Diploma Practice Report

Ivanov Kirill

Brest State Technical University (BSTU) 2006

Contents

1. Introduction

According to this diploma practice I have to develop a new genetic algorithm (GA) for Traveling Sales-person Problem (TSP) (you may see the algorithm in part “The developed GA for TSP” of this report paper).

I had to apply the developed algorithm for real city coordinates.

I had to develop the algorithm to try to get more optimal results with the GA. About my results in part: “The developed GA for TSP”.

And about the report in brief. It consists of the parts:

1. Introduction: the goal of this report.
2. The developed GA for TSP: 1) the developed genetic algorithm for TSP; 2) about results of the developed GA for TSP.
3. Applications of TSP: 1) about TSP; where apply TSP.
4. Why GA for TSP?: 1) why I choosed GA; 2) what is GA in brief.
5. Conclusions: overviews of the diploma practice.

2. Applications of TSP

Much of the work on the TSP is not motivated by direct applications, but rather by the fact that the TSP provides an ideal platform for the study of general methods that can be applied to a wide range of discrete optimization problems. This is not to say, however, that the TSP does not find applications in many fields. Indeed, the numerous direct applications of the TSP bring life to the research area and help to direct future work.

The TSP naturally arises as a subproblem in many transportation and logistics applications, for example the problem of arranging school bus routes to pick up the children in a school district. This bus application is of important historical significance to the TSP, since it provided motivation for Merrill Flood, one of the pioneers of TSP research in the 1940s. A second TSP application from the 1940s involved the transportation of farming equipment from one location to another to test soil, leading to mathematical studies in Bengal by P. C. Mahalanobis and in Iowa by R. J. Jessen. More recent applications involve the scheduling of service calls at cable firms, the delivery of meals to homebound persons, the scheduling of stacker cranes in warehouses, the routing of trucks for parcel post pickup, and a host of others.

Although transportation applications are the most natural setting for the TSP, the simplicity of the model has led to many interesting applications in other areas. A classic example is the scheduling of a machine to drill holes in a circuit board or other object. In this case the holes to be drilled are the cities, and the cost of travel is the time it takes to move the drill head from one hole to the next. The technology for drilling varies from one industry to another, but whenever the travel time of the drilling device is a significant portion of the overall manufacturing process then the TSP can play a role in reducing costs.

The traveling salesman problem (TSP) is one which has commanded much attention of mathematicians and computer scientists specifically because it is so easy to describe and so difficult to solve. The problem can simply be stated as: if a traveling salesman wishes to visit exactly once each of a list of m cities (where the cost of traveling from city i to city j is cij) and then return to the home city, what is the least costly route the traveling salesman can take? A complete historical development of this and related problems can be found in Hoffman and Wolfe (1985).

The importance of the TSP is that it is representative of a larger class of problems known as combinatorial optimization problems. The TSP problem belongs in the class of combinatorial optimization problems known as NP-complete. Specifically, if one can find an efficient algorithm (i.e., an algorithm that will be guaranteed to find the optimal solution in a polynomial number of steps) for the traveling salesman problem, then efficient algorithms could be found for all other problems in the NP-complete class. To date, however, no one has found a polynomial-time algorithm for the TSP. Does that mean that it is impossible to solve any large instances of such problems? Many practical optimization problems of truly large scale are solved to optimality routinely. In 1994, Applegate, et. al. solved a traveling salesman problem which models the production of printed circuit boards having 7,397 holes (cities), and, in 1998, the same authors solved a problem over the 13,509 largest cities in the U.S. So, although the question of what it is that makes a problem "difficult" may remain open, the computational record of specific instances of TSP problems coming from practical applications is optimistic.

How are such problems tackled today? Obviously, one cannot consider a brute force approach. In one example of an 16 city traveling salesman problem -- the problem of Homer's Ulysses attempting to visit the cities described in The Odyssey exactly once -- there are 653,837,184,000 distinct routes, (Grotschel and Padberg, 1993)! Enumerating all such roundtrips to find a shortest one took 92 hours on a powerful workstation. Rather than enumerating all possibilities, successful algorithms for solving the TSP problem have been capable of eliminating most of the roundtrips without ever explicitly considering them.

2.1. Formulations

The first step to solving instances of large TSPs must be to find a good mathematical formulation of the problem. In the case of the traveling salesman problem, the mathematical structure is a graph where each city is denoted by a point (or node) and lines are drawn connecting every two nodes (called arcs or edges). Associated with every line is a distance (or cost). When the salesman can get from every city to every other city directly, then the graph is said to be complete. A round-trip of the cities corresponds to some subset of the lines, and is called a tour or a Hamiltonian cycle in graph theory. The length of a tour is the sum of the lengths of the lines in the round-trip.

Depending upon whether or not the direction in which an edge of the graph is traversed matters, one distinguishes the asymmetric from the symmetric traveling salesman problem. To formulate the asymmetric TSP on m cities, one introduces zero-one variables

$$x_{ij} = \begin{cases} 1 \text{ if the edge } i \to j \text{ is in the tour} \\ 0 \text{ otherwise} \end{cases}$$

and given the fact that every node of the graph must have exactly one edge pointing towards it and one pointing away from it, one obtains the classic assignment problem. These constraints alone are not enough since this formulation would allow "subtours", that is, it would allow disjoint loops to occur. For this reason, a proper formulation of the asymmetric traveling salesman problem must remove these subtours from consideration by the addition of "subtour elimination" constraints. The problem then becomes

$$\min \sum_{j=1}^{m} \sum_{i=1}^{m} c_{ij} x_{ij}$$

$$\text{s.t.} \sum_{j=1}^{m} x_{ij} = 1 \qquad \text{for } i = 1, \ldots, m$$

$$\sum_{i=1}^{m} x_{ij} = 1 \qquad \text{for } j = 1, \ldots, m$$

$$\sum_{i \in K} \sum_{j \in K} x_{ij} \leq |K| - 1 \quad \text{for all } K \subset \{1, \ldots, m\}$$

$$x_{ij} = 0 \text{ or } 1 \qquad \text{for all } i, j$$

where K is any nonempty proper subset of the cities 1,...,m. The cost cij is allowed to be different from the cost cji. Note that there are m(m-1) zero-one variables in this formulation.

To formulate the symmetric traveling salesman problem, one notes that the direction traversed is immaterial, so that cij = cji. Since direction does not now matter, one can consider the graph where there is only one arc (undirected) between every two nodes. Thus, we let xj e {0,1} be the decision variable where j runs through all edges E of the undirected graph and cj is the cost of traveling that edge. To find a tour in this graph, one must select a subset of edges such that every node is contained in exactly two of the edges selected. Thus, the problem can be formulated as a 2-matching problem in the graph Gv having m(m-1)/2 zero-one variables, i.e. half of the number of the previous formulation. As in the asymmetric case, subtours must be eliminated through subtour elimination constraints. The problem can therefore be formulated as:

$$\min 1/2 \sum_{j=1}^{m} \sum_{k \in J(j)} c_k x_k$$

$$s.t. \sum_{k \in J(j)} x_k = 2 \qquad \text{for all } j = 1,\ldots,m$$

$$\sum_{j \in E(K)} x_j \leq |K| - 1 \quad \text{for all } K \subset \{1,\ldots,m\}$$

$$x_j = 0 \text{ or } 1 \qquad \text{for all } j \in E,$$

where J(j) is the set of all undirected edges connected to node j and E(K) is the subset of all undirected edges connecting the cities in any proper, nonempty subset K of all cities. Of course, the symmetric problem is a special case of the asymmetric one, but practical experience has shown that algorithms for the asymmetric problem perform, in general, badly on symmetric problems. Thus, the latter need a special formulation and solution treatment.

2.2. Algorithms

Exact approaches to solving such problems require algorithms that generate both a lower bound and an upper bound on the true minimum value of the problem instance. Any round-trip tour that goes through every city exactly once is a feasible solution with a given cost which cannot be smaller than the minimum cost tour. Algorithms that construct feasible solutions, and thus upper bounds for the optimum value, are called heuristics. These solution strategies produce answers but without any quality guarantee as to how far off they may be from the optimal answer. Heuristic algorithms that attempt to find feasible solutions in a single attempt are called constructive heuristics while algorithms that iteratively modify and try to improve some given starting solution are called improvement heuristics. When the solution one obtains is dependent on the initial starting point of the algorithm, the same algorithm can be used multiple times from various (random) starting points. For an excellent survey of randomized improvement heuristics, see Junger, Reinelt and Rinaldi (1994). Often, if one needs a solution quickly, one may settle for a well-designed heuristic algorithm

that has been shown empirically to find "near-optimal" tours to many TSP problems. Research by Johnson (1990), and Junger, Reinelt and Rinaldi (1994) describes algorithms that find solutions to extremely large TSPs (problems with tens of thousands, or even millions of variables) to within 2% of optimality in very reasonable times. For genetic algorithmic approaches to the TSP, see Potvin (1996), for simulated annealing approaches see Aarts, et al. (1988), for neural net approaches, see Potvin (1993), and for tabu search approaches, see Fiechter (1990). Performance guarantees for heuristics are given in Johnson and Papadimitriou (1985); probabilistic analysis of heuristics are discussed in Karp and Steele (1985); and the development and empirical testing of heuristics is reported in Golden and Stewart (1985).

In order to know about the closeness of the upper bound to the optimum value, one must also know a lower bound on the optimum value. If the upper and lower bound coincide, a proof of optimality is achieved. If not, a conservative estimate of the true relative error of the upper bound is provided by the difference of the upper and the lower bound divided by the lower bound. Thus, one needs both upper and lower bounding techniques to find provably optimal solutions to hard combinatorial problems or even to obtain solutions meeting a quality guarantee.

So how does one obtain and improve the lower bound? A relaxation of an optimization problem is another optimization problem whose set of feasible solutions properly contains all feasible solution of the original problem and whose objective function value is less than or equal to the true objective function value for points feasible to the original problem. Thus we replace the "true" problem by one with a larger feasible region that is more easily solvable. This relaxation is continually refined so as to tighten the feasible region so that it more closely represents the true problem. The standard technique for obtaining lower bounds on the TSP problem is to use a relaxation that is easier to solve than the original problem. These relaxations can have either discrete or continuous feasible sets. Several relaxations have been considered for the TSP. Among them are the n-path relaxation, the assignment relaxation, the 2-matching relaxation, the 1-tree relaxation, and the linear programming relaxation. For randomly generated asymmetric TSPs, problems having up to 7500 cities have been solved using an assignment relaxation which adds subtours within a branch and bound framework and which uses an upper bounding heuristic based on subtour patching, (Miller and Pekny, 1991). For the symmetric TSP, the 1-tree relaxation and the 2-matching relaxations have been most successful. These relaxations have been embedded into a branch-and-cut framework.

The process of finding constraints that are violated by a given relaxation, is called a cutting plane technique and all successes for large TSP problems have used cutting planes to continuously tighten the formulation of the problem. It is important to stress that all successful computational approaches to the TSP utilize facet-defining inequalities as cutting planes. General-type cutting planes of the integer programming literature that use the simplex basis-representation to obtain cuts, such as Gomory or intersection cuts, have long been abandoned because of poor convergence properties.

One of the simplest cuts that have been shown to define facets of the underlying TSP polytope are the subtour elimination cuts. Besides these constraints, comb inequalities, clique tree inequalities, path, wheelbarrow and bicycle inequalities, ladder inequalities and crowns have also been shown to define facets of this polytope. The underlying theory of facet generation for the symmetric traveling salesman problem is provided in Grotschel and Padberg (1985) and Junger, Reinelt and Rinaldi (1994). The algorithmic descriptions of how these are used in cutting plane

approaches are discussed in Padberg and Rinaldi (1991) and Junger, Reinelt and Rinaldi (1994). Parallel processing implementations are presented in Christof and Reinelt (1995) and Applegate, et al. (1998). Cutting plane procedures can then be embedded into a tree search referred to as branch and cut. Some of the largest TSP problems solved have used parallel processing to assist in the search for optimality. As our understanding of the underlying mathematical structure of the TSP problem improves, and with the continuing advancement in computer technology, it is likely that many difficult and important combinatorial optimization problems will be solved using a combination of cutting plane generation procedures, heuristics, variable fixing through logical implications and reduced costs and tree search.

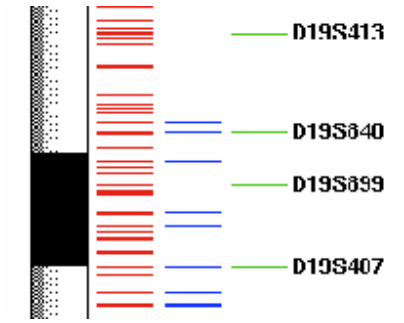
2.3. Applications

2.3.1. Overviews

One might ask, however, whether the TSP problem is important enough to have received all of the attention it has. Besides being a "polytope" of a difficult combinatorial optimization problem from a complexity theory point of view, there are important cases of practical problems that can be formulated as TSP problems and many other problems are generalizations of this problem. Besides the drilling of printed circuits boards described above, problems having the TSP structure occur in the analysis of the structure of crystals, (Bland and Shallcross, 1987), the overhauling of gas turbine engines (Pante, Lowe and Chandrasekaran, 1987), in material handling in a warehouse (Ratliff and Rosenthal, 1981), in cutting stock problems, (Garfinkel, 1977), the clustering of data arrays, (Lenstra and Rinooy Kan, 1975), the sequencing of jobs on a single machine (Gilmore and Gomory, 1964) and the assignment of routes for planes of a specified fleet (Boland, Jones, and Nemhauser, 1994). Related variations on the traveling salesman problem include the resource constrained traveling salesman problem which has applications in scheduling with an aggregate deadline (Pekny and Miller, 1990). This paper also shows how the prize collecting traveling salesman problem (Balas, 1989) and the orienteering problem (Golden, Levy and Vohra, 1987) are special cases of the resource constrained TSP. Most importantly, the traveling salesman problem often comes up as a subproblem in more complex combinatorial problems, the best known and important one of which is the vehicle routing problem, that is, the problem of determining for a fleet of vehicles which customers should be served by each vehicle and in what order each vehicle should visit the customers assigned to it. For relevant surveys, see Christofides (1985) and Fisher (1987).

2.3.2. More in details some real world application examples

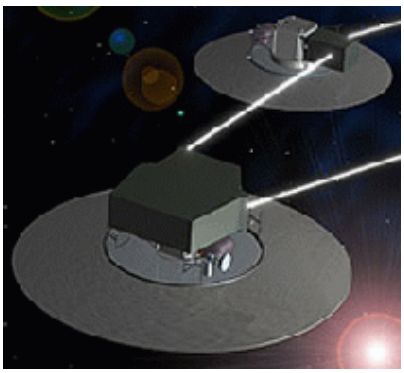
Genome Sequencing



Researchers at the National Institute of Health have used Concorde's TSP solver to construct radiation hybrid maps as part of their ongoing work in genome sequencing. The TSP provides a way to integrate local maps into a single radiation hybrid map for a genome; the cities are the local maps and the cost of travel is a measure of the likelihood that one local map immediately follows another. A report on the work is given in the paper "A Fast and Scalable Radiation Hybrid Map Construction and Integration Strategy", by R. Agarwala, D.L. Applegate, D. Maglott, G.D. Schuler, and A.A. Schaffler.

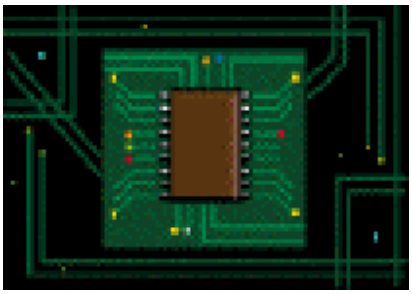
This application of the TSP has been adopted by a group in France developing a map of the mouse genone. The mouse work is decribed in "A Radiation Hybrid Transcript May of the Mouse Genome", Nature Genetics 29 (2001), pages 194--200.

Starlight Interferometer Program



A team of engineers at Hernandez Engineering in Houston and at Brigham Young University have experimented with using Chained Lin-Kerninghan to optimize the sequence of celestial objects to be imaged in a proposed NASA Starlight space interferometer program. The goal of the study it to minimize the use of fuel in targeting and imaging maneuvers for the pair of satellites involved in the mission (the cities in the TSP are the celestial objects to be imaged, and the cost of travel is the amount of fuel needed to reposition the two satellites from one image to the next). A report of the work is given in the paper "Fuel Saving Strategies for Separated Spacecraft Interferometry".

Scan Chain Optimization



A semi-conductor manufacturer has used Concorde's implementation of the Chained Lin-Kernighan heuristic in experiments to optimize scan chains in integrated circuits. Scan chains are routes included on a chip for testing purposes and it is useful to minimize their length for both timing and power reasons.

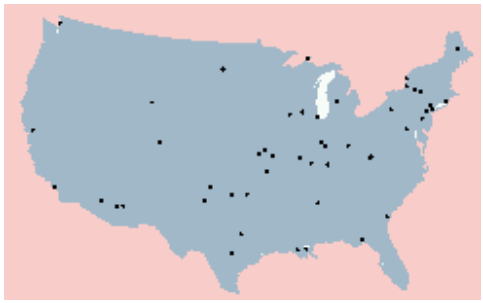
Genome Sequencing

An old application of the TSP is to schedule the collection of coins from payphones throughout a given region. A modified version of Concorde's Chained Lin-Kernighan heuristic was used to solve a variety of coin collection problems. The modifications were needed to handle 1-way streets and other features of city-travel that make the assumption that the cost of travel from x to y is the same as from y to x unrealistic in this scenario.

Touring Airports

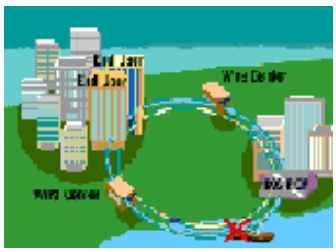
Concorde is currently being incorporated into the Worldwide Airport Path Finder web site to find shortest routes through selections of airports in the world. The author of the site writes that users of the path-finding tools are equally split between real pilots and those using flight simulators.

USA Trip



The travel itinerary for an executive of a non-profit organization was computed using Concorde's TSP solver. The trip involved a chartered aircraft to visit cities in the 48 continental states plus Washington, D.C. (Commercial flights were used to visit Alaska and Hawaii.) It would have been nice if the problem was the same as that solved in 1954 by Dantzig, Fulkerson, and Johnson, but different cities were involved in this application (and somewhat different travel costs, since flight distances do not agree with driving distances). The data for the instance was collected by Peter Winker of Lucent Bell Laboratories.

Designing Sonet Rings



An early version of Concorde's tour finding procedures was used in a tool for designing fiber optical networks at Bell Communications Research (now Telcordia). The TSP aspect of the problem arises in the routing of sonet rings, which provide communications links through a set of

sites organized in a ring. The ring structure provides a backup mechanism in case of a link failure, since traffic can be rerouted in the opposite direction on the ring.

Power Cables

Modules from Concorde were used to locate cables to deliver power to electronic devices associated with fiber optic connections to homes. Some general aspects of this problem area are discussed in the paper "Powering the Last Mile: An Alternative to Powering FITL".

3. Why GA for TSP?

A salesman wants to visit n cities cyclically. He wants to visit each city once and return to the city where he starts. In which way should he visit the cities so that the distance traveled by him will be minimum?

This is a simple combinatorial problem. For a small value of n, one can easily find the solution by having a permutation of the cities which is the minimum distance tour. But for larger n, it would be impracticable as there are (n-1)!/2 ways to visit the cities.

Many approaches have been proposed for the problem. Of these, Genetic Algorithms can produce reasonable solutions in a short span of time.

Physics, Biology, Economy or Sociology often have to deal with the classical problem of optimization. Economy particularly has become specialist of that field1. Generally speaking, a large part of mathematical development during the XVIIIth century dealt with that topic (remember those always repeated problems where you had to obtain the derivative of a function to find its extremes).

Purely analytical methods widely proved their efficiency. They nevertheless suffer from a insurmountable weakness : Reality rarely obeys to those wonderful differentiable functions your professors used to show you.

Other methods, combining mathematical analysis and random search have appeared. Imagine you scatter small robots in a Mountainous landscape. Those robots can follow the steepest path they found. When a robot reaches a peak, it claims that it has found the optimum. This method is very efficient, but there's no proof that the optimum has been found, each robot can be blocked in a local optimum. This type of method only works with reduced search spaces.

What could be the link between optimization methods and artificial life ?

3.1. Evolution and optimization.

We are now 45 millions years ago examining a Basilosaurus :

Figure 2: Basilosaurus

The Basilosaurus was quite a prototype of a whale. It was about 15 meters long for 5 tons. It still had a quasi-independent head and posterior paws. He moved using undulatory movements and hunted small preys3. Its anterior members were reduced to small flippers with an elbow articulation.

Movements in such a viscous element (water) are very hard and require big efforts. People concerned must have enough energy to move and control its trajectory. The anterior members of basilosaurus were not really adapted to swimming4. To adapt them, a double phenomenon must

occur : the shortening of the "arm" with the locking of the elbow articulation and the extension of the fingers which will constitute the base structure of the flipper.

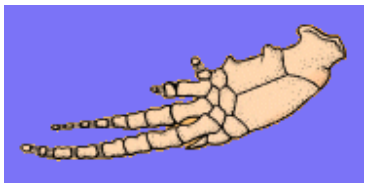


Figure 3: Tursiops flipper

The image shows that two fingers of the common dolphin are hypertrophied to the detriment of the rest of the member.

The basilosaurus was a hunter, he had to be fast and precise. Through time, subjects appeared with longer fingers and short arms. They could move faster and more precisely than before, and therefore, live longer and have many descendants.

Meanwhile, other improvements occurred concerning the general aerodynamic like the integration of the head to the body, improvement of the profile, strengthening of the caudal fin ... finally producing a subject perfectly adapted to the constraints of an aqueous environment.

This process of adaptation, this morphological optimization is so perfect that nowadays, the similarity between a shark, a dolphin or a submarine is striking. But the first is a cartilaginous fish (Chondrichtyen) originating in the Devonian (-400 million years), long before the apparition of the first mammal whose Cetacean descend from5.

Darwinian mechanism hence generate an optimization process6, Hydrodynamic optimization for fishes and others marine animals, aerodynamic for pterodactyls, birds or bats. This observation is the basis of genetic algorithms.

3.2. Evolution and Genetic Algorithms

John Holland, from the University of Michigan began his work on genetic algorithms at the beginning of the 60s. A first achievement was the publication of Adaptation in Natural and Artificial System7 in 1975.

Holland had a double aim : to improve the understanding of natural adaptation process, and to design artificial systems having properties similar to natural systems8.

The basic idea is as follow : the genetic pool of a given population potentially contains the solution, or a better solution, to a given adaptive problem. This solution is not "active" because the genetic combination on which it relies is split between several subjects. Only the association of different genomes can lead to the solution. Simplistically speaking, we could by example consider that the shortening of the paw and the extension of the fingers of our basilosaurus are controlled by 2 "genes". No subject has such a genome, but during reproduction and crossover, new genetic combination occur and, finally, a subject can inherit a "good gene" from both parents : his paw is now a flipper.

Holland method is especially effective because he not only considered the role of mutation (mutations improve very seldom the algorithms), but he also utilized genetic recombination, (crossover)9 : these recombination, the crossover of partial solutions greatly improve the capability of the algorithm to approach, and eventually find, the optimum.

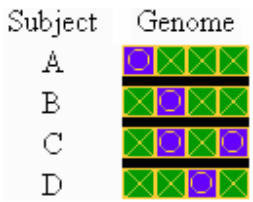
3.3. Functioning of a Genetic Algorithm

As an example, we're going to enter a world of simplified genetic. The "chromosomes" encode a group of linked features. "Genes" encode the activation or deactivation of a feature.

Let us examine the global genetic pool of four basilosaurus belonging to this world. We will consider the "chromosomes" which encode the length of anterior members. The length of the "paw" and the length of the "fingers" are encoded by four genes : the first two encode the "paw" and the other two encode the fingers.

In our representation of the genome, the circle on blue background depict the activation of a feature, the cross on green background depict its deactivation. The ideal genome (short paws and long fingers) is:

The genetic pool of our population is the following one :



We can notice that A and B are the closest to their ancestors ; they've got quite long paws and short fingers. On the contrary, D is close to the optimum, he just needs a small lengthening of his fingers.

This is such a peculiar world that the ability to move is the main criteria of survival and reproduction. No female would easily accept to marry basilosaurus whose paws would look like A's. But they all dream to meet D one day.

We can then see that the principle of genetic algorithms is simple :
Encoding of the problem in a binary string.
Random generation of a population. This one includes a genetic pool representing a group of possible solutions.
Reckoning of a fitness value for each subject. It will directly depend on the distance to the optimum.
Selection of the subjects that will mate according to their share in the population global fitness.
Genomes crossover and mutations.
And then start again from point 3.

The functioning of a genetic algorithm can also be described in reference to genotype (GTYPE) and phenotype (PTYPE) notions10.

Select pairs of GTYPE according to their PTYPE fitness.

Apply the genetic operators (crossover, mutation...) to create new GTYPE.

Develop GTYPE to get the PTYPE of a new generation and start again from 1.

Crossover is the basis of genetic algorithms, there is nevertheless other operators like mutation. In fact, the desired solution may happen not to be present inside a given genetic pool, even a large one. Mutations allow the emergence of new genetic configurations which, by widening the pool improve the chances to find the optimal solution. Other operators like inversion are also possible, but we won't deal with them here.

3.4. Adaptation and Selection : the scaling problem

We saw before that in a genetic algorithm, the probability of reproduction directly depends on the fitness of each subject. We simulate that way the adaptive pressure of the environment.

The use of this method nevertheless set two types of problems :

A "super-subject" being too often selected the whole population tends to converge towards his genome. The diversity of the genetic pool is then too reduced to allow the genetic algorithm to progress.

With the progression of the genetic algorithm, the differences between fitness are reduced. The best ones then get quite the same selection probability as the others and the genetic algorithm stops progressing.

In order to palliate these problems, it's possible to transform the fitness values. Here are the four main methods :

1- Windowing : For each subject, reduce its fitness by the fitness of the worse subject. This permits to strengthen the strongest subject and to obtain a zero based distribution.

2- Exponential : This method, proposed by S.R. Ladd11, consists in taking the square roots of the fitness plus one. This permits to reduce the influence of the strongest subjects.

3- Linear Transformation : Apply a linear transformation to each fitness, i.e. $f' = a.f + b$. The strongest subjects are once again reduced.

4- Linear normalization : Fitness are linearized. For example over a population of 10 subjects, the first will get 100, the second 90, 80 ... The last will get 10. You then avoid the constraint of direct reckoning. Even if the differences between the subjects are very strong, or weak, the difference between probabilities of reproduction only depends on the ranking of the subjects.

To illustrate these methods, let's consider a population of four subjects to check the effect of scaling. For each subject, we give the fitness and the corresponding selection probability.

Subjects	1	2	3	4
Rough Fitness	50/50%	25/25%	15/15%	10/10%
Windowing	40/ 66.7%	15/25%	5/8.3%	0/0%
Exponential	7.14/ 36.5%	5.1/ 26.1%	4.0/ 20.5%	3.32/ 16.9%
Linear transfo.	53.3/ 44.4%	33.3/ 27.8%	20/16.7	13.3/ 11.1%
Linear normalization	40/40%	30/30%	20/20%	10/10%

Windowing eliminates the weakest subject - the probability comes to zero - and stimulates the strongest ones (the best one jumps from 50 % to 67 %).

Exponential flattens the distribution. It's very useful when a super-subject induces an excessively fast convergence.

Linear transformation plays slightly the same role than exponential.

At last, linear normalization is neutral towards the distribution of the fitness and only depends on the ranking. It avoids as well super-subjects as a too homogeneous distribution.

3.5. Conclusion

Genetic algorithms are original systems based on the supposed functioning of the Living12.
The method is very different from classical optimization algorithms13.
Use of the encoding of the parameters, not the parameters themselves.
Work on a population of points, not a unique one.
Use the only values of the function to optimize, not their derived function or other auxiliary knowledge.
Use probabilistic transition function not determinist ones.

It's important to understand that the functioning of such an algorithm does not guarantee success. We are in a stochastic system and a genetic pool may be too far from the solution, or for example, a too fast convergence may halt the process of evolution. These algorithms are nevertheless extremely efficient, and are used in fields as diverse as stock exchange, production scheduling or programming of assembly robots in the automotive industry.

4. The developed GA for TSP

According to this diploma practice I had to execute the next tasks:
1) develop new GA for TSP (4.1);
2) check the algorithm for 6 cities which already has the optimal defined tour and to find if the algorithm works correctly and if it needs to correct it (4.2);
3) apply the algorithm for the task of "52 real towns' location in Berlin" (4.3);
4) give the examples of application of TSP in real world fields (given in 2.3).

4.1. GA description

The amount of cities is *N*.
The number of city is *i (*will call it simply *city).*
The number of current population is *g*.

GENE: *i;*
CHROMOSOME: *g + 1;* at every iteration we take the best chromosome (the shortest tour) of last population and copy it to the first chromosome of the next population.
POPULATION: *g + 1;* then add the next city (its number is the number of the generation) there and copy it to the next chromosome of the next population. But there exchange the *last* gene with its (*last -1*) friend. And all this copy to the next chromosome. Then exchange (*last -1*) gene with (*last -2*) one. And vice versa until exchanging the *second* gene with the *first* one. With such way we create a new population.
SELECTION: The shortest tour of the population
GENERATIONS: *N*

After creating N generations we continue predefined number times:
CROSSOVER: 1 point crossover (but here we get a new chromosome by exchanging the halves of the given one). We crossover 50% best chromosome and create the same number of new ones.
MUTATION: Exchanging two random genes for one chromosome. Mutate all chromosomes.
SELECTION: The shortest tour of the population.

4.2. 6 cities

The algorithm was checked using optimal tour for 6 cities. For any 6 cities it performs. So it neededn't to correct my algorithm.

4.3. "52 real towns' location in Berlin"

And I was advised (by my diploma leader professor Akira Imada) to take these coordinates from the web-site page:
http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95.

There is list of files with real coordinates of different geographical places of the world (for internet download). According to my diploma task I had to take the file *berlin52.tsp* which has the coordinates of 52 towns' location of Berlin (see figure 1).

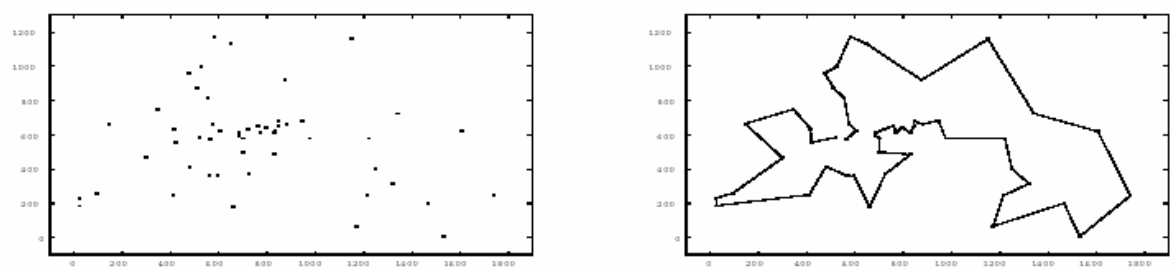


Figure 1: An example of 52 real towns' location in Berlin. Ploted with the data taken from http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95.

And to check my results there is the file berlin52.tour on the same web-page which stores the optimal tour for the coordinates for file berlin52.tsp.

So here the situation is the next: it performs only 80-90% correctly.

5. Conclusions

6. References

1 The main paradigm of Economy (neo-classical) is largely just a wonderful ode to optimization mathematics.

2 That's what the recent "non linearity" revolution learned us.

3- S.J. Gould et al., Le livre de la vie, Seuil, Science ouverte, 1993, pp.186 ss.

4- It is said that basilosaurus reproduced on earth. In that case, posterior members were useful. Harrison R, Bryden M.M. dir., Baleines, dauphins et marsouins. Bordas, 1989.

5- This a very common phenomenon. In that case, we could also speak of the Ichthyosaure, marine reptile of the Mesozoic era whose morphology was closed to shark and dolphin.

6- It's important to understand that Darwinian process doesn't have to lead to optimum. It improves fitness but has nothing to do with optimum.

7- Holland J.H., Adaptation in natural and artificial system, Ann Arbor, The University of Michigan Press, 1975.

8- Goldberg D., Genetic Algorithms, Addison Wesley, 1988.

9- Emmeche C., Garden in the Machine. The Emerging Science of Artificial Life, Princeton University Press, 1994, pp. 114 ss.

10- Heudin J.C., La Vie Artificielle, Hermes, 1994, pp. 91 ss.

11- S.R. Ladd, Genetic Algorithm in C++, 1999-2000. Downloadable book. http://www.coyotegulch.com.

12- "Biological programming" is not limited to AG, another well-known case is neural networks.

13- Goldberg D, idem, pp. 8 ss.