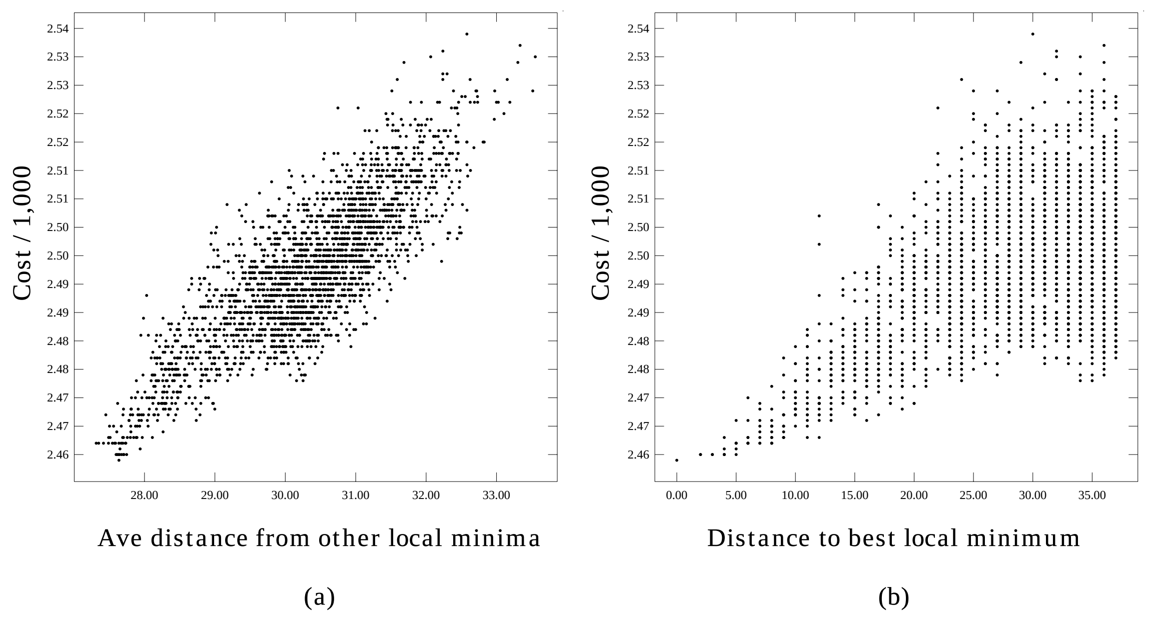

Figure 3: Analysis of 2,500 random locally minimum bisections for graph in $G(150, 0.5)$. The data represent 2,399 distinct local minima.

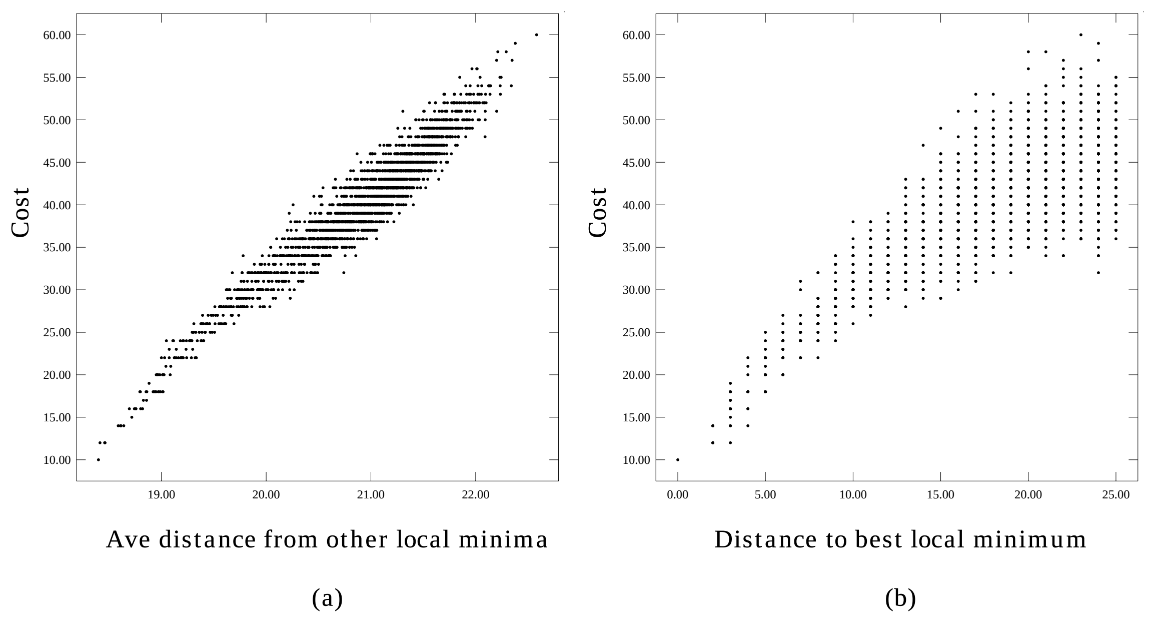

Figure 4: Analysis of 2,500 random locally minimum bisections for graph in $G_{Bui}(100, 4, 10)$. The data represent 2,343 distinct local minima.

3 Exploiting Global Structure: Adaptive Multi-Start

It is natural to wonder whether more effective starting solutions for **Greedy_Descent** can be derived if we assume that a "big valley" structure holds for the set of local minima. In this section, we consider a simple instance of such an *Adaptive Multi-Start* (AMS) methodology and demonstrate its effectiveness in practice.

3.1 A Simple Adaptive Multi-Start (AMS) Heuristic

We have implemented a simple AMS heuristic consisting of two phases:

1. **Phase One:** Generate R random starting solutions and run **Greedy_Descent** from each to determine a set of corresponding *random local minima.*

2. **Phase Two:** Based on the local minima obtained so far, construct *adaptive* starting solutions and run **Greedy_Descent** A times from each one to yield corresponding *adaptive local minima.*

Intuitively, the two phases respectively develop, then exploit, a structural picture of the cost surface. Our AMS heuristic is more precisely described in Figure 5. (Note that henceforth our discussion is couched with respect to the symmetric TSP.)

```
Adaptive_Multi-Start (G,D,R,k,A)
Input: TSP instance $G = (V, E)$ with $|V| = n$
          $D \equiv$ limit on number of descents $\equiv$ CPU budget
          $R \equiv$ number of descents from random starting tours
          $k \equiv$ number of local minima used to construct each adaptive tour
          $A \equiv$ number of descents made from each adaptive starting tour
Output: $t^* \equiv$ best local minimum found after $D$ descents
Local Variables:
          $M \equiv$ set of $k$ best local minimum tours so far
/*   (Phase One)   */
1.  Generate $R$ random local minima
/*   (Phase Two)   */
2.  repeat
3.      Update $M$
4.      $t \leftarrow$ Construct_Adaptive_Starting_Tour($M$)
5.      Run Greedy_Descent($t$) $A$ times
6.      Update $t^*$
7.  until $D$ total descents.
```

```
Subroutine Construct_Adaptive_Starting_Tour(M)
Input: Set $M = \{M_1, \ldots, M_k\}$ of locally minimum tours
Output: Set $t$ of edges forming a new starting tour
S1.  In = union of all edges in set of tours $M$
S2.  $t = \emptyset$.
S3.  Assign weight $w(e_i)$ to each edge $e_i \in$ In
S4.  for $e_i \in$ In in order of decreasing $w(e_i)$ do
S5.      if $e_i$ is a valid tour edge with respect to $t$ then
S6.          $t = t \cup \{e_i\}$ with probability $Pr(e_i)$
S7.  while $|t| < n$ (i.e., $t$ is not yet a tour) do
S8.      Add a randomly chosen valid tour edge from $E \setminus t$
```

Figure 5: **Adaptive_Multi-Start** template.

In the Figure 5 template, the term *descent* denotes a single execution of **Greedy_Descent**. We use D to denote the total number of calls to **Greedy_Descent**. The number of passes through Phase Two (Lines 3-6) is determined by the relationship $passes = \lceil \frac{D-R}{A} \rceil$. In obtaining the results of Section 4, we uniformly use $R = D/2$ (i.e., we spend exactly half our CPU budget in Phase One) and $A = 10$. When $D-R$ is not an exact multiple of A we truncate the final pass in Line 5. The subroutine **Construct_Adaptive_Starting_Tour** always constructs an adaptive starting tour from the set of k best local minima found so far; we use $k = 10$ in our experiments. In the description of **Construct_Adaptive_Starting_Tour**, a *partial tour* is a set of edges that is a subset of the edges in some tour. Given a partial tour t, edge e is a *valid tour edge* with respect to t iff $t \cup \{e\}$ is a partial tour. The experimental results in Section 3.2 below were obtained using the following additional implementation details:

1. In Line S4, $w(e_i)$ is the sum of the inverse tour costs for tours in which e_i appears, i.e.,
$$w(e_i) = \sum_{M_j \ni e_i} \frac{1}{cost(M_j)} \tag{1}$$
2. In Line S6, we use
$$Pr(e_i) = \exp(\frac{w(e_i)}{W} - 1) \tag{2}$$
where $W = \sum_{j=1}^{|M|} \frac{1}{cost(M_j)}$ is the weight of an edge that is contained in all tours in M.
3. In Lines S7-S8, we simply insert random valid edges until t becomes a tour.

(In Line S4, our definition of $w(e_i)$ gives greater weight to edges that are present in many short tours; however, we have obtained similar results using uniform edge weights. In Line S6, while we have also tried other probabilistic weighting methods, our choice allows the probability that e_i is included in t to approach 1 as $w(e_i)$ approaches its maximum possible value. In Lines S7-S8, we have found that a number of other strategies yield very similar results, e.g., adding the shortest valid edge, testing edges according to a fixed order, or following a nearest-neighbor heuristic.)

Given these implementation decisions, **Construct_Adaptive_Starting_Tour** has $O(kn \log n + n^2)$ worst-case run time. Of course, this is dominated by the known exponential worst-case run time of **Greedy_Descent** for 2-opt in the TSP [20].

3.2 Experimental Results

Table 1 compares results of our AMS implementation with results of Random multi-start and Nearest Neighbor (NN) multi-start (i.e., multi-start from initial tours obtained by the nearest-neighbor TSP heuristic). NN tours are suggested in [10] for obtaining initial tours for 2-opt descents; although Bentley [4] has since shown that the greedy TSP tour is a slightly better starting point for a single descent, it may be less appropriate

# Descents used in AMS	Equivalent # Descents Random strategy	Equivalent # Descents NN strategy	Average Tour Cost (AMS)	% Above Held-Karp Lower Bound
1	1	1	8.5404	10.91
50	442	50	7.9835	3.68
100	1508	176	7.9266	2.94
150	> 4500	426	7.8967	2.55
200	> 5700	826	7.8806	2.34
400	> 7800	> 3000	7.8555	2.02
600	> 9100	> 4000	7.8367	1.78
1000	> 10000	> 4500	7.8275	1.66

Table 1: Experimental results for AMS executions on 50 random 100-city Euclidean TSP instances from the unit square. Results show the number of descents needed for Random multi-start or Nearest-Neighbor-based multi-start to achieve the same solution quality. Random multi-start was run for 3,000 descents and Nearest-Neighbor multi-start for 2,000 descents on each of the 50 instances. Entries with ">" are conservative estimates based on linear extrapolation. Average tour costs and their relationship to the expected Held-Karp lower bound on TSP cost (provided by D. S. Johnson [12]) are shown in order to facilitate comparison with other work, e.g., [10].

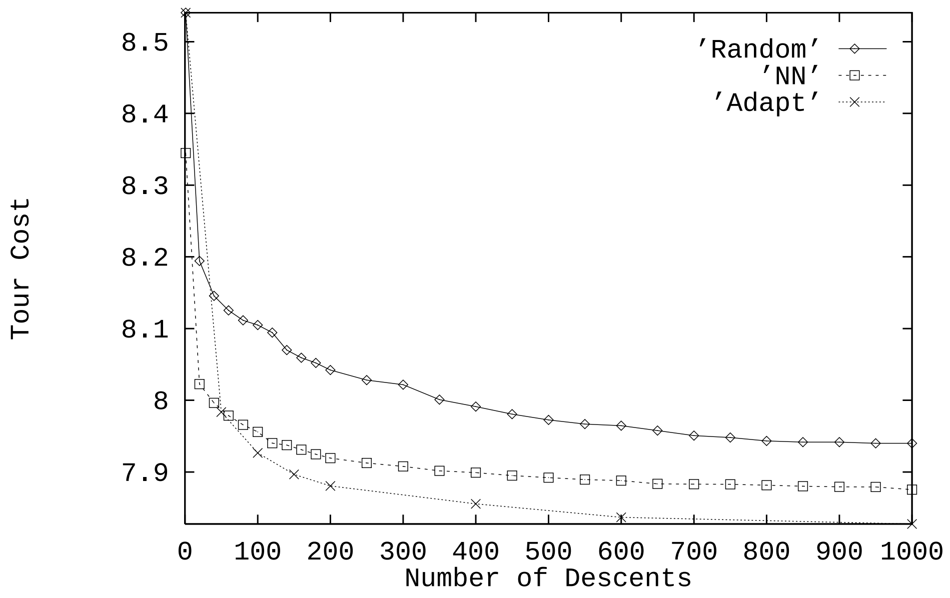

Figure 6: Graphical comparison of AMS, Random multi-start, and Nearest-Neighbor multi-start approaches averaged over 50 random 100-city Euclidean TSPs instances.

for multi-start because there is only one greedy tour of a given instance versus up to n different NN tours. As problem size and the CPU budgets D increase, we obtain significant improvements over the current methods, e.g., for 100-city TSP instances, our adaptive strategy achieves in only 200 descents what random multi-start would require over 5,700 descents to achieve. A more detailed portrait of this data is given in Figure 6.

Table 2 shows the stability of our AMS method, where we define stability to be the standard deviation

	Adaptive		NN		Random	
	mean	std dev	mean	std dev	mean	std dev
D=1			7.713	0.171	8.015	0.265
D=100	7.412	0.030	7.492	0.020	7.597	0.011
D=200	7.393	0.020	7.456	0.019	7.552	0.009

Table 2: Stability comparison of different multi-start strategies on a *single* random 100-city TSP instance. Mean and standard deviation of solution cost are computed for 50 executions of each strategy.

	50-city instance			**100 city instance**		
Starting Tours	**Random**	**NN**	**Adaptive**	**Random**	**NN**	**Adaptive**
Distance to Local Min	47.52 (1.32)	13.20 (1.66)	5.18 (1.93)	97.90 (1.34)	27.16 (3.92)	9.69 (5.25)
Ave Dist to Random Mins	47.58 (1.13)	22.99 (0.82)	13.51 (0.88)	98.08 (1.01)	39.01 (1.14)	31.27 (1.52)

Table 3: Mean (standard deviation) of bond distance between 50 starting tours and the corresponding locally minimum tours found by **Greedy_Descent**. Also given are the mean (standard deviation) of the average bond distance between each starting tour and 100 "random" local minima. Sample size is 50 for all data; single random Euclidean instances are used for both $n = 50$ and for $n = 100$.

of $cost(t^*)$. Measured over 50 separate executions on a single 100-city instance, the stability of AMS is very good, with a standard deviation of only 0.030. Although other multi-start methods can have greater stability, AMS maintains its superiority because of its low average solution cost. AMS is also "stably better" than the other methods. For example, in Table 1 AMS with D=200 gave a superior solution to random multi-start in all instances and to NN multi-start in 47 out of 50 instances.

Finally, Table 3 shows the strong relationship between our adaptive starting tours and the corresponding local minima. For the three types of starting tours (Random, NN, Adaptive), the Table shows average bond distance between the starting tour used by **Greedy_Descent** and the locally minimum tour it returns. The Table also compares these distances with the average bond distance between starting tours and 100 "random" local minima obtained by **Greedy_Descent** from 100 different random starting tours. We see that the position of a *random* starting tour has little effect on the position of the locally minimum tour found by **Greedy_Descent**: in fact, this local minimum is no closer to the starting tour than a "random local minimum". By contrast, the NN and adaptive local minima are not only much closer to their starting tours, but are also significantly closer than other "random" local minima. (The high standard deviation of bond distance from the adaptive starting tour to the corresponding local minimum tour is due to our data collection methodology: to obtain 50 distinct adaptive starting tours using the AMS heuristic with the usual $R = D/2$, $A = 10$, etc. we must use $D \approx 1000$, and the quality of the adaptive initial tours increases significantly over the course of this process. We could also obtain the adaptive initial tours by executing AMS 50 separate times with $D = 20$; this yields slightly higher means and lower standard deviations of 7.46 (1.766) for $n = 50$, and 15.20 (3.339) for $n = 100$.)

4 Conclusions

Multi-start greedy optimization has shown much promise in practical applications, but the traditional random multi-start implementation suffers from a "central-limit catastrophe" when problem size grows large. In this paper we address this difficulty with an *adaptive multi-start* methodology that is based on new insights into global structure of optimization cost surfaces.

For instances of the symmetric TSP and graph bisection, we study correlations between the cost of a local minimum and its average distance to all other local minima (as well as its distance to the best-found local minimum). Our analyses show evidence of a "big valley" governing local minima in the optimization cost surface, and motivate the *adaptive multi-start* methodology. In other words, our evidence suggests a *globally convex* [9] structure for the set of local minima (we may make an analogy to the structure of an integer polytope, which "viewed from afar" may appear to have a single minimum point, but which up close has many local minima). Our results may explain why simulated annealing, tabu search, iterated greed, and other hill-climbing heuristics have been so successful in practice: very good solutions are located near other good solutions.

Based on these insights, our Adaptive Multi-Start (AMS) heuristic uses best-known locally minimum solutions to generate starting points for subsequent greedy descents. Experimental evidence for TSP instances shows significant improvement in run time and solution quality over previous multi-start methods (random- and nearest-neighbor-based). Future work should apply our adaptive approach to other greedy methods (e.g., 3-opt and Lin-Kernighan for TSP) and other combinatorial formulations (e.g., graph partitioning and VLSI circuit placement). Our current study also motivates consideration of alternate neighborhood structures which can induce a "big valley" over the solution space.

Appendix 1. Relation between $d(t, t')$ and $b(t, t')$

Recall from Section 2.1 that $d(t, t')$ is the 2-opt distance and $b(t, t')$ is the bond distance between TSP tours.

Theorem 1. For any two tours t_1 and t_2 of a given TSP instance, $\frac{b(t_1,t_2)}{2} \le d(t_1, t_2) \le b(t_1, t_2)$, with the lower bound being tight.

Proof: The lower bound $d(t, t') \ge \frac{b(t,t')}{2}$ follows easily by noting that a single 2-opt affects only two edges in the tour, and can therefore increase the bond distance by at most two. This bound is tight, since a sequence of j 2-opts from t ($1 \le j \le \lfloor \frac{n}{2} \rfloor$), each of which is applied to two edges remaining from t, will yield t' with $d(t, t') = j$ and $b(t, t') = 2j$.

We depict tours as permutations $< 1, \ldots >$ with city 1 listed first, followed by its *lower-index* neighbor. In proving the upper bound, we use a canonical *t-subtour* representation to express t' in terms of t. A *t-subtour* of t' is a sequence of contiguous cities in t that are also contiguous in t'; we use capital letters to label the

t-subtours, with these labels assigned in alphabetic order according to their positions in t. A t-subtour is defined to be *lower* (*higher*) than another t-subtour if its label is closer to the beginning (end) of the alphabet (e.g., B is lower than C). Because all tours are notated as permutations beginning with city 1, the canonical t-subtour representation always begins with A. A t-subtour that appears in t' in reverse order is denoted by a bar above its label. For example, if $t =< 1,2,3,4,5,6 >$ and $t' =< 1,2,4,3,5,6 >$, then the t-subtour representation of t' is $A\overline{B}C$, where $A =< 1,2 >$, $B =< 3,4 >$, and $C =< 5,6 >$. Any 2-opt in t' which preserves t-subtours will reverse a sequence of contiguous t-subtours in t', e.g., a 2-opt involving the two edges separating $A|\overline{B}$ and $C|A$ in $t' = A\overline{B}C$ will yield the tour $A\overline{C}B =< 1,2,6,5,3,4 >$.

We will prove the following three facts (note that $b(t,t') = 1$ is impossible):

- **Fact 1:** If $b(t,t') = 2$, then $d(t,t') = 1$.
- **Fact 2:** If the t-subtour representation of t' contains a reversed t-subtour, then there is a 2-opt which transforms t' to t'' such that $b(t'',t) < b(t',t)$.
- **Fact 3:** If the t-subtour representation of t' contains a reversed t-subtour, then there is 2-opt which transforms t' into t'' such that either (i) $b(t'',t) \leq b(t',t) - 1$ and t'' contains a reversed t-subtour, or (ii) $b(t'',t) = b(t',t) - 2$ and t'' contains no reversed t-subtour.

The following recipe transforms t' to t using at most $b(t,t')$ 2-opts; the recipe relies directly on Facts 1 and 3. (Fact 2 is used in the proof of Fact 3.) Each 2-opt used in the recipe reduces the bond distance from t' to t by either 0, 1, or 2; and every move that leaves the bond distance unchanged can be paired with a move that decreases the bond distance by 2. Thus, we are left only with proving Facts 1 to 3.

```
1.  If $t'$ contains no reversed $t$-subtours
        arbitrarily reverse a $t$-subtour, leaving the bond distance to $t$ unchanged.
2.  If $t'$ contains at least one reversed $t$-subtour, perform a 2-opt that either
        reduces $b(t,t')$ by 1 or 2 and leaves a reversed $t$-subtour; or
        reduces $b(t,t')$ by 2 and leaves no reversed $t$-subtours. [Fact 3]
3.  If $b(t,t') = 2$ perform a 2-opt which transforms $t'$ into $t$. [Fact 1]
4.  Repeat Lines 1 - 3 until $t' = t$.
```

Proof of Fact 1: If $b(t,t') = 2$, then t' can be represented using t-subtours as either $A\overline{B}$ or $A\overline{B}C$. In either case, a single 2-opt reversing B will transform t' into t.

Proof of Fact 2: Let β represent the lowest t-subtour that is reversed in t', i.e., it appears as $\overline{\beta}$, and let α be the (non-reversed) t-subtour immediately preceding β alphabetically. The basic idea is to reduce the bond distance by bringing α and β next to each other using a single 2-opt. If α occurs before $\overline{\beta}$ in t', then a 2-opt which places β directly after α will reduce the bond distance to t by at least one, i.e., $t' = A \ldots \alpha\mathcal{Z}\overline{\beta} \ldots$ becomes $t'' = A \ldots \alpha\beta\overline{\mathcal{Z}} \ldots$, where $\mathcal{Z}$ represents the sequence of t-subtours between α and β in t'. (For example, if $t' = AC\overline{B}$, then $\beta = B$, $\alpha = A$, and $t'' = AB\overline{C}$.) If α occurs after $\overline{\beta}$ in t', then the 2-opt which places $\overline{\alpha}$ directly after $\overline{\beta}$ will reduce the bond distance to t; i.e., $A \ldots \overline{\beta}\mathcal{Z}\alpha \ldots$ becomes $A \ldots \overline{\beta\alpha\mathcal{Z}} \ldots$.

Proof of Fact 3: Suppose t' contains a reversed t-subtour, and that (i) is not true, i.e., there is no 2-opt which transforms t' to t'', with t'' containing a reversed t-subtour and having $b(t'', t) < b(t', t)$. Let $\mathcal{X}$ be the first maximal contiguous sequence of reversed t-subtours in t'. Suppose t' contains a t-subtour outside $\mathcal{X}$. Then any 2-opt will leave a reversed subtour. Since by Fact 2, there must be some 2-opt move reducing the bond distance to t, it must be that all reversed t-subtours in t' are contained in $\mathcal{X}$. As before, let β be the lowest reversed t-subtour in t', let α be the t-subtour preceding β alphabetically, and let $\mathcal{Z}$ be the sequence of t-subtours between α and β, with η denoting another t-subtour which may be present in t'. The template below shows that the following three conditions must hold when there is no 2-opt move which both reduces the bond distance to t and leaves a reversed t-subtour: (a) α appears before $\overline{\beta}$; (b) $\overline{\beta}$ is located at the end of block $\mathcal{X}$; and (c) α is located immediately before block $\mathcal{X}$. If we let γ denote the highest reversed t-subtour in t' and let δ denote the t-subtour after γ alphabetically (we use $\delta = A$ if γ is the highest t-subtour), a similar argument shows that $\overline{\gamma}$ must be located at the beginning of $\mathcal{X}$, and δ must lie immediately after $\mathcal{X}$. Hence, the resulting tour structure allows a 2-opt from $t' = \ldots\alpha\overline{\gamma\mathcal{Z}\beta}\delta\ldots$ to $t'' = \ldots\alpha\beta\mathcal{Z}\gamma\delta\ldots$, which reduces the bond distance to t by 2. Thus, if t' contains a reversed t-subtour and (i) is false, then (ii) must be true.

Case	If () holds	Then $\exists$ 2-opt satisfying (i)
(a)	(α not before $\overline{\beta}$ in t')	$\ldots\overline{\beta}\mathcal{Z}\alpha\ldots \quad\Rightarrow\quad \ldots\overline{\beta\alpha\mathcal{Z}}\ldots$
(b)	($\overline{\beta}$ not at end of $\mathcal{X}$ in t')	$\ldots\alpha\overline{\mathcal{Z}\beta\eta}\ldots \quad\Rightarrow\quad \ldots\alpha\beta\mathcal{Z}\overline{\eta}\ldots$
(c)	(α not next to $\mathcal{X}$ in t')	$\ldots\alpha\eta\overline{\mathcal{Z}\beta}\ldots \quad\Rightarrow\quad \ldots\alpha\beta\mathcal{Z}\overline{\eta}\ldots$

□

Appendix 2. Distribution of Tours in the TSP

In [15], an analysis of the distribution of tours at each bond distance from a given tour is attributed to D. Gross and M. Mezard. By symmetry, this distribution is the same as the distribution of $b(t_1, t_2)$ between random tours t_1 and t_2. The cited result is that the number of common edges between two random tours t_1 and t_2, i.e., $n - b(t_1, t_2)$, approaches a Poisson distribution with mean $= 2$ as $n \to \infty$. We have studied this distribution somewhat more precisely. In what follows, we show that the number of edges in common has mean $= 2 * \frac{n}{n-1}$. We then describe an exact method for efficiently calculating the $b(t_1, t_2)$ distribution.

Fact 4: The expected number of edges in common between two random tours on n cities is $2 * \frac{n}{n-1}$.

Proof: Each edge between cities occurs in exactly $(n-2)!$ different tours. Since each tour has n edges, the expected number of overlaps between a given tour and all tours in the solution space S is equal to $\frac{n(n-2)!}{|S|} = \frac{n(n-2)!}{(n-1)!/2} = 2 * \frac{n}{n-1}$. □

To compute the $b(t_1, t_2)$ distribution, without loss of generality we compute the distribution of $I(n, k)$, where $I(n, k)$ denotes the number of tours on n cities that have k edges in common with the *identity tour* $\sigma(n) =< 1, 2, \ldots, n-1, n >$. Each $I(n+1, k)$ can be calculated exactly based on the observation that any tour t' of $n+1$ cities is uniquely expressible as the insertion of city $n+1$ into a tour t of n cities.

The basic methodology is that of dynamic programming: given the values $I(n,k)$ for $0 \leq k \leq n$ we can compute all the values $I(n+1,k)$ for $0 \leq k \leq n+1$. For a given n, the entire computation of all $I(n,k)$ values requires only $\Theta(n^2)$ time and $\Theta(n)$ space. We use the term *bond* to denote any adjacency in an n-city tour t that is also contained in $\sigma(n)$, i.e., t has k bonds if $b(t,\sigma(n)) = n-k$, or equivalently if there are k edges in common between t and $\sigma(n)$. We say that we *break* a bond in t when we insert city $n+1$ between the two cities forming the bond in t. One readily sees that breaking a bond will usually decrement by one the number of bonds remaining in t'; similarly, inserting $n+1$ next to city 1 or n will generally increment the number of bonds by one, while other positions for city $n+1$ will generally leave the number of bonds unchanged. The analysis considers eight cases for each k, $0 \leq k \leq n$, corresponding to the possible presence or absence of each of three "special" bonds $(1,2)$, $(1,n)$ and $(n-1,n)$. The reader is referred to [6] for details.

References

[1] D.H. Ackley, *A Connectionist Machine for Genetic Hillclimbing*, Kluwer, 1987.

[2] S. Bacci and N. Parga, "Ultrametricity, Frustration and the Graph Colouring Problem," *J. Physics A: Math. and General* **22**, 3023-3032 (1989).

[3] E. B. Baum, "Towards Practical 'Neural' Computation for Combinatorial Optimization Problems", in J. Denker, ed., *Neural Networks for Computing*, American Institute of Physics, 1986.

[4] Bentley, J.L., "Fast Algorithms for Geometric Traveling Salesman Problems", *ORSA Journal on Computing* **4** (4), 387-411 (Fall 1992).

[5] K. D. Boese and A. B. Kahng, "Best-So-Far vs. Where-You-Are: Implications for Optimal Finite-Time Annealing", to appear in *Systems and Control Letters*, 1993.

[6] K. D. Boese, A. B. Kahng and S. Muddu, "On the Big Valley and Adaptive Multi-Start for Discrete Global Optimizations", *technical report TR-930015*, UCLA CS Department, 1993.

[7] B. Bollobas, *Random Graphs*, Academic Press, 1985.

[8] T. N. Bui, S. Chaudhuri, F. T. Leighton, and T. Sipser, "Graph Bisection Algorithms with Good Average Case Behavior", *Combinatorica*, **2** 171-191 (1987).

[9] T. C. Hu, V. Klee and D. Larman, "Optimization of Globally Convex Functions", *SIAM J. on Control and Optimization* **27** (5), 1026-1047 (1989).

[10] D. S. Johnson, "Local Optimization and the Traveling Salesman Problem, in *Proceedings of the 17th International Colloquium on Automata, Languages and Programming*, July 1990, 446-460.

[11] D. S. Johnson, C. R. Aragon, L. A. McGeoch, and C. Schevon, "Optimization by Simulated Annealing: An Experimental Evaluation Part I, Graph Partitioning", *Operations Research* **37** 865-892 (1989).

[12] D. S. Johnson and E. E. Rothberg, "Asymptotic Experimental Analysis of the Held-Karp Lower Bound for the Traveling Salesman Problem" (to appear).

[13] S. Kauffman and S. Levin. "Toward a General Theory of Adaptive Walks on Rugged Landscapes", *Journal of Theoretical Biology* **128**, 11-45 (1987).

[14] S. Kirkpatrick, C. D. Gelatt, and M. Vecchi, "Optimization by Simulated Annealing", *Science*, **220**, 671-680 (1983).

[15] S. Kirkpatrick and G. Toulouse, "Configuration Space Analysis of Traveling Salesman Problems", *Journal de Physique* **46**, 1277-1292 (1985).

[16] E. L. Lawler, J. K. Lenstra, A. Rinnooy-Kan and D. Shmoys, *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*, Wiley, 1985.

[17] M. Mezard and G. Parisi, "A Replica Analysis of the Travelling Salesman Problem", J. Physique **47** 1285-1296 (1986).

[18] H. Mühlenbein, "Parallel Genetic Algorithms, Population Genetics and Combinatorial Optimization," in *Proc. Intl. Conf. on Genetic Algorithms*, 1989, pp. 416–421.

[19] H. Mühlenbein, M. Georges-Schleuter, and O. Krämer, "Evolution Algorithms in Combinatorial Optimization," *Parallel Computing* 7 (1988), pp. 65–85.

[20] C. Papadimitriou, "The Complexity of the Lin-Kernighan Heuristic for the Traveling Saleman Problem", *SIAM J. Computing* **21** (3), 450-465 (1992).

[21] S. A. Solla, G. B. Sorkin and S. R. White, "Configuration Space Analysis for Optimization Problems", in E. Bienenstock et al., eds., *Disordered Systems and Biological Organization*, Springer-Verlag, 1986, 283-292.

[22] G. B. Sorkin, "Efficient Simulated Annealing on Fractal Energy Landscapes", *Algorithmica* **6**, 367-418 (1991).

[23] G. B. Sorkin, *personal communication*, May 1993.

[24] N. Sourlas, "Statistical Mechanics and the Travelling Salesman Problem", *Europhysics Letters* **2** (12), 919-923 (1986).

[25] N.L.J. Ulder, E.H.L. Aarts, H.-J. Bandelt, P.J.M. van Laarhoven, and E. Pesch, "Genetic Local Search Algorithms for the Traveling Salesman Problem," in *Parallel Problem Solving from Nature* edited by H.-P. Schwefel and R. Männer, Springer-Verlag, 1990, pp. 109–116.

[26] Y. C. Wei and C. K. Cheng, "A Two-Level Two-Way Partitioning Algorithm", in *Proc. IEEE Intl. Conf. on Computer-Aided Design*, Nov. 1990, 516-519.

[27] E. Weinberger, "Correlated and Uncorrelated Fitness Landscapes and How to Tell the Difference", *Biological Cybernetics* **63**, 325-336 (1990).