Optimal resequencing using Genetic algorithm*

by Dmitriy Borodin, Pieter Caluwaerts,
Viktor Gorelik and Alexander Rodyukov
University College of Ghent, Belgium
Dorodnicyn Computing Centre of RAS, Russia
Borisoglebsk State Pedagogical University, Russia

This paper describes special genetic operators for searching an optimal solution (in terms of heuristic methods) in the problems where the solution is represented as a sequence of unique elements (numbers, letters etc, e.g. 42751368 or BDACE). There is a number of problems with similar solution representation: Travelling Salesman Problem (TSP), Graph Theory problems (minimal or maximal path on a graph) etc. Authors try and assume the formalization as follows: we need to find a maximum (or minimum) of an objective function *F(X)*, within a solution pool of $\{x_1, x_2, \ldots, x_n\}$, where *n* is the length of vector *X* and each x_i is unique. In other words, the problem is in finding the sequence of the solution vector *X** components bringing the *minimum* (or *maximum*) value to the objective function *F(X*)*.

Such problems can be solved using analytical techniques such as branch and bound method but the more the number of the vector x components, the more is the polynomial time for finding a solution and from some value of *n* it is nearly impossible to use analytical techniques.

The authors propose to use genetic algorithm (GA) to solve the above mentioned problem as a reliable heuristic technique. GA does not ensure an optimal solution, however it usually gives good approximations in a reasonable amount of time. Genetic algorithms are loosely based on natural evolution and use a "survival of the fittest" technique, where the best solutions survive and are varied until we get a good result. The classical GA process consists of the following steps: 1) Encoding; 2) Evaluation; 3) Crossover; 4) Mutation; 5) Decoding.

A suitable encoding is found for the solution to a problem so that each possible solution has a unique encoding and the encoding is some form of a string. For our problem this condition is satisfied and no solution encoding/decoding is needed, so we use GA *excluding the*

* The research is funded by the European CAPSCHED project

coding/decoding steps. Then the initial population is selected, usually at random. The fitness of each individual in the population is then computed, according to the objective function *F(X)*; that is, how well the individual fits the problem and whether it is near the optimum compared to the other individuals in the population. This fitness is used to find the individual's probability of crossover. If an individual has a high probability (which indicates that it is significantly closer to the optimum than the rest of its generation) then it is more likely to be chosen to crossover. Crossover is where the two individuals are recombined to create new individuals which are copied into the new generation. For our problem the classical GA crossover doesn't work because it produces children with duplicates and missing elements. Some crossovers for such situations have been developed so far, but we choose two which are simple and effective: a variation of the Greedy Crossover (GC) and Partially Matched Crossover (PMC). GC: we choose the first element from one of the parent chromosomes (parent1 = 12345678; parent2 = 85213647). We pick 1 from parent1, child1 = 1*******. We must pick every element from one of the parents and place it in the position it was previously in. Since the first position is occupied by 1, the number 8 from parent2 can not be there. So we must now pick the 8 from parent1, child1 = 1*******8. This forces to put 7 in position 7 and 4 in position 4, as in parent1, child1 = 1**4**78. Since the same set of position is occupied by 1,4,7,8 in parent1 and parent2, we finish by filling in the blank positions with the elements of those positions in parent2: child1 = 15243678, and we deduce child2 from the complement of child1.

In PMC we take two random points (like in 2-point classical crossover) and swap the respective genes on the same positions in both parents: Parents 12|34|5 and 35|21|4 => 3->2 and 4->1 => Children 43215 and 25341 (respectively).

Next some individuals are chosen randomly to be mutated. Classical mutation has also to be replaced with one producing feasible changes to the solution, ie it is possible to swap two randomly selected elements or inverse the order in the elements located between two randomly selected ones. Once this is done, a new generation has been formed and the process is repeated until some stopping criterion has been reached. At this point the process is complete.

Computation experiments performed in Mathcad and Microsoft Visual C# .NET conclude the effectiveness of the described GA.