PRESEQUENCING THE EUCLIDEAN TSP WITH A COMBINED GENETIC ALGORITHM / SPACE FILLING CURVE APPROACH

David M. Tate, Cenk Tunasar and Alice E. Smith
Department of Industrial Engineering
1048 Benedum Hall
University of Pittsburgh
Pittsburgh, PA 15261
412-624-9830

ABSTRACT

The spacefilling curve (SFC) method of Bartholdy and Platzman is an extremely fast heuristic for the Euclidean Traveling Salesman Problem. We show how genetic search over a parametrized family of spacefilling curves can be used to improve the quality of the tours generated by SFC. The computational effort of the search grows more slowly than for comparable heuristics, and the tours obtained by the search provide extremely robust presequences for repetitive problems in which only a subset of all cities will be present in any given problem instance.

INTRODUCTION

The Euclidean Traveling Salesman Problem (ETSP) is a well-known, NP-hard combinatorial optimization problem. Given a set of points in Euclidean space, we seek to find the minimum-length closed tour of those points. Dozens of solution methods have been proposed for this problem; a good survey of both optimum-seeking methods and heuristics is given in the book by Lawler et al [1]. A more recent discussion of fast heuristics is given by Bentley [2].

Choosing a solution method for a given ETSP problem instance (or recurring problem environment) is in itself a multi-criterion optimization problem. The solver must consider the tradeoffs among tour length, speed of execution, and solution robustness. There is thus an "efficient frontier" of solution methods which are not equalled or surpassed in all three of these areas by any one competing solution method. A heuristic which yields mediocre tour lengths may nevertheless be useful for its speed, while a relatively slow heuristic may provide an attractive lower-bound guarantee on solution quality.

One extreme point of this efficient frontier of solution methods is the spacefilling curve heuristic (SFC) of Bartholdi and Platzman [3, 4, 5]. The average solution quality of the method is only fair, and its worst-case performance is relatively bad [6]. Nevertheless, the

method is potentially very useful, for it is the fastest available heuristic for large problems. SFC works by mapping city locations in multi-dimensional Euclidean space onto the unit circle, using the inverse of a closed spacefilling curve. A tour is then found by visiting the cities in the order in which their images appear on the circle. Since no actual distances are calculated or compared, SFC may be used even when the true inter-city distances are unknown, or subject to random error.[1] Storer and Bringhurst used a simulated annealing approach to transform the SFC for full sequences and showed average improvements of 10 to 14 percent over the naive SFC on problems of 10 to 500 cities [7].

After describing how the SFC can be used to generate tours very quickly even for large ETSP instances, Bartholdi and Platzman [5] make the following comment about problems in which not all cities must always be visited:

The essential contribution of the spacefilling curve is simply to provide a linear ordering of all the points that may potentially be included in a problem instance. The [SFC] may be implemented with any such ordering. But, to be most effective, the ordering should be tailored to the distribution from which the problem instances are drawn and the metric used to represent distances between pairs of points (if available!). In general, we might be willing to spend considerable effort creating an effective ordering for a particular problem; this is a design task and so needs to be performed only once.

This introduces the idea of a <u>presequence</u>, also explored by Jaillet [8]. A presequence is simply a sorted list of possible city sites; individual instances of the problem are treated by visiting the cities in the order in which they appear in the presequence. Constructing tours from a presequence is even faster than basic SFC, but will generally result in

---

[1]SFC may be thought of as a limiting case of the Fast Recursive Partitioning (FRP) algorithm alluded to by Bentley [2]. In SFC, the refinement of the Euclidean region into buckets is continued until each city is alone in its bucket. FRP shares many of the operational and performance attributes of SFC.

longer tours. For large, repetitive problem environments in which there are finitely many potential city sites, but only a proper subset of these sites occur in any given problem instance, presequencing becomes an attractive alternative. Bartholdi et al. [4] describe a successful application of SFC presequencing to repetitive vehicle routing problems.

Intuitively, for any given set of possible city sites (and their relative frequencies of occurrence in problem instances) there is some particular presequence which has the lowest expected tour length. In practice, though, finding this presequence is even harder than finding the shortest tour through all possible cities. Nevertheless, it may be possible to find presequences which give significantly better tours than the SFC sequence. Bartholdi and Platzman [5] suggest an iterative search approach, in which the initial SFC sequence is modified by random interchanges, and each new sequence is evaluated by generating some statistically significant number of random problem instances, to which the current presequence is then applied.

This approach has several drawbacks. It is extremely computationally intensive, and the random interchange search of sequence space is very inefficient. We could improve the search by implementing a good heuristic search method in which proposed solutions are evaluated by generating random test problems, but this would still be very slow. Also the set of possible tours is large and ill-behaved, and thus difficult to search effectively even for deterministic problems.

We propose an alternative approach to improving the SFC presequence, based on searching a space of alternative spacefilling curves rather than the space of possible sequences. This approach is faster than the iterative simulation method, and its computational effort grows more slowly as problem size increases. The space it searches is continuous, and the presequences it generates compare extremely favorably not only to the naive SFC presequence, but also to presequences generated by other fast heuristics.

GENETIC ALGORITHMS AND ALTERNATIVE SEARCH SPACES

Genetic algorithms (GA) are a family of heuristic search procedures based on the biological paradigm of natural selection. They were pioneered by Holland [9], deJong [10], and Goldberg [11] in the context of continuous nonlinear optimization, and later extended by various authors [12, 13, 14] to combinatorial problems. For optimization problems, GA search begins with an initial

"population" of feasible solutions, encoded in some convenient data structure. The search proceeds by repeatedly selecting solutions from the population, "breeding" and "mutating" their encodings to produce encodings represent new feasible solutions, adding these new solutions to the population, and "culling" solutions from the population to maintain a stable population size. The selection of "parents" for breeding is made randomly, but with preference given to solutions with better objective function values. As a result, the average solution quality in the population (and, more importantly, the best solution in the population) will tend to improve continuously. A more thorough description of past GA applications to traveling salesman problems is given in [15].

If there were a parametric family of spacefilling curves with a small number of real-valued parameters, such that distinct parameter values gave distinct spacefilling mappings from $\Re$ to $\Re^n$. We could then search the parameter space looking for the particular spacefilling curve which gives the best tour for the complete city set of our problem environment. We might then expect this tour to give better results as a presequence, as well, since it would it some sense "follow" the irregularities in the distribution of cities. Furthermore, we might expect this tour to be more robust with respect to random errors in the locations of cities or the distances between cities. It is this notion that was explored by Storer and Bringhurst [7].

It is easy to find such parametric families of spacefilling curves. Let S be an inverse spacefilling curve mapping a compact region H^n in $\Re^n$ onto B, the boundary of the unit circle; $S: H^n \Rightarrow B$. Now let M be any mapping of H^n onto itself; $M: H^n \Rightarrow H^n$. The composition S^OM defines a new mapping from H^n onto B. If M is invertible, S^OM is the inverse of a spacefilling mapping. Even if M is not invertible, S^OM can be used to define a tour-construction heuristic analogous to SFC, simply by including a tie-breaking mechanism for cities mapped to the same point in B. If we are given a parametric family of mappings $M(\pi)$, where π is a vector of real-valued parameters, $(S^OM)(\pi)$ defines a parametric family of mappings from H^n onto B, any one of which can be used to construct tours on sets of points in H^n. For any fixed set of city locations, we can search the possible values of π to find a particular mapping which results in a short tour. If our family of mappings is sufficiently flexible, this will allow us to find good solutions to ETSPs by searching a space whose size does not grow as a function of the number of cities.

As simple illustration, consider the inverse spacefilling mapping of Bartholdi and Platzman which maps $[0,1] \times [0,1]$ onto B. The spacefilling curve is a recursive

algorithm which can map to any degree of resolution depending on the grid size. If we precede this mapping with the transformation T: $[0,1] \times [0,1] \Rightarrow [0,1] \times [0,1]$ defined by $T(x,y) = (x^\beta, y^\beta)$ for some $\beta > 0$, we obtain a new transformation from $[0,1] \Rightarrow [0,1]$ onto B, which will usually produce different tours than the original SFC. Similarly, different choices of the parameter β will give different tours for the same problem instance. We could thus search over possible values of β to find a better tour than that given by using the spacefilling mapping directly on the original problem data.

Below, we describe a GA implementation which employs a parametric family of spacefilling mappings. The GA seeks to optimize tour length as a function of 3 parameter values, where each real-valued triple of parameter values corresponds to a distinct spacefilling curve. Computational testing shows that the GA method gives modest improvements over SFC for single deterministic problem instances, but that the GA tour used as a presequence compares quite favorably with other presequences, in both average and worst-case tour length.

GA IMPLEMENTATION

To test the effectiveness of the GA approach, we selected two well-known two-dimensional Euclidean TSP instances from the literature. The first is the 318-city problem described by Lin and Kernighan [16]; the second is the 538-city problem described by Padberg and Rinaldi [17]. We also generated several 2-D test problems with uniform distributions, described below.

To implement a genetic search of the type described above, we first needed to identify a parametric family of spacefilling curves. We chose simply to transform the coordinates of the cities in a given problem using some parametric 2-D coordinate transformation $M(\pi)$, and then to map the transformed coordinates onto the unit circle using the inverse spacefilling mapping described by Bartholdi and Platzman [3]. The GA was used to search over possible values of the parameter vector π, looking for a particular coordinate transformation that would yield the shortest tour when the resulting transformed city locations were given as input to the original SFC.

We can think about this approach as follows: for any problem instance, there is some optimal sequence s in which to visit the cities. Furthermore, there are many location vectors $L=(L_1, L_2, ..., L_n)$ such that placing city 1 at L_1, city 2 at L_2, and so forth would cause the original spacefilling curve to pass through the cities in order s. If our parametric family of coordinate transformations is sufficiently general, there will be some combination of parameter values that causes the actual city locations to be mapped close to one of the "optimal" vectors L. By searching the parameter space of the transformation, we can thus hope to find a near-optimal tour of the cities for our particular problem instance.

There is a tradeoff here between small parameter spaces (which are easier to search) and extremely flexible families of transformations (which may involve many parameters). We chose to focus our efforts on simple transformations, hoping to find modest improvements with minimal computational effort. For the most part, we considered independently shifting x and y coordinates, combined with some rotational transformation. After some preliminary testing of various mappings, we decided to perform a complete test of the GA method using one of the simpler transformations: a centered rotational transformation $T(x,y; \alpha, x_0, y_0)$ defined as follows:

$$T(x,y) = (x_0 + (x - x_0)) \cos(\alpha) - (y - y_0) \sin(\alpha),$$
$$y_0 + (x - x_0)) \sin(\alpha) + (y - y_0) \cos(\alpha)) \qquad (1)$$

The three parameters (α, x_0, y_0) comprise the decision variables of the new optimization problem. The transformation consists of rotating the entire data set counter-clockwise by α degrees about the center of rotation (x_0, y_0). This is clearly not a very flexible family of transformations; not even the relative distances among cities will change. Because this simple angle transformation did not allow sufficient richness to reliably find better TSP presequences, we turned to a transformation with more tuneable parameters. After empirical investigation, we found the following transformation to possess greater flexibility than (1) above, while still allowing a search over only three parameters.

$$(x', y') = ((x - x_{min})/(x_{max} - x_{min}), (y - y_{min})/(y_{max} - y_{min})) \qquad (2)$$

Given this 3-parameter search space, we implemented a real-coded GA with a population size of 20. Individual solutions were encoded as triples (α, x_0, y_0) of floating point numbers. The angle α was restricted to lie in the interval $[0, \pi/4]$, while x_0 and y_0 were restricted to lie in the interval $[0.4, 0.6]$. (This restriction was based on empirical observation that centers of rotation outside that range tended to give poor results; we could just as easily have used the range $[0,1]$.) The initial population was formed by generating uniform random coordinates over those ranges. Breeding of solutions consisted of simply averaging the corresponding parameter values of the two selected triples, with selection probabilities favoring higher-ranking solutions in the current population.

Mutation was effected through the introduction of new random solutions. Each generation consisted of the best solution from the previous generation, 14 offspring of parents from the previous generation, and 5 randomly-generated newcomers. For each parameter triple, the corresponding tour was found by rotating the unit-square data set about the specified point, contracting the resulting point set back into the unit square using (2), and then applying the original SFC mapping to the unit circle.

RESULTS

We applied this GA to the two well-known ETSP instances described above. For each problem, we ran 20 replications of the GA for 20 generations each, so that a total of 420 tours (not necessarily distinct) were considered in each run. Each run used a different random number seed, yielding a different realization of the population's evolution. The results of these runs are summarized in Table 1. The GA/SFC method showed a noticeable improvement over the simple SFC solution, and gave results comparable to the Nearest Neighbor (NN) heuristic for problems of this size. This was encouraging, in that the GA/SFC method, with its O(n log n) work per generation, should scale up to larger problems better than NN-based methods. Note the a single GA/SFC run is faster than enumerating a large fraction of all possible NN starting points and selecting the best NN tour among them. Figure 1 gives the relative CPU requirements of the simple SFC, a single random-start NN solution, a single 20-generation GA run, and trying 1% of the possible NN starting points before choosing of those tours.
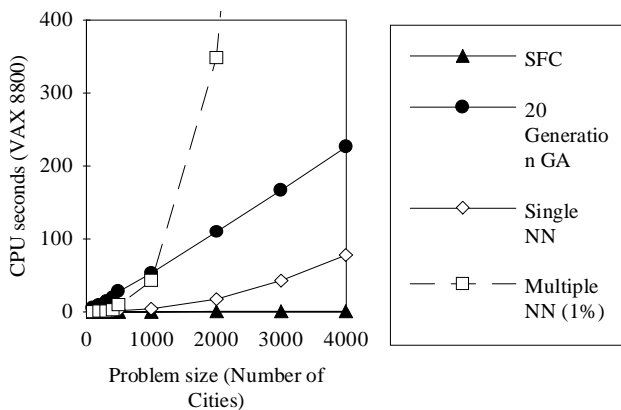

Figure 1. Relative CPU Requirements of Various Fast Heuristics.

We also generated a series of random test problems, ranging in size from 100 to 500 cities, with cities uniformly distributed over the unit square. Optimal tours were not known for these problems, but we were able to compare the average SFC, GA/SFC, and NN behavior over 100 randomly-generated problems of each size, as shown in Table 2. The GA/SFC still yielded significant improvement over the simple SFC, despite the fact that the simple SFC should perform best on uniformly distributed city sets. In addition, the GA/SFC results compared favorably to the NN results, in both best case and average case, for all but the largest problem.

We next tested the performance of the GA/SFC sequence when used as a presequence for subsets of the 532-city problem. We tested five different sequences as presequences: the optimal tour for all 532 cities, the simple SFC tour for all 532 cities, the 532-city tour found by a single GA/SFC run, the 532-city tour found by a single NN run with random starting point, and the best NN tour among all 532 possible starting points. Each of these tours was used as a presequence for 100 randomly-generated subsets of 50 cities, 100 cities, 200 cities, 300 cities, 400 cities, and 500 cities. All cities were assumed to be equally likely to occur in a given problem instance.

Table 3 shows the average performance and standard deviation for each of the five presequences over 100 random problem instances for each subset size. The optimal sequence for the overall city set is clearly the best presequence as well. For the smaller subsets, however, the two SFC-based tours outperformed the NN-based presequences by a wide margin. In particular, the GA/SFC tour was better than the SFC and random-start NN tours for every subset size, and better than the best NN tour for subsets of 400 cities or fewer. This domination grew stronger as the size of the subset in question decreased, as shown in Figure 2. We conjecture that the GA/SFC advantage results from the fact that NN tours are strongly based on local distance information in the overall point set, so that removing many of the points in the overall set also removes the reason for the adjacencies that exist in the NN presequence. The SFC and GA/SFC presequences, on the other hand, are based on global positional information about individual points, and remain more stable in quality as points are removed from the set. If this is true, the GA/SFC method should scale up nicely to extremely large, asymmetrical point sets from which small subsets will be visited in individual problem instances.

CONCLUSIONS

Spacefilling curve heuristics provide an extremely fast way to generate acceptable tours for large Euclidean TSPs. They also produce sequences which make good presequences for repetitive ETSP environments in which

not all cities will be visited in a given tour. Genetic search over parametric families of spacefilling curve allow us to improve the performance of spacefilling curve heuristics in both these areas, at a computational cost that is still relatively low for large problems. The GA/SFC presequences for the problems considered here yielded lower costs and greater robustness than other fast heuristic methods, especially for small subsets of the overall city set.
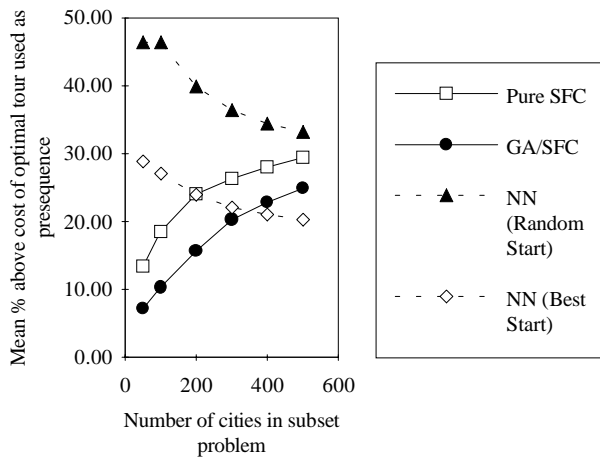

Figure 2. Percent Cost Above Optimal Tour as Presequence.

The GA presented in this paper used a very simple family of coordinate transformations. Future research should be directed toward finding more general families of transformation which can be searched effectively by the GA, yielding larger improvements over the simple SFC and better presequences for large problems. If possible, these transformations should avoid the computationally expensive transcendental functions used in the work presented here, in order to enhance the speed of the method.

Finally, this work may be seen as a special case of a general heuristic approach to optimization, in which heuristic search is used to enhance the solutions produced by constructive or greedy heuristics by searching over possible perturbations of their input data. This approach is loosely related to the "problem space based search neighborhoods" described by Storer et al. [18] in the context of machine scheduling, and can be applied to any optimization problem for which a fast, deterministic, constructive heuristic exists.

REFERENCES

[1] *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*, edited by E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan and D.B. Shmoys, John Wiley & Sons, 1985.

[2] "Fast Algorithms for Geometric Travelling Salesman Problems", Jon Jouis Bentley, *ORSA Journal on Computing*, Vol. 4, No. 4, pp. 387-411, 1993.

[3] "An O(Nlog N) Planar Travelling Salesman Heuristic Based on Spacefilling Curves", John J. Bartholdi and Loren K. Platzman, *Operations Research Letters*, Vol. 1, No. 4, pp. 121-125, 1982.

[4] "A Minimal Technology Routing System for Meals-on-Wheels", John J. Bartholdi, Loren K. Platzman, R. L. Collins and W. H. Warden, *Interfaces*, Vol. 13, No. 3, pp. 1-8, 1983.

[5] "Heuristics Based on Spacefilling Curves for Combinatorial Problems in Euclidean Space", John J. Bartholdi and Loren K. Platzman, *Management Science*, Vol. 34, No. 3, pp. 291-305, 1988.

[6] "Worst-Case Examples for the Spacefilling Curve Heuristic for the Euclidean Travelling Salesman Problem", Dimitris Bertsimas and Michelangelo Grigni, *Operations Research Letters*, 8, pp. 241-244, 1989.

[7] "Heuristics for the Planer Euclidean TSP Based on Space Filling Curves and Simulated Annealing," Robert H. Storer and Andrew Bringhurst, Lehigh University, *Department of Industrial Engineering Working Paper 87-008*, 1987.

[8] *The Probabilistic Travelling Salesman Problem*, P. Jaillet, Ph.D. Thesis, Department of Civil Engineering, Massachusetts Institute of Technology, 1985.

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

[10] *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*, Kenneth deJong, Doctoral Dissertation, University of Michigan, 1975.

[11] *Genetic Algorithms in Search, Optimization and Machine Learning*, David E. Goldberg, Addison-Wesley Publishing, 1989.

[12] "Distributed Genetic Algorithms for the Floorplan design Problem", James P. Cohoon, Shailesh U. Hegde, Worthy N. Martin and Dana S. Richards, *IEEE Transactions on Computer-Aided Design*, Vol. 10, No. 4, pp. 483-492, April 1991.

[13] "Genetic Algorithms and Job Shop Scheduling", John E. Biegel and James J. Davern, *Computers and Industrial Engineering*, Vol. 19, Nos. 1-4, pp. 81-91, 1990.

[14] "Evolution Algorithms in Combinatorial Optimization", H. Muhlenbein, M. Gorges-Schleuter and O. Kramer, *Parallel Computing*, 7, pp. 65-85, 1988.

[15] "Parallel Genetic Algorithms Applied to the Travelling Salesman Problem", Prasanna Jog, Jung Y. Suh and Dirk Van Gucht, *SIAM J. Optimization*, Vol. 1, No. 4, pp. 515-529, 1991.

[16] "An Effective Heuristic Algorithm for the The Traveling Salesman Problem", S. Lin and B.W. Kernighan, *Operations Research*, Vol. 21, No. 2, pp. 498-516, 1973.

[17] "Optimization of a 532-City Symmetric Travelling Salesman Problem by Branch and Cut", M. Padberg and G. Rinaldi, *Operations Research Letters*, Vol. 6, No. 1, pp.1-7, March 1987.

[18] "New Search Spaces For Sequencing Problems With Application To Job Shop Scheduling", R. H. Storer, S. D. Wu and R. Vaccari, *Management Science*, Vol. 38, No. 10, pp. 1495-1509, 1992.

BIOGRAPHICAL SKETCHES

David M. Tate is Assistant Professor of Industrial Engineering at the University of Pittsburgh. He received his PhD from Cornell University. His research interests are in heuristic optimization and facility layout. He is a Senior Member of IIE.

Cenk Tunasar received both BSc and MS degrees in IE and is currently a PhD candidate in the Industrial Engineering Department of the University of Pittsburgh. His interest areas include heuristic techniques for optimization problems with a focus on stochastic search techniques, mainly genetic algorithms and simulated annealing. Application areas of interest include travelling salesman problems, multi-stage lot sizing and other combinatiorial production planning problems.

Alice E. Smith is Assistant Professor of Industrial Engineering at the University of Pittsburgh. Her research interests are in neural network modeling and heuristic optimization, primarily for process control and engineering design. Her research has been funded by NSF, Lockheed Corporation and the Ben Franklin Technology Center of Western Pennsylvania. She is a Senior Member of IIE.

Table 1. Improvement of SFC Tour by Genetic Algorithm.

Problem	Optimal Solution	Pure SFC	NN (*)	Genetic Algorithm Search (#)			
				Average	Minimum	Maximum	CV
318 city	42029	57513	54438	53766	53536	54266	0.003
532 city	87035	117316	111773	114037	113633	114779	0.004

(*) Averaged over 50 random starting cities
(#) 20 generations of population size 20

Table 2. Relative Performance of Heuristics for Random Problems.

Problem	Pure SFC	NN (*)	Genetic Algorithm Search (#)			
			Average	Minimum	Maximum	CV
100 city	966.3	959.1	916.0	915.7	918.7	0.001
200 city	1393.6	1362.4	1359.7	1355.2	1366.8	0.002
300 city	1646.9	1733.3	1627.4	1624.1	1633.5	0.002
400 city	1891.5	1899.9	1873.5	1866.7	1890.0	0.003
500 city	2195.1	2124.5	2141.7	2137.9	2146.0	0.001

(*) Averaged over 50 random starting cities
(#) 20 generations of population size 20

Table 3. Relative Presequence Performance of Heuristics.

Problem		OPT	Pure SFC	GA/SFC	Single NN	Best NN
M50	Avg	34879.00	39554.45	37381.30	51051.78	44958.45
	CV	0.05	0.08	0.06	0.09	0.06
M100	Avg	44989.48	53327.51	49610.91	65880.03	57173.17
	CV	0.05	0.05	0.05	0.06	0.05
M200	Avg	56584.70	70209.49	65453.69	79167.50	70161.95
	CV	0.03	0.03	0.04	0.04	0.03
M300	Avg	63881.90	80681.51	76870.27	87126.63	77984.36
	CV	0.02	0.03	0.03	0.03	0.03
M400	Avg	68999.35	88341.79	84753.76	92725.80	83527.61
	CV	0.02	0.03	0.03	0.03	0.02
M500	Avg	72639.64	94042.93	90756.11	96742.11	87395.35
	CV	0.02	0.02	0.03	0.02	0.02