A Competitive-Cooperative Approach to Complex Combinatorial Search

Michael Norman

Department of Physics, University of Edinburgh, Mayfield Road.
Edinburgh, EH9 3JZ. Scotland.

Pablo Moscato

Caltech Concurrent Computation Program 158-79.
California Institute of Technology.
Pasadena, CA 91125, U.S.A.

and

CeTAD
Universidad Nacional de La Plata
C.C. 75
1900, La Plata
ARGENTINA

ABSTRACT

Combinatorial optimization problems confront us with the problem of searching in a huge configuration space. Associated with each of these configurations is the value of a *utility* function and the optimization task generally consists of finding the configuration which has either the lowest or the highest cost of all. Among the most difficult of these problems are those which belong to the NP-complete class for which, it is conjectured, there does not exist a polynomial-time algorithm that can give optimal solutions. We have designed a Competitive-Cooperative Search to deal with these problems. A number of distinguishable individuals, arranged in a certain topology, compete and cooperate with the other individuals during the search. The method uses techniques from Simulated Annealing and Genetic Algorithms and is inspired by analogies with evolving systems, both social and biological. Its performance is being tested on large Traveling Salesman Problem instances, like the Lin-Kernighan 318-cities, for which the optimal tour is known, among the

10^{655} tours that compose the configuration space. The approach shows natural parallelism in which competition and cooperation between individuals enhances its performance. It is therefore an excellent method for implementation on parallel computers. Owing to the surprisingly good results obtained during the testing, and the generality of the approach, it promises excellent results in other scientific and technological optimization problems.

P.A.C.S. numbers 05.20, 02.50, 87.10

1 Introduction

Combinatorial optimization consists of search within a huge space of configurations. Associated with each configuration is the value of a utility function and the task consists of finding configurations that either maximises or minimises this value. Finding the best configuration of all is almost impossible in many optimization problems. Indeed there is a class of combinatorial optimization problems which are called NP-Complete and which, it is conjectured, can not be solved by any polynomial-time algorithm. In addition to the 300 problems presented by Garey and Johnson [8], the task of finding the ground state of a three-dimensional spin-glass has been proved to belong to the NP-Complete class [2] [1]. More recently, many tasks related to learning in neural networks have been proved to be NP-complete [14].

In 1983, Kirkpatrick, Gelatti and Vecchi presented an algorithm called *Simulated Annealing* (SA) for dealing with these problems. They had previously developed the method in the framework of statistical mechanics [15]. The basis of their algorithm is that it allows movement in the configuration space and that changes occur from one configuration to another according to the probability $p = min(1, e^{-\Delta C/T})$ where ΔC is the difference in the cost function between the two configurations and T is an external control parameter regarded as a *"temperature"* which is decreased through the simulation. The initial value of T is initialized according to a start criterion and many *ad hoc* procedures are used for the initialization and decrement schedule.

Simulated Annealing algorithms are generally composed of two basic parts: the inner and outer loop. In pseudo code, the algorithm is something like the piece of pseudocode depicted in Fig. 1. We tried to put a SA algorithm and the algorithm presented in this paper in the same page. This has been done in order to show the common general structure. At the begining of the the SA method a configuration is generated as starting state. Both algorithms also share a temperature as a control parameter which is updated each time the algorithm executes a complete inner loop. Please note that we called MC step (Monte Carlo step) the inner loop of the cooperative and competitive algorithm we are introducing. Cooperative and competitive phases are interlabeled between MC steps. It is this interaction between solutions that helps to develop global optimization properties.

Simulated annealing is general in that it only needs the specification of the cost function associated with configurations, the moves that may be made between them, and a cooling schedule for modifying the temperature parameter, and so it has become a popular method for diferent optimization problems like the N-city travelling salesperson problem (TSP) [15] [16], the min-cut partitioning [15] and global wiring [27]. Other aplications include least square fitting of many unknowns [26] image analysis [22] [9] and learning

in random boolean networks [6] [5].

One of the principal drawbacks of the simulated annealing approach is a large dispersion shown by the results of optimizations with different random starts. This can be solved using a slow cooling schedule [21], something that requires a large amount of computer time. Although some theoretical results have shed light on this matter [19] it is still a practical difficulty.

1.1 The Traveling Salesman Problem (TSP)

The TSP is generally defined as the task of finding the *cheapest* way of connecting N cities in a closed tour where a cost is associated with each link between cities.

One version of it, which belongs to the NP-complete class, the Euclidean TSP in two dimensions, has been one of the most studied optimization problems. In the Euclidean version, the cost of linking two cities is proportional to the Euclidean distance between them. The *blind* Euclidean version of the TSP, beloved by purists, is a variation on the above where only the length of a tour is known, and no information about the distance between individual cities is used to guide the search.

During the long battle that science has waged with the TSP, only some big problems have been solved to optimality [20] . Nevertheless, for a random uniform distribution of N cities over a rectangular area of R units, an asymptotic expected length formula for the optimal tour has been derived [3]. The expected length L_e of the optimal tour is given by

$$L_e(N, R) = K\sqrt{NR} \tag{1}$$

With computational experiments [23], the value K has been bounded by

$$0.765 \leq K \leq 0.765 + \frac{4}{N} \tag{2}$$

and we also remark the derivation given by Bonomi and Lutton. They give a value of $K = 0.749$ for large N [4]. Many *ad hoc* TSP algorithms have been constructed during the last 50 years [17] and so it represents a challenging test-model for our approach.

2 The Cooperative-Competitive Search

The central idea of the cooperative-competitive approach for searching in large configuration spaces is to use collective properties of a group of distinguishable individuals, which are separately performing the search, to generate

solutions that are better than those which would be obtained by each individual without interactions within the group, or by only one individual during a number of searching attempts equal to the number of members of the group.

Given this description, the approach is especially attractive for being implemented on *multiple instruction multiple data* (MIMD) parallel computers because each individual is trying to solve the complete optimization problem. If each individual node can have the complete instance of a problem, we can assure that each node can be running an individual process. The method would gain from the asynchronicity since the only needs for comunication between nodes arise from the cooperative and competitive phases which do not dominate the computer time employed. The most time consuming procedure is the individual local search and for that a fast heuristic would be suitable.

An Implementation

We have applied the competitive-cooperative search to the blind version of the Euclidean TSP in two dimensions. We regard the present work as an application of a more general approach for optimization problems.

In this case, the individuals are arranged on a ring and each one *searches* locally, *competes* with its two immediate neighbours in the ring, and *cooperates* with individuals which are very distant within the ring. The arrangement introduces a different neighbourhoods for cooperation and competition.

We can see that, for the sixteen element ring shown in Fig. 2, an individual competes with its nearest neighbours in the ring, and cooperates with individuals that are four links away in the ring.

The local search is supplied by Monte Carlo simulated annealing. The cooperative aspect is supplied by a crossover operator identical in form to that used in *genetic algorithms* [13] as applied to the travelling salesperson problem by Grefenstette [11]. The competitive aspect is supplied by a procedure where individuals subsume each other's positions according to their relative fitnesses. The acceptance of the changes involved in *all three* components of the search is governed by temperature, as described below, and this value is subject to a cooling schedule.

Local Search

One step of the local Monte Carlo search process for a tour of N cities can be understood as N attempted rearrangements of the tour. This indicates that we used as our stoping criterion of the MC step the completition of N attempted rearrangements. We are aware of better criteria (see Ref.[10]) and this would deserve more research in the future. This criterion combined with

a different acceptance procedure (see Ref. [18]) would yield a more efficient way of performing the local search, however here we are restricted to the usual procedures.

The moves used for rearrangement are of three different types: the inversion of a sub-tour, the insertion of one city in a different part of the tour, or the insertion of two connected cities in another part of the tour. The first move changes two links and the latter two moves both change three links. The cities and the places of insertion are selected with a random uniform distribution among all possible values and all processors have the possibility of doing all rearrangements, so we have an stochastic Markovian process. Each of the changes is accepted with a probability

$$p(\Delta E_{MC}, T) = \frac{1}{1 + e^{\Delta E_{MC}/T}} \tag{3}$$

where

$$\Delta E_{MC} = \frac{\Delta L}{\sqrt{N}} \tag{4}$$

and ΔL is the change in length produced by the rearrangement. This acceptance procedure has been also used by H. Szu and others in connexion with Boltzmann Machines [24] [25].

Competition

Competition occurs between individuals on the basis of a challenge by an individual currently residing in one location on the ring to an individual in another location. In a given competition phase all individuals both challenge and are challenged, and so are involved in two interactions with neighbours. The competition procedure can be clarified with an example. In the ring shown above, the tour in location 0 would compete with the tour actually in location 1 by an *issued* challenge, and with the tour in location 15 by a *received* challenge. If the challenge to location 1 is successful then the tour in location 1 is removed and it is replaced with a *clone*, an exact copy of tour 0. Clearly tour 0 can itself be replaced by tour 15 if it can not succeed against his challenger (tour 15). The *battle* is decided according to the probability

$$p(\Delta E_{comp}, T) = \frac{1}{1 + e^{\Delta E_{comp}/T}} \tag{5}$$

If each tour is of length L_i. where the sub-index i stands for the sequence number of the location of the tour around the ring, for the competition between 0 and 1 we compute ΔE_{comp} according to

$$\Delta E_{comp}(0,1) = \frac{\Delta L_{0,1}}{N} = \frac{L_0 - L_1}{N} \tag{6}$$

so if tour 0 challenges tour 1, we generate a random number q with uniform distribution in the interval $[0,1]$ and if $q \geq p(\Delta E_{comp}(0,1), T)$ nothing happens but if $q < p(\Delta E_{comp}(0,1), T)$ tour 1 is deleted and replaced with a copy of tour 0.

Cooperation

The cooperation procedure is based upon the crossover operator of genetic algorithms in which components of configurations are exchanged, allowing the combination of subcomponents of successful individual searches into configurations that may develop to be better than either of their generating configurations.

The operator used is that defined as the *order crossover* or OX operator by Goldberg [?]. Of the two configurations to be combined, an arbitrary subtour is chosen from one tour, and inserted into a second. In order that the generated tour should obey the constraint that each city is visited exactly once, the cities that are inserted are excised from their original locations in the second tour, and those cities that were connected on each side of them are re-connected to each other. The result bears a structural relationship to both parents, although the excision of cities means that achieving the subtour often makes significant changes to the tour into which it is inserted.

In contrast to the two and three link changing operations of the Monte Carlo step procedure, the number of links in the second tour which change during crossover may be of any value, up to the number of links it contains.

Cooperation occurs on a similar basis to competition in that a challenge, which may be considered as a proposition in this case, is issued between neighbours in the locality defined for cooperation. A proposition is assessed by the same criteria as a challenge, scaled with temperature in the same way. If the proposition is accepted, crossover is performed between the tours and the resulting "child" replaces the recipient of the proposition. The length of the *result* of crossover is not used to determine its acceptability.

3 The OX operator

In order to describe the type of crossover operator used, we can show how it works with the sequence-example used by Goldberg (see Ref. [12]). We show with an example two parents configurations, tours A and B (see Fig. 1 and Fig. 2), which are the following strings

A = 9 8 4 5 6 7 1 3 2 0
B = 8 7 1 2 3 0 9 5 4 6

As Goldberg describes, we will swap a substring from one of the parents that will be called the donor. This substring is selected randomly with uniform probability of starting and ending sites, and in Goldberg's example, the substring 5 6 7 of donor A is the one that will be mapped into the receiver B. A *matching section* is defined and is graphically represented between two symbols " ' ".

A = 9 8 4 '5 6 7' 1 3 2 0
B = 8 7 1 '2 3 0' 9 5 4 6

The swapping of substring '5 6 7' will be performed in replacement of the '2 3 0' substring in parent B. Since the new individual should have ten different city names (the numbers 0, 1, ..., 9); after the insertion of the substring we must delete the cities of the receptor that were included in the new substring added. Instead of presenting the final result, we will show all its steps. Before swapping, we will delete the cities in the receptor.

A = 9 8 4 '5 6 7' 1 3 2 0
B = 8 * 1 '2 3 0' 9 * 4 *

We have filled the *holes* with stars in the receptor parent B. Now to preserve the relative order in the receiver, we will make a *sliding motion* to leave the holes in the matching section marked in the receiver. Goldberg (Gold. page 174) makes this sliding motion starting in the second crossover site, so after the rearrangement we have

A = 9 8 4 '5 6 7' 1 3 2 0
B = 2 3 0 '* * *' 9 4 8 1

After that, the stars are replaced with the city names taken from the donor A. We get the configuration OX(B,A,2) (see Fig. 3)

A = 9 8 4 '5 6 7' 1 3 2 0
OX(B,A,2) = 2 3 0 '5 6 7' 9 4 8 1

For the purpose of this explanation, we can stop the discussion here, but in the description of the OX operator, the creation of the complementary cross must be added. This would give

$$\text{OX(A,B,2)} = 5\ 6\ 7\ \text{'}2\ 3\ 0\text{'}\ 1\ 9\ 8\ 4$$
$$\text{OX(B,A,2)} = 2\ 3\ 0\ \text{'}5\ 6\ 7\text{'}\ 9\ 4\ 8\ 1$$

From the 10-cities example shown, it seems rather dificult to understand how this crossover operator helps giving a cooperative mechanism for this concurrent search. For this example we have selected two tours which are far from being near optimality. This fact, combined with the small number of cities of the tours under consideration, gave tours that differ significantly from the parents. However, we encourage the reader to make some drawings to understand what happens in other cases and how two tours which are near of "local minima" by using the set of moves described above, merge to form a new child. The effect of the crossover operator is to make "a long jump" in configuration space but *preserving most of the good subcomponents* of the parents tours.

The Optimisation Schedule

Competition, cooperation and local search are interspersed so that a period of local search is followed by a competition phase, another period of local optimisation, a cooperation phase, and then back to local optimisation. As in some implementations of Simulated Annealing, the temperature is initially set to a value where 40 percent of the rearrangements with $\Delta L > 0$ are accepted and reduced using the standard geometrical schedule $T_n = 0.98T_{n-1}$ on the completion of each Monte Carlo step.

The optimisation is judged to have completed if the diversity of the group falls to a low value. To be more specific, samples of the connections of 128 random cities are made in random pairs of individuals within the group . If, for all such pairs, the selected city is connected to the same two other cities then the group is judged to have reached a solution. The sampling is implemented in a MIMD machine by considering only pairs of cities either competing or cooperating, and by globally asynchronously monitoring the diversity of pairs of tours.

The advantage of the cyclical sequence of phases outlined above are first that the results of cooperation do not compete until they have undergone local optimisation to ameliorate the damage caused by the OX operator, and second that if both an individual and its clone are victors of competition, they are allowed to optimise along separate paths before their components are propagated in cooperation.

This said, there is no real reason, apart from simplicity of implementation, that the phases should run synchronously in all localities. Indeed, the method requires only occasional communication between individuals during cooperation and competition and so is not likely to suffer the performance penalties usually associated with message-passing in an asynchronous environment.

Results

In order to study the performace of the method, we performed simulations using a set of 100 cities randomly distributed with uniform density inside a unit square. We found that when the number of individuals was small the costs of final configurations, resulting from different initial random starts, had a large variance such as that expected for the results of simulated annealing. When, however, the number of individuals was of the same order as the number of cities, final configurations were all very short and differed only in a small number of links. In Figure 1 we show some tours obtained using 128 individuals and different random starts.

The configurations displayed are the ones that systematically appear as final results and were not selected as the best solutions in a huge number of simulations. Indeed, had we done this, all four tours would be identical.

In search of a bigger problem to test the method, we looked at the Lin-Kerningham 318-cities TSP. This particular instance of the TSP seems to be very difficult since it contains randomness and clusters on different scales. It also contains one fixed link. It is interesting to remark that while many researchers have sufficient computer facilities to prove their methods on it, they usually generate their own random set of cities.

For this problem, the tour of optimal length among the 10^{655} possibilities has been found, so it is a good test for our method. Instead of defining the tour *ad-hoc* to contain the one fixed link we gave this link a very negative distance value; that is a value equivalent to the the total length of a random tour so we are performing a search between 317!/2 possible configurations. We performed the very computationally expensive experiment of simulating a system composed of 408 individuals. The surprising result is given in Figure 2 and it should be compared with the optimal solution shown in Figure 3.

4 Conclusions

It is clear that the method shows promise. The striking result for the 318-cities problem as well as the performance in the 100-cities case, encourage us to study the properties of the approach and ways of both improving the

results and diminishing the processing time required. The method is sufficiently general to be applied to other problems and as a result of its intrinsic parallelism and its low inter-process communication requirement, it should be considered as a useful strategy for combinatorial optimization in MIMD parallel computers.

References

[1] C.P. Bachas, J. Phys. A **17**, L709 (1984).

[2] F. Barahona et al., J. Phys. A **15**, 673 (1982).

[3] J. Beardwood, J.H. Halton and J.M. Hammersley, Proc. Cambridge Philos. Soc. **55**, 299 (1959).

[4] E. Bonomi and J.L. Lutton, SIAM Review **26** 551 (1984).

[5] O. Bria and P. Moscato, unpublished results.

[6] S. Patarnello and P. Carnevalli, Europhys. Lett. **4**(4), 503 (1987).

[7] G.C. Fox et al, *Solving Problems on Concurrent Processors. Vol 1* (Prentice Hall, Englewood Cliffs, New Jersey, 1988).

[8] M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, San Francisco, 1979.

[9] S. Geman and D. Geman, IEEE Trans. Pattern Analysis Machine Intelligence, **6**, 721 (1984).

[10] L.P.P.P van Ginneken and R.H.J.M. Otten, "An Inner Loop Criterion for Simulated Annealing", Physics Letters A, **130**, number 8,9,; pp. 429-435, (25 July 1988).

[11] *Incorporating Problem Specific Knowledge into Genetic Algorithms* J.J. Grefenstette, in *Genetic Algorithms and Simulated Annealing* L. Davis (ed). London: Pitman (1987).

[12] D.E. Goldberg, *Genetic Algorithms in Search Optimisation and Machine Learning* Addison Wesley (1989).

[13] J.H. Holland, *Adaptation in Natural and Artificial Systems.* Ann Arbor: University of Michigan Press. (1975)

[14] J.S. Judd, *Neural Network Design and the Complexity of Learning* Ph D Thesis, Massachusetts, Amherst, MA. Sep. (1988).

[15] S. Kirkpatrick, C.D. Gelatt and M.P. Vecchi, Science **220**, 671 (1983).

[16] S. Kirkpatrick, J. Stat. Phys. **34**, 975, (1984).

[17] E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan and D.B. Shmoys, *The Travelling Salesman Problem: A Guided Tour of Combinatorial Optimization*, (Wiley-Interscience, Chichester, 1985).

[18] P. Moscato and J.F. Fontanari, "Stochastic versus Deterministic Update in Simulated Annealing", Physics Letters A, **146**, Number 4, pp. 204-208, (21 May 1990).

[19] J. D. Nulton and P. Salamon, Phys. Rev. A **37**, 1351 (1988).

[20] M.W. Padberg and S. Hong, Math. Programming Stud. **12**, 78 (1980).

[21] R.E. Randelman and G.S. Grest, J. Stat. Phys. **45**, 885 (1986).

[22] W.E. Smith, R.G. Paxman, and H.H. Barrett, J. Opt. Soc. Am. A **2**, 491 (1985)

[23] D. Stein, *Scheduling Dial-a-Ride Transportation Systems: An Asympotic Approach* Ph.D. Thesis, Harvard University, Cambridge, MA (1977).

[24] H. Szu, "Nonconvex Optimization by Fast Simulated Annealing", Proceedings of the IEEE, **75**(11), pp. 1538 (1987).

[25] H. Szu, "Fast Simulated Annealing", Physics Letters A, **122**(3-4), pp. 157 (1987).

[26] D. Vanderbilt and S.G. Louie, J. Comput. Phys. **56**, 259 (1984).

[27] M.P. Vecchi and S. Kirkpatrick, IEEE Trans. Comput. Aided Design: Integrated Circuits **CAD-2**, 215 (1983).