An Experience in Building a Parallel and Distributed Problem–Solving Environment

José C. Cunha Pedro Medeiros Vítor Duarte João Lourenço Maria Cecília Gomes
{*jcc, pm, vad, jml, mcg*}*@di.fct.unl.pt*
Departamento de Informática
Faculdade de Ciências e Tecnologia
Universidade Nova de Lisboa
Portugal

Abstract

We describe our experimentation with the design and implementation of specific environments, consisting of heterogeneous computational, visualization, and control components. We illustrate the approach with the design of a problem–solving environment supporting the execution of genetic algorithms. We describe a prototype supporting parallel execution, visualization, and steering. A life cycle for the development of applications based on genetic algorithms is proposed.

1 Introduction

In this paper we describe our work on the development of a problem–solving environment that can exploit the advantages of parallel and distributed processing. Our main motivation is due to the interactions that we have established with other research groups in distinct domains, namely in the environmental sciences, where parallel processing solutions are necessary to support complex simulation processes. However, such solutions must fit within heterogeneous environments which also include visualization components, interactive control components, virtual reality interfaces, and access large and complex databases.

At a higher level one must support coherent methodologies and abstractions that may be used to model the above mentioned complex applications. However, the definition of suitable high-level models and methodologies is not enough. A critical aspect of this research is the need of assessing the design and implementation options through the effective evaluation of the prototypes by end–users. This requires very flexible and extensible experimental test-beds.

So, from the application side there is an increased pressure to develop more adequate software development and execution environments in order to improve the functionalities and performance of real applications. Such tools and environments should help to manage the complexity of large scale systems, their dynamic interactions, as well as to support the building of an application through the reuse and customization of pre-existing off-the-shelf software components. There are tools that should operate at the level of each individual component, as we find in current software engineering environments. Other tools should operate at the inter-component level, in order to support the process of application building, by selecting, evaluating and testing, configuring, activating, interconnecting, and monitoring and controlling the execution of multiple heterogeneous application components.

In this paper, we describe our work on the design and implementation of a problem–solving environment for the parallel execution of genetic algorithms and we propose a methodology for its application.

The rest of the paper is organized as follows. Section 2 briefly presents the motivations and approaches to build problem–solving environments based on genetic algorithms. Section 3 discusses how several distinct components were integrated in order to generate a problem domain oriented environment for the parallel execution of genetic algorithms. Finally, we present some conclusions and outline ongoing work.

2 Problem–Solving Environments based on Genetic Algorithms

Genetic algorithms (GA) [10] are naturally suitable for solving complex optimization problems, as alternative to other search techniques such as hill-climbing or simple enumerative techniques like dynamic programming. Since their proposal many approaches have been made to exploit their implicit parallelism, according to two main models: the Island model [17, 2, 19, 20, 18], and the Neighborhood model (or fine-grain model) [15].

In [1, 4] one can find good surveys of the main approaches to parallel genetic algorithms (PGA). Problem–solving GA based environments aim at providing support for the application of GA to specific problems. They are classified into three main classes, according to [7]. Application-Oriented environments are seen as ”black boxes“ by the end-users, who are typically professional users from a specific problem domain. In some systems, a small range of application domains is supported. Algorithm-Oriented environments are programming systems supporting specific types of GA and their operators. Toolkits-Oriented environments are programming systems which provide a large diversity of supporting tools, as well as several types of GAs. They aim to be applied in general to any application domain, and provide support for the programmer to adapt the GA software according to the application needs. They typically provide several types of tools that can be integrated to build a complex environment.
Application-oriented GA environments are the ultimate goal, because they protect the end–user from the details of low level GA programming. On the other hand, general purpose, toolkit–oriented, environments, although relatively recent, are the most important for research and experimentation with new models (and to support the implementation of application-oriented GA environments).
Distributed systems including GA components can bring increased potentialities beyond the speedup obtained by parallel GA execution. Due to the complexity and heterogeneity of modern computer applications, it is often necessary to subdivide them in multiple subproblems, each possibly solved by GA or other technique, such as Neural Networks, an expert system or a pattern recognition component. In some cases, there are subpopulations in distinct components which are allowed to evolve autonoumosly using different models, but must interact in order to satisfy global constraints of the application problem [13]. In order to support such heterogeneous component-based systems, great flexibility is required in the configuration and activation of each component, the programming of their interactions, and the monitoring and control of their global and individual behavior.

3 A Genetic Algorithm Oriented Heterogenous Environment

In this section we show how we have integrated some distinct components in order to build a problem–solving environment that can be used to solve optimization problems.
GA can be used to solve a wide range of optimization problems in real life situations. Furthermore, they have high computational power requirements, but are easily parallelized [1]. Additionally they are used in applications typically requiring a highly interactive environment. Due to the large number of parameters that affect the behavior of GA and their interrelationship, it is important to offer on-line visualization of evolution of the GA simulation. It is also important to be able to perform modifications in the GA parameters, during execution, depending on the observed behavior.
The environment we have developed has facilities for visualization, in real time, of the simulation evolution and for interactive steering, corresponding to the following three components (see figure 1). A Genetic Algorithm (GA) component supports parallel genetic algorithm execution. A Data Visualization component visualizes the evolution of the GA computation. An Interactive Steering (IS) component supports the interactive steering of the computation.

3.1 The Genetic Algorithm Component

A common approach for the parallelization of genetic algorithms exploits a master-slave scheme to subdivide the evaluation of a whole population of individuals. All the population is managed by a master process that subdivides it in slices (each with a certain number of individuals) and distributes it to the slaves. These slaves evaluate the GA fitness function and return the results to the master (as, for example, in the PGAPack package [14]). Another approach is to scatter the population as a set of independent populations that evolve separately. Each set is called an *island*. In each island the evolution rules and parameters can be different, and, from times to times, some individuals can migrate between islands in order to allow better convergence behavior. Two prototypes were developed using this model, one for PVM [5], the other for MPI-1 [16]. Additional experimentation exploited hybrid GA and Simulated Annealing techniques to better control GA convergence [6].
This experimentation suggests the interest for heterogeneous execution of GA components. It also allows to access and effectively integrate existing GA packages like the PGAPack, requiring MPI-1 support, into an environment where other GA components are evaluated in PVM platforms. In order to experiment with such an idea, an extension to the island model provides an aggregated set of islands that execute in a particular hardware configuration, communicating internally by the most efficient interprocess communication mechanism available. Individuals can migrate within each aggregate as well as between the aggregates by using an interconnection model dealing with heterogeneity.

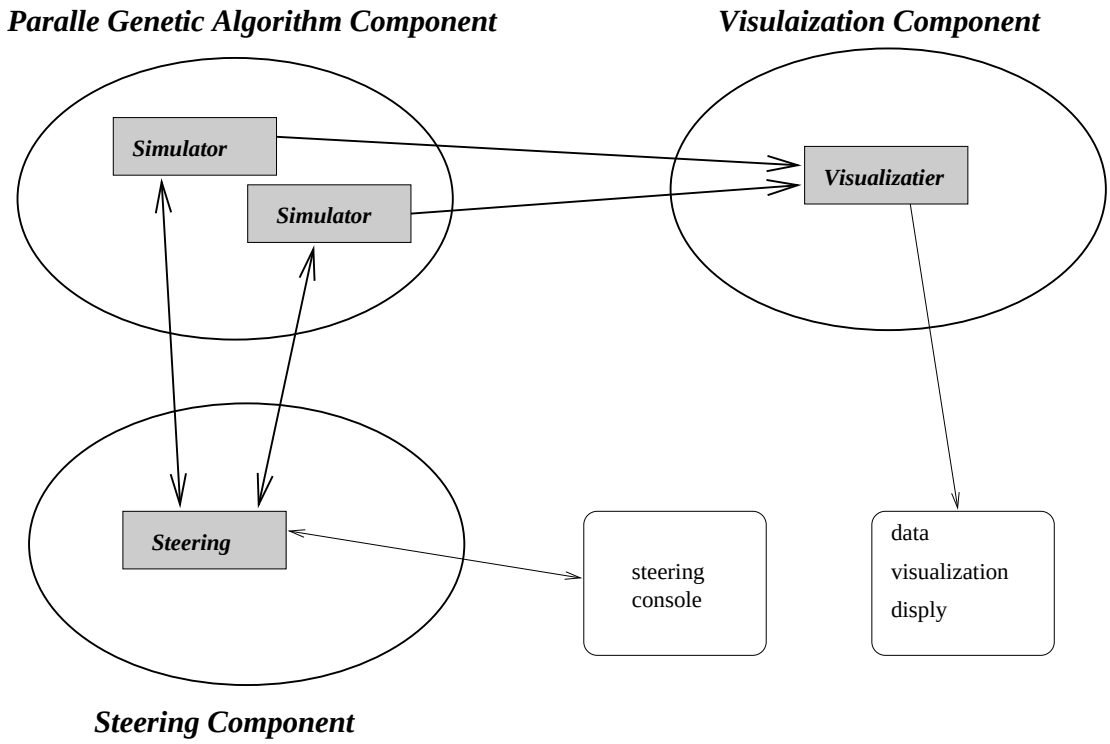

Figure 1: Problem–Environment Architecture

3.2 Data Visualization Component

This component is currently based on the visualization module of an existing sequential system [12]. It displays an on-line graphical evolution of the GA fitness function, for each island, as a function of the population iterations [21].

3.3 The Interactive Steering Component

This component directly supports a steering console where the user can ask about the status of the GA computation, and change the GA parameters for each island, as well as the parameters affecting the migration process [21]. A distributed monitoring system and a debugging tool were also used to support steering services and a resource management services for configuration of the islands and their launching in the distributed architecture. The use of a debugger as a steering tool was first proposed in [9].

4 The Life Cycle of GA–based Application Development

The environment supports the following steps in a life cycle of GA application development:

1. Problem specification using GA model
2. Configuration of the architecture of the environment by component selection (for GA, visualization and control)
3. Component activation and mapping onto the physical platform
4. Initial definition and setup of the parameters of the GA components, depending on the selected type of GA model
5. Start of the execution, with monitoring and steering
6. Control of the execution
7. Analysis of the intermediate or final results

The activities in step 1 of the above life cycle correspond to the specification of each individual (representing a solution to the given problem) using GA structures defined by the GA component.
The final 2 steps may be repeated cyclically until the desired final results are obtained. Depending on the specified modes of operation, the final results may be logged in files for post-mortem statistical processing. In steering mode, the intermediate results are displayed on-line, and the user can dynamically modify the GA parameters for the next generations. In general, the experimentation process may lead the application developer to go back to step 1 and repeat the above cycle with different approaches for problem specification for each step.

5 Conclusions and Current Status

Ongoing work focus on the application of the environment to implement a prototype parallel optimization tool in the field of environmental engineering. The project is a joint effort of the Parallel and Distributed Processing Group of the Computer Science Department and a group from the Department of Environmental Sciences and Engineering of our Faculty. As a marginal but very important effect of this project, our experimentation is producing effective prototypes that can be tested and evaluated by end–users. We hope this will provide us feedback to improve and evaluate our design and implementation options, by following the iterations in the life cycle.

Acknowledgments Thanks are due to Bruno Horta, Luis Duarte, Julio Duarte, Nuno Neves, and Gregzor Fert for their work in the development of several GA prototypes. The work reported in this paper was partially supported by the Portuguese CIENCIA, and the PRAXIS XXI Programme (SETNA Project).

References

[1] CANTÚ-PAZ, E. A survey of parallel genetic algorithms. Tech. Rep. IlliGAL 97003, Illinois Genetic Algorithms Laboratory, 1997.

[2] COHOON, J., HEGDE, S., MARTIN, W., AND RICHARDS, D. Punctuated equilibria: a parallel genetic algorithm. In *Proc 2nd Int Conf on Genetic Algorithms* (1987), pp. 148–154.

[3] DAVIS, L., GREFENSTETTE, J., AND CERYS, D. *GENESIS and OOGA: Two genetic algorithm systems*. TSP Publications, Melrose, MA, 1991.

[4] DORIGO, M., AND MANIEZZO, V. Parallel genetic algorithms: Introduction and overview of current research. In *Parallel Genetic Algorithms: theory and applications*, J. Stender, Ed. IOS Press, 1993, pp. 5–42.

[5] DUARTE, L., AND DUARTE, J. Genetic algorithms in distibuted-memory systems using PVM. Tech. rep., Departamento de Informtica, Universidade Nova de Lisboa, 1996. in portuguese.

[6] FERT, G. Genetic annealing and parallel genetic annealing. Master's thesis, University of Wroclaw / Universidade Nova de Lisboa, 1996.

[7] FILHO, J. R., ALIPPI, C., AND TRELEAVEN, P. Genetic algorithms programming environments. In *Parallel Genetic Algorithms: theory and applications*, J. Stender, Ed. IOS Press, 1993, pp. 65–83.

[8] GREFENSTETTE, J. GENESIS: A system for using genetic search procedures. In *Proc of the Int Conf on Intelligent Systems and Machines*. 1984, pp. 161–165.

[9] GU, W., EISENHAUER, G., ET AL. Falcon: on-line monitoring and steering of large-scale parallel programs. In *Proc. Frontiers'95* (1995).

[10] HOLLAND, J. *Adaptation in natural and artificial systems*. The University of Michigan Press, 1975.

[11] HORTA, B. Optimization using genetic algorithms and parallel processing. Tech. rep., Departamento de Informtica, Universidade Nova de Lisboa, 1994. in portuguese.

[12] HUNTER, A. SUGAL v2.1 user's manual. Tech. rep., University of Sunderland, July 1995.

[13] HUSBANDS, P., AND MILL, F. Simulated co-evolution as the mechanism for emergent planning and scheduling. In *Proc 4th Int Conf on Genetic Algorithms* (1991), pp. 264–270.

[14] LEVINE, D. Users guide to the PGAPack parallel genetic algorithm library. Tech. Rep. ANL-95/18, Math and Computer Science Division, Argonne National Laboratory, Jan. 1996.

[15] MANDERICK, B., AND SPIESSENS, P. Fine-grained parallel genetic algorithm. In *Proc 3rd Int Conf. on Genetic Algorithms* (1989), pp. 428–433.

[16] NEVES, N. Genetic algorithms in distibuted-memory systems using MPI-1. Tech. rep., Departamento de Informtica, Universidade Nova de Lisboa, Sept. 1997. in portuguese.

[17] PETTEY., C., LUTZE, M., AND GREFENSTETTE, J. A parallel genetic algorithm. In *Proc 2nd Int Conf on Genetic Algorithms* (1987), pp. 155–161.

[18] T. STARKWEATHER, T., ET AL. Optimization using distributed genetic algorithms. In *Proc of 1st Int Workshop on Parallel Problem Solving From Nature (PPSN)* (1990), pp. 76–186.

[19] TANESE, R. A parallel genetic algorithm for a hypercube. In *Proc 2nd Int Conf on Genetic Algorithms* (1987), pp. 177–183.

[20] TANESE, R. Distributed genetic algorithms. In *Proc 3rd Int Conf on Genetic Algorithms* (1989), pp. 434–440.

[21] VAZ, R. Computational steering of a parallel genetic algorithm. Tech. rep., Departamento de Informtica, Universidade Nova de Lisboa, Sept. 1997. in portuguese.