A Parallel Genetic Algorithm to Coevolution of the Strategic Evolutionary Parameters

Eduardo Noronha\(^1\), and Celso G. Camilo-Junior\(^1\)

\(^1\)Informatics Institute, Federal University of Goias, Goiania, Goiás, Brazil

Abstract—The strategic choice of parameters (crossover rate, mutation rate, population size, number of generations, among others) in an AG has a direct impact on the success of evolutionary search. Therefore, the definition of good values in the parameters of evolution can optimize the search process population, finding better solutions in shorter times. Thus, a process of self-adaptation constitutes an implicit search with strategic parameters. Considering the characteristics of evolutionary models and their inherent parallelism, this paper presents a coevolutionary algorithm developed in parallel MPI to evolve the parameters of a Multidimensional Knapsack Problem from the evolving capacities of individuals.

Keywords: Genetic Algorithms, Evolutionary Algorithms, Evolutionary Parameters, Coevolution, Knapsack Problem

1. Introduction

The Genetic Algorithms (GA) are intelligent models inspired by biological evolution through adaptive methods; they can find potential solutions without thoroughly considering all possible solutions to the problem.

The simulation of evolution is guided through the instrument changes, called strategic parameters. Many studies have presented research on the choice of these parameters (crossover rate, mutation rate, number of generations, population size) responsible for controlling the evolutionary process of search.

The focus of this work is to find good values for these parameters, through the development of a Parallel Genetic Algorithm (PGA) coevolutionary.

Several meta-heuristics are used: Evolutionary Strategies \([1]\), Genetic Fuzzy Systems \([2]\), Evolutionary Programming and Genetic Programming \([3]\). According to Eiben \([4]\), there are basically two ways to choose the parameters in an evolution: by adjusting the parameters, where they are constant from start to finish the process, or by controlling the parameters. In this case, can be deterministic, adaptive and self-adaptive (where the parameters are themselves subjected to an evolutionary algorithm).

The possibility of multiple processors working together to solve a computational problem allows a differentiated approach to impacts from conception and design of algorithms, from the computing platform, to test the mechanisms employed.

The central point of a parallel system is the division of tasks in a single application that can run simultaneously. There are some computer models that allow for parallel computation, and are divided primarily by the presence or not of a synchronization between the processors and the way of communication between them. Parallel computing models were considered in this context primarily by the characteristic of the GA’s inherently parallel, allowing multiple combinations of parameters were tested in numerous processors simultaneously. For this, was used the MPI environment for simulating coevolution of strategic parameters, with possible extension of the simulations in CUDA.

In relation to the context of application of the model was named the Knapsack Problem. The motivation for choosing this problem is that, besides being a classic combinatorial problem NP-Hard class, represents numerous possibilities of application in real problems. There are different variations of the knapsack problem (KP 0-1, where each item can be inserted into the knapsack only once, KP unlimited, where each item can be inserted more than once, among others). In this work the variation was chosen Multidimensional Knapsack Problem (MKP).

Considering the behavior of self-adaptive evolutionary systems with their inherent parallelism and potential available in parallel computing, the aim of this paper is to present a coevolutionary algorithm developed in parallel MPI is able to evolve the strategic parameters (crossover rate, mutation rate, population size, number of generation) to a MKP. Some instances of problems in the literature \([5]\) were tested and compared.

2. Literature Review

The self-adaptation was originally introduced by Rechenberg \([6]\) and Schwefel \([1]\) to Evolutionary Strategy, and later by Fogel \([7]\) Evolutionary Programming.

The idea that the parameters of an algorithm can also be submitted to the evolutionary process was approached in 1967 by Bagley \([8]\). Various methods and techniques of evolutionary self-adaptation are explored, among which the most common for this type of evolution is the Evolution Strategy \([9]\), \([10]\), \([11]\), \([6]\), \([12]\), \([13]\), also evolutionary programming, but since the 90’s, several studies have been performed Back \([14]\) and Smith \([15]\) showing the capability of doing you with Genetic Algorithms (GA).
In 1970, Weinberg [16] proposed a genetic algorithm external (meta-algorithm) control, responsible for choosing the best strategic parameters for GA slaves. To the master algorithm, each individual in the population represents a possible configuration parameters. To evaluate each individual, the algorithm performs a meta-slave algorithm with that set of parameters, and checks the quality of results. At the end of execution, the individual winner represents the most appropriate configuration parameters. This algorithm has a practical approach Koch [17] for application to problems in the industry.

In his thesis, Potter [18] describes a computational model of cooperative coevolution including the notion of modularity necessary to provide reasonable solutions and evolve interacting coadapted subcomponents. In this approach, subcomponents are represented as species genetically isolated and evolve in parallel. Individuals of each species entering temporarily in collaboration with other members of other species and are rewarded based on the success of the collaboration, in other words, the fitness of the objective function.

In the work of Aguirre et al. [22] is presented a parallel model of cooperation / competition for genetic operators to develop self-adaptation within the AG’s, showing that parallel self-adaptive mutations have better outcomes than serial self-adaptive mutations. The parallelism in this case, is restricted to the application of mutation operators.

In Bartz et. al [19] was developed a method of setting parameters for the output of the algorithm stochastically perturbed Particle Swarm Optimization (PSO) has been applied successfully in many applications. Preuss et. al [20] uses PSO to optimize self-adaptive evolutionary algorithms for binary code, where it is combined classical regression methods and statistical approaches.

A covariance matrix for self-adaptation in evolutionary strategies (CMSA-ES) is presented by Beyer and Sendhoff [21] as an example of successful self-adaptation of real numbers in the solution space. The correlation matrix of mutation sets the rotation axis and scale of change, allowing the adaptation of the characteristics of local fitness. Several operators are worked in the matrix as: mutation, crossover and selection.

3. Proposed algorithm

The main difference in the proposed algorithm presented in relation to the work of literature is the realization of the strategic parameters of the coevolution of MKP Algorithm, using the paradigm of parallel computing. The coevolutionary developed evolutionary process can be decomposed into two subprocesses: master process and slave process.

The slave process is a genetic algorithm that seeks to evolve individuals seeking to find a solution to a particular instance of an MKP. Was chosen to represent the binary representation of individuals. The master process has individuals with 4-position vector representation of real values, as shown in Table 1, with the following meanings respectively: number of generations, population size, crossover rate, mutation rate. The objective of this evolutionary process is to find individuals who return better fitness values when applied to an AG resolver an MKP (runs on master process).

Therefore, the basic purpose of the coevolutionary algorithm developed consists in the evolution of the parameters of the strategy due to changes in attributes of the data structure of each individual. Data structures correspond to individuals (candidate points in the search space). Figure 1 provides a simplified view of the AGP coevolutionary developed.

<table>
<thead>
<tr>
<th>Crossover Rate</th>
<th>Mutation Rate</th>
<th>Population Size</th>
<th>Generations</th>
</tr>
</thead>
</table>

Table 1: Representation of individuals in the Master Process

In generating the initial population of the master process the following parameters were used: population corresponds to number of processors available slaves, 250 generations, crossover rate 0.85 and mutation rate 0.01. The evaluation is based on the objective function of the master process, which consists in the fitness returned by the slave process execution minus a penalty factor, as shown in Figure 2. Each slave process receives as input an individual master processor with parameters that will be used to generate its initial population, and after the evolution of the slave process, the master processor receives a fitness value, called fitness master, one for each of individuals. Therefore, the system can be seen as two evolutionary processes interconnected and fed back,
featuring a coevolutionary system.

![Flowchart of Genetic Algorithm MKP](image)

**Fig. 2: Flow of the Genetic Algorithm MKP of the process slave.**

The stopping criterion established for the AG of the slave process is the number of generations. The desired end result of coevolution is to find individuals with the best fitness values associated master.

Two experiments the composition of the objective function were used in the master processor to perform the co-evolution. The first was based primarily on two components [22]: effectiveness and efficiency of processors to solve an MKP. The effectiveness is represented by the evolving capacities of a slave processor and translated by the slave processor performance evolution, which is the difference of fitness value in the first and last generation. This choice was used to replace the fitness value found by the slave processor, since a combination of strategic parameters with small populations and generations are able to find high values of fitness due to the randomness of the generation of initial population and not for their ability evolutionary. Since efficiency is represented here by the number of searches carried out in solution space, which can be simplified by the product of population size and the number of generations. Thus, one can say that a chromosome of the master processor has better performance than another, as it finds the best values of the objective function.

In another composition of the objective function of the master processor, the processor with the configuration parameters provided by the master processor, perform the evolution for the MKP and return to the master found the best fitness and the amount of search space solution that was used for this gym. With this information the master processor performs calculates the fitness of coevolution by applying a penalty on the fitness of the slave processors directly proportional to the number of searches performed.

The value chosen for the penalty for the tests was 20% of the best results found in the literature.

In relation to the master process, the crossover operator used was the arithmetic crossover [23], which is a strategy n-parent recombination, using the weighted average of the parents. Because it is a real representation of individuals, the mutation of genes is accomplished by choosing a random value for replacement of alleles, considering the probability of mutation.

The next session presents the results of the simulations.

### 4. Analysis of Results

The model proposed in this paper presents a coevolutionary genetic algorithm exploiting the parallelism available in the model of distributed memory parallel computing, implemented in language C and MPI.

The experiments were performed in a staggered platform for distributed memory (MPI). It explored the instances 1 and 2 of the OR-library [5]. The experiments run on a cluster of 7 machines, with the following settings: AMD Phenom 9600 Quad-Core, 4 Gb of RAM, available hard drive partition of 160 Gb.

Experiments were performed with two objective functions in the master processor. The first, according to the evolutionary capacity and efficiency, and the second considering the value of fitness with application of the penalty, based on the number of searches.

After the evolutionary process, two situations can be verified. The first, in which the parameters found in the final representation of the best individuals in the master process is achieved by the very nature of the convergence process. In the second possibility, there are cases in which the best individual has values in their genes (crossover rate, mutation rate, number of generations and population size) forged in the evolutionary process effective. This conclusion is taken from the moment that individuals evolved with a low amount of searches as simulated by 10 times, did not return good slave fitness, reinforcing the randomness as the main influence.

Thus, the analysis of results is performed disregarding the results of searches with a small amount (less than 25%), and focusing on data that showed potentially interesting in this context, ie, search over 75%. Simulations were performed with up to 20 slave processors, the master process generating random values between 0 and 200 for the parameters: population size and number of generations. This configuration allows up to 40,000 a search solution solutions space.

The analysis undertaken aimed at identifying the best settings of the input parameters for the resolution of an MKP through a genetic algorithm. For this, the outputs of AGP were subjected to a data mining (using the tool WEKA) seeking to identify characteristics that allow correlation of evolutionary computing specialists, a combination of the
most suitable configuration of the strategic parameters of the GA to solve an MKP.

Applying the mining algorithm selection of attributes, it was found that the parameters with greater influence and a strong correlation in the fitness master are: population size and mutation rate. The level of influence of these parameters can be seen even more clearly in the cluster analysis performed on clusters by applying the algorithm based on centroids, Kmeans on data obtained from simulations with the first objective function, as shown in Table 2.

Table 2: Kmeans Algorithm applied in the strategic parameters

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Cluster 0</th>
<th>Cluster 1</th>
<th>Cluster 2</th>
<th>Cluster 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Elements</td>
<td>45</td>
<td>35</td>
<td>168</td>
<td>14</td>
</tr>
<tr>
<td>Generations</td>
<td>56.93</td>
<td>42.48</td>
<td>57.42</td>
<td>78.28</td>
</tr>
<tr>
<td>Population Size</td>
<td>17.86</td>
<td>26.28</td>
<td>44.52</td>
<td>16.21</td>
</tr>
<tr>
<td>Crossover Rate</td>
<td>0.2356</td>
<td>0.5911</td>
<td>0.4702</td>
<td>0.7636</td>
</tr>
<tr>
<td>Mutation Rate</td>
<td>0.3967</td>
<td>0.7794</td>
<td>0.4841</td>
<td>0.1857</td>
</tr>
<tr>
<td>Fitness Master</td>
<td>33222</td>
<td>27756</td>
<td>1033</td>
<td>34972</td>
</tr>
</tbody>
</table>

The analysis presents data in five groups with similar characteristics (clusters) corresponding to 23% 13% 25% 6% and 33% of the data. It is observed that the clusters 0 and 3, with the lowest values for mutation rates, however, the highest fitness, indicating an inverse correlation. The same is true for the population size as observed in the same clusters. These clusters are those with the best fitness values. The multiple correlation of the parameters was also analyzed showing that the population size influences the rate of mutation, whereas the reverse is not true.

In an analysis with only two clusters (Table 3), it clearly appears that the cluster 0 performs a greater number of searches in solution space (exploring party), generating a greater computational cost, while in cluster 1, is characterized by individuals who carry a smaller number of searches and fitness values are much higher than in cluster 0. This phenomenon can be explained by the randomness of the initial population that can find some guy with a good fitness value associated, but not necessarily the individuals in this cluster a good performance evolution.

Table 3: Analysis of strategic parameters with two clusters.

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Cluster 0</th>
<th>Cluster 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Search</td>
<td>6866</td>
<td>889</td>
</tr>
<tr>
<td>Crossover Rate</td>
<td>0.434</td>
<td>0.477</td>
</tr>
<tr>
<td>Mutation Rate</td>
<td>0.519</td>
<td>0.4088</td>
</tr>
<tr>
<td>Fitness Master</td>
<td>52</td>
<td>32400</td>
</tr>
</tbody>
</table>

In numerical terms, it is concluded that a large population (number of slave processors) master process would be necessary to ensure an evolutionary process more accurately. The values close to 0.5 in both clusters for both crossover rate and mutation rate to show a lack of greater number of individuals in the population, especially in the mutation rate, these values indicate the need for high achievement of exploitation in search of diversification of candidate solutions. In this case, the use of a larger number of processors, such as the GPU, seems an interesting alternative.

Considering the results with few (less than 10) search (population size versus number of generations), we applied the classification algorithm M5P to obtain the correlation coefficient and the relationship of inference to the value of fitness in terms of strategic parameters 4.

Table 4: Sorting results of the simulations

<table>
<thead>
<tr>
<th>Number of Searches</th>
<th>Search Space (%)</th>
<th>Classification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Between 110 and 1055</td>
<td>Between 0.25% and 2.6%</td>
<td>High (86.87%)</td>
</tr>
<tr>
<td>Between 1055 and 7044</td>
<td>Between 2.6 % and 17%</td>
<td>High (63.30%)</td>
</tr>
<tr>
<td>Up to 7044</td>
<td>Up tp 17%</td>
<td>Low (38%)</td>
</tr>
</tbody>
</table>

To a high number of search (> 7444) the linear model does not return good ratings. The first track, one can observe that the independent term has a high value, noting that the minimum value that the master can take fitness is also high (around 13000). It appears that this is a good range to obtain high values of fitness, since the classification is also of considerable value (86%).

In classification using the algorithm REPTree (Table 2) with good correlation coefficient (0.84), the variable search is the factor that most influences the determination of fitness. The range between 115.5 and 639.5 is the only one that another factor (in this case, crossover) is relevant. Interestingly, the highest value for fitness was obtained for the highest values of crossover in the middle range of searches that match the range of the previous algorithm for best results. Another issue to consider is that both for this algorithm as for the other search values represented are well below average (about 5000) showing that a process where you can search many an algorithm is difficult to obtain good results. This was to be expected because of the randomness of genetic algorithms.

Seeking to penalize high fitness values in processors with low amount of searches, we used the objective function with penalty on the number of searches. Whereas the maximum amount possible is 40.000 search, we have assumed the penalty in the amount of 500 (12.5%) for the maximum amount. Thus, if a processor performs 2000 searches will be penalized 25 in the amount of your fitness.

In Table 5 shows the clustering applied in simulations using penalty function. Were used the instance 1 of [5], choosing only the simulations with fitness values greater than 20.500.

It is observed that there is a relationship already expected
in the second cluster occurs where the least amount of searches, the mutation rate is high, while the crossover rate is very low. In the first cluster, which has a greater amount of searches, the mutation rate is higher, however, the crossover rate also accompanies growth, signaling a behavior in these settings exploration and exploitation. In the first cluster, which represents most of the data, we can see a scenario with a larger amount of searches and crossover rate, however, a smaller value for crossover rate, showing how much more searches to lower the rate of mutation.

Simulations with the same amount of search present values for crossover and mutation as well different. This can be explained because in smaller populations, it is necessary to have higher values of mutation, while larger populations, the mutation rate tends to be lower. Within this context the data is separated into two: populations greater than 100 and less than 100, seeking to identify some patterns. Table 6 presents all clusters with high mutation rates and similar rates of crossover.

Table 5: Clusters for coevolution with search fee.

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Cluster 0</th>
<th>Cluster 1</th>
<th>Cluster 2</th>
<th>Cluster 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Elements</td>
<td>5</td>
<td>13</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>Search</td>
<td>13932</td>
<td>2800</td>
<td>1648</td>
<td></td>
</tr>
<tr>
<td>Crossover Rate</td>
<td>0.394</td>
<td>0.7523</td>
<td>0.1464</td>
<td></td>
</tr>
<tr>
<td>Mutation Rate</td>
<td>0.618</td>
<td>0.5446</td>
<td>0.4755</td>
<td></td>
</tr>
<tr>
<td>Fitness Master</td>
<td>21264</td>
<td>21113</td>
<td>21370</td>
<td></td>
</tr>
</tbody>
</table>

Table 6: Simulations with populations less than 100 individuals.

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Cluster 0</th>
<th>Cluster 1</th>
<th>Cluster 2</th>
<th>Cluster 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Elements</td>
<td>3</td>
<td>7</td>
<td>9</td>
<td>10</td>
</tr>
<tr>
<td>Search</td>
<td>1566</td>
<td>2059</td>
<td>9173</td>
<td>1372</td>
</tr>
<tr>
<td>Crossover Rate</td>
<td>0.52</td>
<td>0.56</td>
<td>0.7</td>
<td>0.38</td>
</tr>
<tr>
<td>Mutation Rate</td>
<td>0.66</td>
<td>0.5</td>
<td>0.74</td>
<td>0.30</td>
</tr>
<tr>
<td>Fitness Master</td>
<td>19783</td>
<td>21175</td>
<td>20405</td>
<td>20402</td>
</tr>
<tr>
<td>Generations</td>
<td>159</td>
<td>77</td>
<td>105</td>
<td>45</td>
</tr>
<tr>
<td>Population</td>
<td>10</td>
<td>28</td>
<td>85</td>
<td>78</td>
</tr>
</tbody>
</table>

5. Conclusions

The objective of this study was to develop a Parallel Genetic Algorithm (PGA) for the coevolutionary strategic parameters, applied to the Multidimensional Knapsack Problem (MKP). The inherent parallel feature of Genetic Algorithm (GA), was chosen computing model for distributed memory implementation of coevolution through MPI.

Through analysis of the simulations was found groups of individuals of strategic parameters, reaching them with a good fitness values due to randomness, and the other group showing the ability of the evolutionary model developed. It created an objective function, which gets the fitness of the master processor through the relationship of efficiency and effectiveness of the AG in the slave processors, and other objective function with penalty is assessed according to the number of searches.

The experiments have shown the potential of parallel computing models, especially MPI, indicating as future work the possibility of promising to test the model on GPU using CUDA implementation, concluding that larger populations tend to master the process to ensure a more evolutionary process accuracy.

References


