Communicating Process Architectures 2003
Jan F. Broenink and Gerald H. Hilderink (Eds.)
IOS Press, 2003

Accurate Calculation of Deme Sizes for a Parallel Genetic Scheduling Algorithm

Michelle Moore
Department of Computing and Mathematical Sciences
Texas A&M University-Corpus Christi
mmoore@sci.tamucc.edu

Abstract. The accuracies of three equations to determine the size of populations for serial and parallel genetic algorithms are evaluated when applied to a parallel genetic algorithm that schedules tasks on a cluster of computers connected via shared bus. This NP-complete problem is representative of a variety of optimisation problems for which genetic algorithms (GAs) have been shown to effectively approximate the optimal solution. However, empirical determination of parameters needed by both serial and parallel GAs is time-consuming, often impractically so in production environments. The ability to predetermine parameter values mathematically eliminates this difficulty. The parameter that exerts the most influence over the solution quality of a parallel genetic algorithm is the population size of the demes. Comparisons here show that the most accurate equation for the scheduling application is Cantú-Paz' serial population sizing calculation based on the gambler's ruin model [1]. The study presented below is part of an ongoing analysis of the effectiveness of parallel genetic algorithm parameter value computations based on schema theory. The study demonstrates that the correct deme size can be predetermined quantitatively for the scheduling problem presented here, and suggests that this may also be true for similar optimisation problems. *This work is supported by NASA Grant NAG9-140.*

1 Introduction

Many cluster computing applications require efficient scheduling of tasks. The scheduling model described here is being developed for use inside a computational parallelisation and optimisation tool that runs on a Beowulf-type cluster of PCs. The cluster is used for a variety of applications in computational science. Several of these applications require "production" runs, where information about execution time and communication time is known. The overall execution time for these projects tends to be large, and computing resources are limited. Time expended to obtain a good schedule for these tasks is well spent.

This research builds on work that examined task scheduling with sequential GAs [11]. In [12], high quality schedules were reported when a distributed GA was implemented by dividing the population of the GA into multiple "demes" with each deme executing on a separate processor. In [13], results obtained by Goldberg [6], Goldberg, Deb, and Clark [7] and recent breakthroughs by Cantú-Paz [1] were applied for the first time to the scheduling problem, with encouraging results. This paper presents an in-depth examination of the sizing equations developed in [7] and [1] in terms of their applicability to the scheduling problem.

The remainder of this paper is organised as follow. Section 2 describes the scheduling

problem model and its representation within the genetic metaphor. Section 3 describes the genetic algorithm operators and parameters that were held constant to provide a stable environment for the experiments. Section 4 presents the three sizing equations and defines the variables of the equations. Section 5 describes the design of the experiments, and the problem-specific variable values used in the sizing calculations. Section 6 discusses the results of the experiments. Section 7 presents conclusions and directions for further study.

2 The Task Scheduling Problem

2.1 The Computational Model

The cluster consists of m processors $(p_1,...,p_m)$ connected by a shared Ethernet bus. Tasks contain an execution time estimate and a communication time estimate (e,c). The n tasks can be represented as $(t_1,...,t_n)$ or as $((e_1,c_1),...,(e_n,c_n))$.

Tasks are created and scheduled by an initial processor designated p_1. Tasks assigned to p_1 do not require the scheduling of communication time. The time required to send the task assignments to the processors is assumed to be constant and is therefore not considered. All tasks executions must complete before the corresponding result communication can be placed on the bus. Any final computations that must occur after all results are communicated back to p_1 are unaffected by the particular schedule, and are not included in the schedule quality evaluations.

The quality or "fitness" of the schedule is the total time required for all tasks to complete execution. An exhaustive search of all possible schedules (even using branch-and-bound) will require time bounded by $O(m^n)$ where n is the number of tasks to schedule, and m is the number of processors to schedule. The addition of a communication bus that also must be "scheduled" increases the complexity of the problem. Since the problem is NP-complete, approximation algorithms for scheduling attempt to create schedules as close to the optimal as possible.

Processors are identical. Tasks are independent. Only one message may be on the bus at a time. The result message is sent as soon as possible after task execution completion. The system is non-pre-emptive. A processor will not remain idle if a task is available. Only one task will execute on a processor at a time.

2.2 The Genetic Metaphor

A schedule is represented as a "chromosome" consisting of n genes, where n is the number of tasks to be scheduled. Each position of the chromosome corresponds to a task, and the values at each position correspond to a processor number. A schedule for 6 tasks on 4 processors might appear as 3 4 2 2 1 2, indicating that t_1 is executed on p_3, t_2 is executed on p_4, and so on. Chromosomes are also referred to as strings. The cardinality of the string alphabet is the number of digit values that may appear on the string. In the example above, the cardinality of the alphabet is 4. Many genetic algorithms use binary alphabets, but in general, we may refer to a χ-ary alphabet where the cardinality is χ.

3 Genetic Operators and Parameters

Genetic Algorithms employ metaphors from biology and genetics to "evolve" an initial population of solutions (schedules in this case) into high quality solutions. In the original serial version of this algorithm, the initial population of schedule chromosome strings is

randomly generated. The fitness, or quality of each schedule is evaluated. The "fitter" schedules are saved and used as a "mating pool". These schedules are paired randomly to create new schedules until there are enough "offspring" to return the population to its original size.

The fitness of an individual chromosome string is evaluated as *fitness = -1 * Schedule-Execution-Time*. In each generation, individuals with fitness greater than or equal to the average fitness are allowed to survive to the next generation. This is a rank-based selection method that combines truncation selection and (γ + λ) selection. The top $1/\nu$ of the population is selected. Then the $\gamma = 1/\nu$ parents mate to create λ offspring. The union of γ and λ constitute the next generation. In this manner, very fit individuals may survive to mate repeatedly.

Mating is accomplished using uniform crossover with a .5 crossover probability. In other words, when creating an offspring from a pair of schedules, the offspring's processor assignment for each task is equally likely to match either parent. The mutation rate is .02. This means that after crossover, the GA allows for a .02 probability for each task of being reassigned to a randomly chosen processor. The process of selection, mating, crossover, and mutation iterates for a set number of generations, predetermined by the user of the scheduler.

To allow schedules to be generated more quickly, the serial scheduling algorithm was rewritten in LAM MPI to execute on multiple cluster processors. In the distributed algorithm, the initial population of schedules is distributed among the available cluster nodes. Each of these subpopulations, or demes, is allowed to evolve in isolation for a set number of generations called an "epoch". After each epoch, the more fit individuals of each deme are copied and distributed among the other demes, replacing the least fit individuals. After each migration, the local iterations of selection, mating, crossover, and mutation resume. After the execution of the preset number of generations in an epoch elapse, a new migration occurs. Again, the amount of time this continues is predetermined by the user of the scheduler.

Migration occurs after selection and before mating every three generations. The migration rate is set to the maximum that can occur for the deme with the most survivors after selection. Migration follows what is called a "best-worst policy", since migrants take the place of the least fit individuals in each deme.

4 The Sizing Equations

Three sizing equations were applied, and the resulting schedules were analysed. Detailed explanation of the derivation of the first equation can be found in [7]. The second and third equations are described extensively in [1]. Equation (1) and Equation (2) were designed for serial GAs, but the population size for each deme, n_d can be found by dividing the serial population size n by the number of demes r. Equation (3) was designed specifically for parallel genetic algorithms. Consideration of collateral noise (the noise of other partitions when deciding between best and second best building block) is built into the equations. External noise may exist if fitness cannot be directly calculated. The scheduling application did not require adjustments for external noise.

In the past, the determination of population and deme sizes for practical applications of GAs required a great deal of what was frequently termed "empiricism", i.e., trial and error. The three equations below represent state-of-the-art sizing estimations based on statistical analysis of the behaviour of parallel genetic algorithms for simple theoretical applications. Goldberg, Deb, Clark, Cantú-Paz, and others have based their in-depth analysis on the somewhat controversial schema theorem. The focus of this study of the equations was not to confirm or dispute the underlying validity of the theorem, but to determine if any of the

equations could be adapted to provide useful a priori sizing values for the cluster scheduler.
$n_d = n / r$

Equation (1) $n = 2 z^2 \chi^k m' (\sigma^2_{rms} / d^2)$
Equation (2) $n = -2\chi^{k-1} ln(\alpha) (\sigma_{bb} sqrt(\pi m') / d)$
Equation (3) $n_d = \chi^k (ln(1 - P_{bb})) / ((sqrt(\delta)\tau' + 1) ln(q/p))$

These equations and their application to the scheduling problem are discussed individually below.

4.1 Equation (1)

$$n = 2 z^2 \chi^k m' (\sigma^2_{rms} / d^2)$$

The symbols used in Equation (1) are defined below.

(1.1) α ~ probability of making the wrong choice between two competing schema

(1.2) $z(\alpha)$ ~ ordinate of a unit one-sided, normal deviate at α

(1.3) z^2 ~ square of a one-sided, normal deviate at α

(1.4) χ ~ cardinality of the chromosome string alphabet

(1.5) k ~ building block size, the number of fixed positions in the schema

(1.6) lval ~ length of the chromosome (schema)

(1.7) m ~ number of partitions in the string, number of building blocks, lval / k

(1.8) m' ~ (m - 1)

(1.9) f_{max} ~ maximum fitness function value possible

(1.10) f_{min} ~ minimum fitness function value possible

(1.11) σ^2_f ~ overall variance of all schemas for each fixed position, $= \sum_{i=1}^{m} \sigma^2_{fi}$
$\approx (f_{max} - f_{min})^2 / 12$. [7]

(1.12) σ^2_{rms} ~ root mean squared of the sub function variances, $= \sigma^2_f / m$

(1.13) d ~ signal difference for peak to peak (mean to mean) function difference between the best and the second best competing individuals, this affects the likelihood of choosing the better individual

4.2 Equation (2)

$$n = -2\chi^{k-1} ln(\alpha) (\sigma_{bb} sqrt(\pi m') / d)$$

The symbol π represents the actual constant value 3.14159...

The additional symbol used in Equation (2) is defined below.

(2.1) σ_{bb} ~ sqrt (σ^2_f) the average building block standard deviation

4.3 Equation (3)

$$n_d = \chi^k(ln(1- P_{bb})) / ((sqrt(\delta)\tau'+1) ln(q/p))$$

The additional symbols used in Equation (3) are defined below.

(3.1) δ ~ degree of a deme, the number of neighbours it has

(3.2) τ ~ number of epochs, the generations between migrations constitute an epoch

(3.3) τ' ~ $(\tau - 1)$

(3.4) q ~ probability of losing a copy of the best building block between generations

(3.5) p ~ $(1 - q)$, $\approx 1/2 + \psi / (sqrt(2\pi))$, where $\psi = d / (\sigma_{bb}\, sqrt(2m'))$

(3.6) Q ~ required problem solution quality, $= (1 - \alpha) m$

(3.7) Pbb ~ probability that one partition in one deme is correct, probability of success per deme, $\approx Q / m$ - sqrt (ln (r)) / sqrt (2m)

5 Experiment Design

In order to evaluate the quality of the schedules that were produced, actual optimal schedules were found for the data sets. Because of the exponentially increasing amount of time required to find the comparison optimal schedules, the number of tasks in the tests sets was kept small. Data sets consisted of 20 normally distributed, randomly generated (e,c) task pairs. The mean of the execution time was 25 with a standard deviation of 2%. Three communication means were used, 10, 25, and 40, (std = .02) to vary the relative effects of communication overhead. The sizing program generated population and deme sizes for each of the equations using probabilities of success of .7, .75, .80, .85, .90, .95, .99. or corresponding z^2 of 0.2809, 0.4489, 0.7056, 1.0816, 1.6641, 2.7225, 5.4289. Five runs were averaged for each communication mean at each confidence level. This was done in order to smooth out stochastic variance. Size values were generated at each confidence level, with each of the three communication means.

The following variable values were set.

r	=	4
χ	=	3
k	=	1
lval	=	20
m	=	20
m'	=	19
d	$\approx$	$(1 - \mu_e) / 2$
f_{max}	=	if $\mu_c = 10$, lval $\mu_e + \mu_c$
		if $\mu_c = 25$, lval $\mu_e + \mu_c$
		if $\mu_c = 10$, lval $\mu_c + \mu_e$

f_{min} = if $\mu_c = 10, 1/3$ lval $\mu_e + 2\mu_c$
if $\mu_c = 25, 1/2$ lval $\mu_e + \mu_c$
if $\mu_c = 40, 1/3$ lval $\mu_c + \mu$

τ = 3
δ = 3
p = .508743

6 Results

The population sizes were tested in the parallel genetic scheduling algorithm. Figures 1-3 compare the expected solution quality as predicted by the equations to the average measured solution quality at each deme size. An expected quality of 70% meant that using the deme size calculated at the 70% confidence level was expected to produce schedules able to execute in an amount of time exceeding the optimal by 30%. The quality achieved with deme sizes from Equation 2 follows the expected values the most closely. Figure 4 shows the deme sizes calculated by each of the equations. Both Equations 2 and 3 are very low compared to those calculated by Equation 1. Since larger demes take longer to process, all other things being equal, Equations 2 and 3 would both appear better. Figure 1 indicates that Equation 1 overestimates the required population size at the lower quality levels. However, Equation 3 underestimates the deme size at the 99% quality level, and overestimates at lower levels (Fig. 3). In terms of deme size and closeness to quality estimates, Equation 2 appears superior.

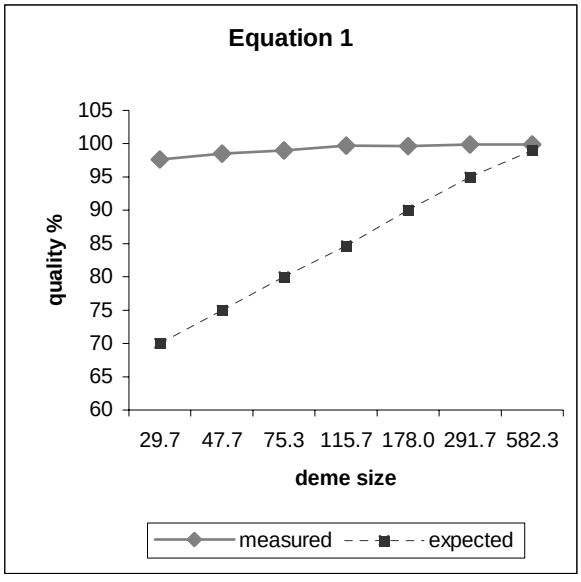

Figure 1: Performance of Equation 1

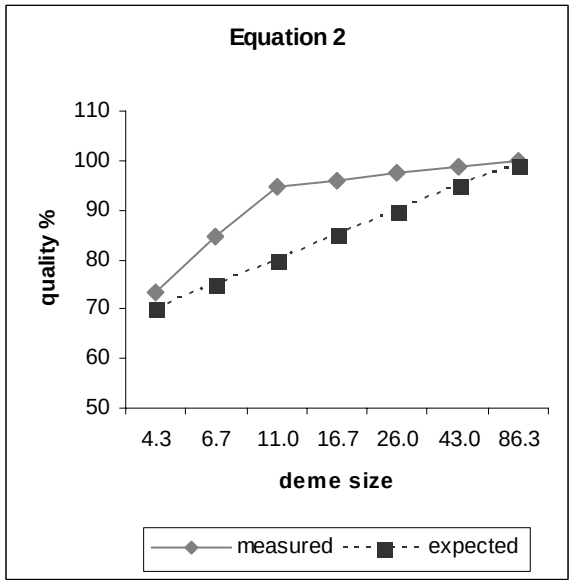

Figure 2: Performance of Equation 2

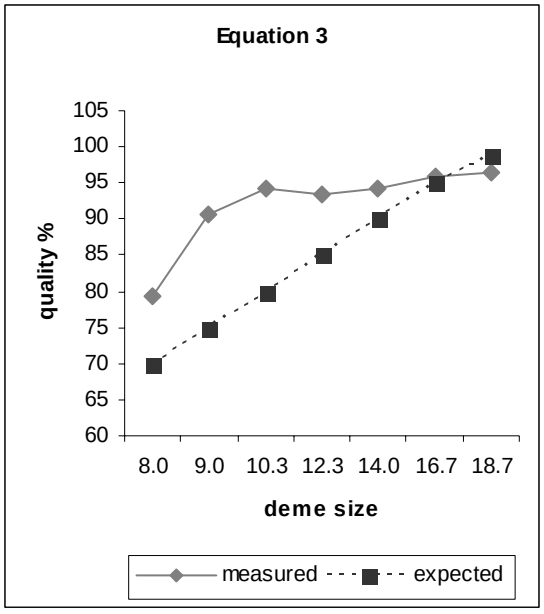

Figure 3: Performance of Equation 3:

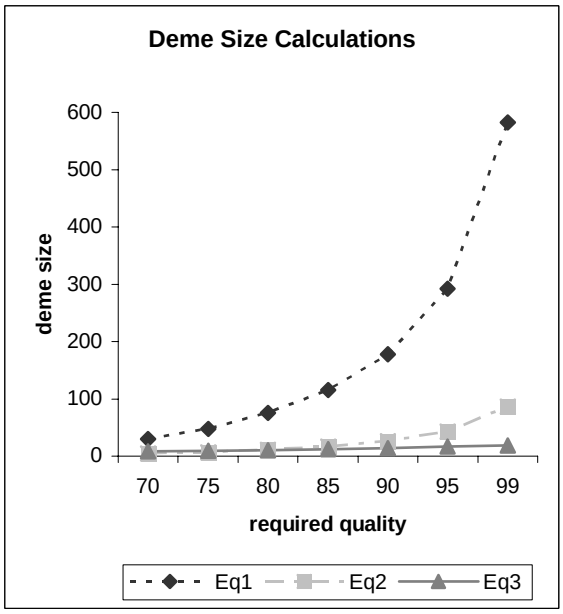

Figure 4: Comparison of Deme Size

Viewing Figure 5, it is important to note that all of the equations lead to deme sizes that produce good schedules. The farthest miss of required accuracy was Equation 3 which reached 95% accuracy when 99% was needed. The other two equations specify deme sizes that always allow the scheduler to meet the 99% accuracy requirement, with the scheduler finding the optimal schedule frequently, and average schedules better than 99.5% in all other cases.

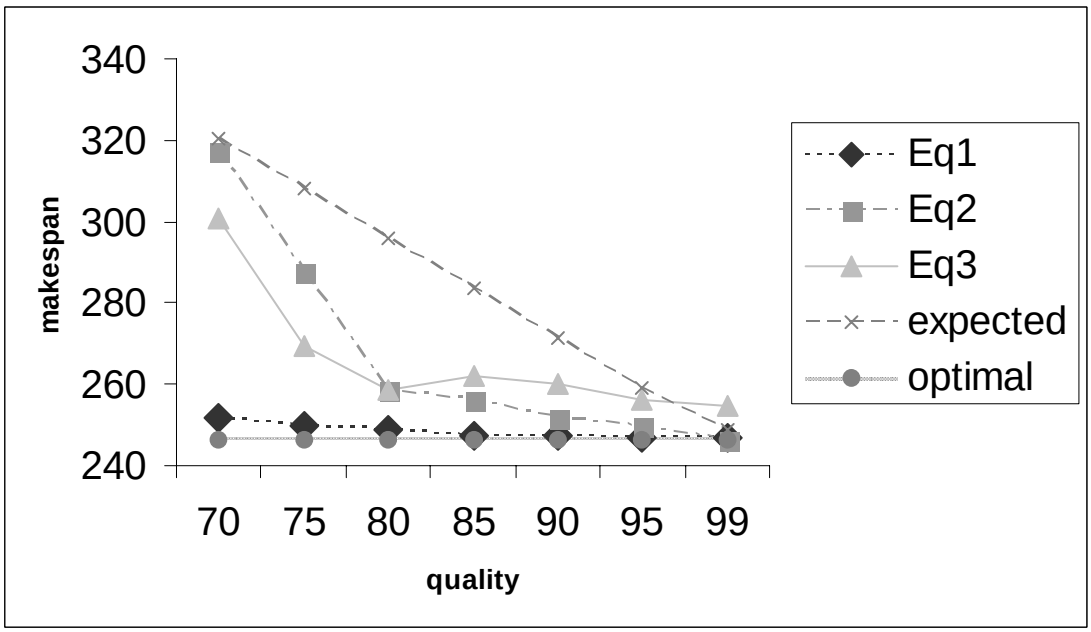

Figure 5: Schedule Execution Times Compared to Optimal and Expected

7 Conclusions and Future Work

Equation 2, which combined analysis using the gambler's ruin model and schema theory, provided the best deme sizes when the population sizes it generated were divided by the number of demes. The accuracy of the scheduling algorithm was as good as or better than that expected in all cases. In addition, the measured quality was closer to the expected solution quality than that of the other two equations. Equation 2 is a more accurate model and is successful at attaining desired quality levels without overestimating population size. This means that schedules using the sizing results will take only the time needed to provide the desired result quality. This can be especially important in time-critical applications. In addition, the ability to determine population sizing values mathematically rather than by trial and error will reduce the time required to determine schedules for new applications.

Other than the communication time mean and the execution time mean for the data, all values required by the equations can be calculated using theoretical analysis or knowledge of the problem domain. This suggests that the equations can be applied to problems where very little is known about the specific problem data. If the mean is known, function max and function min can be estimated. Where max and min are known, the mean can be estimated.

There are many directions for future work. There are various equations for deciding in advance the optimal values of many of the other parallel GA parameters. These will also be applied to the parallel scheduling problem. In addition, it will be interesting to see if calculated size values are equally successful with multi-objective, multi-constrained applications.

Acknowledgements

Research Assistants Brian McCord, Tzintzuni Garcia, William Jackson, Simon San Miguel, and Jason Picarazzi have contributed and continue to contribute to this project. *This work is supported by NASA Grant NAG9-1401*.

References

[1] Cantú-Paz, E., *Efficient and Accurate Parallel Genetic Algorithms,* Dordrecht, The Netherlands: Kluwer Academic Publishers Group, 2000.

[2] Coffman, E., Introduction to Deterministic Scheduling Theory, *Computer and Job-Shop Scheduling Theory,* New York, NY: John Wiley and Sons, 1976.

[3] De Jong, K. and Spears, W., Using Genetic Algorithms to Solve NP-complete Problems, *Proceedings of the Third International Conference on Genetic Algorithms,* 1989, 124-132.

[4] Eshelman, L., Caruana, R. and Schaffer, J., Biases in the Crossover Landscape, *Proceedings of the Third International Conference on Genetic Algorithms,* 1989, l0-19.

[5] Goldberg, D., *Genetic Algorithms Search, Optimization, and Machine Learning,* Reading, MA: Addison-Wesley, 1989a.

[6] Goldberg, D., Sizing Populations for Serial and Parallel Genetic Algorithms, *Proceedings of the Third International Conference on Genetic Algorithms,* 1989b, 70-79.

[7] Goldberg, D., Deb, K., and Clark, J., Genetic Algorithms, Noise, and the Sizing of Populations, *Complex Systems, 6,* 332-362.

[8] Holland, J., *Adaptation in Natural and Artificial Systems*, Ann Arbor, MI: University of Michigan Press, 1975.

[9] Horowitz, E. and Sahni, S., Exact and Approximate Algorithms for Scheduling Nonidentical Processors, *Journal of the ACM*, vol. 23, no. 2, 1976, 317-327.

[10] Hou, E., Hong, R. and Ansari, N., Efficient Multiprocessor Scheduling Based on Genetic Algorithms, *Proceedings of the 16th Annual Conference of the IEEE Industrial Electronics Society*, 1990, 1239-1243.

[11] Kidwell (Moore), M., Using Genetic Algorithms to Schedule Distributed Tasks on a Bus-Based System, In Forrest, S. (Ed.), *Genetic Algorithms: Proceedings of the Fifth International Conference*, San Mateo, CA: Morgan Kaufmann, 1993.

[12] Moore, M., Parallel Genetic Algorithms to Find Near Optimal Schedules for Tasks on Multiprocessor Architectures, In Chalmers, A., Mirmehdi, M., and Muller, H. (Eds.) *Communicating Process Architectures 2001*, Concurrent Systems Engineering Series, IOS Press, vol. 59, 2001, pgs, 27-36.

[13] Moore, M., An Accurate and Efficient Parallel Genetic Algorithm to Schedule Tasks on a Cluster, *Proceedings of the International Parallel and Distributing Processing Symposium*, 2003.

[14] Salleh, S. and Zomaya, A., *Scheduling in Parallel Computing Systems*, Dordrecht, The Netherlands: Kluwer Academic Publishers Group, 2000.

[15] Syswerda, G., Uniform Crossover in Genetic Algorithms, *Proceedings of the Third International Conference on Genetic Algorithms*, 1989, 2-9.