Hybrid Genetic Algorithm with Exact Techniques applied to TSP

Carlos Alberto Rossel Jahuira
Systems Engineering, University of San Agustin
Arequipa – Peru

Ernesto Cuadros Vargas
ICMC, Universidade de São Paulo
Sao Carlos-SP- Brazil

Abstract

This paper presents the hybridization of the Genetic Algorithm (GA) by using ideas enclosed in exact techniques, like the Branch and Bound (B&B), Minimal Spanning Tree and Backtracking Algorithms. It also has been used the Divide and Conquer principle. Two crossovers operators were proposed to propagate the good schema. These operators together with mutation and one method to generate the initial population are based on the algorithms mentioned above.
This Hybrid GA was applied to the well-known combinatorial optimization problem Traveling Salesman Problem (TSP). In almost all cases, the optimal solution was found in few generations with quite a few individuals.
The Hybrid GA was implemented in software by using Visual C++ 6.0 together with VTK library (tool for scientific visualization). VTK was used to build a visual environment which allows to see how is working the Hybrid GA.

1 Introduction

Genetic Algorithms (GAs) have been used successfully in optimization problems. However, this technique doesn't have a good performance as much as another techniques oriented to particular problems, so, a way to save this problem is by the combination of several techniques. There has been proposed one method to generate the initial population, two crossover operators and one mutation operator. The method to generate the initial population is based on the Minimal Spanning Tree philosophy. First crossover operator is based on the Backtracking, and B&B algorithms; the second, on the Minimal Spanning Tree idea; and mutation, on the Minimal Spanning Tree idea, and the Divide and Conquer principle. Both crossover operators allow propagate good schema. Selection operator is based on roulette wheel.

2 Literature Overview

2.1 Genetics Algorithms

GAs are search algorithms based on the mechanics of natural selection and natural genetics [1]. A GA works over a population. Each individual of the population represents a solution to the problem that we want to solve. Each individual should be represented as a bit string (0's and 1's). This string is called c*hromosome. Fitness* is a value that represents how good is a solution. The *function fitness* is the function we want to optimize and it allows calculating the fitness for each individual. The best solutions will have a high fitness, so that they will have a higher probability to survive than bad individuals. A simple GA is composed by 3 operators: reproduction, crossover and mutation. The reproduction operator is an artificial version of selection natural: the fittest individuals will have the highest probability to survive. This operator may be implemented in algorithm form in a several ways. One of them is the roulette wheel [1]. Crossover consists in the information exchange between 2 individuals to create offspring. Mutation consists in the modification of the encoded information in the chromosome. Both, crossover and mutation operators, are driven by rates that indicate how often this operators are applied.

2.2 The Traveling Salesman Problem (TSP)

A salesman is required to visit each of n given cities once and only once, starting from any city and returning to the original place of departure. What tour should he choose in order to minimize his total travel distance? The distances between any pair of cities are assumed to be known by the salesman.

The Traveling Salesman Problem is one of the most widely studied problems in combinatorial optimization [9]. The problem is easy to state, but hard to solve. Mathematically, the problem may be stated as follows:
Given a 'cost matrix' $C = (c_{ij})$, where c_{ij} represents the cost of going from city i to city j, (i, j = 1,..., n), find a permutation $(i_1, i_2, i_3, ..., i_n)$ of the integers from 1 through n that minimizes the quantity $c_{i_1 i_2} + c_{i_2 i_3} + ... + c_{i_n i_1}$.
At first look it seems easy to solve, but a deeper analysis reveals the search space is a permutations set of n cities, so that, its size is *n!*. The number of possible solutions to the symmetric TSP is *(n-1)! / 2*.

2.3 Hybridization

When problem-specific information exists, it may be advantageous to consider a GA Hybrid. GAs may be crossed with various problem-specific search techniques to form a hybrid that exploits the global perspective of the GA and the convergence of the problem-specific technique [1]. GAs have a lower performance than other techniques oriented to a specific problem. To save this inconvenient is recommended to apply the tactic of the *hybridization*, where other techniques are incorporated ([2], [3] and [4]).

2.4 Branch and Bound Algorithm (B&B)

In some problems, it is possible to reduce the search by excluding parts of the search space which cannot give better than the current best known solution. This is the idea behind branch and bound. It is usually used for finding the global optimum. It is only applicable to problems with a sufficiently well defined structure. Because it finds the global optimum it has an exponential time complexity on *NP*-hard problem and is thus feasible only for "small instances". In some cases B&B can be used in conjunction with other heuristic search mechanisms which provide a strong initial bound [5].

2.5 Minimal Spanning Tree (MST)

You desire to create a path net to connect a number of towns. Due to budget limitations, the total distance of paths to build must be the least that allows the direct or indirect connection of traffic between different towns.
The algorithm needs to start with any node and to connect it with the nearest node. These two nodes forms a *connected set C* and the others nodes forms the *unconnected set* $\hat{C}$. Then, a node is chosen from the unconnected set, which is the nearest (with the shortest branch length) to any node from connected set. The chosen node is eliminated from $\hat{C}$ and it is joined to *C*. This process repeats until the unconnected set is empty [6].

2.6 Backtracking

It is a method of solving combinatorial problems by means of an algorithm which is allowed to run forward until a dead end is reached, at which point previous steps are retraced and the algorithm is allowed to run forward again. Backtracking can greatly reduce the amount of work in an exhaustive search [7].

2.7 Divide and Conquer Principle

Maybe the most important technique applied to design efficient algorithms is the strategy called Divide and Conquer, which consists in the decomposition of a problem of size n in smaller problems, so that with the solution of every problem is possible to build easily one solution to the whole problem [8].

2.8 Genetic Algorithms to the TSP

Many operators have been proposed for the TSP. Some of them are: Partially Matched Crossover (PMX) [1], Edge Recombination Crossover (ERX) [10], Distance Preserving Crossover (DPX) [11], Edge Assembly Crossover (EAX) [12]. It is possible to find a comparison between different operators in [13].
These algorithms have been tested with several TSP instances. Some of the them with instances up hundreds of cities from the TSPLIB[14]. One fact that was found in many algorithms is that they use too many individuals and large number of generations is required in order to find a good or optimal solution. The problem with this is that we required more memory. In [15] a Hybrid GA with B&B Technique was proposed in order to get a better performance. It was used to solve instances up to 100 cities.
Most works on GAs aim to mix this technique with another heuristics. On the other side in this paper we will see how it is possible to mix different some exacts techniques with GAs.

3 Hybrid Genetic Algorithm

3.1 Encoding the solutions

To encode a valid tour it was used the path representation [1], e.g.: the tour 5-1-6-4-2-3 is represented by the vector (5,1,6,4,2,3). Where each number represents a node or city and the sequence, the order in which must be traveled over.

3.2 Initial population

To initialize the population the Minimal Spanning Tree idea was used. The method implemented differs from the traditional algorithm in one thing: when we add a new node to the connected set we just consider a node which is the nearest to the initial or end node of the connected set. In Fig. 1.a we have to connect 7 nodes. We start with any node, for example node 2. Now we choose the nearest node to 2. It is node 3. These two nodes form the connected set. The remaining nodes form the unconnected set. Now we look for the nearest node to the extreme nodes of the partial tour, in this case nodes 2 and 3. Node 4 is the nearest to node 3. We add it to connected set. In Fig. 1.c we can see new partial tour, where the extreme nodes are 2 and 4. We continue until all nodes are in the connected set.

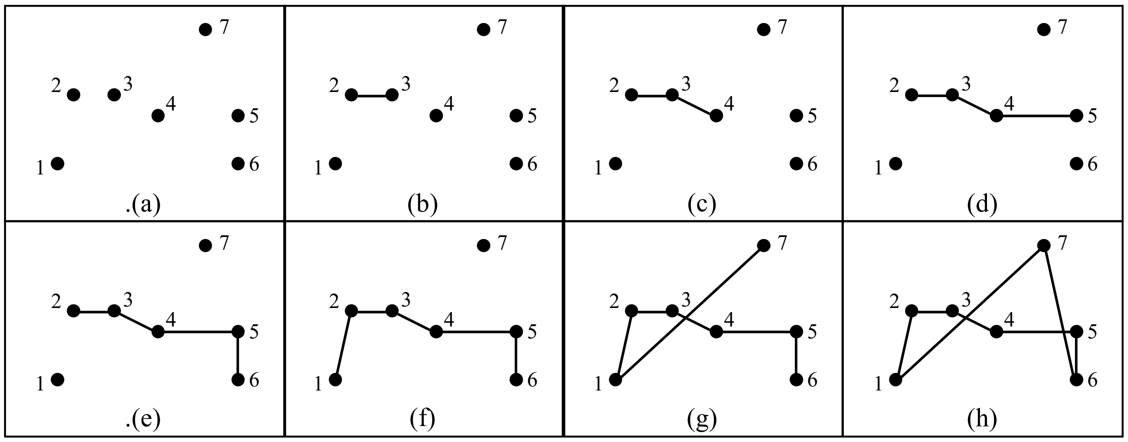

Fig. 1. Method to create an individual

Finally, when we have connected all nodes, we link the first with the last node of the partial tour. We can appreciate this in Fig. 1.h. Although this is not the best solution, it is much better than a solution formed randomly by a permutation. From now on, we will refer to this method as Population Generator based on MST idea (PGMST).

3.3 Hybrid GA Operators

3.3.1 Selection Operator

Consider a population with 5 individuals. We choose the shortest tour length. Then, we divide it by each tour length and the result will be the fitness of an individual. This is shown in Table 1.

Table 1. Selection Method based on Roulette Wheel

Individual (tour)	Length	Fitness	$\sum$ Fitness
1	50	50/50 = 1.00	1.00
2	80	50/80 = 0.63	1.63
3	90	50/90 = 0.55	2.18
4	70	50/70 = 0.71	2.89
5	100	50/100 = 0.50	3.39

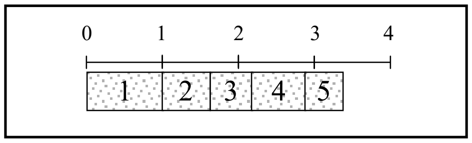

Fig. 2. Probabilities for each individual

In Fig. 2 probabilities for each individual from Table 1 are shown graphically. The first individual is the best and that's why it has the highest fitness, so it will have the highest probability to survive when parents are selected randomly.

3.3.2 Crossover Operators

There have been implemented two kinds of crossover operators. Both have in common the use idea of the schemata theory: Good schema will survive in the next generations. Therefore, having this idea in mind we choose 2 parents and detect what parts of the string are common. Once we have all common segments in both parents, we eliminate them together with their intermediate nodes. Then, we apply to the remaining nodes the method PGMST.

Figures 3.a and 3.b shows 2 possible solutions to a TSP instance of size 7. This first step is to identify the common segments in both parents. In this case there is only one common segment which is 1-2-3-4, see figures 3.c and 3.d. Now we delete the common segment's edges and fusion the remaining edges from both parents into one graph. This can be appreciated in Fig. 3.e. It also has been added a new edge which links the first node with the last node from the common segment. Now we look for a new tour considering only the parent's edges. For this purpose, Backtracking, and B&B Algorithms have been used. We can see a new tour in Fig. 3.f (thick lines). Finally, we add the common segment, which was deleted, to the new tour, as we can see in Fig. 3.g

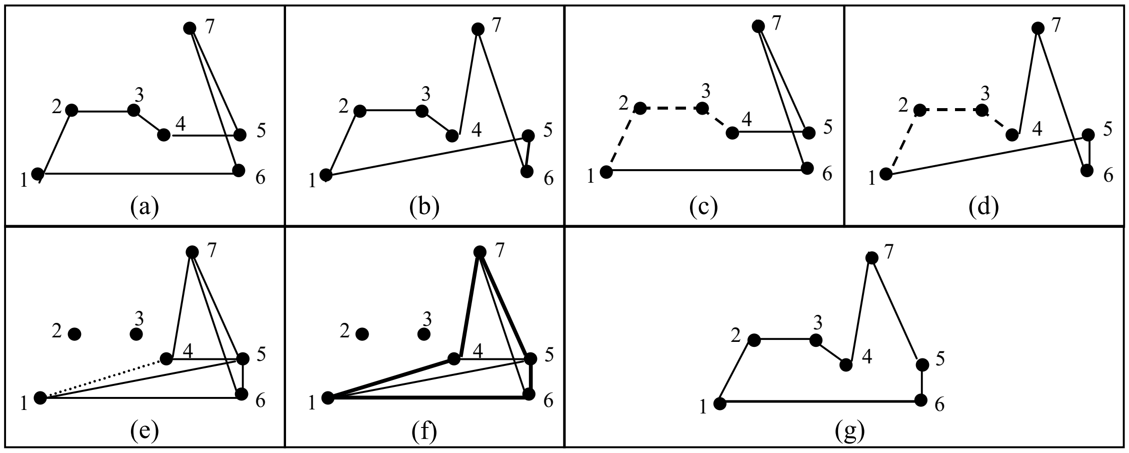

Fig. 3. First Crossover Operator

The second crossover operator also detects the common segments in both parents, but the child will not be created considering only parent's edges. Suppose we want to apply this operator to the parents of the last example. There is only one common segment which is formed by nodes 1, 2, 3 and 4. We delete the intermediate nodes of the segment which are nodes 2 and 3. Now we link node 1 to node 4 and delete all parent's edges. It is shown in Fig. 4, where nodes 2 and 3 appear in gray.

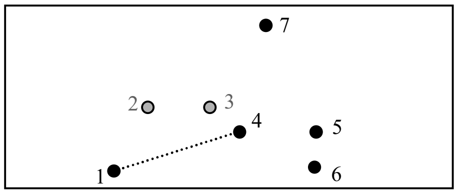

Fig. 4. Second Crossover Operator

Now we apply the method PGMST to the remaining nodes. These nodes are 1, 4, 5, 6 and 7. When a tour is found, the edge that links the node 1 and 4 is replaced by the common segment.

In both operators is important to realize that they always look for common segments. Then, they are replaced by an edge which links the first to the last node of each segment. These segments will always be present in the child. Therefore, when 2 good parents are crossed their child will inherit the "good schema".

The first operator only looks for new children considering the parent's edges. However, there is the possibility to add just one new edge. The second considers the possibility to add many new edges. It provides diversity in order to do a global search.

3.3.3 Mutation Operator

What this operator makes is to reorder the chromosome structure. This operator takes a chromosome, divide it into fixed length segments and then reorders each segment in order to find a segment which represents a smaller path length. In Fig. 5.a we have an individual, who has been selected for the process of mutation. We choose an integer

number randomly. Suppose 7 is the selected number. Now we divide the chromosome into parts of size 7. The chromosome only have 11 nodes, so that, we only have one segment of size 7. We choose any node as initial node. For example, node 4. Now we choose our segment of size 7. It may be 4-3-2-1-5-6-10 or 4-7-9-8-11-10-6. Let's take the last segment. This is the segment formed by the edges a, b, c, d, e and f. Now we delete the remaining nodes and all edges, see Fig. 5.c. Then, we push in a vector the initial and last node of the segment they are nodes 4 and 6, respectively. The next step is to apply the method PGMST. We choose the nearest node to 4 or 6 and add it to the vector. The nearest is node 11, which is close to node 6. We add it; this is shown in Fig. 5.d. Now we search for the nearest node to node 4 or node 11. The nearest is 7. We add it, see Fig. 5.e. We continue this process. In Fig. 5.f, we can see how the vector is changing until all nodes have been connected. The tour completed is shown in Fig. 5.g, but this not the entire chromosome. In Fig. 5.h, we can see that chromosome have been completed. The segment that was deleted at the beginning of the process has replaced the edge that linked node 4 and 6.
From now on, we will refer to the first crossover operator as B&BBX (Branch & Bound-Backtracking Crossover) and to the second as MSTX (Minimal Spanning Tree Crossover).

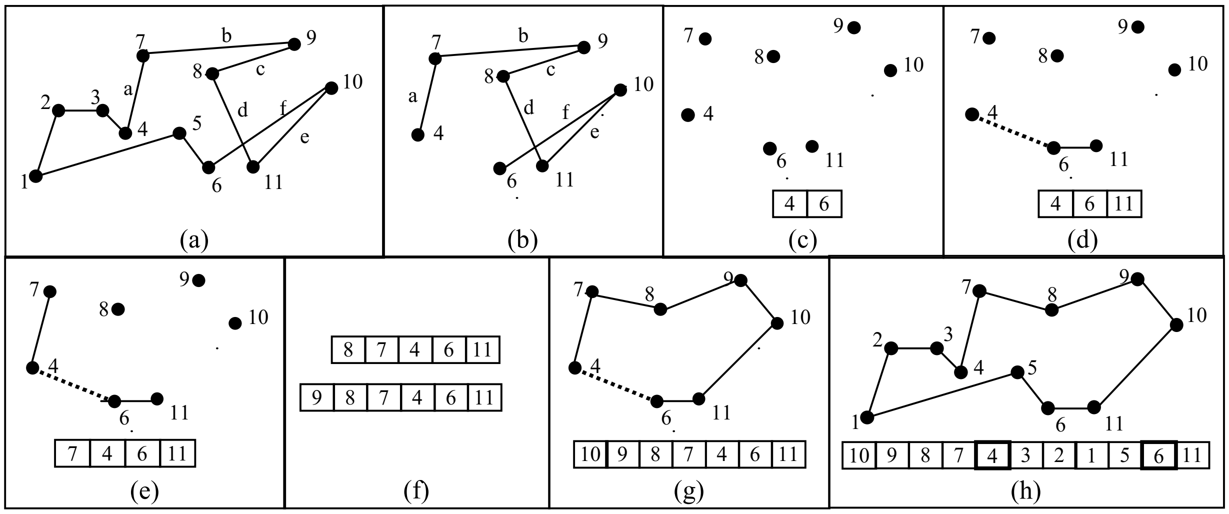

Fig. 5. Mutation Process

4 Results

Trials realized so far include several TSP instances obtained from [10]. Before explaining results, it is important to mention that experiments have not finished yet. Since there are several parameters that drive the operators performance, more trials are required in order to realize a detailed comparison with another genetic algorithms. However, it is possible to mention some important results.
All trials have been carried out on simple PC Pentium 100MHz with 40MB RAM. Hybrid GA has been tested with instances up to 225 cities.
Applying only B&BBX it was possible to find the optimal solution or with an error rate of 0.3% for instances up to 105 cities. Performance decreased for larger problems because of backtracking. With MSTX optimal solutions or with an error rate of 0.4% were found.
Elitism was an important factor. The best individual of a generation was copied to the next generation. Besides, new individuals were inserted into population periodically in order to provide diversity. Working both crossover operators at the same time made possible to get a better performance. B&BBX was applied with a probability 0.4 and MSTX, with 0.6. Mutation was always applied to best individual of each generation. The error rate was 0.17% in the worst case and quite a few individuals were used in each trial obtaining good solutions between 6 and 30 generations for instances up to 105 cities.
In latest trials, only MSTX was used and mutation operator was applied to every individual with probability 0.4, see Table 2. It was noticed that quality of solutions improved because mutation operator was applied to every individual in the population.

Table 2. Some results applying only MSTX and mutation to every individual

Instance	Number of cities	Population size	Number of generations	Crossover operator	Hybrid GA	Best Known Solution	Error (%)
TSP225	225	6	25	MSTX	3935.35	3916	0.49
KroA100	100	15	20	MSTX	21285.44	21282	0.016

Trying trials with different values for each parameter can make possible to get better results. Besides, it's possible to make fits to the application in order to improve the performance. Additionally, let's remember that Hybrid GA run on a simple Pentium 100 MHz, 40MB RAM. Therefore, we can expect better results when we try it on a powerful machine, where Hybrid GA will be tested with larger instances.

To propagate *good schema*, by copying the common segments from parents to a child, worked very well. Experiments showed that in most crossovers (88% of the first three generations), children were better than their parents.

Another authors have tested their algorithms with hundreds of cities. Combining GAs with another heuristics has been most successful approaches. However, in this paper we have shown that combining GAs with exact techniques can give good results as much as another algorithms. Still we can not say how good is the Hybrid GA presented here, because more trials with larger instances are required. However, at this moment we can say this Hybrid GA can give good solutions as much as most successful genetic algorithms for instances up to 225 cities. A good thing about this Hybrid GA is that an optimal or good solution is found in few generations with quite a few individuals.

About the application's interface, it was possible to implement a visual environment which shows how are working the different techniques that were mixed. It also allowed detecting some problems that were not detected before.

5 Conclusions

It is possible to fusion evolutionary and exact techniques within a cooperative framework in order to get a better performance. This kind of approach can give good results as others approaches (GAs with others heuristics).

To apply the B&B technique with adequate criteria to guide the search could be beneficial. It's important to consider advantages and drawbacks of the techniques to use, so that, it's possible to detect all variables which must be managed adequately. It is very important to control the search when exact techniques are being used, because it may reduce the computational effort.

The idea to propagate good schema, by copying the common segments from parents to a child, worked as it was expected. Therefore, in this way it is possible to follow researching new operators that try to propagate "intentionally" good schema. In fact, this is one of the main contributions of this work.

To provide a visual environment, which shows how a Hybrid GA is working, can be too much useful. This can allow detecting problems in specific situations that are not appreciated when we just see the best solution of a generation.

Applying an adequate elitist strategy is very important, because this can avoid falling in optimal local solutions. It's not enough to insert new individuals when rate of identical individuals is high. A good tactic could be to insert new individuals periodically in order to provide diversity to the population. This is good when quite a few individuals are used in trials.

References

1. Goldberg, D. (1989). Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley.
2. Davis, L. (1991). Hand-Book of Genetic Algorithms. New York:Van Nostrand Reinhold.
3. Geyer-Schultz, (1997). Fuzzy Rule-Based Expert Systems and Genetic Machine Learning. Physica-Verlag.
4. Burke, E. , Elliman, D. , & Weare, R. (1995). A Hybrid Genetic Algorithm for Highly Constrained Timetabling Problems. Proceedings of 6th International Conference on Genetic Algorithms (ICGA'95), pp 605-610.
5. Prugel-ennett, A.(2000). Branch and Bound.
http://www.isis.ecs.soton.ac.uk/isystems/evolutionary/notes/evol/Branch_Bound.html
6. Taha, H.(1989). Operations Research, an Introduction. Macmillam Publising, 5a ed.
7. Skiena, S. (1990). Backtracking and Distinct Permutations, Addison-Wesley.
8. Aho, A., Hopcroft, J., & Ullman J. (1983). Data Structures and Algorithms. Addison-Wesley.
9. Lawler, E. , Lenstra, J. , Rinnooy Kan, A. & Shmoys, D. (1985). The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization. Wiley, New York.
10. Whitley, D., Starkweather, T., & Shaner, D. (1991). The Traveling Salesman and Sequence Scheduling Quality Solutions using Genetic Edge Recombination. In Handbook of Genetic Algorithms. New York:Van Nostrand Reinhold.
11. Freisleben, B. & Merz, P. (1996). Genetic Local Search Algorithm for Solving Symmetric and Asymmetric Traveling Salesman Problems. Proceedings of IEEE International Conference on Evolutionary Computation, IEEE-EC 96, IEEE Press, pp. 616-621.
12. Nagata, Y. & S. Kobayashi. (1997). Edge Assembly Crossover: A High-power Genetic Algorithm for the Traveling Salesman Problem. Proceedings of Seventh International Conference on Genetic Algorithms (ICGA'97), East Lansing, Michigan, pp. 450-457.
13. Larrañaga, P., Kuijpers, C. M. H., Murga, R. H., Inza, I. & Dizdarevic, S. (1999). Genetic Algorithms for the Traveling Salesman Problem: A Review of Representations and Operators. Artificial Intelligence Review, pp. 129-170.
14. Reinelt, G. TSPLIB, http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/
15. Cotta, C., Aldana, J. F., Nebro, A. J., & Troya, J. M. (1995). Hybridizing Genetic Algorithms with Branch and Bound Techniques for the Resolution of the TSP. Artificial Neural Nets and Genetic Algorithms, D.W. Pearson, N.C. Steele, R.F. Albrecht (editors), Springer-Verlag, Wien New York, pp. 277-280.