A Distributed Genetic Algorithm for Multivariable Fuzzy Control

Derek A. Linkens H. Okola Nyongesa

Introduction

The traditional approach to multiple parameter optimization in GA practice is to combine the coding of the parameters into a single compound bit-string; the so-called *concatenated binary mapping.* This approach has some shortcomings; the GA is a competition-based technique that has a natural tendency to evolve one winner which in complex problems yields a solution that is a better on some parameters than the others. An extension to the simple GA, called Vector Evaluated Genetic Algorithm (VEGA), has been used in multiobjective optimization where one is not interested in a single solution, but a family of optimal solutions. In VEGA each member of the population is evaluated and assigned a weighted fitness value dependent on how it relates to each objective criteria. The reproduction plan then develops groupings within the populations for each of the objectives to be optimized, ensuring that the improvement of one objective does not adversely affect the others. This, however, requires large population sizes and can be quite ineffiecient. In cases where the complex task is divisible into simpler optimization problems, a better solution set may be obtained using parallel genetic algorithms to search for the optimal solution to each sub-problem.

Multivariable Fuzzy Control problem

Our problem is to derive fuzzy control rules for multivariable processes in which there are interactions between the effects of the inputs. Considering a 2-input 2-output process, the usual fuzzy control approach will derive rules that use 4 input state variables and 2 outputs. These rules can be divided into two groupings, for each output. It can further be shown that a "complex fuzzy controller" with 2*M-inputs and N-outputs can be implemented by not more than M*N 2-input 1-output "simple fuzzy controllers". Thus, to avoid rule-bases of very large dimensions the control algorithm can be simplified by using decomposed rule-bases. In this case two simple rule-bases deal with the direct effects of the inputs, and another two deal with their interactions. The rules that deal with the interactions are however, dependent on those used for the direct actions. Learning of the fuzzy control rules is then divisible into tasks of discovering these decomposed rule-bases.

Distributed Genetic Algorithms

Parallel Genetic Algorithms have been used to deal with the need for larger population sizes, costly evaluation functions or simply to speed up the learning process. One of the most common parallelization models is to subdivide the population into isolated groups, each operating its own GA but solving the same problem. Migration of good individuals is carried out from time to time between the sub-populations. Studies have shown this form of parallelization to be in many cases superior to a GA using a single population, although this is not true in all cases [3]. Our approach to the multi-variable fuzzy control problem is a Distributed Co-operating GA (DCGA), so called to differentiate from the other types of parallel GAs. In a DCGA, different sub-populations are

---

Department of Automatic Control & Systems Engineering,
University of Sheffield,
Sheffield S1 4DU, UK.

maintained but the evaluation functions are also different from each other, since they represent decomposed partial problems. Each sub-population is only concerned with finding the best solution to its part of the composite problem, using the best available solutions from the other GAs. Thus, instead of occasional migration of individuals between the sub-populations, candidate solutions are communicated between the sub-populations at the beginning of each generation. As an example, a GA learning the rules to control the interaction between inputs will use the best rules derived for their direct actions.

Distributed Genetic Algorithms are a learning environment in which the learning agents cooperate to perform a particular task. The advantages of a distributed learning system include,

- A centrally organized system may have to deal with an overwhelming amount of data. In the 2-input 2-output multivariable system considered this can be a total of about 256 fuzzy rules, while a decoupled agent deals with a maximum of 64 rules.
- Changes in a complex system will more often be local, for example occurring in a particular control loop or region of operation. These changes are more easily identified and dealt with in a distributed system.
- A global optimum is obtainable by reaching local optima, although this is not necessarily true in all cases.

Implementation of Distributed Co-operating Genetic Algorithm (DCGA)

Parallel distributed genetic algorithms are best suited to a parallel station, although it is feasible that the concept of task distribution can be realized on a single processor serial machine. Pseudo-parallelism can, for example, be achieved on a Unix system using task threading, with *fork* commands. One processor is assigned to each learning agent, operating a GA population. Where more processors are available task forces can be formed to speed up the processing of the genetic plan. In real-time derivation of the rules, each of the learning agents can in addition operate parallel populations that partition their input spaces [2].

We have compared the distributed co-operating GA approach with the traditional GA using one population with a concatenated coding representing all the rules. Previous studies of similar problems have used this approach [1]. We also give a comparison with a series distributed approach, where the partial solutions are derived in sequence. This is, of course, only applicable where there are no two-way communication between the learning agents. The factors of performance that we are interested in include; execution times, number of trials to obtain a good solution, number of evaluations to reach convergence, the process control performance of the best solutions, population size effects. Because the structures of the GAs in the different approaches are different, it is not possible nor is it desirable to make an objective comparison based on these factors. We can, however, make some observations. Total execution time per generation, for the same population sizes, is slightly more in the DCGA than using one composite population. However, much larger population sizes and hence more generations are required using a composite GA. A learning agent which depends on other agents does not reach a final solution until those on which it depends have converged. Thus, the number of evaluations to convergence is larger than operating a single learning agent. The performance of the rules obtained in the DCGA is more robust than that of those obtained from a single population, for the same learning criteria. This can be attributed to the fact that over the learning cycle a distributed learning agent will be presented with candidate solutions from other co-operating agents, many of which are not good, and hence it learns to cope with them. Finally, the application of the genetic operators (reproduction, crossover and mutation) is critical to the DCGA. We have devised a genetic plan that delivers rapid improvement in candidate solutions, while at the same time allowing for continued experimentation with new trials. This is achieved with an elitist plan that retains all members of a population until an offspring appears that is better. In this way we can afford high rates of mutation to produce new trials without degrading the on-line performance.

The interacting environment in this study is modelled by simulation of a multivariable anaesthesia environment, that is, closed loop administration of two drugs to control muscle relaxation and the depth of unconciuosness in surgical patients. The published pharmacokinetics and pharmacodynamics of the muscle relaxant drug, atracurium, are given by a third-order transfer function with a dead time of 60 seconds and nonlinearity describable by a Hill equation. The depth of unconciuosness is controlled using a drug, isoflurane, whose effect on mean arterial pressure (MAP) can be represented by a first-order transfer function with a dead time of approximately 30 seconds.

References

[1] Karr, C.L., et al(1989); Improved fuzzy process control of spacecraft autonomous rendezvous using a genetic algorithm. SPIE Intelligent Control and Adaptive Systems Vol. 1196, pp 274-288.

[2] Linkens D.A., Nyongesa H. O. (1992); Real-time acquisition of fuzzy rules using genetic algorithms. IFAC/IFIP/IMACS Int. Symp. AIRTC-92, preprints pp599-605.

[3] Starkweather, T., et al(1991); Optimization using distributed genetic algorithms. Proc. 1st Workshop Parallel Problem Solving from Nature, PPSN 1, pp176-85.

Printed and published by the IEE, Savoy Place, London WC2R 0BL, UK.