DISTRIBUTED GENETIC ALGORITHMS WITH AN APPLICATION TO PORTFOLIO SELECTION

A. Loraschi
SIGE Consulenza S.P.A., Milano, Italy
A. Tettamanzi
Universita' degli Studi di Milano
Dip.to di Scienze dell'Informazione, Milano, Italy
Marco Tomassini
Centro Svizzero di Calcolo Scientifico, Manno
and Laboratoire de Systèmes Logiques
Ecole Polytechnique Fédérale de Lausanne
Switzerland
and
Paolo Verda
Université de Fribourg, Institut d'Informatique
Fribourg, Switzerland

Abstract

This paper presents a PVM-based coarse-grained distributed genetic algorithm implemented on workstation clusters. After successfully evaluating the algorithm with standard test functions, we apply it to a hard real-world portfolio selection problem. The distributed version easily outperforms sequential genetic algorithms and shows promise for difficult management applications.

1 Distributed Genetic Algorithms

Genetic Algorithms (GAs) have enjoyed an increasing popularity as reliable stochastic optimization, search and rule-discovering methods in the last few years. The original formulation of GAs by Holland and others in the seventies was a sequential one. This approach made it easier to reason about mathematical properties of the algorithms and was justified at the time by the lack of adequate software and hardware. However, it is clear that GAs offer many natural opportunities for parallel implementation [1]. There are several possible parallel GA models, the most popular being the fine-grained or *grid* models [2], and the coarse-grain or *island* models. In the grid models, large populations of individuals are spatially distributed on a low-dimensional grid and individuals interact locally within a small neighborhood. Massively parallel SIMD machines are especially suited for the grid model.

In the island model the population is subdivided into smaller subpopulations which evolve independently and simultaneously according to a standard GA. Periodic migrations of some selected individuals between islands allow to inject new diversity into converging subpopulations. Microprocessor-based multicomputers and workstation clusters are well suited for the implementation of this kind of parallel GA. Since we are targeting real-world industrial problems, concerns of availability, cost and portability in non-academic environments prompted us to implement the island model using workstation clusters and the PVM environment [3]. Being coarse-grained, the island model is less demanding in terms of communication speed and bandwidth, which makes it a good candidate for a cluster implementation. We employed an SPMD (Single Program Multiple Data) model in which different subpopulations exchange groups of individual in a loosely synchronous manner. The following is an outline of the basic algorithm as it has been proposed for example in [4,5,8]:

```
1 - initialize P subpopulations
     of size N each
2 - for each subpopulation do in
     parallel for M generations

       evaluate and select
       individuals by fitness.

       recombine individuals.

       mutate individuals.

       send K<N best individuals to
       a neighbouring subpopulation.

       receive K individuals from a
       neighbouring population.

       replace K individuals in
       the subpopulation.

3 - if termination criterion met
     then stop, otherwise go to 2
```

We started from two well-known public domain GA packages: GENEsYs [6] and Genitor [7]. After the code parallelization, the first step is to establish suitable values for a number of parameters of the distributed algorithm. At the very least, one has to define the topology of the communicating processes, the size N of the subpopulations, the migration frequency M, the number of individuals K involved in the migration and the individual replacement policy. One problem with the distributed/parallel GAs is that there are no rigorous prescriptions for setting those parameters, the only practical way being experimental. We found the following values by trial and error, running the distributed algorithm on a few standard test functions. The size of the subpopulations was between 100 and 200, the migration frequency was 10 generations and the number of individuals exchanged 5 to 10% of the subpopulation size. Not surprisingly, these values turned out to be comparable to those used in other studies [4, 5]. Besides, our experiments showed that small variations affect the results in a negligible manner. We experimented with only one process topology: the ring, and we tested two exchange policies: simply passing individuals to the next process in the ring, alternating directions at each swap, and passing individuals "modulo" the swap number (see [8]). The "modulo" swap gave the best results and we retained it for the subsequent tests. The replacement policy was extremely simple: the new K individuals displaced the worst K individuals of the receiving population. Here again other solutions are possible.

The goal of the next step was to test the distributed approach in order to see if it offered real advantages with respect to the standard sequential algorithm. To this end, we selected the following four hard standard test functions: Schwefel's function, low autocorrelation binary sequences, fully deceptive function and Griewank's function. They are described in detail in [6]. The task is to find the absolute minimum of each function in a given domain. In order for the results to be comparable, we experimentally fixed the number of function evaluations for each problem and imposed that both the sequential and the distributed algorithm made this same number of function evaluations. For each problem we ran both the sequential and distributed GA 30 times and took the average and best results over the whole number of experiments.

We found that, although the distributed algorithm was not optimized in any significant way, it consistently outperformed the sequential version, both on the average and best values. In all cases the distributed algorithm found the global optimum much more frequentely than the sequential one within the given number of function evaluations and its speed of convergence was always greater. Our workstation cluster was under normal usage by other people during all of our tests, giving rise to strong fluctuations in computing time. Furthermore, PVM processes are given low priority. But this is exactly the way in which these clusters are best used, since what one wants to maximize is total system throughput and PVM nicely exploits idle time on the machines. Under these conditions one can hardly talk about speedups in the usual way. We did some time measurements nevertheless and found that with eight Sun Sparc workstations the average speedup was about 4.5. Had we run the tests during nighttime, the results would have certainly been better.

These results confirm the findings of other authors [1,4,5,8] and show that distributed GAs are not only advantageous in terms of computing time, they also often solve problems more easily than the sequential algorithm and we remarked that they can better help track multiple optima. Qualitatively, this can be attributed to a reduction of premature convergence due to the exchange of fresh genetic material. This effect could perhaps be enhanced by using different GA parameter settings for different island but we did not experiment with this approach. In the next section we will put the distributed algorithm to a more stringent test: the portfolio selection problem, a difficult real-world optimisation problem.

2 Distributed Portfolio Selection

This section deals with an application of GAs to some issues raised by recent research in Portfolio Theory (PT) concerning the use of non-trivial indices of risk. The central problem of PT has to do with selecting the weights of assets in a portfolio that minimize a certain measure of risk for any given level of expected return. According to mainstream PT [9], a meaningful measure of risk is given by the variance of the distribution of portfolio returns. However, while leading to elegant analytical results and easy practical implementation, this approach fails to capture what an investor perceives as the essence of risk, that is the chance of incurring a loss. In this respect, the economic literature has recently proposed alternative measures of risk; an interesting family of measures, defined as lower partial moments [10], refers to the down-side part of the distribution of returns and has therefore become known under the name of downside risk.

A number of difficulties are encountered when trying to apply this new approach even to simple problem instances. First of all, the nature of these risk measures prevents one from devising general analytical solutions and quadratic optimization techniques cannot be applied because the shape of the objective surface is in general non-convex. Moreover, the typical sizes of real-world portfolio selection problems are of the order of several tens or even hudrends of assets; expected return and risk are calculated by using historical series made up of hundreds of historical returns. Even the software packages that apply the conventional mean-variance approach through quadratic optimization suffer limitations as the size of the problem grows beyond a certain threshold. Things become more involved as it is usual practice among fund managers to impose various constraints on their optimal portfolios.

We have carried out experiments applying genetic algorithms to the portfolio selection problem using the downside risk approach on a sequential machine [11], and obtained satisfactory results. However, the structure itself of the portfolio selection problem suggests a distributed architecture for a suitable evolutionary algorithm. The problem can be viewed as a convex combination parametric programming problem [12], in that an investor wants to minimize risk while maximising expected return. As the level of risk increases, the expected return attached to optimal portfolios draws a convex non-decreasing curve, which is called efficient frontier, which is the set of Pareto-optimal, i.e. non-dominated, portfolios. In other words, on the efficient frontier, a larger expected return corresponds to a greater risk. This can be expressed as a two-objective optimization problem:

$$\min_{\mathbf{w}}\{\mathrm{Risk}(\mathbf{w})\}$$

$$\max_{\mathbf{w}}\{\mathrm{Return}(\mathbf{w})\}$$

subject to

$$\sum_{w_i \geq 0} w_i = 1$$

These two objectives can be parametrized to yield a parametric objective

$$\min_{\mathbf{w}}\{\lambda\,\mathrm{Risk}(\mathbf{w}) - (1-\lambda)\,\mathrm{Return}(\mathbf{w})\},$$

where parameter λ is a trade-off coefficient ranging between 0 and 1. When $\lambda = 0$ the investor disregards risk and only seeks to maximize expected return; when $\lambda = 1$ the risk alone is minimized, whatever the expected return. Since there is no general way to tell which particular trade off between risk and return is to be considered the best, optimizing a portfolio means finding a whole range of optimal portfolios for all the possible values of the trade off coefficient; the investors will thus be able to choose the one they believe appropriate for their requirements. A natural way to achieve that in an evolutionary setting is to have several distinct populations evolve for a number of trade-off coefficient values. The greater this number, the finer the resolution with which the investor will be able to explore the efficient frontier. Because it is likely that slight variations in the trade-off coefficient do not significantly worsen a good solution, a natural way to sustain the evolutionary process is to allow migration or cross-breeding between individuals belonging to populations corresponding to values of the trade-off coefficient that are close together. This suggests a distributed implementation where populations are linearly arranged according to their relevant trade-off value. The distributed genetic algorithm discussed in the previous section has been slightly adapted to the present problem. The only difference being the subpopulations topology: a two-way line in which exchanges take place only between nearest neighbours i.e., those with similar trade-off coefficients. The results have been very encouraging. Fig. 1 shows a fully developed efficient frontier for a portfolio with 53 assets.

It is clearly seen that the distributed version can find portfolio weightings that offer a significantly better return for a given risk level over nearly all the risk/return region of interest. We also experimented with very large portfolios

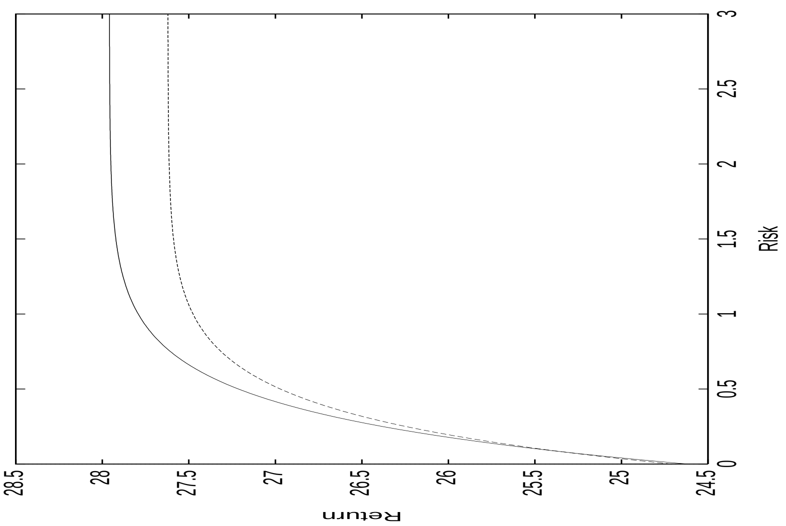

Figure 1: Efficient frontier for a 53 assets portfolio. Sequential (dashed line) and distributed (thin line).

(more than 100 assets). In all cases, the distributed GA outperformed the sequential algorithm. In a selection problem with 140 assets the distributed algorithm running on a normally loaded Sun Sparc cluster with 10 machines has been able to develop acceptable solutions in a few hours time, while we had to stop the sequential version after two days, before any reasonable curve had formed.

3 Conclusions

PVM-based distributed genetic algorithms proved to be a useful tool for solving difficult optimization problems. Both standard hard test functions and a difficult financial problem were optimized faster and more reliably by the distributed GAs than by their sequential counterparts. This opens up a whole new range of possibilities for hard financial applications, whereby larger and more realistic problem instances can be dealt with in reasonable time. The distributed GA is only slightly more complicated than the sequential version and the PVM implementation can make use of existing workstations, thus efficiently exploiting available CPU time. This makes this kind of widely available architecture more affordable than dedicated parallel computers for non-academic applications.

In the future we will pursue the application of the distributed GA to other important financial and industrial problems. The algorithm presented here is of the loosely synchronous type. We think that an asynchronous approach would be more efficient since a given subpopulation would not exchange individuals at fixed times irrespective of its internal degree of uniformity, but only when some convergence level has been attained. Work is planned for experimenting with this model.

Acknowledgement

We are grateful to Professor Gianni Degli Antoni of the Computer Science Department of the University of Milano for useful discussions and his continuing support.

References

1. H. Mühlenbein: *Parallel Genetic Algorithms, Population Genetics and Combinatorial Optimization*, in Proc. of the Third Int. Conf. on Genetic Algorithms, J. D. Schaffer (Editor), Morgan Kauffman, 416, 1989.

2. B. Manderick and P. Spiessens: *Fine-Grained Parallel Genetic Algorithms*, in Proc. of the Third Int. Conf. on Genetic Algorithms, J. D. Schaffer (Editor), Morgan Kauffman, 428, 1989.

3. PVM3 User's Guide and Reference Manual, Oak Ridge National Laboratory, Tennessee, 1993.

4. J.P. Cohoon, S.U. Hegde, W.N. Martin and D. Richards: *Punctuated Equilibria: a Parallel genetic Algorithm*, in Proc. of the Second Int. Conf. on Genetic Algorithms, J.J Grefenstette (Editor), Lawrence Erlbaum Associates, 148, 1987.

5. R. Tanese: *Parallel genetic Algorithm for a Hypercube*, in Proc. of the Second Int. Conf. on Genetic Algorithms, J.J Grefenstette (Editor), Lawrence Erlbaum Associates, 177, 1987.

6. T. Bäck: *A User's Guide to GENEsYs 1.0*, University of Dortmund, Department of Computer Science, 1992.

7. D.L. Whitley: *Getting Started with Genitor*, Computer Science Department, Colorado State University, 1990.

8. T. Starkweather, D. Whitley and K. Mathias: *Optimization Using Distributed Genetic Algorithms*, in Parallel Problem Solving from Nature, Lecture Notes in Computer Science Vol. 496, H.-P. Schwefel and R. Männer (Editors), Springer-Verlag, 176, 1991.

9. H. M. Markowitz: *Portfolio Selection*, New York, John Wiley & Sons, 1959.

10. H. V. Harlow: *Asset Allocation in a Downside-Risk Framework*, Financial Analysts Journal, Sep/Oct, 30, 1991.

11. S. Arnone, A. Loraschi and A. Tettamanzi: *A Genetic Approach to Portfolio Selection*, Neural Network World 3(6), 597, 1993.

12. R.E. Steuer: *Multiple Criteria Optimization: Theory, Computation and Application*, John Wiley & Sons, New York, 1986.