Genetic Neighborhood Search

Juan José Domínguez[1], Sebastián Lozano[2] , Marcos Calle[2]

[1] E. S. Ingeniería, Dpto. Lenguajes y Sistemas Informáticos, University of Cadiz
C/ Chile, nº1, 11003 Cádiz, Spain
juanjose.dominguez@uca.es
[2] E. S. Ingenieros, Dpto. Organización Industrial y Gestión de Empresas, University of Sevilla
Camino de los Descubrimientos, s/n, 41092 Sevilla, Spain
slozano@us.es mcalle@esi.us.es

Abstract. The Genetic Neighborhood Search (GNS) is a hybrid method for combinatorial optimization problems. A main feature of the approach is the iterative use of local search on extended neighborhoods, where the better solution will be the center of a new extended neighborhood. We propose using a genetic algorithm to explore the extended neighborhoods. Computational experiences show that this approach is robust with respect to the starting point and that high quality solutions are obtained in reasonable time.

1 Introduction

Combinatorial optimization deals with the problem of minimizing (or maximizing) a function where the solution set is often represented by a set of decision variables, whose values can have certain ranges [1]. A solution is represented by a value assigned to these variables. To find a globally optimal solution consists in finding a solution between the set of feasible solutions. Mathematically, a problem of combinatorial optimization is defined by the pair *(F, f)* [2]:

$$min\ f(T), \quad T \in F \tag{1}$$

Where F is the set of feasible solutions for the problem, and the function $f(T)$ measure the quality of the solution, such that:

$$f : T \to \Re \tag{2}$$

Many heuristic methods for combinatorial optimization problems are based in local search. The use of a local search algorithm implies the definition of a neighborhood. A neighborhood function N is a mapping,

$$N : F \to 2^F \tag{3}$$

Which defines for each solution $i \in F$, a set $N(i) \subseteq F$ of solutions, called the neighborhood of F, that are in some sense close to i. The meaning of "close to i" is that they can be reached from i by a single *move*. This single-move concept can be considered as a first-order neighborhood while performing several consecutive moves

P.M.A. Sloot et al. (Eds.): ICCS 2002, LNCS 2331, pp. 544–553, 2002.

allows the definition of extended neighborhood, the order of the extended neighborhood corresponding to the number of consecutive moves. A move is an operator, which transforms one solution onto another via small modifications [3]. A solution i is locally optimal with respect to the neighborhood N, if:

$$f(i) \leq f(x), \forall x \in N(i) \quad (4)$$

A local search process can be viewed as the procedure of minimizing the cost function f in a number of successive steps in each of which the current solution i is being replaced by a solution j such that:

$$f(j) < f(i),\ j \in N(i) \quad (5)$$

There are many different ways to conduct local search in the neighborhoods [1]: *Best improvement local search* replaces the current solution with the solution that improves most in cost after searching the whole neighborhood, and *first improvement local search* accepts a better solution when it is found.

The strategy presented with the name *Genetic Neighborhood Search* (GNS) is a new heuristic based in local search but of a special mode, where a Genetic Algorithm (GA) is used for explore the neighborhoods ([3] and [4]).

The rest of this paper is structured as follows. In section 2 the basic GNS algorithm is depicted. In section 3 we give an account of the application of the method to the Symmetric Traveling Salesman Problem. Finally, in section 4, some conclusions are drawn.

2 Genetic Neighborhood Search

The Genetic Neighborhood Search algorithm we propose has the following schema: given a initial solution, x^0, and an extended neighborhood N of order L around x^0, the GA proposed will perform an exploration of this neighborhood. The best solution obtained by the GA will become the center of a new extended neighborhood which will, in turn, be explored using the GA. The GNS is an iterative improvement local search, which pseudocode description follows:

```
Optimization: x^0 × L × A × • → x*
k ← 0
optimum ← LOCAL
Neighborhood ← L
do
   x^(k+1) ← GNS (Neighborhood, x^k)
   if ( Convergence (x^k, x^(k+1), •)) then Neighborhood ← L
   else if (Can_Increase (A))
   then Neighborhood ← Neighborhood + L
   else optimum ← GLOBAL
   k ← k + 1
while  (optimum = LOCAL)
return x^k
```

Note that when a GNS doesn't improve the center of the neighborhood, the order of the neighborhood is increased and this can be done up to A times. The objective is to avoid the local optimum. The maximum order of the extended neighborhood will be:

$$L_{max} = L \times A \tag{6}$$

The order of the neighborhood L is an input parameter of the algorithm. Since it determines the size of the search region, other parameters of the algorithm depend of the value of L. So, the size of the population and the number of generation of the GA are proportional to L:

$$Size\_Pob = L/\sigma$$
$$N\_Gen = Size\_Pob * \theta \tag{7}$$

2.1 Representation

The individuals are coded as a succession of L movements from the center point of the neighborhood. The movements are dependent of the specific combinatorial problem. In the section 3 we show how the individuals can be coded for the Traveling Salesman Problem.

2.2 Fitness

The evaluation of the individual consist in calculate the value of the objective function for each move in case of a maximization problem and minus the objective function for a minimization problem. The better solution in the trajectory of the individual is the value of the fitness. Such that, if $Fitness_{i,j}$ is the value of the objective function for the individual i after the movement j, the fitness of the individual is:

$$Fitness_i = max\left\{Fitness_{i,j},\ j = 1,...,L\right\} \tag{8}$$

Note that the fitness of the individual is not the value in the last movement. Moreover, since the individual represents L possible solutions, the GA population implicitly contains $Size\_Pob \times L$ solutions.

2.3 Operators

The GA employs three operators for mutation and two for crossover. A steady state GA has been implemented. In each iteration either mutation or crossover is applied, so the probability of mutation (P_M) and the probability of crossover (P_C) are $P_M + P_C = 1$.

Each operator for mutation has a probability, such that the sum of these probabilities is P_M. The three operators for mutation are:

- *Exchange Mutation* consists in to exchange the position of some pairs of movements. The number of exchange is $L \times P_{EM}$, where P_{EM} is the probability of this operator.
- *Modify Mutation* consists in alter the value of some movements. The number of modify is $L \times P_{MM}$, where P_{MM} is the probability of this operator.
- *Mutation Directed by Fitness* consists in applying the Modify Mutation but only to the movements following the move that defines the fitness of the individual. The number of mutation is $L \times P_{MDF}$, where P_{MDF} is the probability of this operator.

Each operator for crossover has a probability, such that the sum of these probabilities is P_C. The operators for crossover are:

- *Uniform Crossover* consists in to generate a new individual with the movements, select randomly, of the two parents.
- *Crossover Directed by Fitness* consists in exchanging only the movements following the move that defines the fitness of the individual. Only the better child is introduced in the new population. The figure 1 shows this operator, where the gray zones are the moves prior to the one that defines the fitness.

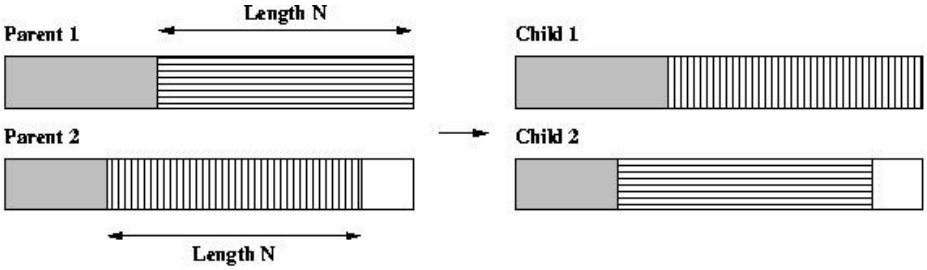


Fig. 1. The Crossover Directed by Fitness.

2.4 Selection

The operators for mutation and crossover require the selection of the parents, and the individual to be deleted from the population. In both operators (crossover or mutation), the selection of individual for replacement is fitness-based, where the individuals with worse fitness have a bigger probability of selection than the individuals with better fitness.

In the case of mutation, the individual necessary for the operator is randomly selected, i.e. the selection is not fitness-based. In the crossover, the two progenitors are selected proportional to their fitness, namely using the roulette wheel procedure [4], where the individuals with worse fitness have a smaller probability of selection than the individuals with better fitness.

2.5 The Genetic Algorithm

The pseudocode of GA employ for the exploration of a neighborhood is the following:

```
GNS: S_POP × P_C × P_CU × P_M × P_MI × P_MA × G × L × A × x^0 → x*
t ← 0
P_t ← Initialize_Population (S_POB, L, x^0)
while (t < G)
do
   t ← t + 1
   op ← Select_Operator (P_C, P_CU, P_M, P_MI, P_MA)
   [ind1, ind2 ] ← Select_Individual (P_t, op)
   new_ind ← Apply_Operator (P_t, op, ind1, ind2)
   old_ind ← Select_Individual_Deletion (P_t)
   Insert_Individual (P_t, old_ind, new_ind)
done
return Best_Individual (P_t)
```

The stopping criterion is the maximum number of generations or a consecutive number of generations without obtaining a better solution that the center of neighborhood, whichever occurs first.

3 Application to the Symmetric Traveling Salesman Problem

There are many variations to the Traveling Salesman Problem (TSP)[6]. In this report, we consider the classic Symmetric TSP. The problem is defined by N cities and a symmetric matrix D, of dimensions $N \times N$, where $d_{ij}=d_{ji}$ gives the distance between any two cities i and j, $i,j=1,...,N$. The goal in TSP is to find a tour of minimal length which visits each city exactly once [1]. That is, the objective is to find a cyclic permutation μ on the N cities, such that minimize the cost of the tour:

$$F(\mu) = \sum_{i=1}^{N-1} d(c_{\mu(i)}, c_{\mu(i+1)}) + d(c_{\mu(n)}, c_{\mu(1)}) \qquad (9)$$

3.1 The 2-exchange move

In section 2.1 we explain that the individuals on GNS are a succession of L movements. In the TSP, the move employed is the *2-exchange*. It consists in randomly select two cities, P and Q, in such a way as the arcs $(P,\Pi(P))$ and $(Q,\Pi(Q))$ are deleted, and the new arcs (P,Q) and $(\Pi(P),\Pi(Q))$ are introducing. The figure 2 shows this movement.

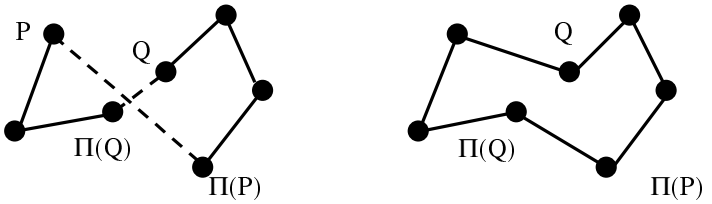


Fig. 2. The 2-exchange move.

The coding of this type of movement is a vector of two components, which represent the two arcs to delete. Each arc is coded with a vector of two elements: the origin city and the direction of the arc. Although the Symmetric TSP the tour has not a direction, for the codification of individuals the tour has a direction. So, the code for the direction of an arc has a value of 1 if it's the same direction, or 0 in other case. The figure 3 shows a tour, where the direction is clockwise. In this example, the arc between the cities 3 and 7 is coded *[3,1]*, while that the arc between the cities 3 and 2 is *[3,0]*.

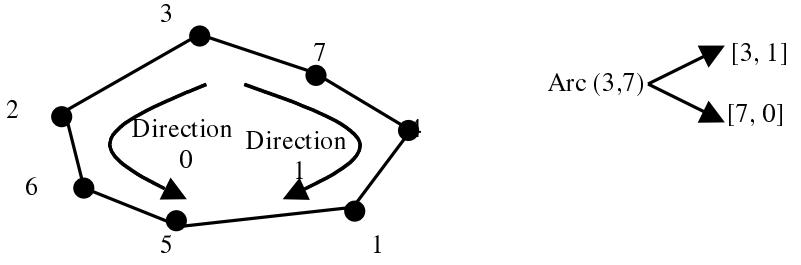


Fig. 3. The coded of an arc in a movement

The problem of this representation is that an arc can have two codes. The solution consists in using the code involving the lower-labeled city. In the example of the figure 3, the arc between the cities 3 and 7, with coded *[3,1]* and *[7,0]*, is only represented by *[3,1]*.

3.2 Fitness of the individuals

Each movement of an individual has a cost due to the two arcs deleted and the two new arcs. So, the individual *i* has an increment in the objective function due to the movement *j*:

$$\Delta_{ij} = d_{p,q} + d_{\Pi(p),\Pi(q)} - d_{p,\Pi(p)} - d_{q,\Pi(q)} \quad \textbf{(10)}$$

The value of the individual i in the k GNS associated to the movement j is, where the center individual of the neighborhood is x^k:

$$Fitness_{i,j} = F(x^k) + \sum_{h=1}^{j-1} \Delta_{ih} + \Delta_{ij} \tag{11}$$

The fitness of the individual is the best value in the L movements:

$$Fitness_i = max\{Fitness_{i,j},\ j = 1,...,L\} \tag{12}$$

3.3 Computational experiences

To test the performance of the GNS, the TSPLIB [7] was used. All the tests have been run on a 450 MHz PC Pentium III. What we report here are the results obtained with the proposed approach for some of the problems tested. A thorough study to benchmark GNS will be the subject of another paper.

Figure 4 shows the results obtained by the GNS in 100 runs performed with uniform probability for each operator and an order of the neighborhood L=15, with different values of the population size, for the ST70 problem (with 70 cities). The quality of the solution is measured as the percentage error over the optimal solution:

$$\frac{Solution\_Cost - Optimal\_Cost}{Optimal\_Cost} \times 100 \tag{13}$$

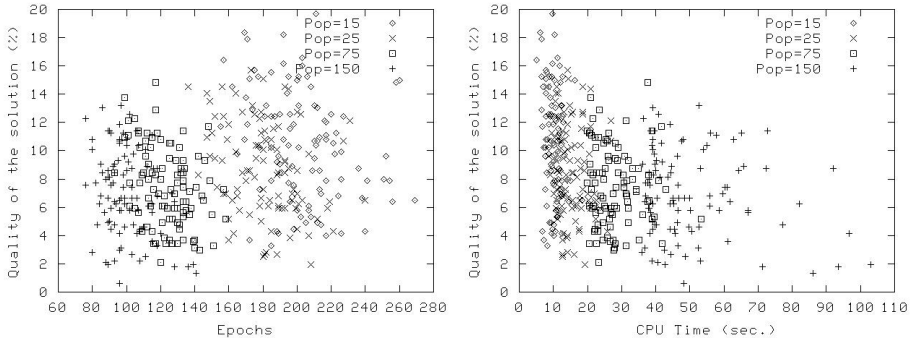
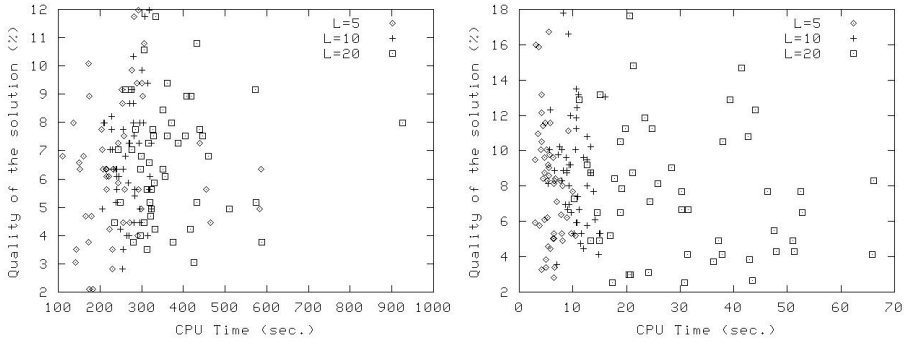


Fig. 4. Epochs and CPU time vs quality of solution for different size of population, ST70 problem.

Note that a small population size requires more epochs than a larger size. Obviously, a large population requires more computing time because it needs to evaluate more points and more generations (equation 7). However, a rise in the computing time doesn't always translate into better solutions unless an effective trade-off between intensification and diversification is maintained. The computational experiences show that the optimum population size is around 15-20% of number of cities.

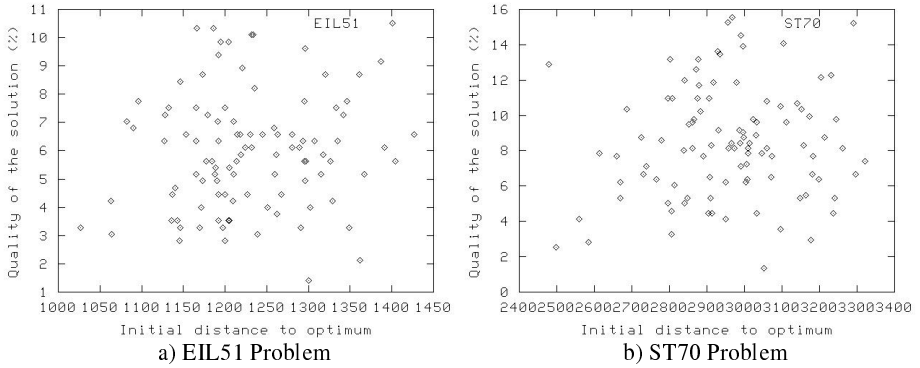
The order L of the extended neighborhood is the main parameter of the algorithm. Figure 5 shows, for problems EIL51 and ST70 (51 and 70 cities, respectively), the results obtained by GNS for fifty runs carried out, using uniform probability for each operator, a population size of 20% of number of cities and different values of the order of the neighborhood.



a) EIL51 Problem b) ST70 Problem

Fig. 5. CPU time vs quality solution by different order of neighborhood.

Note that a large neighborhood order seems to lead to better solutions than a smaller neighborhood order. This may be due to GNS with a large neighborhood order neighborhood more effectively avoiding getting trapped in local optima. However, a small order of neighborhood has a lower computing time. It is thus necessary to reach a trade-off between computing time and quality of the solution. The computational experiences show that the optimal neighborhood order is between 10 and 15, allowing for at most 6 or 10 rises.

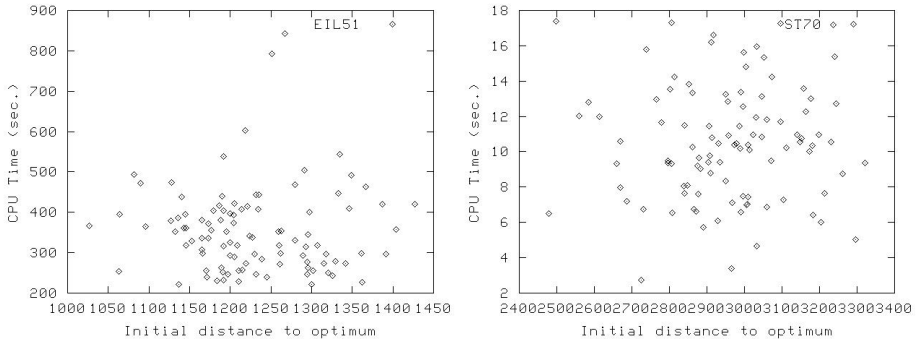


a) EIL51 Problem b) ST70 Problem

Fig. 6. Initial distance to optimum vs quality solution for 100 random starting point.

The proposed algorithm depends on the initial solution or starting tour. Figure 6 show the quality of solution obtained with GNS for EIL51 and ST70 starting from 100 random tours and the figure 7 shows the CPU time versus the initial distance to the optimum. The population size used is 20% of number of cities (i.e. 10 and 15 respectively). The neighborhood order is 10 and the probabilities of the different

operators are uniform. Note that, independently of the starting point, GNS reaches a good approximate solution to the optimum. Similarly, the CPU time required, does not depend too much on the starting point.



a) EIL51 Problem b) ST70 Problem

Fig. 7. Initial distance to optimum vs CPU time for 100 random starting point.

Table 1 shows the results obtained with some problems of TSPLIB for 100 runs. The probabilities employed are P_C=0.8 (P_{CDF}=0.64 y P_{UC}=0.16) y P_M=0.2 (P_{EM}=0.04, P_{MM}=0.04 y P_{MDF}=0.12). The size of the population is fixed to 20% of the number of cities, except in KROA200 that it is the 15%.

Table 1. Problem TSPLIB computational results.

Problem	No. Iterations	CPU Time	Average error (%)	Population size	Neighborhood order
EIL51	147	9.7	3.2	10	5
ST70	169	11.2	5.6	15	10
KROA200	1057	249.2	7.8	30	10
PCB442	1213	523.5	13.5	80	15

4 Conclusions

This paper deals with a new approach to combinatorial optimization, consisting in performing successive extended local searches using a Genetic Algorithm. Genetic Neighborhood Search explores an extended neighborhood of a determinate order (L>1) and considering the best solution obtained by the GA as the center of a new extended neighborhood. The originality of the method is the exploration of higher-order neighborhoods using a GA and the evaluation of the fitness of the individuals as the best solution found along the L-move trajectory. Both the codification of the moves as well as the mutation and crossover operators depend on the problem being solved. The experiments carried out for the symmetric TSP show that GNS is robust

with respect the starting point and generally obtain solutions of good quality in a reasonable CPU time.

References

1. Aarts, E., Lenstra, J.K.: *Local Search in Combinatorial Optimization*. John Wiley & Sons, Ltd. (1997)
2. Michalewicz, Z., Fogel, D.B.: *How to solve it: Modern Heuristics*. Springer-Verlag, Berlin Heidelberg New York. (1998)
3. Voudouris, C., Tsang, E.: *Guided Local Search*. Technical Report CSM-247, Dep. of Computer Science, University of Essex. (1995)
4. Goldberg, D.E.: *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, Reading. (1989)
5. Michalewicz, Z.: *Genetic Algorithms + Data Structures = Evolution Programs*. 3rd edn. Springer-Verlag, Berlin Heidelberg New York. (1996)
6. Lawler, E.L., Lenstra, L.K., Rinnooy Kan, A.H.G., Shmoys, D.B. (Eds.): *The Travelling Salesman Problem: A guided tour in combinatorial optimization*. John Wiley & Sons. (1985)
7. Reinelt, G.:TSPLIB 95. http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95 (1995)