Journal of Convergence Information Technology
Volume 5, Number 9. November 2010

Merging Nash Equilibrium Solution with Genetic Algorithm : The Game Genetic Algorithm

[1]Massimo Orazio Spata, [2]Salvatore Rinaudo
[1,] *STMicroelectronics, massimo.spata@st.com*
[2,]*STMicroelectronics, salvatore.rinaudo@st.com*
doi:10.4156/jcit.vol5. issue9.1

Abstract

In this paper, it has been integrated Nash Equilibrium solution of Game Theory with Genetic Algorithms (GA) to optimize performance of a job scheduler, in order to simulate topology and sizing of Analog Electrical Circuits simulation. We proposed a new method for performance problems solving of Genetic Algorithms applied to Electronic Design Automation (EDA) simulator tool optimization. This optimal solution process is formulated as a non-cooperative Game in order to solve GA performance problem more efficiently and effectively. For these reasons, it has been created a new integrated Algorithm named Game Genetic Algorithm (GGA). A flow chart of the algorithm is presented to investigate the feasibility of the above approach.

Keywords*: Genetic Algorithm, Game Theory, Scheduling, Parallel Computing*

1. Introduction

In many fields of research, simulators have become a fundamental instrument to determine the effects on the output of a determined model by varying input parameters. Therefore, the reliability of a simulator depends on the accuracy of the model parameters.

For example, in the event of a simulation of a microelectronics device, in order to model its behavior the demands parameters are connected to physical phenomenon like mobility, recombination and so on. To optimize the characteristics of bipolar transistors, and particularly, when describing an electrical device through an equivalent circuit, the parameters are connected with the electrical members into the circuit [1]. So in industrial applications, it would be opportune to have a global algorithm optimizer, like for example the GGA algorithm which decreases the global job simulation completion time but increases the total job throughput. This paper presents an implementation of a Genetic Algorithm integrated with Nash Equilibrium solution of Game Theory. Through this implementation we have tried to optimize the overhead (that is ratio between GGA scheduling time and total completion time) of a generic Genetic Algorithm, applying it to scheduling problems of simulation for EDA tools.

2. Strategic Game Form and Nash Equilibrium solution

In [2, 3, 10] a strategic game form *G*, with two players is *(X,Y,E,h)* where:

- *X*,*Y*,*E* are sets
- *h* is a function defined in $X \times Y$ with value in *E*
- *X* represents available choices for player 1; idem for *Y* respect to player 2.
- *E* represents a set of possible strategies for the game
- *h* is an output function providing results achieved on players' choices. So, if player 1 chooses *x* and player 2 chooses *y*, get *h(x,y)* results
- players are mobile agents [10]

In a game *G*, you need to know the players' preferences for the different elements of *E*. A fast and simple mode to describe these preferences is to use "utility functions" *u(x)* [5, 6]. For example, for

player 1 assume it is given a function u defined in E and with values in $\mathbb{R}$, interpreting $u(e') \geq u(e'')$ as an expression of: player 1 prefers outcome e' to outcome e'' [4, 5, 6].

So, having a game form (X,Y,E,h) and two utility functions (for both players) (w,v), this is the expression form of the game:

$$(X,Y,E,h,w,v)$$

Through composition operator $\circ$, let be $f_1 = w \circ h$ and $f_2 = v \circ h$, so we will obtain (X,Y,f_1,f_2) or a strategic game form for two players (1 and 2), as defined in [3].

Where:

- $f_1, f_2 : X \times Y \rightarrow \mathbb{R}$

A Nash Equilibrium for $G=(X,Y,f_1,f_2)$ is $(x^*,y^*) \in X \times Y$ that:

- $f_1(x^*,y^*) \geq f_1(x,y^*) \quad \forall x \in X$
- $f_2(x^*,y^*) \geq f_2(x^*,y) \quad \forall y \in Y$ [3, 10]

3. Genetic Algorithm methodology

The natural selection genetic rules provide their evolution direction to a population of biological organisms in order to adapt its characteristics to the environment. The best individuals are selected through environment adaptation characteristics, since they have better probability to survive and to reproduce. This selection grows from generation to generation allowing increasing average adaptation of the population.

From a computational point of view, to solve difficult optimization problems, Genetic Algorithms apply the above paradigm. Each possible solution of a problem is represented as an individual (in our case an agent), whose strength is the quality of the solution. Starting from a population of individuals or solutions, the evolution mechanisms is applied cyclically to find better solutions to the problem.

We focus our attention on the design of this algorithm providing the pseudo code. The fundamental operation to start the algorithm is generating the initial population. The population consists of a set of entities, each of which corresponds to a possible solution to the problem. The population represents the whole space search algorithm, which can be generated in several ways. In general, we can take the random generation path, which is supplied to the algorithm by generating the maximum size of the population, therefore corresponds to the number of entities that the population should contain. In this approach, work mainly on bit set objects: this means that algorithm generates random binary strings and submits them for processing. Once the algorithm has completed its work, the entity selected as the best solution is converted into an object model to interpret the found solution. A second method for the generation of the population could be established from a dataset. In this approach, the algorithm can provide a source of data that corresponds to the model objects defined above, and it will be to verify the accuracy of the dataset and convert the same into binary strings. It is possible to define different algorithms for different types of Genetic Selection allowed by the GA algorithm. The solution we propose is the selection according to the Nash Equilibrium of the Game Theory.

4. Integration of the Game Theory and Genetic Algorithm: the Nash Selection

The idea is to bring together Genetic Algorithms and Nash strategy in order to make the Genetic Algorithm build the Nash Equilibrium as in [7, 8]. We are going to show how such merging can be achieved with n players trying to optimize n different objectives.

Initial individuals population P are selected according to the criterion of Nash Equilibrium previously described applied to the fitness function. For example, the value of fitness of all players will be placed on a matrix $n \cdot n$ where it is calculated by the Nash Equilibrium. Then the best chromosomes are copied into the new population. This method may grow quickly because the performance of genetic algorithms' filters the best solutions. Below is the pseudo code algorithm:

```
procedure NashSelection(P)
begin
   for all n player agents of P(t) do
      f_k = fitness value of player agent k
   find Nash Equilibrium between the f_1 ... f_n value
  of player agents
  all agents' point of Equilibrium are selected
end
```

Let's see an example, with $n = 2$ players. Let X strategies set for player 1 and Y strategies set for player 2. X corresponds to the subset of variables handled by Player 1, and optimized along f_1. Y corresponds to the subset of variables handled by Player 2 and optimized along f_2. In this case, Nash Equilibrium is the solution for a dual objective optimization problem. Thus, following Nash Theory rules, Player 1 chooses with respect to the first criterion f_1 by modifying X, while Y is fixed by Player 2. Symmetrically, Player 2 chooses with respect to f_2 modifying Y while X is fixed by Player 1 [3, 10].

The next step consists in creating a new population P for both players. Player 1's optimization task is performed by fitness function f_1 on population 1 (P_1), whereas Player 2's optimization task is performed by fitness function f_2 on population 2 (P_2).

Nash equilibrium is reached when neither Player 1 nor Player 2 can further improve their criteria, as described in [3, 8, 10, 16] for the Prisoners' Dilemma problem.

5. The fitness function and the objective function for scheduling problems

Suppose that m machine M_i $(i = 1,\dots, m)$, should schedule n jobs J_j $(j = 1,\dots, n)$, with $m < n$, for every jobs J will be generated an agent as described in [10, 16]. For every job a job allocation of time periods to one or more machine, is commonly named *schedule*. A schedule is allowed if, for every time t, a job is scheduled exactly by a machine. A schedule is optimal if minimizes (or maximizes) a given criteria: the objective function.

Cost f_j for every job J_j, often is calculated considering one of the above variables, or the product between weight w_j and *the completion time* C_j, which is the job range of time required to complete the job J_j. The objective function can be any cost function f_j, $\forall j = 1,\dots, n$. Common objective functions are typically expressed in the form $\Sigma_j f_j$, or $max_j f_j$.

The common job scheduling criteria objective functions are:

- *total weighted completion time* $\Sigma_j w_j C_j$,
- *max job completion time or makespan* $C_{max} = max_j C_j$
- *max lateness* $L_{max} = max_j L_j$

For the implemented model described in this paper, as objective function, we used the *max job completion time or makespan*.

The goal of the presented scheduling algorithm is to assign each job J to a right machine M. In other words, its goal is to get a sequence of the jobs over the machines, minimizing the makespan i.e. the total completion time of all jobs.

In order to solve this problem we should consider these constraints:

- job preemption is not allowed
- every machine m can process only one job J at a time.
- all the jobs can be available at time $t=0$.
- the transportation time is ignored.

6. The GGA algorithm

This is the applied GGA scheduling algorithm customization through the Nash Equilibrium solution [11, 12, 13]:

```
procedure GGA
begin
    t ← 0
    i ← 0
   initialize a chromosome population P_i(t)
    while do
       evaluate(P_i(t)) fitness function f_i
       NashSelection(P_i(t)) agents and insert in P_i
       if(max_i f_i on P_i) then
           break
       else
           select P_i(t) agents and insert in P
           apply Crossover to P making P_i+1
        apply Mutation to P_i+1 agents making P_i+2
        t ← t+1
        create P_i(t) selecting for replacement agents
           from P_i+2 and P_i(t-1)
        i ← i+1
end
```

The *crossover* genetic function will be applied to the selected individual population, after the reproduction. Here, the Precedence Preserving Order-based crossover (POX) strategy is adopted, as described in [17]. The applied *mutation* genetic function (which will be used after the crossover function) applies the Precedence Preserving Shift mutation (PPS) strategy, as described in [18].

The flow chart in Figure 1 shows a description of the GGA algorithm:

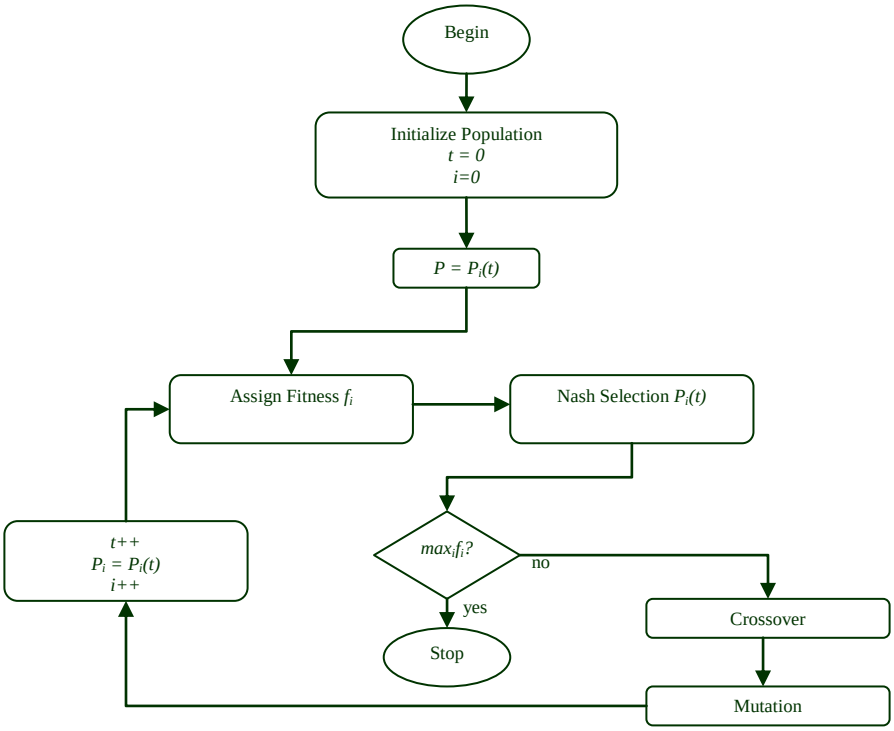

Figure 1. A working principle flow chart of the GGA

7. Experimental results

Experimental tests, has been executed with the "mprun" Eldo [1] command simulator and the monitoring of these jobs demonstrates the improvement of makespan values between a generic Genetic Algorithm without Nash Selection, and the same algorithm with Nash Selection integration, named Genetic Game Algorithm. As an initial setup for simulator we chose 20 compute server, 100 Monte Carlo jobs and 1 Eldo input file.

Each run is an electrical simulation of an operational amplifier that measures the gain bandwidth product at 100 KHz, the phase margin, the positive and negative slew-rate, varying the compensation capacitor of the amplifier model and the load capacitor in agreement with Gaussian distributions; for each run its simulation time was measured.

The initial load conditions of compute server are identical for both experiments with the two different algorithms (GA and GGA).

Clusters used for the experimentation tests make use of three types of resources Linux 32 bit, Linux 64 bit, and Unix Sun Solaris. Their characteristics can be summarized as follows:

- Class Type Linux 64 bit: bi-processors Servers AMD Opteron 64 bit, Red Hat Enterprise 4.0 Update 8 OS and 16 GB of RAM
- Class Type Linux 32 bit: bi-processors Servers Xeon 32 bit, Linux Red Hat Enterprise 3.0 Update 5 OS and 4 GB of RAM
- Class Type Sun Solaris: bi-processors Servers V240 SparcIIIi, with Solaris 8 OS and 8 GB of RAM.

The Local Resource Management System installed and used for experimentation is LSF (a Load Balancing engine produced by Platform). Every Worker Node pertaining to a queue, and it has one slot per processor (job container for only one job) to deny co-allocation of two jobs on the same CPU per Worker Node.

Jobs completion time, average queue length, and average waiting time in queue has been measured through a monitoring system with a web portal interface developed in java and php language.

The Figure 2 shows differences between a generic Genetic Algorithm and the Genetic Game Algorithm jobs makespan:

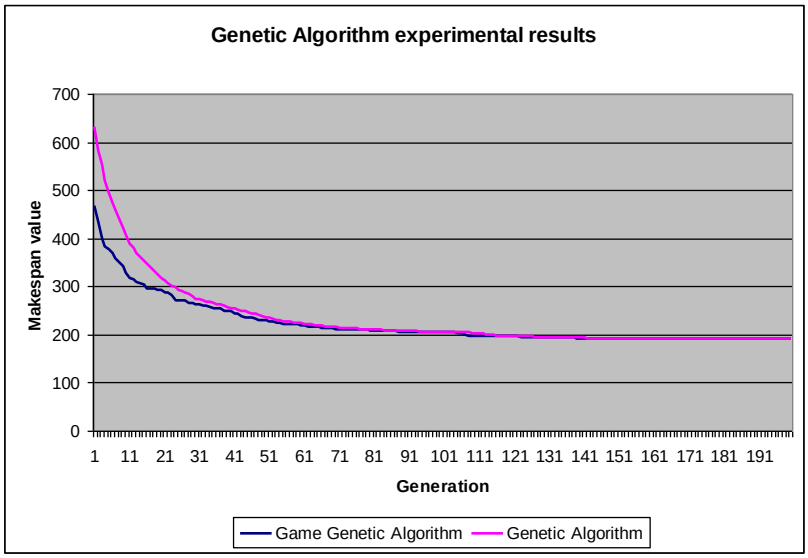

Figure 2. Experimental results regarding the makespan improvement of the GGA algorithm for a Monte Carlo simulation

8. Conclusions and future work

In this paper, we have developed an improved Genetic Algorithm for job scheduling optimization problem. From the experimental point of view, we have found that our improved GGA algorithm is better than the old Genetic Algorithm in terms of overhead. With our algorithm, we can get better result, especially in the initial populations, but the overhead of the scheduler is not yet so negligible for scheduling problems.

Possible future applications of the presented algorithm are mobile phone or distributed systems. Paying particular attention to resource availability discovery and monitoring.

9. Acknowledgements

The author wishes to thank his company, STMicroelectronics, for this research work opportunity.

10. References

[1] Mentor Graphics Corp. *Eldo simulator web page*. http://www.mentor.com/products/ic_nanometer_design/custom_design_simulation/eldo/index.cfm
[2] J. Nash Equilibrium points in n-person games.. *Proceedings of the National Academy of Sciences*, 36, 1950.
[3] F. Patrone (Author) “Decisori razionali interagenti” Una introduzione alla Teoria dei Giochi - Edizioni PLUS, Pisa, 2006 ISBN: 88-8492-350-6
[4] Coley, D. A. , “ An Introduction To Genetic Algorithms For Scientists And Engineers ” , World Scientific publishing Co. Pte. Ltd, 1999.
[5] Goldberg, D.E.. Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley, 1989.
[6] I.A.Ismail , N.A.El_Ramly , M.M.El_Kafrawy , and M.M.Nasef Game Theory Using Genetic Algorithms, Proceedings of the World Congress on Engineering 2007 Vol I WCE 2007, July 2 - 4, 2007, London, U.K.
[7] Sefrioui, M. (1998). Algorithmes Evolutionnaires pour le calcul scientijque. Application I 'electrvmagnetisme et la mecanique des fluides numriques. PhD thesis, Universite Pierre et Marie Curie, Paris.
[8] Sefrioui, M.; Perlaux, J.; , "Nash genetic algorithms: examples and applications," Evolutionary Computation, 2000. Proceedings of the 2000 Congress on , vol.1, no., pp.509-516 vol.1, 2000 doi: 10.1109/CEC.2000.870339
[9] Conca, P.; Nicosia, G.; Stracquadanio, G.; Timmis, J.; , "Nominal-Yield-Area Tradeoff in Automatic Synthesis of Analog Circuits: A Genetic Programming Approach Using Immune-Inspired Operators," Adaptive Hardware and Systems, 2009. AHS 2009. NASA/ESA Conference on , vol., no., pp.399-406, July 29 2009-Aug. 1 2009 doi: 10.1109/AHS.2009.32
[10] Spata, M.O.; Rinaudo, S.; "Agents Based Intelligent Grid Matchmaking System for EDA Tools on a Cluster Grid," Complexity in Engineering, 2010. COMPENG '10. , vol., no., pp.124-128, 22-24 Feb. 2010
[11] D.B. Fogel, "Evolutionary computation: toward a new philosophy of machine intelligence", IEEE Press, New York, 1995 ISBN 078035379X
[12] J.R Koza, "Genetic programming: on the programming of computers by means of natural selection", MIT Press, Cambridge, MA, 1992 ISBN 0262111705
[13] Alden H. Wright, "Genetic Algorithms for Real Parameter Optimization", 1998
[14] Allahverdi, A., & Mittenthal, J. (1994). Scheduling on M parallel machines subject to random breakdowns to minimize expected mean flow time. Naval Research Logistics, 41(5), 677-682.
[15] Graham, R., and E. Lawler, E. Lenstra, A. Rinnooy Kan, Optimization and approximation in deterministic sequencing and scheduling: a survey, Annals of Discrete Mathematics, 5 (1979) 287-326.

[16] Massimo Orazio Spata, Salvatore Rinaudo , "A scheduling Algorithm based on Potential Game for a Cluster Grid ", JCIT: Journal of Convergence Information Technology, Vol. 4, No. 3, pp. 34 ~ 37, 2009.
[17] F. Pezzella, G. Morganti, G. Ciaschetti, “A genetic algorithm for the Flexible Job-shop Scheduling Problem”, Computers & Operations Research, 2008, 35(2008). 3202-3212.
[18] Lee KM, Yamakawa T, Lee KM, “A genetic algorithm for general machine scheduling problems”, International Journal of Knowledge-Based Electronic, 1998, 2. 60–6.