OPTIMIZATION BY SIMULATED ANNEALING: A PRELIMINARY COMPUTATIONAL STUDY FOR THE TSP

Christopher C. Skiscim
The MITRE Corporation
McLean, VA USA

Bruce L. Golden
The University of Maryland
College Park, MD USA

In recent papers by Kirkpatrick et al. (1982,1983), an analogy between the statistical mechanics of large multivariate physical systems and combinatorial optimization is presented and used to develop a general strategy for solving discrete optimization problems. The method relies on probabilistically accepting intermediate increases in the objective function through a set of user-controlled parameters. It is argued that by taking such controlled uphill steps, from time to time, a high quality solution can be found in a moderate amount of computer time. This paper applies an implementation of the proposed algorithm to the TSP for various size networks. The results show the algorithm to be inferior to several well-known heuristics in terms of both solution quality and computer time expended. In addition, set-up time for parameter selection constitutes a major burden for the user. Sensitivity of the algorithm to changes in stopping rules and parameter selection is demonstrated through extensive computational experiments.

1. INTRODUCTION

Combinatorial optimization entails a large class of mathematical programming problems whose underlying structure is inherently discrete in nature. Problems in graph theory and integer programming, such as the traveling salesman problem,the network design problem, and the knapsack problem, fall into this class. This class of problems has received a great deal of attention in the literature due to the large number of practical problems that it includes. However, many of these problems have been shown to be NP-hard (see Garey and Johnson (1979)) with the consequence that it is highly unlikely that an exact algorithm, whose running time is polynomially bounded in the size of the problem, exists. Accordingly, a great deal of research has been devoted to finding efficient, approximate solution methods that exploit some special structure of the particular problem.

Recently, Kirkpatrick *et al.* (1982,1983) have argued that all combinatorial problems possess a common structure, namely, that they are large multivariate discrete systems with many degrees of freedom. An analogy is made between the behavior of large physical systems and combinatorial problems with the result that one could apply results from classical statistical mechanics to combinatorial optimization.

Statistical mechanics concerns itself with analyzing aggregate properties of large numbers of atoms in liquids or solids. The behavior is characterized by random fluctuations about a *most probable behavior*, namely the average behavior of the system at that temperature. An important question is obviously: *What happens to the system at extremely low temperatures?* The low temperature state may be referred to as the ground state of the system and is the lowest energy state. In physical systems, the probability of a certain configuration occurring is calculated from a Boltzmann distribution. The potential energy of the system, the measure of interest, is determined by considering the potentials between a given element and its next nearest element in a given configuration. Since low temperature states are very rare, experiments that reveal the low temperature state of a material are performed by a process referred to as *annealing*. The material under study is first melted and then the temperature is slowly lowered with a long time spent at temperatures near the freezing point. The period of time at each temperature must be sufficiently long to allow a thermal equilibrium to be realized. Otherwise, certain random fluctuations will be frozen into the material and the true low energy state will not be realized. The process can be likened to growing a crystal from a melt. If the temperature is lowered too quickly, the result may be glass or a crystal with many defects.

The annealing process is usually simulated using a Monte Carlo procedure like the one developed by Metropolis *et al.* (1953). In this procedure, the thermal motion of atoms in contact with a heat bath at a given temperature is simulated. The procedure is simply stated:

Given a configuration of the elements of the system, randomly displace the elements, one at a time, by a small amount and calculate the resulting change in the energy, ΔE. If $\Delta E \leq 0$ then accept the displacement and use the resulting configuration as *the starting point for the next iteration. If $\Delta E \geq 0$ then the displacement is accepted with probability $P(\Delta E) = exp(-\Delta E/k_b T)$ where T is the temperature and k_b is Boltzmann's constant.*

The probabilistic aspect is implemented by comparing $P(\Delta E)$ with a random variable drawn from a uniform distribution on the (0,1) interval. Repetition of this step continues until equilibrium is achieved. At that point, the temperature is lowered and the procedure repeated.

The analogy we are seeking now presents itself. If we view the discrete elements of a combinatorial optimization problem as the *atoms* of a physical system, we are seeking a configuration that gives us the lowest energy state, or ground state. The energy of the system is calculated by the objective function. Heuristic methods, which rely on iterative improvement, continually seek a rearrangement that lowers the objective function as much as possible from one iteration to the next. When looked at from a statistical mechanics viewpoint, this is similar to rapidly quenching the system from a high temperature to a very low temperature with the result that solutions of inferior quality may be obtained. The following serves to illustrate this concept with respect to the Traveling Salesman Problem (TSP).

Suppose we apply the 2-OPT branch exchange procedure (Lin (1965)) to a tour with the option of accepting an increase in the tour length with some probability. Figure 1A depicts a ten node example with a starting tour. We note that this tour is 2-Optimal, that is, no decrease in the tour length is possible under the allowed transformation. We are thus at a local minimum with a tour length of 271.16 units.

Now consider exchanging arcs (7,3) and (4,5) with arcs (4,3) and (7,5). This results in an increase in the tour length of 8.9 units (see Figure 1B), but suppose that the probability is such that we accept the exchange and continue executing the 2-OPT procedure. This would lead us to the solution shown in Figure 1C with a tour length of 263.74 units. So we see that accepting an intermediate increase in the tour length can lead to a better solution.

Simulated annealing gives us a mechanism for accepting increases in a controlled fashion. At each temperature setting, we can accept an increase in the tour length with a certain probability. It is possible that accepting an increase will reveal a new configuration that will avoid a local minimum or at least a bad local minimum. This is clearly shown in Figure 1. Note that we always accept a decrease in the tour length. The effect of the method however, is that one descends slowly. By controlling these probabilities, through the temperatures, we are in essence simulating many random starting solutions in a controlled, simultaneous fashion. An analogy similar to this is well-known in statistical mechanics.

Kirkpatrick *et al.* argue that an iterative improvement scheme can be incorporated into the Metropolis procedure and used as a general strategy for solving combinatorial optimization problems. Instead of always rejecting a rearrangement that increases the objective function, we now accept it with some small probability. It is argued that taking controlled uphill steps allows one to break away from configurations leading to locally optimal solutions and hence increases the likelihood of obtaining a higher quality solution, eventually.

In the following sections, we apply this strategy to the traveling salesman problem, perhaps the most celebrated of combinatorial optimization problems. We formally define the Metropolis procedure as applied to the TSP and state the assumptions made in implementing the approach. The procedure is then applied to several problems of varying size and their solutions are compared to the optimal solution (where available) or to solutions generated by heuristics known to produce high quality tours. Sensitivity of the procedure with respect to several control parameters is examined closely.

2. *THE ALGORITHM*

The attractiveness of using the simulated annealing approach for combinatorial optimization problems is that transitions away from a local optimum are always possible when the temperature is nonzero. As pointed out by Kirkpatrick *et al.*, the temperature is merely a control parameter so we no longer require Boltzmann's constant. In keeping with Kirkpatrick *et al.*'s terminology, we will refer to this control parameter as a temperature. In discrete optimization problems, it controls the probability of accepting a tour length increase. As such, it is expressed in the same units as the objective function. In implementing the approach, the 2-OPT procedure developed by Lin (1965) is used for rearranging a tour.

In order for the method to function well, a proper annealing schedule must be developed. An annealing schedule is defined to be the sequence of temperatures and the amount of time at each to reach equilibrium for that temperature. Given this schedule of temperatures, $S = \{ t_1, t_2, ..., t_n \}$ with the sequence obeying $t_1 > t_2 > ... > t_{n-1} > t_n$, we formally define the algorithm (*SA*):

```
Step 0.  Generate a random TSP tour.
         Set $i \leftarrow 1$.

Step 1.  Execute one step of the 2-OPT
         algorithm and evaluate the change
         in the objective function, $\Delta C$,
         as a result of the exchange.
         If $\Delta C < 0$ go to Step 3; otherwise
         go to Step 2.

Step 2.  ($\Delta C \geq 0$).
         Select a random variable $\beta \in U(0,1)$.
         If $\beta < P(\Delta C) \equiv exp(-\Delta C/t_i)$,
         go to Step 3.
         If $\beta \geq P(\Delta C)$, then reject the 2-OPT
         exchange and go to Step 1.

Step 3.  ($\Delta C < 0$ or $\beta < P(\Delta C)$).
         Accept the 2-OPT exchange and compute
         the new value of the objective function.
         Go to Step 4.

Step 4.  If equilibrium, with respect to
         temperature $t_i$, is reached
         set $i \leftarrow i+1$.
         If $i > n$ STOP; otherwise
         go to Step 1.
```

Two implementation issues, crucial to the success of the method, are: What is an equilibrium and how does one construct an annealing schedule? In the unabridged version of the *Science* article, Kirkpatrick *et al.* (1982), give the following guidance on when a steady state or equilibrium is reached: *At each temperature, the simulation must proceed long enough for the system to reach a steady state. The sequence of*

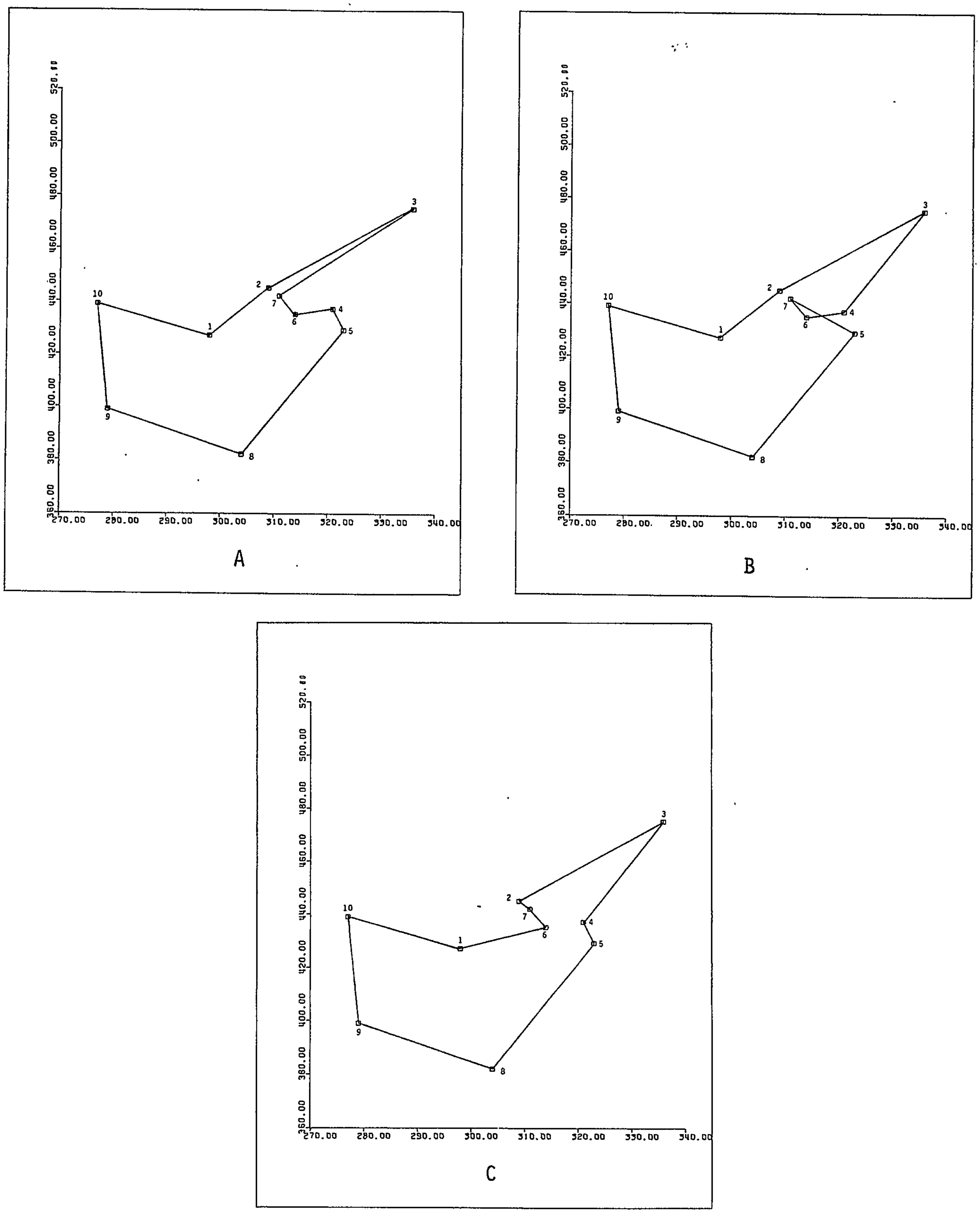


FIGURE 1
EFFECT OF ACCEPTING AN INCREASE IN TOUR LENGTH

temperatures and the amount of time required to reach an equilibrium at each temperature can be considered an annealing schedule. We performed a rather lengthy set of computational experiments in an attempt to observe a steady state and gain insight into the construction of an annealing schedule.

Recall that the 2-OPT is a branch exchange procedure in which a pair of arcs in a feasible tour are exchanged for a pair of arcs *not* in the tour as long as the result remains a tour and the length of the tour is less than the length of the previous tour. If an exchange is made, we term the exchange a *swap.* In the simulated annealing method, recall, we are permitted to make exchanges that increase the tour length. We refer to these as swaps also.

Starting with an arbitrarily high temperature and random tour, the method was executed with the resulting tour length observed after each swap or attempted swap. After a number of swaps had been made, the tour lengths were observed to cycle. The cycle consisted of making a swap that increased the tour lengths subsequently followed by a swap that decreased the tour length to a previously encountered value. That is, tour lengths oscillated between two or more values.

Based on these experiments, a steady state was defined as a set of tour lengths that is not systematically decreasing — a reduction beyond that is thus highly unlikely. We refer to this condition as being *at equilibrium at temperature* t_i. This is obviously related to how long one lets the procedure run for a given t_i.

Because the algorithm relies on uniformly distributed random variables, we want the 2-OPT to proceed long enough for the random element to work, however, excessive execution of the 2-OPT would greatly increase the running time. Based on our observations, we derived a method for equilibrium testing. Central to the understanding of our method is the concept of an *epoch.*

Before testing for an equilibrium, we allow the 2-OPT to effect a number of swaps, this number specified *a priori.* We define this interval between equilibrium testing to be an *epoch.* An epoch thus consists of *n* attempted swaps.[†] After execution of an epoch, the resulting tour length is saved and we test for an equilibrium. The test consists of comparing the tour length from the most recent epoch with the tour lengths from *all* previous epochs at a specified temperature. If the tour length from the most recent epoch is sufficiently close to any previously observed tour length (at the given temperature), we declare the system to be at an equilibrium point. At this stage, the next temperature is selected and the procedure is repeated.

The procedure thus consists of executing the 2-OPT (allowing for tour *increases*) for a number of epochs (the length of which is defined by the maximum number of swaps we permit) and testing for an equilibrium at the completion of each epoch. Thus, a number of epochs elapse before equilibrium. Following Kirkpatrick *et al.*, we *always* permit the system to reach equilibrium before the next temperature is selected. We now state the algorithm, termed ***SA'***, incorporating the equilibrium testing or stopping rule:

Define the set $L^t = \{ l_1, l_2, \ldots, l_k \}$ as the tour lengths observed during an epoch at temperature t.

```
Step 0.  Set $j \leftarrow 0$.
         Set $L^t \leftarrow \emptyset$.

Step 1.  Execute steps 1 through 3 of (SA) for one
         epoch. Call the resulting tour length $L$.

Step 2.  Equilibrium Test. If $|l_j - L| \leq \varepsilon$ for
         any $l_j \in L^t$ then
         go to Step 3; otherwise, perform the
         following updates:

         $j \leftarrow j+1$

         $l_j \leftarrow L$

         $L^t \leftarrow L^t \cup l_j$ .

         Go to Step 1.

Step 3.  Select the next temperature and go to
         Step 0.
```

Construction of the annealing schedule was the next task and it proved to be a very burdensome process. Kirkpatrick *et al.* offer little guidance other than trial and error. However, care must be taken so that large decreases in the objective function are avoided at any given temperature. According to the physical analogy, large decreases at any temperature are likely to result in descent to a local minimum that might make it difficult to exit from. The extreme case is a single application of the 2-OPT procedure at the zero temperature.

In order to determine an annealing schedule, a set of computational experiments were conducted on a 100 node problem taken from Krolak, Felts, and Marble (1971). We found that the initial temperature could easily be calculated by substituting the largest possible arc length for $-\Delta C$ in the expression $exp(-\Delta C/t_i)$ and solving for t_i using a probability of .99. The process consists of starting with the high temperature and then slowly reducing it by a fixed amount while observing the changes in the objective function. If the objective function is found to decrease significantly as the temperature drops from say t_1 to t_2 then that interval is further subdivided.

Despite our persistent efforts to find a schedule that lowered the objective function slowly, the experiments revealed that there seems to be one critical temperature at which a large decrease in the objective function inevitably occurs. The decrease was observed to be fairly uniform preceding and following this critical temperature. When this interval was further subdivided, the effect was to either defer the large decrease until a later temperature, or keep it at the same temperature. It was also observed that the final tour length, that is, the tour length resulting from the execution of the entire procedure, was only slightly changed by the process of interval subdivision. As we shall see later, the amount of computational effort to run a reasonable size problem is not insignificant, making the schedule selection process quite important.

[†] *The maximum number of swaps (n) per epoch is a parameter which can be varied. We examine the sensitivity of the method to variation in n later in this paper.*

			TEST CONDITION									
No. of Nodes	Best Known Solution	Swaps/ Epoch	Base (% over)	CPU* (secs)	+1 (% over)	CPU* (secs)	-1 (% over)	CPU* (secs)	Restart (% over)	Condition	CPU* (secs)
		50	.00	40.86	.27	43.11	.26	61.97	.41	-1	40.86
25	297.03	25	.00	20.97	.31	25.22	.00	26.29	1.08	B	20.13
		15	.00	17.77	.00	15.92	.00	18.74	1.08	B	13.78
		5	.00	10.15	.00	8.91	$\boxed{.00}$	$\boxed{7.53}$	.00	-1	12.80
		50	1.60	41.87	3.31	60.55	.41	150.61	.41	-1	146.79
32	390.89	25	1.60	42.55	3.70	30.05	.41	95.63	.41	-1	89.03
		15	1.92	30.93	2.46	28.57	.41	62.22	.00	+1	40.53
		5	1.60	16.65	6.04	14.38	$\boxed{.41}$	$\boxed{28.95}$	.15	-1	25.19
		50	3.47	79.10	6.26	59.94	3.13	102.20	1.86	B	73.71
40	234.36	25	1.27	62.44	2.73	73.94	8.35	66.09	1.27	B	44.83
		15	1.11	43.99	.59	41.10	$\boxed{.00}$	$\boxed{55.44}$	.00	-1	39.69
		5	7.82	36.99	4.76	39.20	4.85	35.33	4.85	-1	32.32
		50	4.08	361.11	3.76	360.69	5.33	126.50	3.38	B	196.24
50	264.15	25	6.75	150.64	7.71	116.30	7.04	218.26	4.79	-1	102.26
		15	$\boxed{2.15}$	$\boxed{79.39}$	3.35	75.24	3.91	108.71	2.15	+1	59.53
		5	5.47	70.01	6.19	129.78	3.90	70.01	3.90	-1	44.27
		50	4.16	342.67	2.12	285.78	4.89	288.08	2.81	-1	93.74
55	283.98	25	5.19	321.84	4.06	101.95	1.91	429.48	1.91	+1	100.91
		15	1.62	119.88	$\boxed{1.37}$	$\boxed{189.41}$	2.34	104.19	2.34	+1	71.23
		5	2.58	105.76	2.34	108.48	2.22	136.03	2.22	-1	52.86

* IBM 4341 Best Solution for problem $\boxed{\text{Highlighted}}$

TABLE 1
SUMMARY OF COMPUTATIONAL RESULTS

3. *COMPUTATIONAL RESULTS*

In order to evaluate the algorithm, a set of test problems of varying size was generated using a uniform random number generator to locate the nodes. Internode costs were determined by the Euclidean distances. All problems were generated in the same rectangular space. The experimentation revealed that the same annealing schedule was found to work well across varying problem sizes as long as the problems remained in the same rectangular space. The procedure is thus somewhat stable in this regard. The problems were tested varying the number of swaps per epoch and slightly altering the annealing schedule. Finally, the set of five 100 node problems taken from Krolak, Felts, and Marble (1971) was tested. For all problems, the annealing schedule was determined by calculating the initial temperature and then experimenting with various temperature reductions between that value and zero. The schedule was deemed acceptable when it produced a fairly uniform decrease in the tour length between epochs. The rule of thumb that we adopted was to let each annealing schedule consist of 25 temperatures.

Because the number of swaps per epoch affects the running time, we varied this parameter in order to examine not only its effect on running time, but also its effect on the quality of the solution. All solution times, unless otherwise noted, are virtual CPU seconds on an IBM 4341 running under VM/CMS. Solution times include only time spent executing the modified 2-OPT procedure and exclude all input-output times and distance matrix calculations.

In addition to varying the number of swaps per epoch, we examined the effect of small changes to the annealing schedule by defining two perturbation conditions. The first of these, termed the **+1** condition consists of executing the procedure with the experimentally determined annealing schedule except that we *add* one to every temperature greater than one. In a similar fashion, we define the **-1** condition by *subtracting* one from every temperature greater than one. Finally, for all problem sizes and all test conditions the procedure was executed using, as a starting tour, the tour resulting from an application of the simulated annealing algorithm. This condition was termed **Restart.** The annealing schedule used for the Restart condition was a second experimentally derived schedule that started at a much lower temperature than the first schedule. The initial temperature for the Restart condition was calculated by substituting the average arc length from the resulting tour in the probability expression and solving for the temperature. Determination of the other temperatures was as previously described.

The results for the randomly generated test problems, for all test conditions, are summarized in Table 1. The Best Known Solution (column 2) used for purposes of comparison was determined by repeated application of the OROPT branch exchange procedure (see

			TEST CONDITION								
No. of Nodes	Best Known Solution	Swaps/ Epoch	Base (% over)	CPU* (secs)	+1 (% over)	CPU* (secs)	-1 (% over)	CPU* (secs)	Restart (% over)	Condition	CPU* (secs)
60	290.28	50	4.69	290.86	3.39	181.02	$\boxed{.00}$	$\boxed{829.88}$	.01	-1	475.62
		25	3.42	181.02	.16	599.93	3.99	224.88	.16	+1	205.85
		15	.32	359.06	.95	287.08	1.93	201.68	.32	B	118.47
		5	3.71	163.57	4.89	129.47	3.42	141.44	3.36	B	51.91
70	300.21	50	$\boxed{.00}$	$\boxed{1296.91}$	5.96	1314.25	.69	1217.23	.01	B	472.39
		25	3.17	511.25	5.04	332.78	2.04	777.39	2.04	-1	151.11
		15	3.75	603.43	4.39	344.24	4.93	406.67	3.75	+1	157.64
		5	8.28	287.23	5.94	354.23	4.16	232.72	4.16	-1	89.02
75	303.09	50	5.32	1000.88	3.66	984.62	$\boxed{2.23}$	$\boxed{921.44}$	3.27	+1	441.61
		25	3.62	485.22	6.36	423.21	8.87	482.56	3.62	B	221.52
		15	10.60	449.33	12.59	416.22	12.94	421.63	8.87	-1	191.44
		5	2.59	384.39	2.37	401.21	6.24	432.23	2.37	B	152.66
80	314.15	50	6.82	495.10	5.78	1168.02	4.53	1058.32	3.36	+1	254.35
		25	12.84	563.21	4.51	898.35	7.08	777.81	5.00	+1	138.13
		15	5.78	495.22	3.73	421.54	5.11	558.07	4.22	+1	123.75
		5	$\boxed{1.90}$	$\boxed{401.11}$	9.22	509.38	7.44	455.58	1.90	B	148.23
85	323.45	50	3.65	1710.89	.24	1820.62	2.06	1611.23	1.44	+1	282.11
		25	8.57	1002.11	2.31	1142.75	$\boxed{.20}$	$\boxed{1278.77}$	.04	-1	221.11
		15	7.83	1222.48	3.20	942.48	5.39	1124.93	1.58	-1	428.62
		5	4.69	848.56	5.29	641.22	2.08	600.59	1.85	-1	132.34

* IBM 4341   Best Solution for Problem $\boxed{\text{Highlighted}}$

TABLE 1
SUMMARY OF COMPUTATIONAL RESULTS
(CONCLUDED)

Or (1976)). Note that in these runs we adopted a strict definition of equilibrium in that we set ε equal to unity. We later present results based on a relaxation of this definition.

In evaluating the results, we seek to answer several questions. *First*, what is the relationship between problem size and running time? Kirkpatrick *et al.* report that the computational effort grows as N (the number of nodes), or as a small power of N. *Second*, how do the solution quality and running times compare with other heuristics for the TSP? *Third*, what effect do small perturbations of the annealing schedule have on running time and solution quality? *Finally*, does a variation in the number of swaps per epoch have a significant impact on the solution quality? We examine these topics next.

4. ANALYSIS OF RESULTS

We first examine the impact of problem size on the computational effort. Considering each test condition independently, an examination of Table 1 reveals, not surprisingly, that as the number of swaps per epoch is lowered, the solution time generally decreases. Also, examining the solution times for a fixed number of swaps per epoch, we find that solution time increases as the problem size increases. Figures 2, 3, and 4 display these relationships graphically for each test condition.

These graphs show that as the number of swaps per epoch is lowered, the solution time becomes less. Clearly, five swaps per epoch is most attractive from the point of view of solution time for all test conditions. In order to quantify some of the above observations, a least squares fit of the solution time as a function of problem size was performed for all test conditions. Simple power functions were used. Tables 2, 3, and 4 summarize the results.

For the unperturbated schedule, see Table 2, excellent fits are evidenced with no r^2 below .89. We see that for all number of swaps per epoch conditions the computational effort scales at a rate greater than cubic.

Looking at the +1 condition results, see Table 3, the fits are similar. Three of the four r^2 values are greater than or equal to .90 with the remaining r^2 value only slightly less at .88. Again the rate of increase is greater than cubic.

The fits for the -1 condition, shown in Table 4, are not as clear. Here we see that the the highest r^2 value occurs at 15 and 5 swaps per epoch. Looking at Figure 4, we see a rather large variability in the solution times as the problem size increases for both the 25 and 50 swap per epoch conditions. This variability contributes to the fits. Note that for 15 and 5 swaps per epoch, the rate of increase for the power function is, again, cubic or approximately cubic.

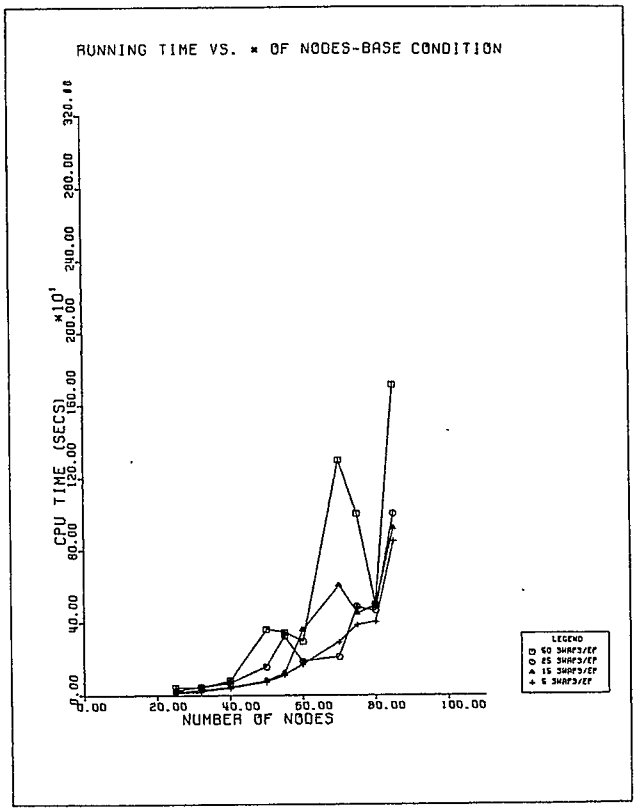


FIGURE 2
PLOT OF RUNNING TIME VS. NUMBER OF NODES – BASE CONDITION

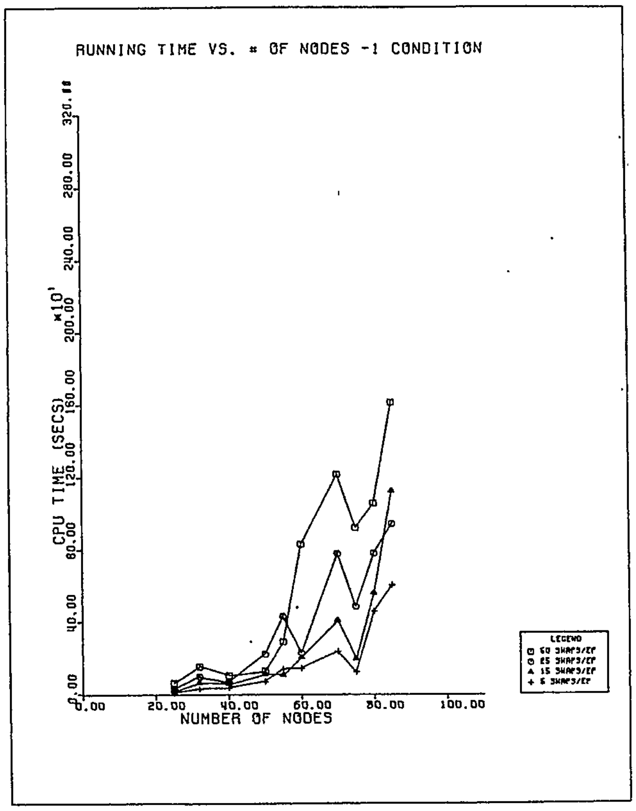


FIGURE 4
PLOT OF RUNNING TIME VS. NUMBER OF NODES – –1 CONDITION

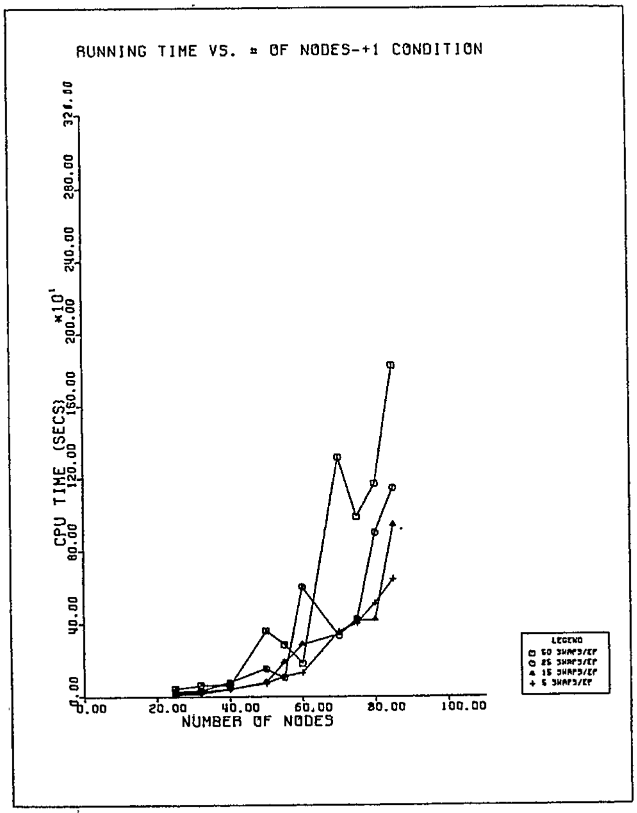


FIGURE 3
PLOT OF RUNNING TIME VS. NUMBER OF NODES – +1 CONDITION

Swaps/ Epoch	$y = a\,x^b$		
	r^2	a	b
50	.89	$1.19 \cdot 10^{-3}$	3.12
25	.96	$1.08 \cdot 10^{-3}$	3.04
15	.93	$2.20 \cdot 10^{-4}$	3.40
5	.98	$8.92 \cdot 10^{-5}$	3.53

x = Problem Size

y = Solution Time (cpu secs)

TABLE 2
RESULTS OF CURVE FITTING – BASE CONDITION

In examining the second question, we wish to compare these results with those of a specialized heuristic for the TSP. The method we choose is the CCAO procedure developed by Stewart (1977) because of its ease of implementation, fast running time, and its documented ability to produce high quality solutions (See Golden and Stewart (1983)). The CCAO heuristic is a hybrid procedure that uses the convex hull of points as the starting subtour and inserts nodes using a combination of the Greatest Angle method (Norback and Love (1979)) and the cheapest insertion criteria. Finally, a branch exchange heuristic, the OROPT procedure (Or (1976)), is used as a post-processor. The results of applying this heuristic to the ten test problems are summarized in Table 5.

Table 5 shows that the CCAO procedure failed to

Swaps/ Epoch	$y = a\,x^b$		
	r^2	a	b
50	.88	$8.07 \cdot 10^{-4}$	3.23
25	.90	$5.73 \cdot 10^{-4}$	3.20
15	.96	$3.59 \cdot 10^{-4}$	3.25
5	.98	$6.63 \cdot 10^{-5}$	3.61

x = Problem Size

y = Solution Time (cpu secs)

TABLE 3
RESULTS OF CURVE FITTING – +1 CONDITION

Swaps/ Epoch	$y = a\,x^b$		
	r^2	a	b
50	.85	.01	2.71
25	.68	.01	2.47
15	.93	$1.33 \cdot 10^{-3}$	2.95
5	.98	$1.43 \cdot 10^{-4}$	3.41

x = Problem Size

y = Solution Time (cpu secs)

TABLE 4
RESULTS OF CURVE FITTING – –1 CONDITION

No. of Nodes	Best Known Solution	% over	Cpu (secs)
25	297.03	.00	0.76
32	390.89	.30	1.28
40	234.36	.00	3.17
50	268.01	.00	5.77
55	280.15	1.37	7.17
60	290.28	.00	16.61
70	300.21	.00	27.88
75	303.09	.32	23.78
80	314.15	.00	30.59
85	323.45	.70	44.59

TABLE 5
COMPUTATIONAL RESULTS FOR CCAO HEURISTIC

find the best known solution on only four out of ten problems with the percentage above the best known solution being less than 1.5% for those problems. The running times are also very reasonable. The curve fitting experiments show that run time as a function of number of nodes fits the power function $y = 1.1E\text{-}05\ x^{3.41}$ with an r^2 of .98. While the computational effort of the CCAO scales closely with that of the simulated annealing approach, the rate of increase is always less and, as we will show, its behavior, in terms of both accuracy and efficiency, is considerably more consistent than that of the simulated annealing method.

In order to rigorously examine the effects of perturbations of the annealing schedule and swaps per epoch, we apply tests from nonparametric statistics. Such methods have been used to compare heuristic procedures for combinatorial optimization problems with respect to a number of criteria (see Wasil, Golden, and Assad (1983)). These tests are particularly suited to this application as they overcome the problems associated with large fluctuations in the data that may occur across problem sets. We use the nonparametric test due to Friedman (see Conover (1980)) with $\alpha = .05$.

Focusing on solution quality, we test the following null hypothesis for each of the three annealing schedules (Base, +1, –1)

H_0: ***The variation in the number of swaps per epoch produces identical results***

against the alternative

H_1: ***At least one swaps per epoch condition tends to yield better results than at least one other condition.***

As Table 6a shows, we do not reject the null hypothesis for any of the test conditions. From Table 1 and Table 6a, we conclude that none of the number of swaps per epoch conditions produces consistently good results. Figures 5, 6, and 7 graphically depict the variability resulting from altering the number of swaps per epoch.

Considering the effect of perturbing the annealing schedule, we test the following null hypothesis for each number of swaps per epoch condition

H_0: ***The three test conditions, Base, +1, –1, yield identical solutions***

against the alternative

H_1: ***At least one test condition tends to yield better solutions than at least one other test condition.***

Table 6b summarizes the results and shows that we do not reject the null hypothesis. As Figure 8 shows, perturbing the annealing schedule produces varied and rather unpredictable results.

In a similar manner, we test for differences in running time by first testing the following null hypothesis for each of the three annealing schedules (Base, +1, –1)

H_0: ***The variation in the number of swaps per epoch produces identical running times***

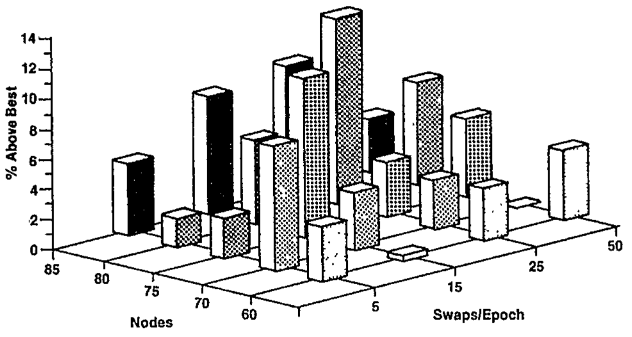
	TEST CONDITIONS		
	BASE	+1	–1
Result	Accept H_0	Accept H_0	Accept H_0

(a)

	SWAPS PER EPOCH			
	50	25	15	5
Result	Accept H_0	Accept H_0	Accept H_0	Accept H_0

(b)

TABLE 6
STATISTICAL TESTS OF SOLUTION QUALITY



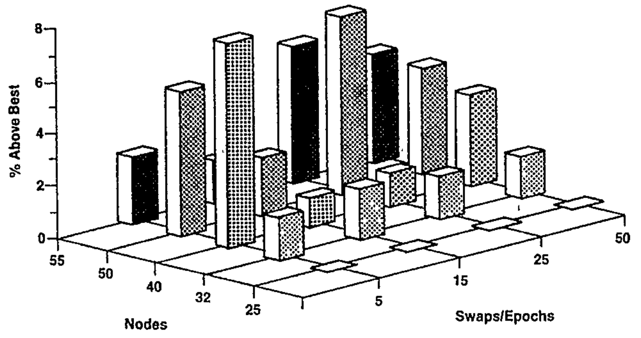
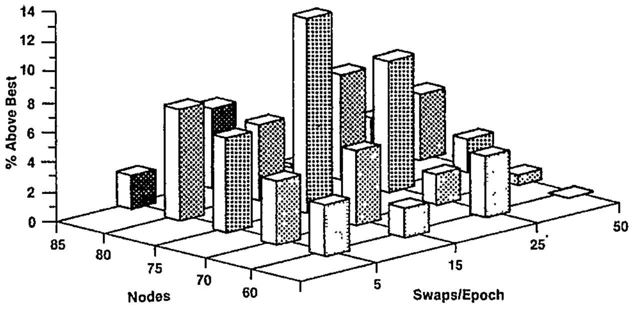


FIGURE 5
VARIABILITY IN SOLUTION QUALITY – BASE CONDITION



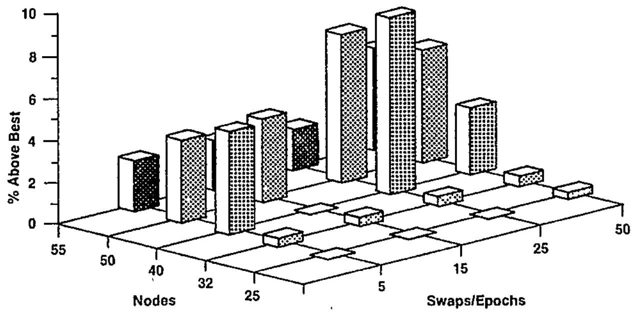
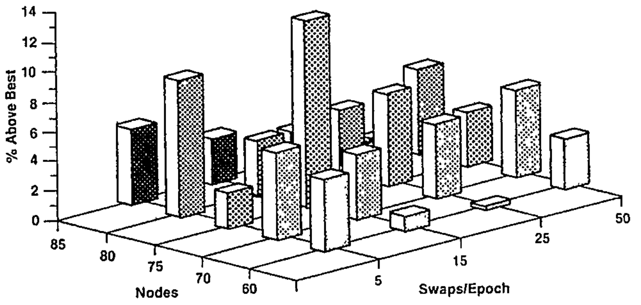


FIGURE 7
VARIABILITY IN SOLUTION QUALITY – –1 CONDITION



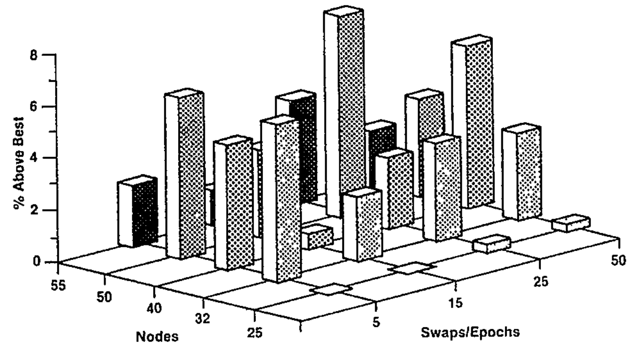


FIGURE 6
VARIABILITY IN SOLUTION QUALITY – +1 CONDITION

against the alternative

H_1: ***At least one swaps per epoch condition yields a larger running time than at least one other condition.***

The results, shown in Table 7a, indicates that we reject the null hypothesis for all test conditions. Furthermore, a test for multiple comparisons (Conover (1980)), points out the significantly different pairs. As expected, running time generally decreases as the number of swaps per epoch is decreased.

To examine the effect that perturbing the annealing schedule has on running time, we test the following null hypothesis

H_0: ***The three test conditions, Base, +1, -1, yield identical running times***

against the alternative

H_1: ***At least one test condition tends to yield a higher running time than at least one other condition.***

Table 7b shows that for 15 swaps per epoch, the null hypothesis is rejected with the significantly different pairs displayed in the appropriate columns. Clearly, for some swaps per epoch conditions, the method shows a sensitivity in running time to slight changes in the annealing schedule. The reason for this is not entirely clear.

The last test condition we examined was termed **Restart**. In this setting the tour resulting from the application of simulated annealing was used as the starting tour for yet another application of the method with a different annealing schedule. This was done for each test condition and number of swaps per epoch. The results are summarized in Table 1.

The data for **Restart** in Table 1 reports the percentage above the best known solution and the test condition that yields the *best* result. We note that in less than ten percent of the cases did **Restart** produce tour lengths greater than those with which it was started. Also, in only four cases did the solution time of the Restart condition exceed the solution time of the randomly started problem.

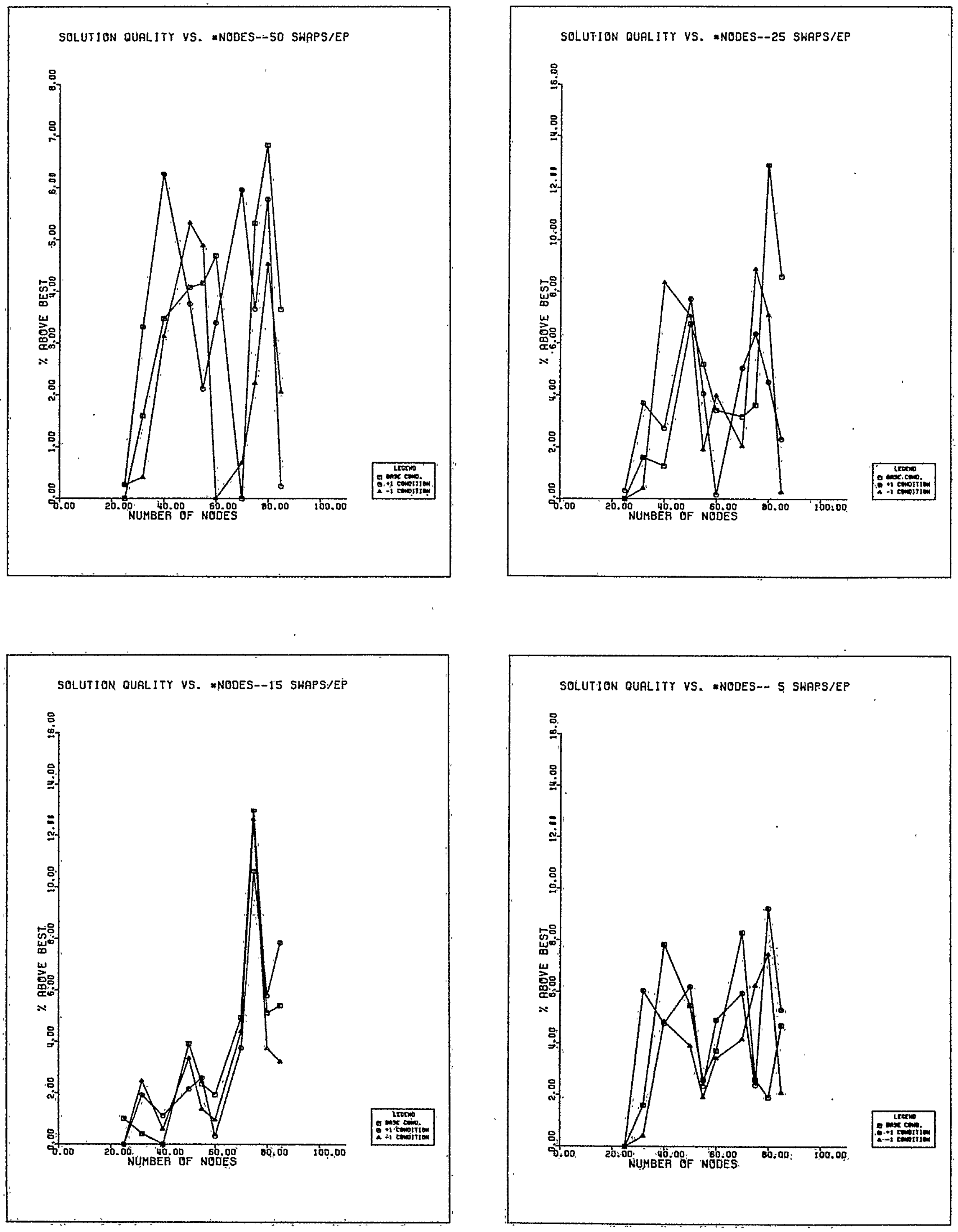


FIGURE 8
VARIABILITY AMONG TEST CONDITIONS

	TEST CONDITIONS		
	BASE	+1	-1
Significantly Different Pairs	(5,15)	(5,50)	(5,25)
	(5,15)		(5,50)
	(5,50)		(15,50)
			(15,50)

(a)

	SWAPS PER EPOCH			
	50	25	15	5
Significantly Different Pairs		(BASE, -1)	(BASE, -1)	
		(+1, -1)	(+1, -1)	

(b)

TABLE 7
STATISTICAL TESTS OF RUNNING TIMES

Since the simulated annealing method for the TSP is based on the 2-OPT procedure, we now wish to compare the performance of repeated applications of the 2-OPT procedure and the simulated annealing approach. For each problem listed in Table 1, we executed the 2-OPT from 15 randomly generated initial tours and recorded the cumulative cpu seconds expended and the percentage above the best known solution thus obtained every five executions of the algorithm. This permits us to compare the performance of the 2-OPT and simulated annealing methods for somewhat comparable expenditures of computational effort. Table 8 reports the outcome.

We perform the comparison by matching, as closely as possible, the cpu times from the best answer obtained by the simulated annealing approach with a comparable cpu expenditure by the 2-OPT algorithm. We note that this comparison is biased in favor of the simulated annealing method as we are not adding the set-up times or the experimentation with number of swaps per epoch and other user controlled parameters required to produce the solution. On only *one problem out of ten* did the simulated annealing approach outperform the 2-OPT procedure.

In order to reduce the solution times for the simulated annealing approach, we decided to experiment with modifying the definition of equilibrium as expressed in Step 2 of ***SA'***. Instead of requiring a nearly exact match of a current tour length to any one of the previously encountered tour lengths, we now require that the *percentage* difference in the tour lengths be less than a specified amount. Specifically, we replace Step 2 of ***SA'*** with

Step 2a. *Equilibrium Test.* If $(l_j - L)/L \le \varepsilon$ $(0 < \varepsilon < 1)$, for any $l_j \in L^t$ then go to ***Step 3***; otherwise perform the updates as in ***Step 2*** of ***SA'***.

In general, we expect faster convergence as ε is increased. In order to examine the effect that ε has, we tested the five 100 node problems taken from Krolak, Felts, and Marble (1971) with various ε settings using 50 swaps per epoch. The results are displayed in Table 9. We note that adopting the previous definition of equilibrium produced solution times in excess of one cpu *hour* so the times reported in Table 9 represent a substantial improvement.

Statistical tests, similar to those described

No. of Nodes	Best Known Solution	2-OPT Runs	Best (% over)	CPU (secs)	No. of Nodes	Best Known Solution	2-OPT Runs	Best (% over)	CPU (secs)
		5	3.70	2.57			5	1.53	200.12
25	297.03	10	2.89	7.22	60	290.28	10	1.53	403.31
		15	0.00	10.29			15	0.00	604.92
		5	0.00	8.64			5	1.46	380.14
32	390.89	10	0.00	18.42	70	300.21	10	0.69	766.19
		15	0.00	28.24			15	0.00	1152.34
		5	2.33	31.96			5	3.86	533.25
40	234.36	10	0.78	68.82	75	303.09	10	2.86	1056.50
		15	0.00	104.82			15	1.51	1598.25
		5	0.62	84.94			5	2.41	673.45
50	264.15	10	0.62	174.17	80	314.15	10	0.81	1373.43
		15	0.00	260.32			15	0.81	1905.98
		5	0.59	122.65			5	3.10	737.48
55	283.98	10	0.00	255.39	85	323.45	10	1.56	1495.93
		15	0.00	386.12			15	0.82	2257.08

TABLE 8
REPEATED APPLICATION OF THE 2-OPT ALGORITHM

			ε = 25%		ε = 10%		ε = 5%		ε = 1.0%		ε = 0.5%	
PROBLEM No.*	NUMBER OF NODES	OPTIMAL SOLUTION	% ABOVE OPTIMAL	CPU** (MINS)	% ABOVE OPTIMAL	CPU** (MINS)	% ABOVE OPTIMAL	CPU** (MINS)	% ABOVE OPTIMAL	CPU** (MINS)	% ABOVE OPTIMAL	CPU** (MINS)
24	100	21282	3.35	16.92	$\boxed{1.34}$	$\boxed{22.60}$	1.37	29.59	3.63	17.35	1.98	25.67
25	100	22148	3.93	18.79	1.70	17.70	$\boxed{1.40}$	$\boxed{20.22}$	4.21	33.16	10.00	17.08
26	100	20749	7.75	19.30	$\boxed{1.48}$	$\boxed{28.89}$	4.48	29.40	2.07	30.46	2.60	46.01
27	100	21294	4.32	20.52	$\boxed{3.43}$	$\boxed{22.98}$	5.00	31.58	9.60	22.69	9.61	21.98
28	100	22068	4.33	16.49	$\boxed{2.39}$	$\boxed{18.90}$	9.66	14.37	7.55	27.75	2.87	20.22

* KROLAK, FELTS, AND MARBLE (1971)

** VAX 11/780

BEST SOLUTION IN ROW IS $\boxed{\text{HIGHLIGHTED}}$

TABLE 9
COMPUTATIONAL RESULTS FOR VARIOUS ε VALUES

Problem No.*	No. of Nodes	% Above Optimal	Cpu** (mins)
24	100	7.53	13.97
25	100	4.32	14.25
26	100	6.56	13.94
27	100	3.56	8.85
28	100	5.28	16.04

* From Krolak, Felts, and Marble (1971)

**VAX 11/780

TABLE 10
COMPUTATIONAL RESULTS FOR 50 SWAPS PER EPOCH AT EACH TEMPERATURE

Problem No.*	No. of Nodes	CCAO % Above Optimal	Cpu** (mins)
24	100	0.01	1.80
25	100	2.76	2.02
26	100	0.83	1.74
27	100	1.35	1.70
28	100	1.72	1.94

* From Krolak, Felts, and Marble (1971)

**VAX 11/780

TABLE 11
COMPUTATIONAL RESULTS FOR CCAO HEURISTIC

previously, revealed that an ε of 10% consistently outperformed all others with respect to solution accuracy. There was no significant difference among running times for the various ε values, however, if one computes the total solution time to solve the five problems at each ε value, the times increase as ε decreases.

In an attempt to further reduce the solution times, we relax the equilibrium condition by only considering 50 swaps at each temperature setting. Again, we use the five 100 node problems. Table 10 reports the results. Statistical tests show that the running times for this condition are significantly less than those reported on in Table 9. The accuracy, however, was significantly *worse* than the ε = 10% condition only.

Comparing these results with those of the CCAO heuristic, Table 11 shows that the solution times are consistently small fractions of the time required by the simulated annealing method. In only one case, did the simulated annealing method turn in a higher quality solution.

5. CONCLUSIONS

This paper has presented a preliminary computational study of the Kirkpatrick *et al* approach to combinatorial optimization as applied to the TSP. Through an analogy to physical systems and statistical mechanics, a general heuristic method for combinatorial optimization is proposed. This method is intended to serve as a framework within which problem-specific methods can be embedded. The results of our investigation can be summarized as follows:

- Using a definition of equilibrium similar to the one advocated by Kirkpatrick *et al.*, the computational effort grows faster than cubically. The CCAO heuristic is faster and consistently produces tours of higher quality.

- Despite persistent efforts, we failed to find a set of annealing schedule parameters that performed consistently well.

- The solution times displayed a marked sensitivity to minor perturbations in the annealing schedule and to variations in the number of swaps per epoch.

- For comparable amounts of computer time, repeated application of the 2-OPT algorithm outperformed the simulated annealing method on *nine out of ten* randomly generated test problems.

- For the **Restart** condition, there was no significant improvement in accuracy over the initially generated simulated annealing solution.

- A slightly modified definition of equilibrium resulted in substantially reduced computation time for the 100 node problems. Interestingly, an ε of 10% significantly outperformed all other ε values with respect to solution accuracy.

- Further relaxation of equilibrium, in which we consider 50 swaps per epoch at each temperature, turned in the lowest solution times for the 100 node problems. In terms of accuracy, this alternative was no worse than any of the other ε conditions except for $\varepsilon =$ 10%.

In sum, we found the simulated annealing procedure to be sensitive to a number of control parameters and stopping rules and we were unable to find an implementation strategy that consistently performed well. Furthermore, the CCAO heuristic and repeated application of the 2-OPT algorithm outperformed the simulated annealing procedure for a comparable amount of computational effort.

To be fair, we should point out that traveling salesmen problems have been studied by an extremely large number of management and computer scientists. It is perhaps unfair to expect a new approach such as simulated annealing to compete with the best of an almost endless array of TSP heuristics without extensive fine tuning. With this in mind, as well as an appreciation for the notion of randomizing in order to avoid inferior local minima, we plan to pursue the following topics in future research:

- Experiment with larger problems.

- Construct an annealing schedule by reducing the temperature by a fixed percentage at each step.

- Examine a number of alternative rules for equilibrium.

- Experiment with different density functions for the probability of accepting an increase in tour length. It would also be interesting to investigate probabilistically accepting small decreases in the objective function as well as increases.

- Study the effect of probabilistically increasing the objective function only upon reaching or getting close to a local minimum.

Acknowledgment

The authors wish to thank The MITRE Corporation and the University of Maryland for financial assistance.

References

W. R. Stewart, "A Computationally Efficient Heuristic for the Traveling Salesman Problem", **Proceedings of the 13th Annual Meeting of Southeastern TIMS**, Myrtle Beach, S.C., 75-83 (1977).

W. R. Stewart, "A Computational Comparison of Five Heuristic Algorithms for the Euclidean Traveling Salesman Problem", **Evaluating Mathematical Programming Techniques**, (Ed., John Mulvey), New York, Springer-Verlag, (1982).

E. Wasil, B. Golden, and A. Assad, "Alternative Methods for Comparing Heuristic Procedures on the Basis of Accuracy", University of Maryland, Management Science and Statistics Working Paper **#MS/S 83-012** (1983).

W. J. Conover, **Practical Nonparametric Statistics**, New York, John Wiley and Sons (1980).

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

B. L. Golden and W. R. Stewart, "The Empirical Analysis of TSP Heuristics", forthcoming in **The Traveling Salesman Problem** , E. Lawler, J.K. Lenstra, and A. H. G. Rinnooy Kan, eds., (1983).

S. Kirkpatrick, C. D. Gelatt, Jr., and M. P. Vecchi, "Optimization by Simulated Annealing", IBM Computer Science/Engineering Technology Report, IBM Thomas J. Watson Research Center, Yorktown Heights, New York (1982).

S. Kirkpatrick, C. D. Gelatt, Jr, and M. P. Vecchi, "Optimization by Simulated Annealing", **Science, 220,** No. 4598, 671-680 (1983).

P. Krolak, W. Felts, and G. Marble, "A Man-Machine Approach Toward Solving the Traveling Salesman Problem", **Communications of the ACM, 14,** 327-334 (1971).

S. Lin, "Computer Solutions of the TSP", **Bell Systems Technical Journal, 44,** 2245-2269 (1965).

N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller, "Equation of State Calculations by Fast Computing Machines", **Journal of Chemical Physics, 21,** 1087 (1953).

J. P. Norback and R. P. Love, "Geometric Approaches to Solving the Traveling Salesman Problem", **Management Science, 23,** 1208-1223 (1977).

I. Or, "Traveling Salesman-Type Combinatorial Problems and Their Relation to the Logistics of Blood Banking", Ph.D. Thesis, Dept. of Industrial Engineering and Management Sciences, Northwestern University (1976).