Volume 2, Issue 8, August 2012 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Comparison of processor scheduling algorithms using Genetic Approach

Vikas Gaba, Anshu Prashar
Department of Computer Science and Engineering
Haryana College of Technology and Management
Haryana, India

Abstract— Operating system efficiency is affected by several factors. Scheduling of CPU is one of the critical factor that affects the efficiency. In case of process scheduling we allocate processes to processor in a specified sequence so that efficiency of the system is maximized. In this paper genetic algorithm is used to find a solution for CPU Scheduling. Genetics algorithm is based on the principle of genetics and evolution Using efficient encoding, crossover and mutation operators we accomplish our objective. The population is encoded with the suitable encoding scheme .Our aim is to find out the minimum average waiting time for the given number of processes And result of genetic algorithm is then compared with the First Come First Serve and Shortest Job first scheduling algorithm.

Keywords- CPU scheduling, NP Hard, genetic algorithm, crossover, mutation,selection.

I. Introduction

Scheduling problems are typically NP-Complete [2] and so there are no known algorithms that can find an optimal solution in polynomial time. Therefore, for sizeable scheduling problems finding the optimal solution may not be practical, and we resort to techniques that find reasonable solutions in an acceptable timeframe.There are a number of scheduling algorithms based on which we can evaluate the various scheduling criteria viz. avg. waiting time, turnaround time etc. Among them First-Come First-Served (FCFS) Scheduling, Shortest-Job-First (SJF) Scheduling, Priority Scheduling, Round Robin (RR) Scheduling, Multilevel Queue Scheduling are of much importance and are widely used for scheduling of jobs in a processor. Evolutionary algorithms (EA's) are population-based optimization algorithms that use biology-inspired mechanisms like mutation, crossover, natural selection, and survival of the fittest in order to refine a set of solution candidates iteratively [BAC97, BAC96]. The advantage of evolutionary algorithms compared to other optimization methods is their black box character that makes only few assumptions about the underlying objective functions. The most popular technique in evolutionary computation research has been the genetic algorithm. The different types of operators on which genetic algorithm is made are selection operators, reproduction operators (i.e. crossover and mutation operators). These operators are the main foundations over which a genetic algorithm is based. This study is an effort to develop a simple general algorithm (genetic algorithm based) for obtaining optimal or near optimal schedules for Single Processor Scheduling Problems with minimum computation effort even for large sized problems and comparing the minimum average waiting time using different scheduling algorithms (like FCFS, SJF and scheduling using genetic algorithm).

II. Optimal Solution Using Genetic Approach

John Holland first proposed genetic algorithms in the 1960s . The genetic algorithm is a search algorithm based on the mechanics of natural selection and natural genetics [6]. As summarized by Tomassini [5], the main idea is that, in order for a population of individuals to adapt to some environment, it should behave like a natural system. Using Tomassini's terms [5], genetic algorithms (GA's) consider an optimization problem as the environment where feasible solutions are the individuals living in that environment.

A. *Optimization and Optimization Problem*

In mathematics and computer science, optimization, or mathematical programming refers to choosing the best element from some set of available alternatives. In the simplest case, this means solving problems in which one seeks to minimize or maximize a real function by systematically choosing the values of real or integer variables from within an allowed set. This formulation, using a scalar, real-valued objective function, is probably the simplest example.The generalization of optimization theory and techniques to other formulations comprises a large area of applied mathematics. More generally, it

means finding best available values of some objective function given a defined domain, including a variety of different types of objective functions and different types of domains.
An optimization problem can be represented in the following way:

Given*:* a function f: $A \longrightarrow$R from some set A of some elements to the real numbers.

Sought*:* an element x_0 in A such that $f(x_0) \leq f(x)$ for all x in A ("minimization") or such that $f(x_0) \geq f(x)$ for all x in A ("maximization").

Such a formulation is called an optimization problem or a mathematical programming problem. Many real-world and theoretical problems may be modeled in this general framework. Problems formulated using this technique in the fields of physics and computer vision may refer to the technique as energy minimization, speaking of the value of the function f as representing the energy of the system being modeled

B. Genetic algorithms

A genetic algorithm (GA) is a search technique used in computing to find exact or approximate solutions to optimization and search problems [3]. Genetic algorithms are categorized as global search heuristics. Genetic algorithms are a particular class of evolutionary algorithms (EA) that use techniques inspired by evolutionary biology such as inheritance, mutation, selection, and crossover.

C. CPU SCHEDULING

Multiprogramming operating system allows more than one process to be loaded into the executable memory at a time and for the loaded process to share the CPU using time multiplexing. Part of the reason for using multiprogramming is that the operating system itself is implemented as one or more processes, so there must be a way for the operating system and application processes to share the CPU. Another main reason is the need for processes to perform I/O operations in the normal course of computation. Since I/O operations ordinarily require orders of magnitude more time to complete than do CPU instructions, multiprogramming systems allocate the CPU to another process whenever a process invokes an I/O operation.

Simple Genetic Algorithm

An algorithm is a series of steps for solving a problem. A genetic algorithm is a problem solving method that uses genetics as its model of problem solving. It's a search technique to find nearby solutions to optimization and search problems [7]. GA handles a population of possible solutions. Each solution is represented through a chromosome, which is just an abstract representation. Coding all the possible solutions into a chromosome is the first part, but certainly not the most straightforward one of a Genetic Algorithm. A set of reproduction operators has to be determined, too. Reproduction operators are applied directly on the chromosomes, and are used to perform mutations and recombination over solutions of the problem. Appropriate representation and reproduction operators are really something determinant, as the behavior of the GA is extremely dependant on it [8].

Selection is done by using a fitness function. Each chromosome has an associated value corresponding to the fitness of the solution it represents. The fitness should correspond to an evaluation of how good the candidate solution is. The optimal solution is the one, which maximizes the fitness function. Genetic Algorithms deal with the problems that maximize the fitness function. Once the reproduction and the fitness function have been appropriately defined, a Genetic Algorithm is evolved according to the same basic structure. It begins by generating an initial population of chromosomes. The gene pool should be as large as possible so that any solution of the search space can be engendered. Generally, the initial population is generated randomly[8]. The basic cycle of genetic algorithms is shown in Figure 1.

It shows that the genotypes are used in the reproduction operations whereas the values of the objective functions F_i are computed on basis of the phenotypes in the problem space X which are obtained via the genotype-phenotype mapping "gpm". The basic genetic algorithm is as follows:

- [start]: Genetic random population of n chromosomes (suitable solutions for the problem).
- [Fitness]: Evaluate the fitness f(x) of each chromosome x in the population.
- [New population]: Create a new population by repeating following steps until the new population is complete,
- [selection]: select two parent chromosomes from a population according to their fitness (the better fitness, bigger are the chances to get selected).
- [crossover]: With a crossover probability, cross over the parents to form new offspring (children). If no crossover was performed, offspring is the exact copy of parents.
- [Mutation]: With a mutation probability, mutate new offspring at each locus (position in chromosome).

•[Accepting]: Place new offspring in the new population.
• [Replace]: Use newly generated population for a further sum of the algorithm.
• [Test]: If the end conditions are satisfied, stop and return the best solution in current population.
• [Loop]: Go to step2 for fitness evaluation.

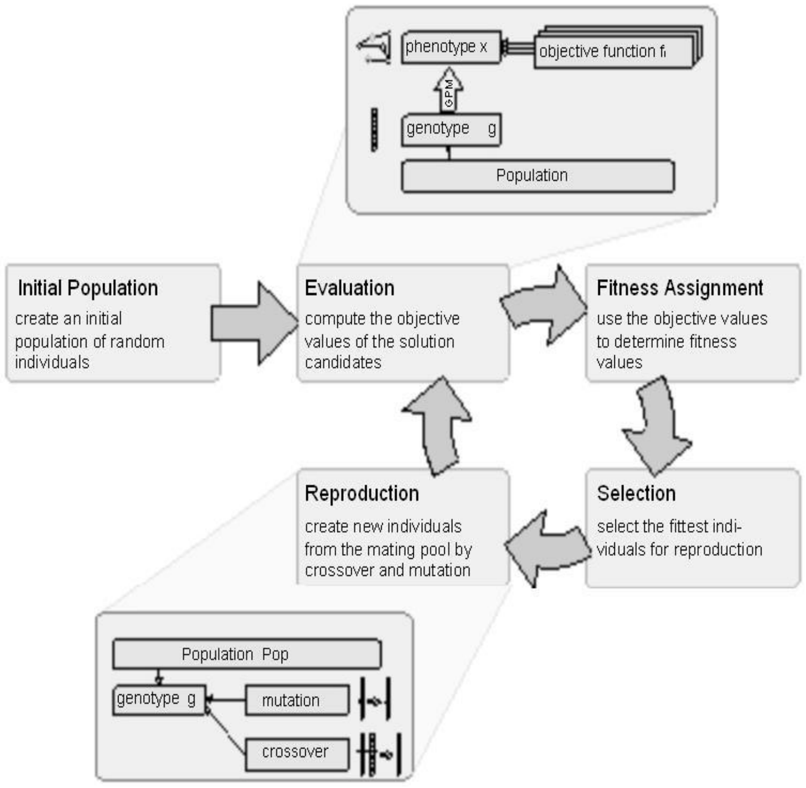

phenotype x
objective function fi
GPM
genotype g
Population
Initial Population
create an initial population of random individuals
Evaluation
compute the objective values of the solution candidates
Fitness Assignment
use the objective values to determine fitness values
Reproduction
create new individuals from the mating pool by crossover and mutation
Selection
select the fittest individuals for reproduction
Population Pop
genotype g
mutation
crossover

Figure 1: The basic cycle of genetic algorithms.

III. PROBLEM STATEMENT

In this, there is a number of processes (jobs) that are to waiting be processed by a single-processing system. The processor is assigned a job and remains busy until all the processes have been processed. Once a process is allocated the processor, remaining processes are forced to wait in the ready queue till their turn comes. The objective is to find the schedule, which follows following constrains:
1. Processor has to process every job, without leaving any job unprocessed.

2. Once a job is executed completely, processor is allocated to next job. Each job should be processed only one time.

3. No preemption is allowed, i.e. processor can't switch between processes before completion of currently executing process.

4. The total waiting time that for all the jobs till processor runs all the jobs in the ready queue should be minimum .

Here the problem is comparing the performance of different scheduling algorithms. As already discussed previously, genetic algorithm is one of the optimization techniques that can be used to solve the problems of function maximization. It can be said as a search procedure inspired by principles from natural selection and genetics [9]. It is often used as an optimization method to solve problems where very little is known about the objective function. The operation of the genetic algorithm is very simple. It starts with a population of random individuals, each corresponding to a particular candidate solution to the problem to be solved. Then, the best individuals survive, mate, and create offspring, originating a new population of individuals. This process is repeated a number of times, and usually leads to better and better individuals.

IV. GA RELATING OUR PROBLEM DOMAIN

The steps of applying GA relating our problem at hand are:

1. Choosing an Encoding scheme.
2. Choosing Fitness function.
3. Choosing Operators.
4. Choosing Parameters.
5. Choosing an Initialization method and Stopping criteria

A. Encoding scheme

The application of a genetic algorithm to a problem starts with the encoding. The basis of genetics in nature is a chromosome.

A particular solution to the problem can then be represented by a specific assignment of values to the decision variables. The set of all possible solutions is called the search space and a particular solution represent a point in that search space. Various types of encoding schemes are:

1. Binary encoding
2. Permutation encoding
3. Value/Real encoding
4. Tree encoding
5. Octal encoding &
6. Hexadecimal encoding.

For our problem value encoding is required.

Real/Value encoding

Direct value encoding can be used in problems where some complicated value such as real numbers are used. Use of binary encoding for this type of problems would be very difficult. In value encoding, every chromosome is a string of some values. Values can be anything connected to problem, form numbers, real numbers or chars to some complicated objects.

Chromosome A	1 7 8 9 4 6 11 2 3 5 10 15 14 13 12 16
Chromosome B	ABDJEIFJDHDIERJFDLDFLFEGT
Chromosome C	(back), (back), (right), (forward), (left)

Figure 2: Value encoding [9]

Various examples of encoded chromosomes relating to our problem are:

[1 7 3 2 4 5 6 10 9 8]

[12 10 8 11 6 4 3 2 5 1 7 9]

The next step after encoding is to select the fitness function.

B. Fitness function

The next step is to specify a function that can assign a score to any possible solution. The score is a numerical value that indicates how well a particular solution solves the problem. The score is the fitness of the individual solution .The task of the GA is to discover solutions that have higher fitness values among the set of all possible solutions. A fitness function must be designed for the problem under consideration. For a solution (Chromosome), the fitness function must return a single numerical value, which is proportional to the "fitness" of that solution. Our aim is to find out individual having minimum average waiting time. So the individual who has minimum average waiting time is fittest as compared to other. So, The fitness function of a Solution S_i is given by

$$\text{Fitness }(Si) = \frac{\sum_{j=1}^{n} wj}{n}$$

Where wj is the waiting time of the process j.

n is the total no. of processes.

C. Operators

Once the encoding and the fitness function are specified, the implementer has to choose selection and genetic operators to evolve new solutions to the problem being solved. The selection operator simulates the "survival-of-the-fittest". There are various mechanisms to implement this operator, and the idea is to give preference to better individuals. Selection replicates individuals with high fitness values and removes individuals with low fitness values.

Selection

Selection is the process of choosing two parents from the population for crossing. After deciding on an encoding, the next step is to decide how to perform selection i.e., how to choose individuals in the population that will create offspring for the next generation and how many offspring each will create.

Selection is a method that randomly picks chromosomes out of the population according to their evaluation function. The higher the fitness function, the more chance an individual has to be selected. The selection pressure is defined as the degree to which the better individuals are favored.

Selection has to be balanced with variation form crossover and mutation. Too strong selection means sub optimal highly fit individuals will take over the population, and it reduces the diversity needed for change and progress, too weak selection will result in too slow evolution.

The Figure 3 shows the basic selection process.

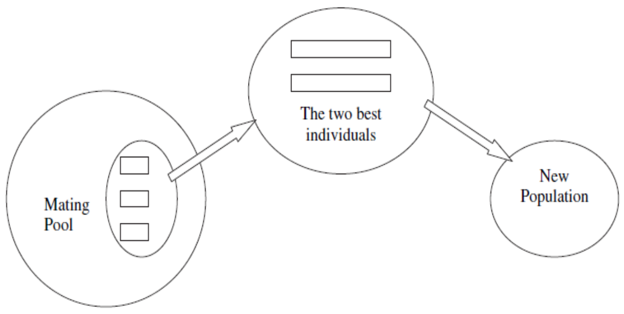


Figure 3 Selection [7]

Roulette wheel selection

We use Roulette selection which is one of the traditional GA selection techniques.. The principle of roulette wheel selection is a linear search through a roulette wheel with the slots in the wheel weighted in proportion to the individual's fitness values. A target value is set, which is a random proportion of the sum of the finesses in the population. The population is stepped through until the target value is reached. The expected value of an individual is that fitness divided by the actual fitness of the population. A slice of the roulette wheel is assigned to each individual, the size of the slice being proportional to the individual's fitness. Here N is the number of individuals in the population so the wheel is spun N times. On each spin, the individual under the wheel's marker is selected to be parents for the next generation.

This method is implemented as follows:

1. total expected value of the individuals in the population is summed . Let it be T.

2. Repeat N times:

- A random integer 'r' between o and T is Chosen.
- Loop through the individuals in the population, summing the expected values, until the sum is greater than or equal to 'r'. The individual whose expected value puts the sum over this limit is the one selected.

Roulette wheel selection is easier to implement but is noisy. The rate of evolution depends on the variance of fitness's in the population.

Crossover

Crossover is the process of taking two parent chromosomes and producing from them a number of offspring's. After the selection (reproduction) process, the population is filled with better individuals. Reproduction makes exact copies of good strings but does not create new ones. Crossover operator is applied to the mating pool with the hope that it creates a better offspring [7].

Crossover is a recombination operator that works in three steps:

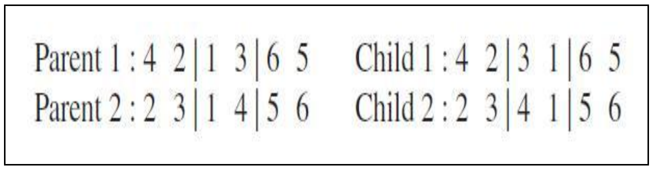
i. The reproduction operator selects at random a pair of two individual strings for the mating/crossover.

ii. A cross site or cross point is selected at random along the string length.

iii. Finally, the position values are swapped between the two strings following the cross site.

Ordered crossover

Ordered two-point crossover is used when the problem is order based, for example in U-shaped assembly line balancing etc [7]. Given two parent chromosomes, two random crossover points are selected partitioning them into a left, middle and right portion. The ordered two-point crossover behaves in the following way: first child inherits its left and right section from parent 1, and its middle section is determined by the genes in the middle section of parent 1 in the order in which the values appear in parent 2 [7]. A similar process is applied to determine the second child. This is shown in Figure 4 below:



```
Parent 1 : 4  2 | 1  3 | 6  5     Child 1 : 4  2 | 3  1 | 6  5
Parent 2 : 2  3 | 1  4 | 5  6     Child 2 : 2  3 | 4  1 | 5  6
```

Figure 4 Ordered crossover

Mutation

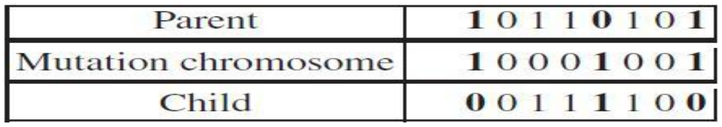
The drawback of algorithm being trapped in local minimum is prevented by Mutation. Recovery of lost genetic materials as well as for randomly distributing genetic information is the vital role of Mutation. To preserve genetic diversity in the population mutation acts as a background operator..

Mutation of a bit involves flipping a bit, changing 0 to 1 and vice-versa. There are various types of mutation schemes such as:

- Flipping
- Interchanging
- Reversing
- Uniform mutation &
- Non-uniform mutation.

Flipping

Flipping of a bit involves changing 0 to 1 and 1 to 0 based on a mutation chromosome generated. The Figure 5 below explains mutation-flipping concept.

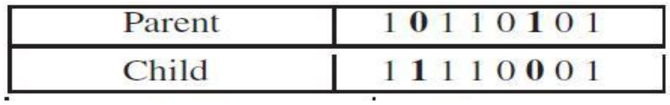


Parent	**1** 0 1 1 **0** 1 0 **1**
Mutation chromosome	**1** 0 0 0 **1** 0 0 **1**
Child	**0** 0 1 1 **1** 1 0 **0**

Figure 5 Mutation Flipping [7]

Interchanging

Two random positions of the string are chosen and the bits corresponding to those positions are interchanged. This is shown below in Figure 6

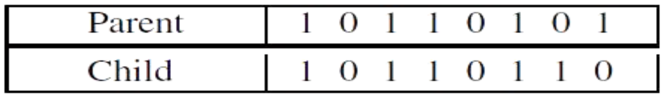


Parent	1 **0** 1 1 0 **1** 0 1
Child	1 **1** 1 1 0 **0** 0 1

Figure 6 Mutation Interchanging [7]

Reversing

A random position is chosen and the bits next to that position are reversed and child chromosome is produced. This is shown below in Figure 7.



Parent	1 0 1 1 0 1 0 1
Child	1 0 1 1 0 1 1 0

Figure 7 Reversing [7]

For our problem, interchanging is used.

D. Parameters

With an encoding, a fitness function, and operators in hand, the GA is ready to enter in action. But before doing that, the user has to specify a number of parameters such as population size, no. of processes [9].

For crossover the probability is 100% i.e. with every iteration/algorithm run crossover is performed always. Whereas the mutation operation performed after every five consecutive algorithm runs.

E. Initialization method & stopping criteria

Following the initialization step, each individual is evaluated according to the user's specified fitness function. Thereafter, the GA simulates evolution on the artificial population of solutions using operators that imitate the survival-of-the-fittest and principles of natural genetics such as recombination and mutation [9].
A number of criteria can be chosen for this purpose, including among others, a maximum number of generations or time has elapsed, some predefined fitness value is reached, or the population has converged considerably [9]. For our problem, stopping criteria chosen is no. of iterations and initialization method is random generation of population.

V. RESULT

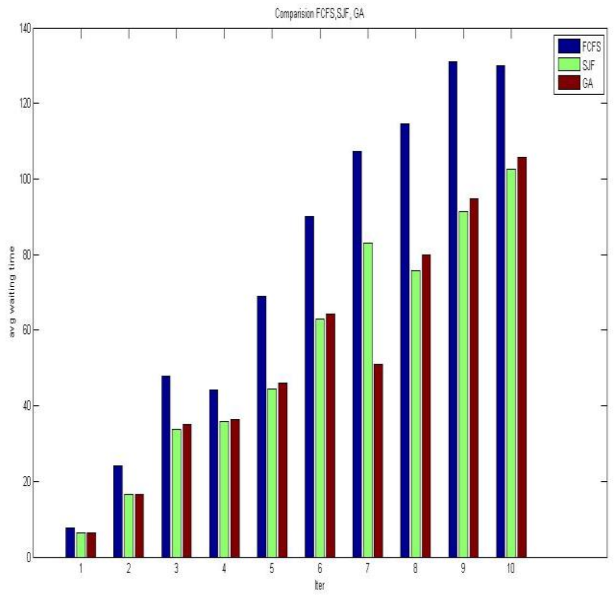


Figure 8 :Comparison

Three algorithms for CPU scheduling are implemented The procedure followed while implementing it is first started with randomly generating the population. This process in GA has been called as encoding the input population. Then a selection technique has been employed over the encoded population and after that crossover is performed. After the crossover operation, mutation is performed, but after every 5 iterations. Then this operation is repeated for the required number of iterations.

The bar graph shown here is about the results showing the **average waiting time** of various algorithms.
The results that came out from genetic algorithm are same or nearly same result as SJF.

VI. CONCLUSION

Waiting time optimization/minimization is the main point of discussion in this paper. This paper is based on implementing a scheduling based Genetic algorithm concepts and making a comparison of it with SJF and FCFS scheduling algorithms based on the average waiting time of these algorithms at different number of iterations.

Basically GA is one of the better function optimization methods generally employed now days. Population is randomly generated. The encoding scheme for this problem of function maximization is value/real encoding. And so a

different type of crossover method is applied for crossing the chromosomes in the population for producing better offspring's. And finally interchanging mutation is used while implementing this algorithm.
In this dissertation mainly we solve the problem of CPU scheduling using the genetic algorithm . Our algorithm gives same or nearly same result as SJF. After all the experimentation and implementation, results that came out are like, best one is genetic algorithm.

VII. REFERENCES

[1] H.Nazif(2009). *A Genetic Algorithm on Single Machine Scheduling Problem to Minimise Total Weighted Completion Time*. European Journal of Scientific Research,Vol.35 No.3, pp.444-452

[2] Garey, M. & Johnson, D. (1979). *Computers and Intractability: a theory of NP- Completeness.* W.H.Freeman, San Francisco Proceedings of the Second International Conference on Genetic Algorithms, pp. 252-256

[3] Back, T., Fogel, David B. & Michalewicz, Z. (Eds.) (1997). *Handbook of Evolutionary Computation*. Computational Intelligence, Library Oxford University Press in cooperation with the Institute of Physics Publishing / CRC Press, Bristol, New York, ring bound edition. ISBN: 0-7503-0392-1, 978-0-75030-392-7, 0-7503-0895-8, 978-0-75030-895-3. [Also available online at:http://books.google.de/books?id=n5nuiIZvmpAC]

[4] Back, T. (1996). *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. Oxford University Press US. ISBN: 0-1950-9971-0, 978-0-19509-971-3.[Also available online at: http://books.google.de/books?id=EaN7kvl5coYC]

[5] Tomassini, M. (1999). *Parallel and Distributed Evolutionary Algorithms: A Review*. In Miettinen, K., Makela, M., & Periaux, J. (Eds.), *Evolutionary Algorithms in Engineering and Computer Science* (pp. 113 - 133). Chichester: J. Wiley and Sons.

[6] Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Boston: Addison-Wesley

[7] Sivanandam, S. N. & Deepa, S. N. (2008). *Introduction to Genetic Algorithms*. Springer

[8] Davis, L. (1991). *Handbook of Genetic Algorithm*. Von Nostrand Reinhold, Newyork.

[9] Lobo, Fernando Miguel (2000).*The parameter-less genetic algorithm: rational and automated parameter selection for simplified genetic algorithm operation.* International Conference on Genetic Algorithms, in Lisboa.

: