also in <u>Case-Based Reasoning Research and Development</u>, *Lecture Notes in Artificial Intelligence, M. Veloso & A. Aamodt (eds), pp401-410, Springer Verlag, 1995.*

On the use of CBR in optimisation problems such as the TSP

Pádraig Cunningham *, Barry Smyth **, Neil Hurley **

**Department of Computer Science, Trinity College Dublin, Ireland

**Hitachi Dublin Laboratory, Trinity College Dublin, Ireland

Abstract. The particular strength of CBR is normally considered to be its use in weak theory domains where solution quality is compiled into cases and is reusable. In this paper we explore an alternative use of CBR in optimisation problems where cases represent highly optimised structures in a huge highly constrained solution space. Our analysis focuses on the Travelling Salesman Problem where difficulty arises from the computational complexity of the problem rather than any difficulty associated with the domain theory. We find that CBR is good for producing medium quality solutions in very quick time. We have difficulty getting CBR to produce high quality solutions because solution quality seems to be lost in the adaptation process. We also argue that experiments with CBR on transparent problems such as the TSP tell us a lot about aspects of CBR such as; the quality of CBR solutions, the coverage that cases in the case-base offer and the utility of extending a case-base.

1 Introduction

Normally in CBR cases represent high quality solutions in weak theory domains. CBR is useful in situations where analytical models of interactions in the problem are difficult to determine and cases represent compiled quality solutions that are adaptable. It is a basic tenet of CBR that this solution quality should survive the adaptation process.* This paper examines solution reuse in a very different context; we explore case reuse in optimisation problems where cases can represent highly optimised structures in a large highly constrained search space.

There has already been some CBR research on scheduling, a particularly important category of optimisation problem. Two distinct approaches to case-based scheduling can be identified from the literature. The first one is used in Koton's SMARTplan (1989). Cases are used to propose preliminary schedules which are then adapted (refined and repaired) to satisfy the target schedule requirements. The second approach does not uses cases to propose schedules, but instead to adapt schedules proposed by other methods (for example, Sycara & Miyashita, 1994). In other words cases can encode actual schedules (the first approach) or they can encode repair procedures (the second approach). In this research we are interested in the first approach where the cases represent actual optimised structures in the problem domain.

So far we have looked at graph traversal problems (shortest path) and Travelling Salesman Problems (TSP). These are not weak theory domains because these

---

* Typical examples of this are; CAPLAN/CBC as described in (Muñoz, Paulokat & Wess; 1994) or Déjà Vu (Smyth & Cunningham, 1992)

problems are easy to model and the way in which solution components contribute to good quality solutions is easily understood. The problems are difficult because they involve a search through a huge highly constrained solution space. The fact that in optimisation problems solution components may be reusable in other situations suggest that CBR may be applicable. The transparency of these problems compared to problems in weak theory domains means that an analysis of the application of CBR in these areas can tell us a lot about CBR itself. In particular, it can tell us something about the quality of CBR solutions, the coverage that cases in the case-base offer and the utility of extending a case-base.

In this paper we focus on the use of CBR on Travelling Salesman Problems. Our findings have been mixed. Our CBR solution has been excellent from the point of view of speed but the solution quality has not been great. The CBR system produces good but not excellent solutions very quickly. In the next section we present a description of our CBR solution for the TSP. The cases are produced using Simulated Annealing (SA) and the performance of CBR system is compared with SA. The SA algorithm we use is described in an Appendix at the end of the paper. In section 3 we present a statistical analysis of case coverage. The performance of the CBR system is analysed in detail in section 4. The paper concludes with a summary of our findings and an indication of some future directions for research.

2 A Model for CBR in TSP

The basic idea behind a CBR approach to a problem like TSP is to shift problem complexity from the time domain into the space domain; that is, by storing cases we expect to find significant improvements in problem solving time. There is the worry however that the combinatorics of TSP will force the need for prohibitively large case-bases, in order to achieve significant performance improvements, thereby rendering a CBR solution infeasible in all but the simplest of TSP domains. We will defer discussion on the size problem to section 3 and describe our model of case reuse here.

The TSP problem involves sequencing a set of objects in order to minimise a cost function. There are many manifestations of this problem; for instance the problem of scheduling N jobs on a single machine where job set-up time is sequence dependent is a TSP (Cunningham & Browne, 1986). This is an example of a non Euclidean TSP in that solutions cannot be graphed in 2-dimensional space. This scheduling problem is also asymmetric in that the 'distance' from A to B is probably not equal to the 'distance' from B to A. The context of this scheduling problem motivates a case based solution. Typically schedules will be produced on a weekly or monthly basis and the set of jobs to be scheduled may be similar to a set scheduled in the past. The CBR solution described here is valid for non-Euclidean asymmetric TSPs even though the examples presented in the paper are shown as graphs.

In Figure 1 we show a 'world' of 100 cities, an optimised tour of 40 cities and a target problem of 40 cities. There is an overlap of 19 cities between the base and the target. In T-CBR the size and distribution of this overlap is the measure of similarity used in case retrieval. The first step in the adaptation process is to produce a skeleton tour from the overlap of the base and target; this is shown in the Figure. The first CBR solution is produced by adding the remaining cities on the target to this tour.

The algorithm for this is as follows:-

```
function Add-cities(tour, rem-cities)
{
     if rem-cities {
            best-city ← Select-best-city(tour, rem-cities)
            rem-cities ← Remove-city(best-city, rem-cities)
            tour ← Insert-city(tour, best-city)
            Add-cities(tour, rem-cities)
     }
}
```

The `Select-best-city` function takes each remaining city in turn and finds the point on the tour where it can be inserted with the minimum cost. The best city is the one that can be inserted in the tour with least cost. This first CBR solution is shown in Figure 2.

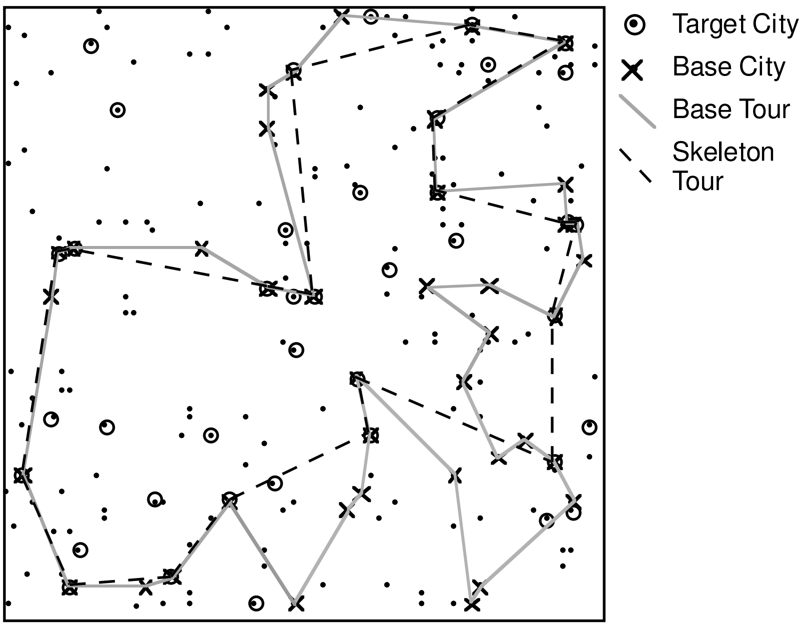

Fig. 1. A 'world' of 100 cities, a target problem of 40 cities and a base with 19 cities in common with the target.

This T-CBR solution is 365 units in length. The SA algorithm produced a solution of 338 units. This T-CBR solution is 8% longer than the SA solution. This adaptation mechanism is inspired by the geometric techniques for solving TSPs used by Norback and Love (1976). However it will work for non geometric or non symmetric problems. T-CBR goes on to improve this initial solution by examining each pair of cities in turn and testing to see if reversing the section of path between these two points produces an improved solution. We call this gradient descent 'Freezing' by analogy with what happens in Simulated Annealing.

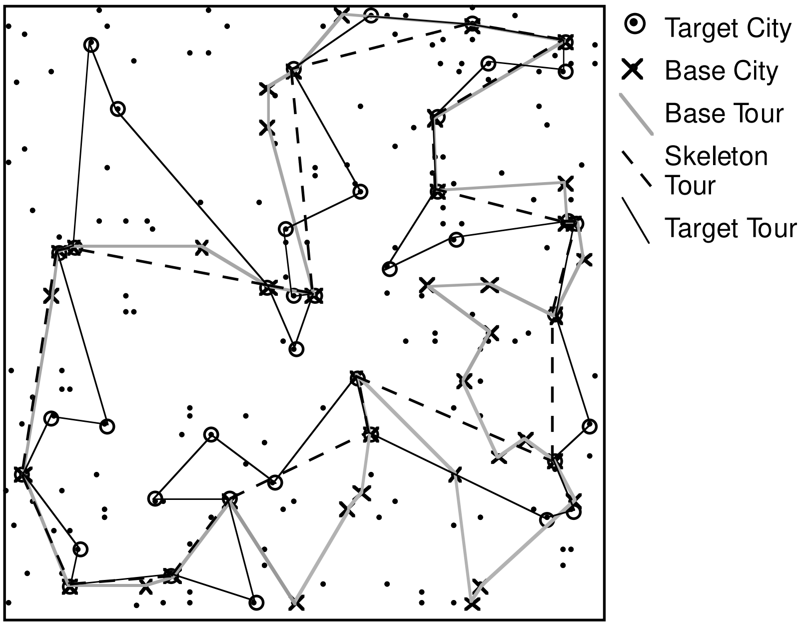

Fig. 2. The first CBR solution for the target shown in Figure 1.

3 An Analysis of Case Coverage

The size of the case-base depends on a number of quantities: n, the number of cities; r the tour lengths of cases and target problems; and k the desired overlap between a target problem and a retrieved case. That is:-

*For a domain of **n** cities and a tour size of **r**, how many cases are required to ensure the presence of at least one case that shares **k** or more cities with the target?*

First of all the total number of possible tours of size r in this domain (of n cities) is given by equation 1.

(1) Total tours $= \binom{n}{r}$

The total number of tours that share exactly k elements is shown in equation 2. The first factor is the total number of ways of choosing k cities from tours of r cities, and, given that k cities have now been fixed, the second factor is the total possible ways of filling the remaining r-k tour positions.

(2) Total tours sharing k cities $= \binom{r}{k} \bullet \binom{n-k}{r-k}$

The total number of tours that share k or more cities is thus given by equation 3.

$$(3) \qquad \text{Total tours sharing } k \text{ or more cities } = \sum_{i=k}^{r} \binom{r}{i} \bullet \binom{n-i}{r-i}$$

For a particular target specification of r cities, the probability of picking a random tour that shares at least k cities with this target is shown in equation 4; this is termed the probability of success and is denoted by S.

$$(4) \qquad S = \frac{\sum_{i=k}^{r} \binom{r}{i} \bullet \binom{n-i}{r-i}}{\binom{n}{r}}$$

Now in a CBR scenario, a case-base of C random cases means that we have essentially C attempts to find a successful tour, a tour that overlaps by at least k cities with the target. We would like the probability of failure (in other words the probability of there not being a suitable case) to be less than some fraction ε. So if $(1\text{-}S)^C$ is the probability of a finding no such case over C trials* then the relationship between ε and S is the simple one shown in equation 5.

$$(5) \qquad (1 - S)^C < \varepsilon$$

So from this we can form the function in equation 6 that computes the case-base size necessary to ensure that a suitable case is present for every target problem, all but $\varepsilon \bullet 100\%$ of the time.

$$(6) \qquad C = \frac{\log(\varepsilon)}{\log(1 - S)}$$

Figure 3 (A) & (B) illustrate how this function behaves for various values of *n, r,* and *k*. In each graph, the error value, ε, is kept static at 0.05, and n is varied from 50 to 500 in increments of 50, a separate curve is drawn for each n value and marked with that value. Both graphs plot the case-based size on the y axis, which is logarithmically scaled.

Figure 3(A) plots case-base size against the tour length (that is the case and target sizes). The desired overlap between target and case, k, is assumed to be 30% of the tour length. As expected there are large variations in the size of the case-base as n and r increase. It is interesting that for this 30% overlap restriction, the size of case-base needed for a given n peaks when r is 12.5% of n and trails off rapidly as r increases towards n. For example, for n = 500, the case-base size offering 30% overlap, peaks at $r = 40$, where over 10^5 cases are needed to provide the desired problem coverage. However, when the tour size is 100, less than 600 cases are needed.

---

* (1- S) is the probability of not selecting a successful tour in one attempt so $(1\text{-S})^C$ is the probability of failure after C consecutive attempts.

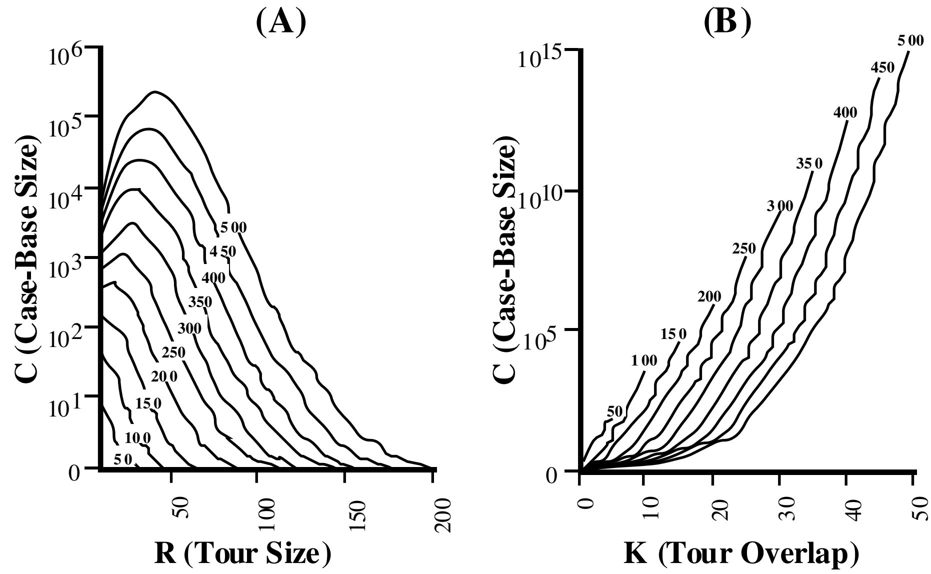

Fig. 3. Variations in case-base size.

Figure 3(B) plots the case-base size against varying overlap. For each curve *r* is fixed at 20% of *n* and the overlap, *k*, is varied from between 0 and *r/2*. So for example, the curve corresponding to *n* = 500 corresponds to a system where the tour length (that is the sizes of the cases and the target problems) is 100, and a varying overlap of between 0 and 50 cities. Again, as expected the case-base size rises exponentially with *k*. In a system where *n* is 500, *r* is 100, and *k* is 50 (that is 50% of the tour length), a case-base of over 1014 cases is required. However, as was mentioned above, if an overlap of 30 cities is acceptable for this system then less that 600 cases are needed.

A pessimistic interpretation of these results might suggest that CBR and TSP do not mix well, owing to the enormous case-bases required to provide adequate coverage and significant performance improvements. However this is not necessarily true in general, and there exist a great many TSP problems which *are* amenable to a case-base solution, at least in the sense that the case-base is of an acceptable size. In particular, this is true if the route size, *r*, is greater than roughly 20% of *n* and *k* is less that about 40% of *r*.

4 Evaluation

In order to evaluate the T-CBR process an artificial world of 500 cities was created and a case-base with 600 tours of 100 cities each was constructed. Good solutions for these tours were produced using the SA algorithm described in the Appendix. The case-base size was chosen to offer good target-base overlap. In fact, on average, about 10 cases are found that overlap the target by 29 or more cities.

The evaluation that we describe here has two parts. In the first part the potential of T-CBR to produce very quick medium quality solutions is considered. In this evaluation it is compared with SA and a Myopic algorithm that selects a starting city at random and chooses the nearest remaining city at each step. This is the standard quick solution to the TSP. This evaluation is described in section 4.1. We then attempted to use T-CBR to produce good quality solutions by seeding a Low Temperature version of the SA algorithm with solution produced from the case-base. This is described in section 4.2.

4.1 Producing quick & nasty solutions

In this situation we are evaluating five alternatives;

- the full Simulated Annealing
- the initial T-CBR solution
- the initial Myopic solution
- the T-CBR with Freezing
- the Myopic with Freezing

One hundred target tours were generated at random and solutions were produced using each of these algorithms. The average solution times and tour lengths are shown in Figure 4. The SA solution was used as a base-line for comparing the alternatives. The CBR+Freezing produces solutions that are within 6% of the SA solutions but in one tenth of the time. The Myopic+Freezing is extremely fast but the solutions are over 11% longer than the baseline. So CBR scores well as a means of producing medium quality solutions quickly.

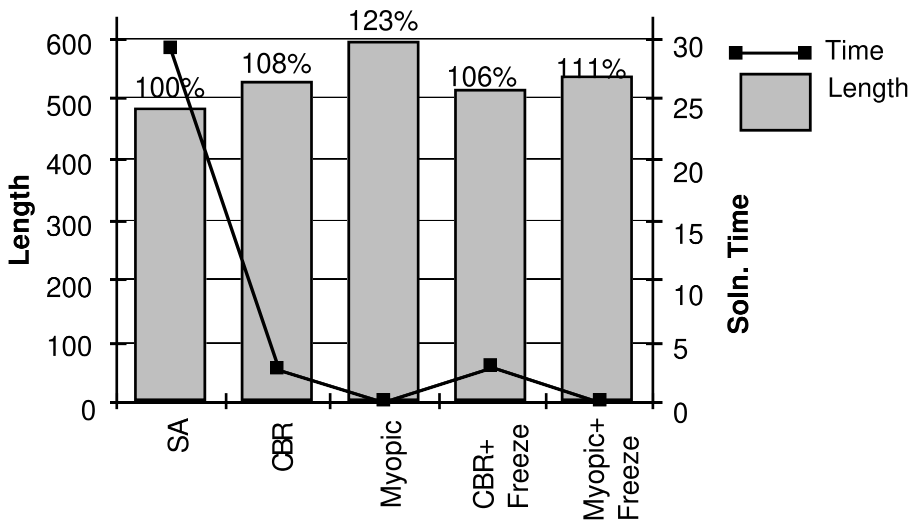

Fig. 4. Costs and lengths associated with different algorithms

4.1 Producing good quality solutions

The obvious question arising from this evaluation is whether T-CBR can be improved to produce high quality solutions. The strategy we came up with was to use the initial

CBR solution to seed a Low Temperature version of the Simulated Annealing algorithm (LTSA). After experiments with different temperatures we settled on a starting temperature of 1.0 for the LTSA. Temperatures much below this were akin to Freezing the solution while values greater than 1.0 allowed the structure of the good solution produced by T-CBR to deteriorate.

The experiment reported here describes tests over a range of target tour lengths from 40 to 160, with 20 tests being done at each length. This also illustrates the flexibility of T-CBR where base cases can be adapted to targets of different lengths. The main results of this experiment are shown in Figure 5. T-CBR+LTSA performs quite well with solutions found on average 2.5 times faster than SA. These solutions are now within 1-3% of the SA solutions. This would be great news for CBR except that we discovered that Myopic+LTSA (Low temperature SA seeded with a Myopic solution) produced solutions of similar quality. Myopic+LTSA is faster than T-CBR+LTSA and does not need a case base. This suggests that the main contribution to the good solution is the LTSA and not the CBR.

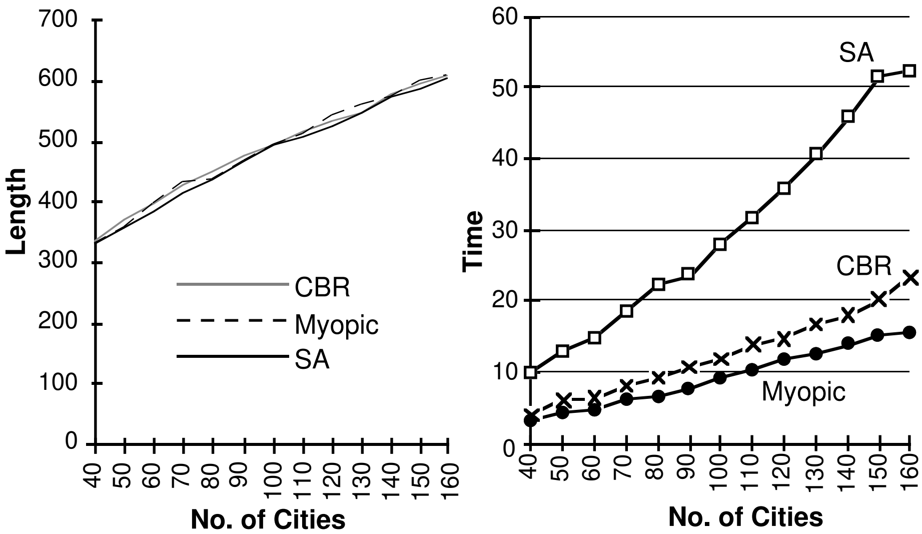

Fig. 5. A comparison of the cost and quality of solutions from the three high quality techniques.

5 Discussion

We have described a CBR solution to the TSP that provides good but not great solutions in very quick time. This is not surprising because the effect of the CBR solution is to shift the problem complexity from the time domain into the space domain. Indeed we have pointed out that this CBR solution is not practicable for many TSP problems because the required case-bases can be prohibitively large.

The more interesting issue raised in this paper is the fact that our CBR technique does not manage to produce solutions of high quality. Some of the solution quality is lost in the adaptation process. We have described a technique combining case retrieval and

Low Temperature Simulated Annealing that does produce very good solutions but we have argued that this is more due to the power of the SA than the contribution of the retrieved case.

It seems that the experience of research on Genetic Algorithms (GA) is informative in this regard. What is happening in T-CBR is that the retrieved case represents a highly optimised structure that is disrupted or broken in the adaptation process. GA research emphasises the importance of building blocks in the GA process. These building blocks are manipulated blindly in the process of reproduction and crossover to produce compositions of building blocks that are highly optimal. It is recognised that it is important for individual building blocks not to be broken in the crossover process (Goldberg, 1989). In our adaptation process the solutions are manipulated naïvely and building blocks representing optimal structure are broken. However, this is probably inevitable where there is so little intersection between the base and target problems. The base case provides a skeletal solution for the target problem but the detailed optimal structure from the base case is not transferred into the target.

This analysis suggests that a change in approach to the reuse of base cases might be fruitful. Instead of considering the base case as the source of a skeleton on which a target tour can be built several base cases could be used to provide tour segments to build into a good quality target solution. This alternative approach seems promising but there are new problems associated with combining tour fragments taken from different cases.

Conclusions

CBR can be used to produce medium quality solutions to optimisation problems. In the approach described here the base case provides a skeletal solution on which a solution for the target problem can be built. Efforts to use CBR to produce high quality solutions for optimisation problems need to focus on ensuring that solution building blocks survive the optimisation process. This might be achieved by using several cases to contribute to the target solution.

We have argued that the naïve manipulation of the solutions in the adaptation process results in the target solutions being of lower quality than the solution in the base case. This is evident in T-CBR because of the transparency of the problem domain. It is interesting to speculate on how prevalent this is in other CBR applications in domains that are not so transparent.

References

Cunningham P., Browne J., (1986) A LISP-based heuristic scheduler for automatic insertion in electronics assembly, *International Journal of Production Research*, Vol.24, No.6, pp1395-1408.

Goldberg D.E., (1989) *Genetic Algorithms in Search Optimization & Machine Learning*, Addison Wesley, Reading Massachusetts.

Kirkpatrick S., Gelatt C.D., Vecchi M.P., (1983) Optimization by Simulated Annealing, *Science*, Vol. 220, No. 4597, pp671-680.

Koton, P. (1989) SMARTplan: A Case-Based Resource Allocation and Scheduling System. *Proceedings of the Case-Based Reasoning Workshop*, pp 285-289, Florida, USA.

Muñoz H., Paulokat J., Wess S., (1994) Controlling Nonlinear Hierarchical Planning by Case Replay, in working papers of the *Second European Workshop on Case-based Reasoning*, pp195-203, Chantilly, France.

Norback J., Love R., (1977) Geometric Approaches to Solving the Travelling Salesman Problem, *Management Science*, July 1977, pp1208-1223.

Smyth B., Cunningham P., (1992) Déjà Vu: A Hierarchical Case-Based Reasoning System for Software Design, in Proceedings of *European Conference on Artificial Intelligence*, ed. Bernd Neumann, John Wiley, pp587-589.

Sycara, K. & Miyashita, K. (1994) Case-Based Acquisition of User Preferences for Solution Improvement in Ill-Structured Domains. *Proceedings of the 12th National Conference on Artificial Intelligence*, pp. 44-49. Seattle, USA.

Appendix A: The Simulated Annealing Algorithm for the TSP

The Simulated Annealing Algorithm is a modification of a basic gradient descent search based on some ideas from statistical mechanics (Kirkpatrick, Vecchi & Gelatt, 1983). A notion of temperature is introduced into the process and the system is gradually cooled to *freeze* at a near optimal solution. The key modification to standard gradient descent is the possibility that some dis-improvements in solution are accepted. The probability of this is greater at higher temperatures (see Step 3 below). This allows the gross features of the solution to be determined at high temperatures while details are worked out at low temperatures. The details of our implementation are as follows:-

1. Set $temp \leftarrow \dfrac{\sqrt{NCITY} \bullet longest - dist}{10}$

   $nsucc \leftarrow 10 \bullet NCITY$

   $ntries \leftarrow 100 \bullet NCITY$

   Generate a pseudorandom feasible solution *T*.

2. Generate a new feasible solution *T'* by reversing a section of *T*.
   $nt \leftarrow nt + 1$.

3. $\Delta E \leftarrow E_{final} - E_{initial}$ (where *E* is the cost of the solution)

   If $\Delta E \leq 0$ accept the move;

   If $\Delta E > 0$ accept the move with prob. $P(\Delta E) = e^{-\Delta E/T}$

   $ns \leftarrow ns + 1$.

4. Repeat from 2 while $nt < ntries$ and $ns < nsucc$

5. When *nt* reaches *ntries* or *ns* reaches *nsucc* drop temperature (i.e. $temp \leftarrow temp \bullet 0.9$), set *ns & nt* to 0 and start a new loop from 2.

   Terminate on the first pass that produces no successes.