MULTI-PROCESSOR PARALLEL GENETIC ALGORITHM IN MATLAB

Ivan Sekaj

Institute of Robotics and Cybernetics,
Faculty of Electrical Engineering and Information Technology,
Slovak University of Technology in Bratislava,
812 19 Bratislava, Ilkovičova 3, Slovak Republic

Abstract

A parallel genetic algorithm is presented which can be implemented on a multi-processor system (multi-processor machine, cluster, GRID, cloud). We have used a computer cluster with tenth of processors (processor-cores) for parallelization of the genetic algorithm. Independent algorithms, each running on a processor core and each solving the same optimization problem, are cooperating in order to obtain the solution of the defined problem. The presented project was solved in Matlab using our Genetic algorithm toolbox and the parallel computing tools of Matlab which are the Parallel computing toolbox and the Distributed computing server. Selected optimization tasks have been solved using a computer cluster. In the presented paper the structure of the parallel GA is described as well as the realization of the algorithms and the use of parallel software tools of Matlab.

1 Introduction

Genetic algorithms [1,2] belong to the group of evolutionary algorithms, which represent powerful computation approaches capable to solve complex search/optimisation problems in various practical application domains as mathematics, economy/finance, technology, construction, control engineering and others. Their algorithms mimic the biological evolution, which is in its basic a very simple but efficient optimisation process. The only difference is that while the evolution in nature needs millions of years, the evolution in computer is able to solve problems in seconds, minutes, hours, but sometimes also much more. The basis of the algorithm is the many times repeated evaluation of a group of optimised objects (of their mathematical models) and random changes of their parameters using simple operations as mutations (random change of a single individual) and crossover (random combination of features of two individuals). The more successful individuals are then selected on a probabilistic base to the next computation cycles. The better have a bigger chance to survive.

In case of solving complex problems the evaluation of a single potential solution (evaluation of a program, simulation, etc.) can take several seconds or minutes. During an algorithm run we need to use ten thousands or even millions of such evaluations. In this case the computation time can take hours or even days. A natural way to deal with such cases is parallelisation of the evolutionary algorithm [3,4,5,6,7]. In this paper we introduce a possible realisation of a parallel genetic algorithm in Matlab. The Parallel computation toolbox of Matlab is used as well as the Genetic toolbox [8]. Note that also the Global optimisation toolbox of Matlab can be used. The structure and the function of the genetic algorithm are described illustration result is presented.

2 The genetic algorithm

The genetic algorithm (GA) is the most frequently used evolutionary algorithm with a wide spectrum of applications. The algorithm is operating over a population of individuals. Population is a group (say 30 or 100) potential solutions of the problem. The block scheme of the algorithm is depicted in Fig.1. The algorithm starts with the initialisation of the population, which is normally the random generation of individuals (chromosomes). Each chromosome consists of a string of parameters (genes), which are to be found to minimise the required cost function (fitness). Next each individual of the population is to be evaluated according the fitness function and the terminating conditions are tested. Next the best individuals (at least the single best one) are copied into the new population and parent individuals are selected. These parents are mutated and crossed over. Examples of mutation and

crossover are in the upper part of the Fig.1. Finally the new population is completed from new children and selected unchanged individuals and the entire cycle is repeated. Result is the best individual of the population in the last computation cycle (in last generation).

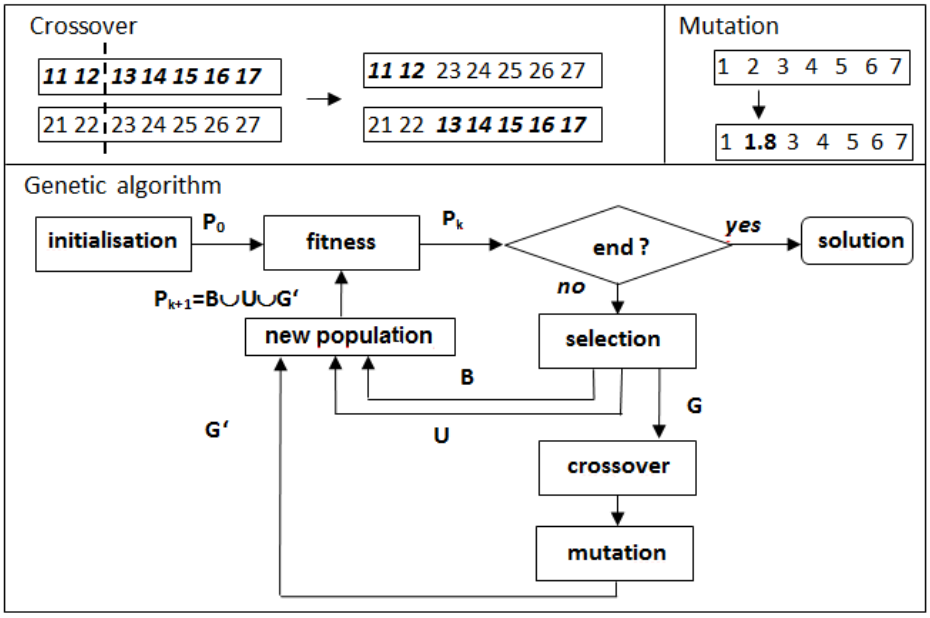

Figure 1: Genetic algorithm and its basic operations

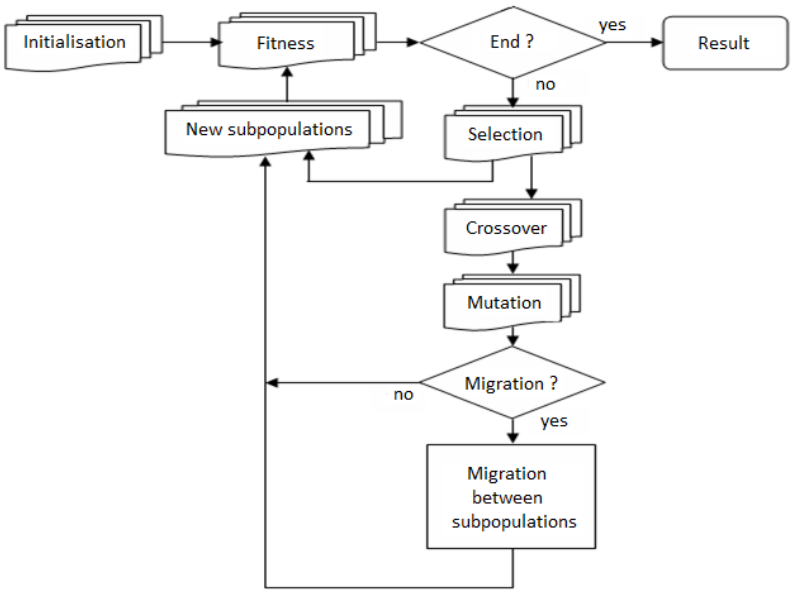

Figure 2: Possible realisation of a parallel GA

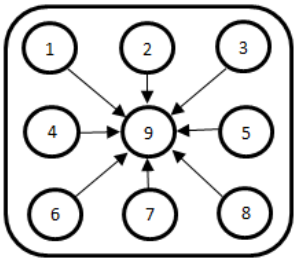

Figure 3: Migration topology of a core, circles are particular populations with own GA

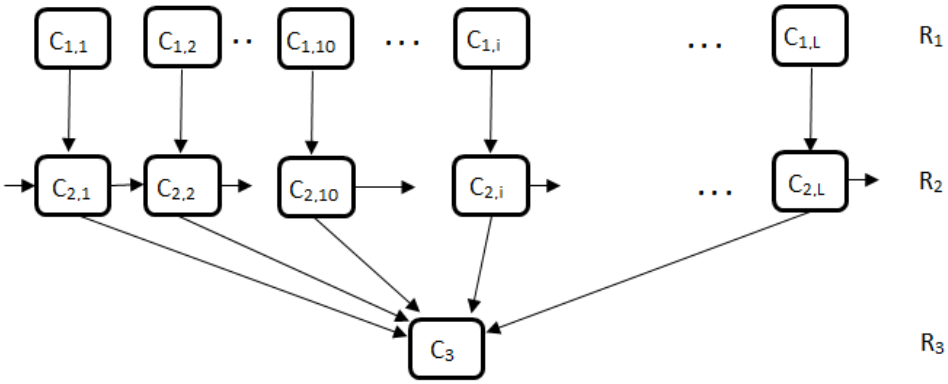

Figure 4: Architecture of the MPGA and migration topology between cores

3 Parallel genetic algorithm

The block scheme of a possible parallelisation of the GA is in Fig.2. The main idea of the parallel genetic algorithm (PGA) is that several populations are calculated contemporarily in separated algorithms (islands). Each algorithm is computed separately during defined time (predefined number of generations). After this period all separated algorithms will exchange a single individual or more individuals between the islands and the separated computation period is repeating. The exchange of the individuals is called migration. The number of islands and the topology of the migration connections between the islands depend on the application and the user experience. In our case a two level multi-processor parallel structure of the PGA has been proposed. Such structure can be used for solving very complex problems. It consists from a lower level PGA (Fig.3) and an upper level PGA (Fig.4). Let us call such two-level PGA a Multi-processor parallel GA (MPGA). Each island in the lower level PGA is marked by circle with a number and it is connected with migration links to the central island No.9. The migration represents in our case the copying of the best individual of the source island (source population) and replacing a randomly selected individual in the central island. Each computation node of the upper level PGA is called a core and it is represented by the lower level PGA. The arrows in the upper level PGA represent the migration linkage which mechanism is similar as above described for the lower level PGA. Important parameters of the MPGA are the migration periods in the lower and upper level PGAs. How to set-up their values will be discussed in part 4. The selection of the PGA or MPGA architecture and their parameters depend on the characteristics and complexity of the particular application and from the experiences of the user. More information can be found also in [7].

4 Realization of the MPGA in Matlab

The Parallel computation toolbox (PCT) of Matlab has been used for parallelization of the MPGA task into more computer cores. Additionally if processor cores of more than a single computer are required, the Distributed computation server of Matlab is needed. Note, that the processor core and the core of the MPGA structure are two different things. But the ideal case (from point of view computation speed) is if each MPGA core can be assigned to a single processor core.

The *parfor* type of cycle of the PCT has been used to parallelize the computation of each core to more processing units (processor cores). The algorithm can be described by the following Matlab pseudo-code:

```
while termination condition are not met
   parfor core=1:cores_number
            generation=0;  migration_counter=0;
            while generation<genenration_number
                  generation=generation+1;
                  migration_counter= migration_counter+1;
```

```
                calculate 1 step of GA in all islands 1,2…,9 of the actual core;
                if migration_counter==migration_period
                        migrate from 1,2,…,8 to 9;              % see Fig.3
                        migration_counter=0;
                end;
        end;
    end;
    migrate between all cores according the migration topology;     % see Fig.4
end;
```

5 Experiment

The presented MPGA algorithm has been used for solving various optimisation problems in field mathematics (multimodal test functions) as well as in control engineering applications. For controller design purposes the Simulink was used as part of the fitness function evaluation procedure [9]. In Fig.5 results of minimization of the Eggholder test function are shown. The function is defined as

$$f(X)=\sum_{i=1}^{n-1}\left(-x_i\sin\left(\sqrt{\left|x_i-(x_{i+1}+47)\right|}\right)\right)--\sum_{i=1}^{n-1}(x_{i+1}+47)\sin\left(\sqrt{\left|x_{i+1}+47+\frac{x_i}{2}\right|}\right)$$

It is a multimodal function with many local minima, and the global minimum is hard to find. In the presented experiment the Eggholder function of 10 variables is considered. The global minimum is *F(X)=-8291.2*. The MPGA used has following parameters: number of cores in row 1 or row 2 is *L=20*. Total number of cores is 41. Migration period between cores was set to 300, which is the mean time (number of generations) where the fitness function evolution process starts to stagnate. Migration period between the islands 1-8 to island 9 was set to 25 generations. This value was set experimentally. The mutation rate in the GA of each island was set to 0.1 and the crossover rate was 0.8. The graphs in Fig.5 represent the evolution of the cost function in each generation in the island 9 of the central core C_3 and in selected cores in the first row ($C_{1,1}$ and $C_{1,2}$) and in the second row ($C_{2,1}$ and $C_{2,2}$) (see Fig.4). The proposed MPGA configuration was able to find the global optimum in all repeated experiments. The result is the best individual in the central island No.9 in the last core C_3 in the last generation. Its evolution is represented by the thick solid line.

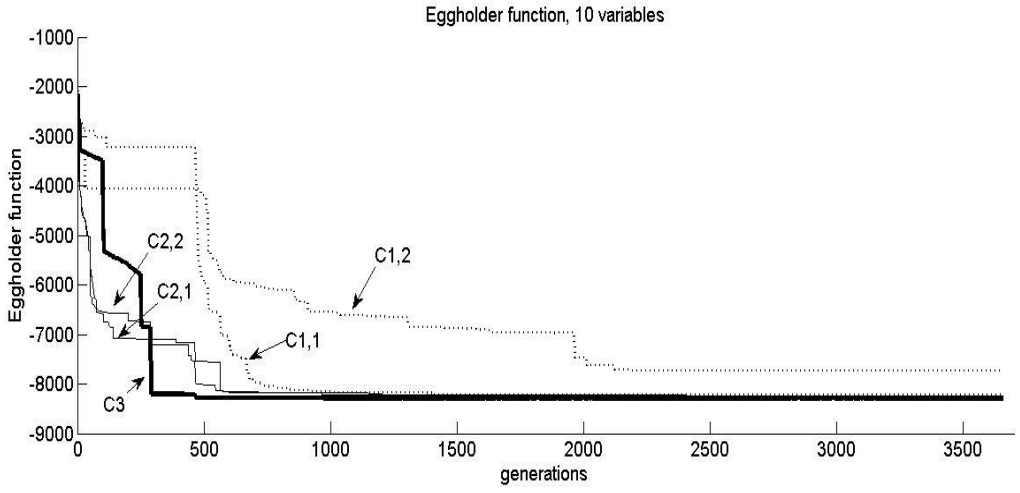

Figure 5: Evolution of the fitness function during minimisation of the Eggholder function with 10 variables

6 Conclusion

Evolutionary algorithms are powerful search/optimisation methods, which are able to solve complex practical problems with many searched parameters, problems with known or unknown internal structure in various application domains. Their main drawback is the high computation

effort/time required. This drawback can be eliminated using parallelisation of the algorithms. Parallel evolutionary algorithms can work on several computation units (multi-processor machines, clusters, GRIDs, clouds) in Matlab using the Parallel computing toolbox and the Distributed computing server. In the presented paper a possible way of parallelisation of complex task is proposed. Such architecture is able to solve tasks with a high order of complexity in acceptable time.

References

[1] A. E. Eiben, J. E. Smith. *Introduction to Evolutionary Computing*. Springer, 2003

[2] I. Sekaj. *Evolučné výpočty a ich využitie v praxi*. IRIS Bratislava, 2005 (in Slovak)

[3] I. Sekaj, M. Linder, D. Pernecký. *Experimental Comparison of Selected Types of Parallel Evolutionary Algorithms*. Springer, In ECTA 2011 and FCTA 2011: Proceedings of the International Conference on Evolutionary Computation Theory and Applications and International Conference on Fuzzy Computation Theory and Applications, Paris, France, 24-26 October, pp. 296-302, 2011

[4] E. Alba, M. Tomassini. *Parallelism and Evolutionary Algorithms*, IEEE Trans. On Evolutionary Computation, Vol. 6, NO.5, October, 2002

[5] E. Cantú-Paz. *A Summary of Research on Parallel Genetic Algorithms*. IlliGAL Report No. 95007, July 1995

[6] E. Cantú-Paz. *Migration Policies, Selection Pressure, and Parallel Evolutionary Algorithms*. IlliGAL Report, 1995 or Springer, 2001

[7] I.Sekaj. M.Oravec. *Paralelné evolučné algoritmy*. Chapter in book: V.Kvasnička a kol., Umelá inteligencia a kognitívna veda III, STU Bratislava, 2011 (in Slovak)

[8] I.Sekaj, M.Foltin. *Matlab toolbox – Genetické algoritmy*. Konference Matlab 2003, Humusoft Praha, 2003 (in Slovak)

[9] I. Sekaj. *Control algorithm design based on evolutionary algorithms*. In: Chugo, D., Yokota, S. (ed.), Introduction to Modern Robotics. iC Press, Hong Kong. iConcept Press Ltd., 2011. - ISBN 978-0980733068. - S. 251-266, https://www.iconceptpress.com/books/introduction-to-modern-robotics/11000022/

Ivan Sekaj
ivan.sekaj@stuba.sk