Improving the Performance of Multi-start Search on the Traveling Salesman Problem

Charles R. King, Mark McKenney, and Dan E. Tamir

Texas State University
Department of Computer Science
{charles.king, mckenney, dt19}@txstate.edu

Abstract. Constructive multi-start search algorithms are commonly used to address combinatorial optimization problems. Multi-start algorithms recover from local optima by restarting, which can lead to redundant configurations when search paths converge. In this paper, we investigate ways to minimize redundancy using record keeping and analyze several restart algorithms in the context of iterative hill climbing with applications to the traveling salesman problem. Experimental results identify the best performing restart algorithms.

Keywords: Combinatorial optimization, traveling salesman problem, iterative hill climbing, multi-start algorithms

1 Introduction

Intuitively, combinatorial optimization problems (COPs) compute the "most favorable" outcome to a problem from a set of possible outcomes. A most favorable outcome may be the outcome with the highest profit, lowest cost, shortest distance, etc., depending on the problem being solved. For example, shipping companies have used COPs to find delivery routes that minimize left turns; by eliminating left turns, drivers spend less time in traffic waiting to turn, and use less fuel, resulting in monetary savings for the company. Clearly, COPs have great practical importance.

Many COPs are NP complete and characterized by exponential growth [1-3]; i.e., as the size of problem parameters grows, the number of potential solutions grows exponentially, causing an exhaustive search of outcomes to be impractical. A *heuristic search* technique is often used in order to assign weights to outcomes, or partial outcomes, such that partial solutions that appear to be leading to poor solutions are not considered, thus, drastically reducing the search space of outcomes. A drawback to heuristic techniques is that the optimal solution may not be found, but often a very good solution can be guaranteed. Therefore, heuristic techniques are applicable only when a sub-optimal solution is acceptable, or when the faster computation times of heuristic algorithms for COPs are required.

It is well known that many heuristic algorithms explore the same partial solution (i.e., a state in the state space) more than one time, thereby wasting computational resources. Several researchers have proposed record keeping mechanisms as a solution to this problem [4, 5], which we further investigate in this paper. In this

sense, record keeping resembles Tabu search where the Tabu list includes a subset of the states encountered so far [6, 7]. It differs from Tabu in that the objective of short term Tabu memory (e.g., a cache) is to explore the solution space more intelligently rather than to speed the search [7]. The goal of long-term Tabu is to enable revisiting explored search states in order to attempt to find a better local optimum [6, 7].

In [4], we analyzed the performance of a cache as a means for record keeping. In [8], we investigated several other record keeping mechanisms such as dedicated memory, and Bloom filters [9]. In this paper, we investigate the performance of record keeping multi-start COP algorithms using various restart algorithms where the record keeping mechanism is a cache. We investigate these in the context of the travelling salesman problem (TSP), using iterative hill climbing (IHC) heuristic search algorithm as the multi-start heuristic procedure [2, 3, 10]. The TSP is chosen since it is a classic, well-understood COP with many practical applications [10].

In this paper, we investigate the performance of six construction (restart) algorithms used in the context of IHC with randomly generated and benchmark TSP problems. The performance of the restart algorithms is empirically determined and the algorithms are compared for performance and solution quality.

The second contribution of this paper is to determine the amount of redundancy associated with various construction algorithms, and utilize record keeping techniques to reduce duplicate path exploration. We show empirically that a good choice of construction algorithm and record keeping mechanism provides improved convergence time. The evaluation of construction algorithms and the use of record keeping have been studied in previous work [4, 8]. Our work differs from that work in that we consider a larger group of construction algorithms, we provide a more in-depth study of the effects of different cache configurations, and we provide a comparison of cache performance to unbounded memory record keeping.

2 Construction Algorithms

In the case of the TSP, IHC can be implemented as a *constructive multi-start search*, in which multiple solutions are computed by constructing solutions from an initial solution. At each step, the concept of a *neighborhood* is used in which a neighborhood of a solution s is defined as the set of solutions S that are generated by making a minor modification, denoted a *move,* to s [11, 12]. For example, a TSP tour can be adjusted by an operation such as 2-opt which results in exchanging the visitation order of two cities [2]. The IHC algorithm proceeds by choosing the best $s' \in S$ as the next step. The move s is then assigned s' and the process is repeated until no improvements to s can be made. This solution is referred to as the locally optimal solution. In general, the algorithm repeatedly restarts with a new starting configuration until a sufficiently good solution is reached, or a set running time has expired. The goal of this paper is to determine the performance of various restart algorithms in generating solutions to the TSP in the context of iterative hill climbing and record keeping. The restart algorithms evaluated, described in details in ref. [8], are: 1) *Greedy Enumeration*, 2) *Greedy Jump*, 3) *GRASP* [12], 4) *Nearest Neighbor* [2], 5) *Clarke-Wright* [2], and 6) *Random*. Identical neighborhood generating algorithms are used for all implementations.

2.1 Record Keeping

A problem with the IHC using a constructive multi-start search method is that the same solution may be reached from multiple starting configurations. Therefore, many tours chosen are duplicate tours that have been explored in previous iterations of the algorithm. One method to combat this problem is to utilize record keeping mechanisms that record some or all tours that have been previously computed. Thus, if a tour is generated that has been considered previously, the algorithm restarts with a new starting configuration. We investigate two models of record keeping [4, 8]: 1) *Unbounded Memory,* 2) *Software Emulation of a Cache* [13]. We use set associative caches with small set sizes in our experiments.

3 Experiments

All experiments were performed using a set of TSP graphs from TSPLIB [14] and random graphs with a maximum of 100 vertices. The set of graphs consists of:

- 10 randomly-generated Euclidean graphs, 100 vertices each
- 10 random non-planar graphs, 100 vertices each
- 18 benchmark instances from TSPLIB [14], vertex counts of 16-100
- 400 random Euclidean graphs of 100 vertices each.

3.1 TSP Solution Quality

The experiment results are summarized in Table 1. Each row in the table shows the average quality of the solutions found by each of the algorithms in each of the experiments as a percentage of the best tour found by Concorde; lower numbers indicate better quality, with 100.0 being the best tour discovered. Note that the best tour may be found by multiple construction algorithms.

In the first set of experiments the IHC algorithm is executed using 10 randomly generated non-planar graphs. The average performance of the algorithms relative to the Concorde results, which are known to be Optimal with respect to this set of graphs, is shown in the first row of Table 1. Lower numbers are better with an ideal being 100 The second experiment runs the IHC algorithm using each of the construction algorithms over ten randomly generated planar Euclidean graphs are summarized in the second row of Table 1. The third experiment uses the TSPLIB benchmarks as the input for the construction algorithms [14].

The results of the experiments show that the nearest neighbor algorithm performs the worst of the six construction algorithms, and that the greedy based approaches perform well regardless of the structure of the input. Similarly, the Clarke-Wright algorithm performs relatively poorly regardless the structure of the input graph. The random algorithm performed the best of the six algorithms on two of the data sets, but its poor performance on the non-planar graph data set brings its average performance to be less than the greedy approaches, and indicates that it is sensitive to the structure of the input graph.

3.2 Construction Algorithms and Tour Redundancy

The next set of experiments examines the quality and redundancy associated with some of the construction algorithms. Because the greedy algorithms perform among the best despite the structure of input graphs and due to the fact that the random restart does not exploit the record keeping efficiently, we perform this set of experiment only on them. Figure 1a shows the average redundancy and tour qualities produced when running IHC with the greedy construction algorithms. The average quality of all algorithms is nearly identical, but the greedy-jump algorithm has far less redundancy than the others. Figures 1b, 1c, and 1d show the distribution of solution quality for 400 random graphs solved using Concorde, GRASP, and GE. All the graphs show a distribution that is close to normal distribution [10], where Concorde has the best mean and lowest variance. The mean and variance of GRASP and GE are almost identical and close to the mean and variance obtained by Concorde. Since GRASP has slightly better redundancy we used it as the subject for the next experiment.

Table 1: Average performance of the construction algorithms

	Greedy	Jump	GRASP	Random	CW	NN
Random graphs	**100.08**	**100.90**	**101.42**	**119.62**	**119.08**	**149.19**
Random Euclidean graphs	**100.83**	**100.83**	**100.50**	**100.29**	**102.80**	**110.43**
TSPLIB benchmarks	**100.66**	**100.37**	**100.73**	**100.18**	**101.75**	**109.72**

3.3 Experiments with Record Keeping

Under the IHC algorithm, the generation of a tour that has been previously considered results in generating tours that have also been previously considered, and eventually leads to a local maximal tour that has been previously computed. Therefore, the obvious solution is to record all computed tours so that duplicates can be detected. In general, this is not feasible for large TSP instances, so we employ a cache mechanism to limit the size of memory used by the IHC algorithm. However, we first use unbounded memory record keeping on 50 random Euclidean graphs, of 100 vertices each, in order to determine an experimental upper bound on IHC speedup using record keeping. For IHC runs of 100,000 iterations, the average speedup using unbounded memory record keeping was 10.9. The same speedup is obtained with a 64,000 blocks cache (cache of 896KB). Figure 2 shows the speedup obtained with different cache configurations [2, 8, 13]. It is apparent that even with a small cache size 16KB, we still achieve a speedup of 6.7.

4 Conclusion

In this paper, we evaluated the use of six different construction algorithms with the IHC method of solving the TSP. Our experimental results show that the greedy, greedy-jump, and GRASP construction algorithms all perform well. The greedy-jump construction algorithm also generates significantly fewer duplicate tours than

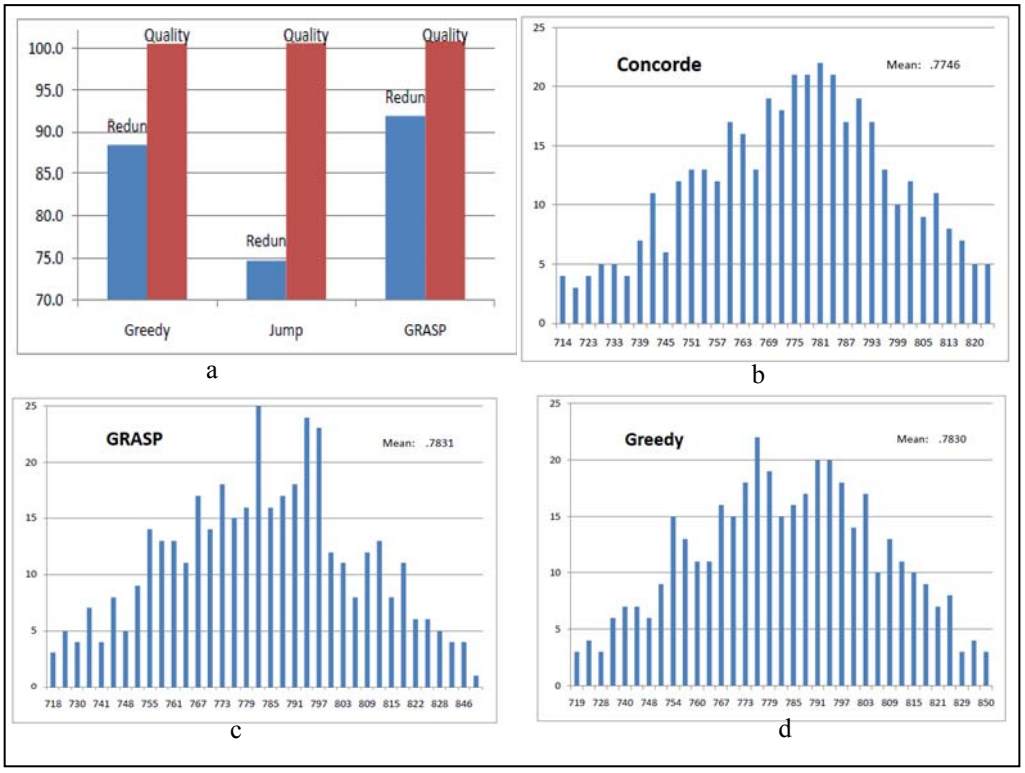

Fig. 1. **Average tour Quality and Redundancy**

the other algorithms, indicating that it may be more robust in situations where many local minima occur. Furthermore, we show that utilizing a record keeping mechanism modeled on cache memory is effective at improving IHC efficiency and can provide a speed up of up to 10.9x in the given configuration. Moreover, even when the memory is only large enough to hold a small percentage of duplicate tours generated, cache still provides a significant performance improvement. Thus, regardless of the complexity of a TSP configuration and memory limits, even a small cache is useful. This work has applications in the study of the traveling salesman problem and in combinatorial optimization problems in general.

Future work involves further investigation of redundancy in the construction algorithms. We plan to investigate stochastic mechanisms to minimize redundancy and study its affects on algorithm performance and quality of the generated tours.

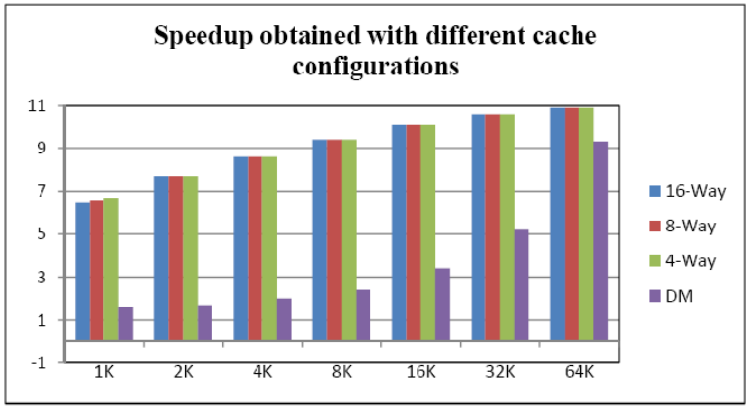

Fig. 2. **Speedup obtained with different cache configurations**

5 References

1. Cormen, T.H., et al., *Introduction to Algorithms*. 2001, Cambridge, Massachusetts: MIT Press.
2. Johnson, D. and L. McGeoch, *The Traveling Salesman Problem: A Case Study*, in *Local Search in Combinatorial Optimization*. 1997, John Wiley & Sons: Chichester. p. 215-310.
3. Garey, M.R., Johnson, D.S, *Cmputers and intractability: A guide to the theory of NP-completeness*. 1979: W.H. Freeman.
4. Karhi, D.T., D. E. *Caching in the TSP Search Space*. in *Next-Generation Applied Intelligence* 2009. Tainan, Taiwan: Springer.
5. Allen, D. and A. Darwiche. *Optimal Time-Space Tradeoff in Probabilistic Inference*. in *Proceedings of the 21st international joint conference on artificial intelligence*. 2003.
6. Glover, F. and M. Laguna, *Tabu Search*. 1997: Kluwer Academic Publishers.
7. Glover, F., Taillard, E., de Werra, D.,, *A User's Guide to Tabu Search.* Annals of Operations Research 1993. **41**(1-4): p. 3-28.
8. King, C.R., *Improving the performance of constructive multi-start search using record keeping*, in *Computer Science*. 2010, Texas State University: San Marcos.
9. Dawson, E. and D. Wong, *Information Security Practice and Experience*. 2007: Springer.
10. Applegate, D.L., et al., *The Traveling Salesman Problem, A Computational Study* 2006, Princeton, New Jersey: Princeton University Press.
11. Ambite, J. and C. Knoblock, *Planning by Rewriting.* Journal of Artificial Intelligence Research, 2001: p. 207-261.
12. Pitsoulis, L.S. and M.G.C. Resende, *Greedy Randomized Adaptive Search Procedures*, in *Handbook of Applied Optimization*. 2001, Oxford University Press. p. 168-183.
13. Hennessy, J.L. and D.A. Patterson, *Computer Architecture, A Quantative Approach*. 2007, San Francisco, California: Morgan Kaufmann Publishers, Inc.
14. Reinelt, G., *"TSPLIB -A traveling salesman problem library,"* ORSA Journal on Computing, 1991. **3**(4): p. 376-384.