal Road function was of length 128. The schemas used were of the form shown in Figure 2 thought the smallest schema consisted of strings of ones of length 4. The models were tested using the discrete event simulation that is described in Section 4.1. The models were run 30 times using 1 through 1000 slave nodes. In this initial experiment the artificial heterogeneous evaluation time was incorporated by adding a delay to the evaluation time in the discrete event simulator. The added delay was selected from a uniform distribution from 0 to 50 seconds.

4.4 Evaluation Speed-Up vs Slave Nodes

The first characteristic of interest comparing these two models is how quickly these two models can evaluate solutions as the number of slave nodes is increased. Figure 3 shows the time it takes to evaluate 10,000 individuals. At 50 slave nodes, the synchronous model levels off and sees no more improvement. This is consistent with the predicted limitations of scaling the synchronous model. The asynchronous model, however, continues to speed up when scaling the number of slave nodes. Figure 4 shows how much speedup is acheived given a certain number of slave nodes. This graph is generated by comparing the time it took to run with a single slave node and the time it took to run with a given number of slave nodes. This graph shows that as the number of slave nodes is increased that the improvement isn't one to one in the asynchronous model. This is due to the evaluation of the initial population. When the number of slave nodes is more than λ in the asynchronous model the evaluation of the initial population take nearly 50 seconds. If this time is adjusted by removing the evaluation of the initial population and replacing it with regular evaluations, the asynchronous algorithm does achieve a one to one increase in efficiency.

4.5 Convergence Time vs Slave Nodes

It has been determined that the evaluation on individuals scales well in the asynchronous model and extremely poorly in synchronous models. However, does not mean that the asynchronous model will converge on the solution more quickly than the synchronous model which is a better way to determine which model is superior. Figure 5 shows the time necessary to converge witha given number of nodes. As with the previous results, the synchronous model levels off

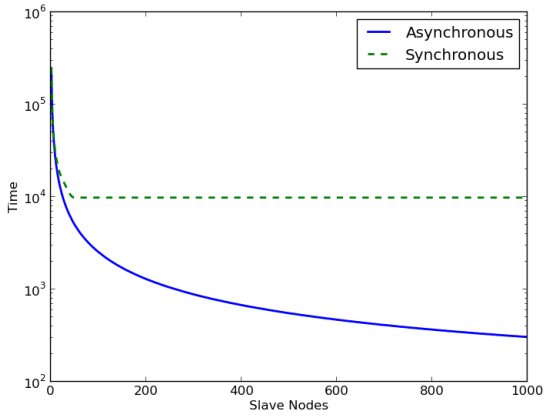

Figure 3: Plot of evaluation time vs number of slave nodes

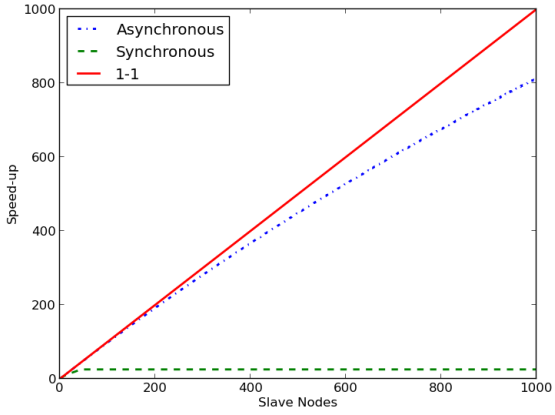

Figure 4: Plot of speed-up of evaluation time vs number of slave nodes

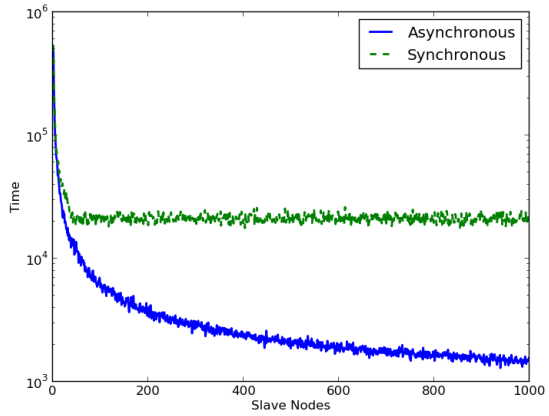

Figure 5: Plot of time until convergence vs number of slave nodes

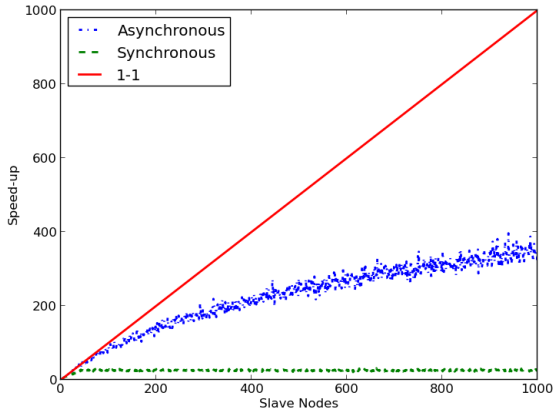

Figure 6: Plot of speed-up of convergence time vs number of slave nodes

once it has exceeded 50 nodes which is expected. Figure 6 better shows the performance of the asynchronous model. As seen in this figure, the effieciency of the asynchronous model is not as good as Figure 4 showed that it could theoretically be. The reason that the asynchronous model is not as efficient is the increase in evaluations needed to converge on the solution as the number of slave nodes increases as demonstrated in Figure 7. While the number of evaluations needed to converge on the solution is much larger for the asynchronous model than the synchronous model, the asynchronous model converges faster on wall time, which is more important for real world applications.

5. GENE-BASED HETEROGENEOUS EVALUATION TIME

While the initial experiments described above demonstrates how the asynchronous and synchronous evolutionary models compare to each other under artificial conditions, there are properties of the asynchronous model that are not shown. It was predicted that the asynchronous model would have an implicit parsimony pressure that pressured the evaluation time of the individuals. In the previous experiment this could not be observed because the evaluation time was not dependent upon the individual being evaluated. This secondary experiment will show the presence of the implicit parsimony pressure by manually encoding the evaluation time into the gene of the individual. By encoding the evaluation time into the gene it allows the Royal Road function to simulate other problems in which the evaluation time is implicitly encoded into the individual.

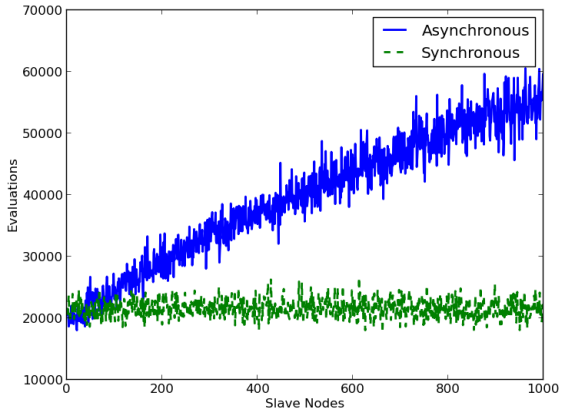

Figure 7: Plot of number of evaluations until convergence vs number of slave nodes

To encode the evaluation time into the individual, a separate gene was inserted into the individual to represent the evaluation time. This floating point number would be randomly assigned in the initial population and then would be passed to the children individuals similarly to how the standard gene is. The crossover operation for this new gene would be a uniform crossover where the child individual randomly gets one of the parents' evaluation time. The mutation operation is a gaussian mutation with a mean of 0 and a standard deviation of 0.2. If the mutation causes the evaluation time to leave the bounds of the random initialization bounds it is set to the bound it offended.

5.1 Elitist Parsimony Pressure

It is predicted that this change in the way the heterogeneous evaluation time is determined for a given individual will not change how the synchronous model performs on average. However it should affect the performance of the asynchronous model. It should cause an elitist parsimony pressure. This parsimony pressure will cause the average evaluation time to be pressured to decrease. The pressure will increase as the number of slave nodes increases. Because this parsimony pressure is elitist, this means that both models still do selection strictly on fitness of the solution. This means the best individualss will always survive in the population unlike most other parsimony pressures which can cause high-quality individuals to be discarded if they are slow running.

5.2 Evaluation Speed-Up vs Work Nodes

To see how the affect the gene-based heterogeneous evaluation times had on the two models, the time necessary to evaluate 10,000 individuals is examined for each of the mod-

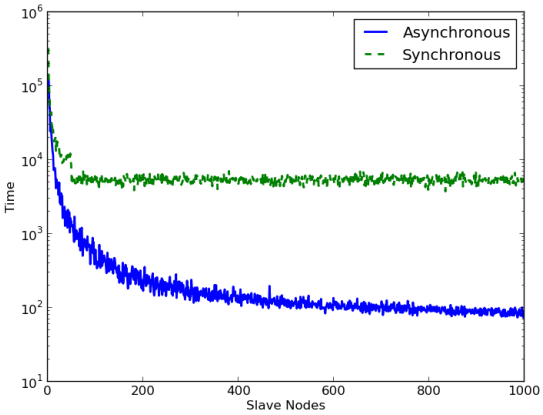

Figure 8: Plot of evaluation time vs number of slave nodes with gene-based heterogeneous evaluation time

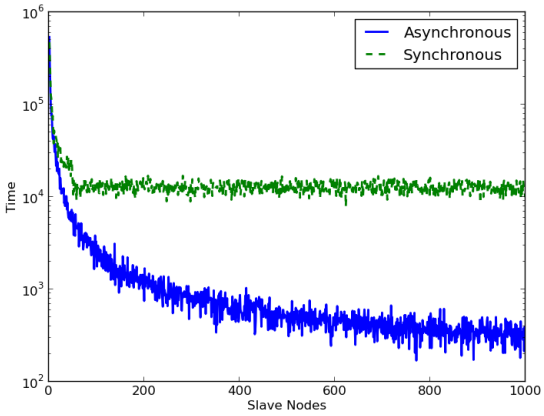

Figure 10: Plot of convergence time vs number of slave nodes with gene-based heterogeneous evaluation time

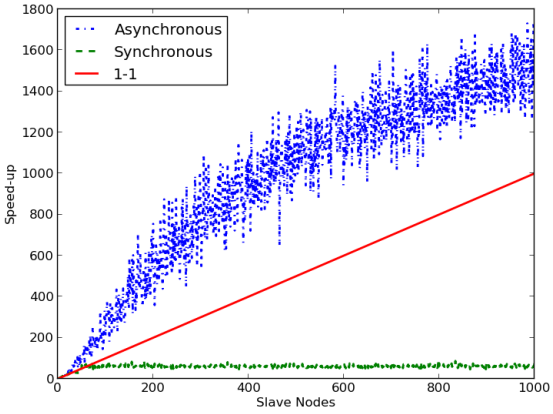

Figure 9: Plot of speed-up of evaluation time vs number of slave nodes with gene-based heterogeneous evaluation time

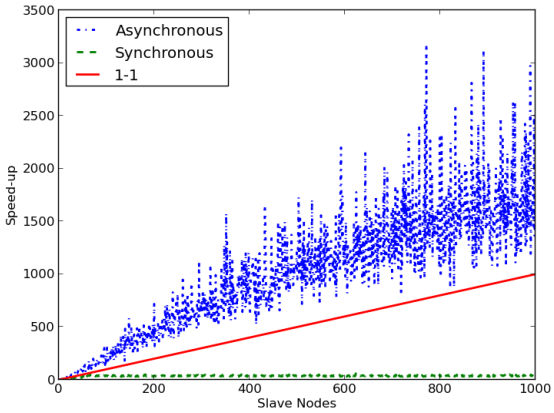

Figure 11: Plot of speed-up of convergence time vs number of slave nodes with gene-based heterogeneous evaluation time

els. Figure 8 shows the results of this experiment. Figure 9 shows the speed-up as the number of slave nodes is scaled. These results are computed by comparing the runtime of a given model using one node and a given number of slave nodes. As can be seen in the graph, with the gene-based heterogeneous evaluation times, the speed-up of the evaluation time is well over the one to one line. Thus, using ten slave nodes will give you more than a ten times speed-up when strictly comparing the evaluation time.

5.3 Convergence Time vs Work Nodes

The previous experiment showed that the more realistic gene-based heterogeneous evaluation times created a parsimony pressure on the population which allowed the asynchronous model to evaluate individuals much faster than with the prior heterogeneous evaluation time method. This experiment was extended to see how the gene-based heterogeneous evaluation time would similarly affect the population when the models were run until convergence instead of

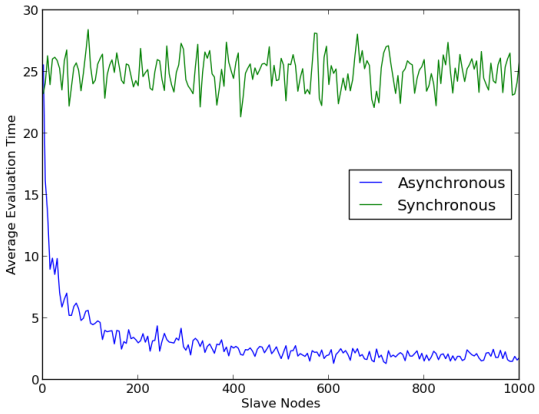

Figure 12: Plot of the average evaluation time in the final population as the number of slave nodes is increased

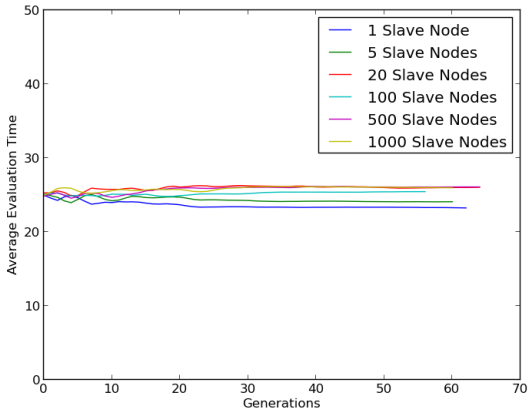

Figure 13: Plot of the average evaluation time of the population during the run in the synchronous model

a constant number of evaluations. Figure 10 shows the converge time of the two models as the number of slave nodes is scaled. Figure 11 shows the speed-up of the convergence time as the number of slave nodes is increased. This figure shows that even while running until convergence, the asynchronous model still exeeds the one to one line.

5.4 Analysis of Elitist Parsimony Pressure

The gene-based hetereogeneous evaluation time proposed in the previous set of experiments allows the simulation to act more like a problem where the evaluation time is implicitly part of the gene. The results when compared to the experiments where the evaluation times were not encoded in the genes show that there is a parsimony-like pressure acting on the population. This parsimony pressure is unlike standard parsimony pressure as it does not affect the fitness of the individual. This means the parsimony pressure is elitist as the ordering of the solutions does not changed based on the evaluation speed of the individual.

To gain more insight into the elitist parsimony pressure, analysis was done of the average evaluation times of the population as the number of slave nodes is scaled. First the evaluation time of the final populations are analyzed to determine how scaling te number of slave nodes affects the elitist parsimony pressure. The second analysis is to see how the elitist parsimony pressure acts on the population during a run.

The first analysis done was to see what the average evaluation time of the final population of the models as the number of slave nodes is increased. This will demonstrate how the parsimony pressure affects the population as the number of slave nodes is scaled. Figure 12 shows the average evaluation time of the individuals in the final population for each given run. These results are averaged over 100 runs for each number of slave nodes tested. As can be seen, there is a general trend down to roughly 1.5 seconds. From this graph it appears as though the parsimony pressure is increased as the number of slave nodes is increased.

The second analysis done was to see how the parsimony pressure affected the evaluation time of a population during the run. This was done by calculating the average evaluation time of the population sporadically through the run. For accurate results, 100 runs were averaged to generate the following figures. Figure 13 shows a plot of the average evaluation time of the population during the run of the synchronous model. As can be seen in this graph, the average evaluation time stays near 25 seconds regardless of the number of slave nodes used. Figure 14 shows a plot of the average evaluation time of the population during the run of the asynchronous model. This plot shows that at the beginning of the run starts at 25 seconds, then is pressured down until it converges. The time at which these runs converge decreases as the number of nodes in increased which corresponds to Figure 12.

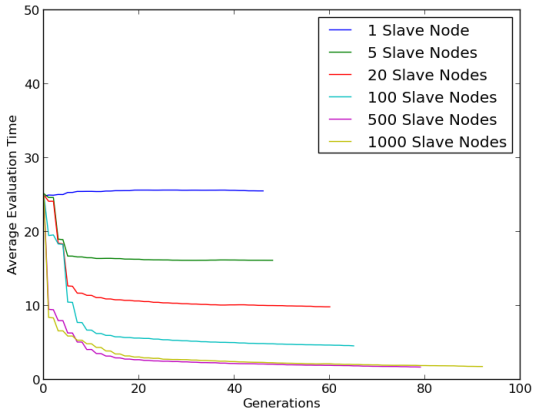

Figure 14: Plot of the average evaluation time of the population during the run in the asynchronous model

It has been established that the asynchronous model causes a elitist parsimony pressure as the number of slave nodes. This elitist parsimony pressure pressures the individuals in the population to become faster. While these previous experiments have show that this pressure exists, the scale at which this pressure can act is not necessarily as strong as shown in these experiments. In real problems, the heteroge-

neous evaluation time exists implicitly in the gene instead of explicitly in the gene as we have done in these experiments. It is predicted that the elitist parsimony pressure introduced by the asynchronous model would be similar to standard parsimony pressure in effectiveness, though there is uncertainty in how the variability in the parsimony pressure would act on real problems.

6. CONCLUSIONS

This paper has shown that APEAs, by removing the generational synchronization step, can be scaled much more efficiently than synchronous PEAs. Also, by continually creating individuals instead of in batches, this effiecient scaling can be continued well beyond the limits of batched creation. These features do however cause the necessary number of evaluations to be increased, this penalty is outweighed by the speed at which these evaluations can be completed. Meaning that when time is the driving constraint, that using APEAs is more efficient use of resources than synchronous PEAs.

This paper also demonstrated that the predicted "elitist" parsimony pressure is induced by using the asynchronous model mentioned in this paper. This "elitist" parsimony pressure attempts to minimize the evaluation time of the individuals in the population without affecting the fitness of the individual as with many standard parsimony pressures. This allows the selection operators to act based on the true fitness of the individual.

7. FUTURE WORK

While this paper has demonstrated the advantages of using the asynchronous model versus the synchronous model when attempting to parallelize and scale these models, there are still many other variations of the asynchronous model. Many modifications can be made to the population mechanics that may help or hinder the asynchronous model. A study into the various ways to handle population mechanics would greatly help the field of asynchronous evolution.

Another aspect of the asynchronous model that should be studied further is the elitist parsimony pressure. The elitist parsimony pressure has been shown to exist when the heterogeneous evaluation time is explicitly placed into the gene, though a study should be done to ensure that when when the heterogeneous evaluation time is implicitly in the gene. If the elitist parsimony pressure affects evolution when the heterogeneous evaluation time is implicitly in the gene, a study should be done to see the extent that the parsimony pressure has on a population.

8. REFERENCES

[1] E. Alba. Parallel evolutionary algorithms can achieve super-linear performance. *Information Processing Letters*, 82(1):7–13, 2002.

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

[3] E. Alba and J. M. Troya. Analyzing synchronous and asynchronous parallel distributed genetic algorithms. *Future Generation Computer Systems*, 17(4):451–465, 2001.

[4] A. W. Churchill, P. Husbands, and A. Philippides. Tool sequence optimization using synchronous and asynchronous parallel multi-objective evolutionary algorithms with heterogeneous evaluations. In *Evolutionary Computation (CEC), 2013 IEEE Congress on*, pages 2924–2931. IEEE, 2013.

[5] J. J. Durillo, A. J. Nebro, F. Luna, and E. Alba. A study of master-slave approaches to parallelize nsga-ii. In *Parallel and Distributed Processing, 2008. IPDPS 2008. IEEE International Symposium on*, pages 1–8. IEEE, 2008.

[6] M. Mitchell, S. Forrest, and J. H. Holland. The royal road for genetic algorithms: Fitness landscapes and ga performance. In *Proceedings of the first european conference on artificial life*, pages 245–254. Cambridge: The MIT Press, 1992.

[7] M. Oussaidene, B. Chopard, O. V. Pictet, and M. Tomassini. Parallel genetic programming and its application to trading model induction. *Parallel Computing*, 23(8):1183–1198, 1997.

[8] M. Yagoubi and M. Schoenauer. Asynchronous master/slave moeas and heterogeneous evaluation costs. In *Proceedings of the fourteenth international conference on Genetic and evolutionary computation conference*, pages 1007–1014. ACM, 2012.

[9] M. Yagoubi, L. Thobois, and M. Schoenauer. Asynchronous evolutionary multi-objective algorithms with heterogeneous evaluation costs. In *Evolutionary Computation (CEC), 2011 IEEE Congress on*, pages 21–28. IEEE, 2011.