Adaptive Sizing of Populations and Number of Islands in Distributed Genetic Algorithms

Johan Berntsson
School of Software Engineering and Data Communications
Queensland University of Technology
QLD 4001, Australia
j.berntsson@qut.edu.au

Maolin Tang
School of Software Engineering and Data Communications
Queensland University of Technology
QLD 4001, Australia
m.tang@qut.edu.au

ABSTRACT

Deciding the appropriate population size and number of islands for distributed island-model genetic algorithms is often critical to the algorithm's success. This paper outlines a method that automatically searches for good combinations of island population sizes and the number of islands. The method is based on a race between competing parameter sets, and collaborative seeding of new parameter sets. This method is applicable to any problem, and makes distributed genetic algorithms easier to use by reducing the number of user-set parameters. The experimental results show that the proposed method robustly and reliably finds population and islands settings that are comparable to those found with traditional trial-and-error approaches.

Categories and Subject Descriptors: D.2 Software Engineering: Miscellaneous

General Terms: Performance Reliability Algorithms

Keywords: genetic algorithms, internet computing, population sizing, adaptation

1. INTRODUCTION

Genetic algorithm (GA) is a general optimization method which is easy to use and apply to a wide range of problems [1]. However, the performance of the GA depends on a number of parameters including representation, crossover and mutation rates, population size, and selection pressure. For efficiency reasons it is often advantageous to parallelize and distribute the GA over several processors, which typically introduces even more parameters, such as the number of islands, and migration parameters. Determining the proper parameter set is a non-trivial task which depends on the nature of the problem and its representation, and the GA operators. Many practitioners rely on trying various combinations of the parameters. This approach obviously requires a lot of computation which sometimes is larger than the time used for solving the problem itself. An alternative approach is to use adaptive methods that adjust the parameters according to observed performance during the run of the GA.

Finding the proper population size for a given GA problem is of crucial importance for good performance, and this paper outlines an adaptive technique that automatically searches for good settings of island population sizes and the number of islands, by the means of competition in distributed genetic algorithms (DGA). While other parameters such as crossover, mutation rates, and migration policy are important, they are also fairly tolerant, while a proper population sizing is crucial and can to some extent overcome suboptimal settings of other parameters. If the population size or the number of islands is too small the GA will convergence prematurely, and too large values are inefficient. A manual approach to population sizing could be to first try with a small population and number of islands, and then increase each parameter until no further gain in performance is detected. The proposed method works in a similar manner, but instead of working in an ad-hoc way, the population adapter automates the process and hides it from the user. It is applicable to any problem, and makes DGAs easier to use by reducing the number of user-set parameters and aiding in construction of robust GA applications with good performance.

GECCO'05, June 25–29, 2005, Washington, DC, USA.
ACM 1-59593-010-8/05/0006.

2. PROPOSED METHOD

The basic idea of the population adapter is to run several distributed GAs with competing sets of number of islands/island size settings in parallel. Let n denote the number of islands, and d denote the island size, which is the same for all islands. The total population is $n * d$. At any given time three DGAs (denoted DGA_0, DGA_1, and DGA_2) are run in parallel with the following set of parameters: $< n, d >, < n, d * 2 >, < n * 2, d >$ This sets up a competition between the basic DGA_0, DGA_1 with more islands, and DGA_2 with bigger island population sizes. The DGAs are allowed to run until one of the DGAs overtake its competitors, or the termination criterion has been met.

The method has similarities to the parameter-less GA [2], but with significant modifications: (i) both *population size*, and *number of islands* are adjusted in parallel, (ii) the population adapter is both *competitive* and *collaborative*, and (iii) the population adapter terminates automatically when no further improvements are found.

2.1 Competitive Evaluation

Evaluation of DGAs can be cut if they are being overtaken by other DGAs, or converging. Overtaking is detected by comparing average fitness of a DGA with DGAs with bigger total population size, since it is unlikely that the smaller DGA with lower average fitness will succeed in getting better optimal results than the bigger DGA. The convergence criterion is problem dependent, and should be set by the user.

Seeding	Success rate	Evaluations		Island size, number of islands						
		Mean	Stddev	80,8	80,16	80,32	160,8	160,16	160,32	320,8
Yes	10/10	407555	93437	-	-	1	3	3	2	1
No	10/10	463970	205637	1	2	1	4	2	-	-

Table 1: Population adapter with the Royal Road problem

Once convergence or overtaking is detected, the population adapter takes the following action:

- DGA_0 converges
  no action.
- DGA_1 and DGA_2 converged
  All DGAs restarted.
- DGA_1 overtakes DGA_0
  $d = d*2$, $DGA_1 \rightarrow DGA_0$, DGA_1 and DGA_2 restarted (using seeding).
- DGA_2 overtakes DGA_0
  $n = n*2$, $DGA_2 \rightarrow DGA_0$, DGA_1 and DGA_2 restarted (using seeding).

The intuition behind this algorithm is that three hypothesis are evaluated in parallel, and when a DGA overtakes its competitors, it is an indication of the need to adjust the parameter set in the direction suggested by the winning DGA. DGA_1 is testing if more islands are beneficial, and when it overtakes the other DGAs, the routine increases the number of islands in the next round of competitions. DGA_2, which is testing the benefit of increasing the population size on each island, works in the same way.

2.2 Collaborative Restart

The basic case of restarting an DGA is to simply reinitiate the population on each island. This may be inefficient, since each newly restarted DGA will need time to catch up with its competitors even if its population sizing parameter set is better. As an alternative the population adapter can use *seeding*. With seeding, each island in the newly restarted DGA reinitiates its population, but also inserts the best individuals from the other DGAs. For instance, if DGA_1 is restarted, each island in DGA_1 is seeded with the best individual from DGA_0 and DGA_2. In this way, the DGAs collaborate to give new DGAs a bias toward promising regions of the search space, which makes the population adapter more efficient.

2.3 Termination

The population adapter terminates the DGAs when there is a relatively slim chance of finding a better set of sizing parameters than the current. The termination criterion should not reply on problem specific parameters, such as a known global optima, or a maximum number of generations. Rather, the population adapter uses the convergence status of each DGA. Since it is quite common that DGAs converge in early trials because of the small values of n and d, it is not possible to terminate as soon as a convergence has been detected. The population adapter therefore does not terminate until (i) each DGA has converged at least once, and (ii) no new best solution is found during the detection phase. If a new best solution is detected, then the best solution so far is updated and the termination detection process is reset. In addition to the general termination criterion, problem specific knowledge can be used.

2.4 Experimental Results

Table 1 summarizes the outcome of ten runs of the population adapter, using Holland's Royal Road problem. To provide a reference for the population adapter experiments a series of experiments with manual settings were conducted, which suggested that a total population size of 2560 is required for good performance. The population adapter successfully finds the optimal value for all runs, but there are clear differences in the number of evaluations and the selected parameters, depending on whether seeding is used or not. Without seeding, most runs lead to comparatively small population sizes. With seeding, bigger population sizes are favoured, which is found in the manual experiments to be advantageous. This is reflected in the number of evaluations, where seeding has a lower average number of generations needed with a significantly lower standard deviation, suggesting that seeding leads to more reliable results. Additional experiments with a hard real-world VLSI floorplanning problem show similar results.

3. DISCUSSION AND CONCLUSIONS

This paper has presented a population adapter for distributed genetic algorithms that uses an adaptive approach to finding a good combination of number islands and island population sizes. The method relies on a race between DGAs using competing parameter settings, but uses collaborative method to jump-start new competitions and increase robustness and efficiency. Although the experiments were carried out of standard GAs, the method can be applied to more advanced GAs, such as messy GAs or the Bayesian optimization algorithm, without any significant modifications.

The population adapter requires about three times the effort needed to solve the problem when the optimal population sizing is known beforehand. This overhead is not unreasonable, since a manual approach would certainly require a fair number of trials as well. The population adapter has the additional benefit of relieving the user from the tedious testing process, and performing the tests in a pre-defined, systematic manner.

4. REFERENCES

[1] D. E. Goldberg. Genetic and evolutionary algorithms come of age. *Communications of the ACM*, volume 37(3), pages 113–119, ACM Press, 1994.

[2] G. R. Harik and F. G. Lobo. A parameter-less genetic algorithm. In W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. Jakiela, and R. E. Smith, editors, *Proceedings of the Genetic and Evolutionary Computation Conference*, volume 1, pages 13–17. Morgan Kaufmann, 1999.