Apple Carving Algorithm to Approximate Traveling Salesman Problem from Compact Triangulation of Planar Point Sets

Marko Dodig[1], Milton Smith[2]
Industrial, Manufacturing and Systems Engineering
Texas Tech University
Lubbock, TX, USA

***Abstract*—We propose a modified version of the Convex Hull algorithm for approximating minimum-length Hamiltonian cycle (TSP) in planar point sets. Starting from a full compact triangulation of a point set, our heuristic "carves out" candidate triangles with the minimal Triangle Inequality Measure until all points lie on the outer perimeter of the remaining partial triangulation. The initial candidate list consists of triangles on the convex hull of a given planar point set; the list is updated as triangles are eliminated and new triangles are thereby exposed. We show that the time and space complexity of the "apple carving" algorithm are $O(n^2)$ and $O(n)$, respectively. We test our algorithm using a well-known problem subset and demonstrate that our proposed algorithm outperforms nearly all other TSP tour construction heuristics.**

Keywords*—TSP; heuristics; combinatorial optimization; computational geometry; compact triangulation

I. Introduction

In this article we examine the following tour-construction heuristic for the planar TSP: take a compact triangulation of the planar set and then find the minimum Hamiltonian cycle embedded in the triangulation by progressively removing triangles of minimal Triangle Inequality measure until n-2 triangles remain. We call this heuristic "apple carving" as this descriptor accurately describes the triangle removal process which is the basis of the algorithm. Possibility of using well-known triangulations such as Greedy and Delaunay to generate heuristic tours was already explored by Reinelt [1], Stewart [2], and Letchford and Pearson [3]. These authors looked at triangulations as presenting a "good" subset of edges and utilized well-established TSP solutions engines like CONCORDE to solve for TSP. Our research is different in that we (a) utilize newly introduced Greedy Compact Triangulation (GCT) proposed recently by Dodig and Smith [4], and (b) utilize a modification of Convex Hull Heuristic on GCT triangles to approximate TSP.

Our paper is organized as follows. First, we formally define the TSP and review the present state of its solution algorithms. Second, we introduce our approach. Third, we present our experimental methodology and review our experimental results. Finally, we highlight our conclusions and outline future research steps.

II. Literature Review

A. Traveling Salesman Problem

Traveling salesman problem (TSP) is perhaps the best-known and most-researched problem in combinatorial optimization. In its general form we are given a collection of cities and the distance to travel between each pair of them, and the problem then is to find the shortest route to visit each city and to return to the starting point [5]. TSP belongs to the class of NP-hard problems; in other words no polynomial-time algorithm exists that can solve the problem optimally in polynomial time, regardless of its complexity (i.e. the number of cities in the tour). The best result to date is a solution method, discovered in 1962, that runs in time proportional to $n^2 2^n$ [6]. TSP has been fascinating both researchers and general public for more than sixty years. In 1954, three researchers from Rand Corporation had solved a long-standing public challenge to find the shortest tour through 48 US state capitals and DC, shown in Fig. 1 [5].

In purely mathematical terms, TSP is the problem of finding a Hamiltonian tour (cycle) of minimum weight in a complete edge-weighted graph. In our research, we consider a symmetric TSP, or STSP, in that we assume that edge-costs are symmetric, or, equivalently, that the graph is undirected. A special case of the TSP is obtained when the vertices of the graph correspond to points in the Euclidean plane, and distance between any two points is equal to the Euclidean distance between the corresponding points. The Euclidean TSP is a special case of the metric TSP, in which the costs obey the triangle inequality. Metric TSP was found to be strongly NP-hard [7]. Related to, but distinct from, the Euclidean TSP is the planar graph TSP which is the focus of our research. This is the version of the TSP in which a planar graph G = (V, E) is given, with weights on the edges of E, and one seeks the minimum cost tour which uses only edges in E. Not only is this problem NP-hard, it is NP-hard even to test if a planar graph is Hamiltonian [7].

There is a multitude of planar TSP solution algorithms; few are exact algorithms, and many are heuristic algorithms. Since planar TSP is NP-hard, exact algorithms are exponential and heuristic algorithms are polynomial; selecting between exact or heuristic algorithms to solve for TSP presents a clear case of precision and time trade-off.

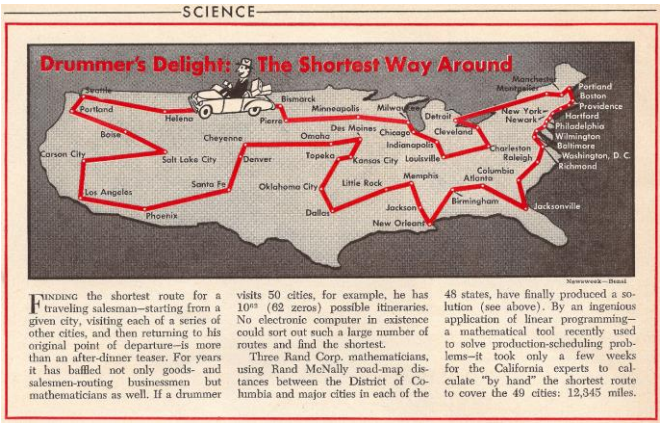

Fig 1. Newsweek Coverage of 49-City Tour through United States [5].

B. Exact Algorithms

Branch-and-bound algorithm is an exact algorithm based on the IP formulation of TSP. This algorithm consists of two steps, (a) branching, which means splitting the problem into sub-problems, and (b) bounding, which means calculating lower and/or upper bounds for the objective function value of the sub-problem. The branching is performed in the following algorithm by separating the current subspace into two parts using the integrality requirement. Using the bounds, unpromising sub-problems can be eliminated. LP-relaxation of the problem is formed by relaxing integer requirements. In the algorithm, a list of sub-problems is maintained. A sub-problem is fathomed (totally solved) and removed from the list only when it has an integer solution that is best so far and becomes the new incumbent solution, or its optimum LP-solution objective is worse than the current incumbent value, or its LP-problem is infeasible.

Held-Karp algorithm is a dynamic programming algorithm utilizing graph theoretical representation of TSP. In a way, it is an intelligent brute force method in that it utilizes recursive formulation to find minimal distance paths between points. It was proposed independently by Bellman [6] and by Held and Karp [8]. This algorithm utilizes an optimization property of TSP in that every sub-path of a path of minimum distance is itself of minimum distance, which is easily proven by contradiction. The algorithm computes the solutions of all sub-problems, starting with the smallest, and looks up solutions already computed when requiring solutions for smaller problems. At the end, computing minimum distance tour means using the final equation to generate the initial node, and then repeating for all other nodes. Held-Karp is exhaustive, in that all sub-problems need to be solved; it has the time complexity of $O(2^n n^2)$ and the space complexity of $O(2^n n)$.

C. TSP Heuristics

In simplest terms, TSP heuristics can be divided into two distinct categories. Tour construction heuristics execute a sequence of operations until a valid tour is obtained, at which point the heuristics stop and report the constructed tour. Tour improvement heuristics start with a valid tour (an output of a tour construction heuristic, for example) and iteratively improve the tour cost, typically via local search, until some stopping criterion is reached [5]. Solution quality of tour improvement techniques far exceeds quality of solutions achieved by tour constructions [5].

Nearest Neighbor heuristic is perhaps the best-known tour construction heuristics [9]. It starts with a random city, adds the nearest non-visited city, and keep adding new non-visited cities in the same fashion until all cities are included. When all of the cities are included it returns to the initial city. It has the time and the space complexity of $O(n^2)$ and $O(n)$, respectively [10].

Greedy heuristic gradually constructs a tour by repeatedly selecting the shortest remaining edge and adding it to the tour as long as it does not create a cycle with less than n edges nor increase the degree of any node (city) to more than two [10]. Greedy heuristic has the time complexity of $O(n \times \log_2 n)$, which makes it more efficient than Nearest Neighbor [10]. The space complexity of Greedy matches that of Nearest Neighbor heuristic [10].

Cheapest Insertion heuristic starts with the shortest edge which becomes the initial sub-tour. Then it selects a city not in the current sub-tour, having the shortest distance to any one of the cities in the sub-tour. It finds an edge in the sub-tour such that the cost of inserting the selected city between the edge cities will be minimal, and keeps inserting shortest-distance remaining cities until none remain. Cheapest Insertion has the time complexity of $O(n^2 \times \log_2 n)$ and is more computationally intensive then Nearest Neighbor and Greedy [11].

Convex Hull heuristics starts by finding the convex hull of a point set and making it an initial sub-tour. For each remaining point it finds its cheapest insertion, adds the city with the least cost/increase ratio, and keeps repeating this process with remaining points until none remain. It is also more computationally intensive with the time complexity of $O(n^2 \times \log_2 n)$ [12].

Christofides heuristic builds a minimal spanning tree (MST) of the planar point set. It then creates a minimum-weight matching (MWM) on points having an odd degree, adds the MST together with the MWM, creates an Euler cycle from the combined graph, and finally traverses it taking shortcuts to avoid already included points. This heuristic has the best worst-case performance guarantee of all TSP heuristics as it never produces tours worse than 1.5 times the optimal [13]. On the other hand, it has the time complexity equal to $O(n^3)$ [13].

Match-Twice-and-Stitch heuristic [14] uses two sequential minimum-weight matchings to construct the cycles. The first matching returns the usual minimum-cost edge set with each point incident to exactly one matching edge. The second matching returns the minimum-cost edge set with each point incident to exactly one matching while ignoring the edges found in the first matching. The first phase results in multiple sub-tours. The second phase stitches the constructed cycles to form the TSP tour, with the exact (slow) and approximate (fast) patching procedure to join two cycles. A minimum spanning tree (MST) calculation determines a way to stitch all cycles into a tour. It is the best construction heuristics reported, with the different versions of the heuristic reporting average tour lengths between 4.8% (slowest) to 7.1% (fastest) over HK bound. It has the time complexity of $O(n^2)$ [14].

Tour improvement algorithm such as 2-opt removes two edges from the feasible tour and reconnects the two paths created if the new tour will be shorter. There is only one way to reconnect the two paths and still have a valid tour. It continues removing and reconnecting the tour until no 2-opt improvements can be found. Algorithm works the same for any path connecting k points, however the time performance severely lags starting at 5-opt. Its worst-case performance guarantee is known, as it is guaranteed to produce results not more than two times the optimal [10]. The main weakness of the 2-opt tour improvement heuristic is that it covers local improvements for pairs of 2 nodes only. This was subsequently addressed in newer k-opt algorithms, where k > 2, chief among them the Lin-Kernighan heuristic with the time complexity of $O(n^{2.2})$ [10].

Solutions generated by TSP heuristics are typically compared to the Held-Karp (HK) lower bound. This lower bound is the solution to the LP relaxation of the IP formulation of the TSP, which can be found in polynomial time by using the Simplex method and a polynomial constraint-separation algorithm [15]. A HK lower bound averages about 0.8% below the optimal tour length [15]; however, its guaranteed lowest bound is only 2/3 of the optimal tour. Fig. 2 summarizes typical performance of the most-significant TSP heuristic algorithms. 2-opt, 3-opt, and Lin-Kernighan heuristics are the tour improvement heuristics, and all of the others are tour construction heuristics.

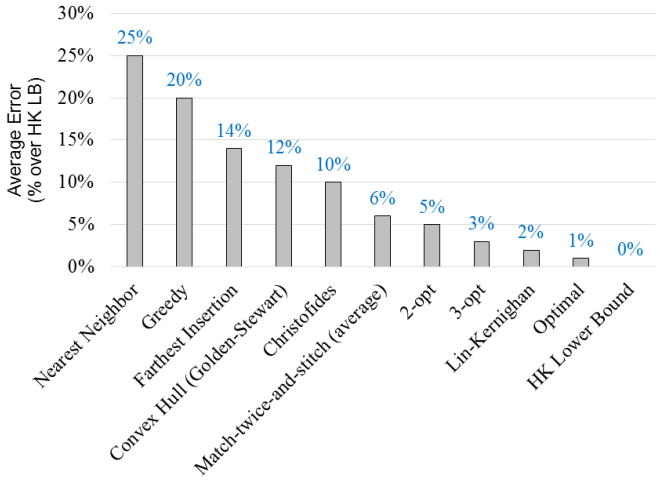

Fig 2. Typical Performance of Best-known Heuristics [10], [14].

III. Our Approach

A. Improved Greedy Compact Triangulation (iGCT)

iGCT of a planar point set S is created by GCT Algorithm [4]. This algorithm progressively inserts most-compact empty triangles into the triangulation not intersecting empty triangles in S previously inserted and achieves local optimality by performing weight-reducing edge flipping [4]. Compactness of an empty triangle Δ_T with area $A(\Delta_T)$ and perimeter $P(\Delta_T)$ in planar point set S is measured as follows [16]:

$$CI(\Delta_T) = \frac{4\pi A(\Delta_T)}{[P(\Delta_T)]^2} \quad (1)$$

Dodig and Smith showed that GCT approximates Minimum Weight Triangulation (MWT) in a variety of planar point set configurations, thereby making its edges compelling candidates for our proposed TSP heuristic [4]. MWT is defined as the full triangulation of a planar point set S having the lowest total edge length out of all full triangulations of a planar point set S. Dodig and Smith have also confirmed that the optimal TSP solution is frequently fully embedded in iGCT (61% of the time), and that the minimum perimeter polygon fully contained in iGCT is nearly optimal, or 0.36% longer than optimal. Fig. 3 shows full embeddedness of the optimal TSP tour in iGCT for *berlin52*, one of the TSPLIB problems for which the optimal TSP is known.

B. Apple Carving Algorithm

There are 2n - h - 2 triangles in both iGCT and MWT triangulations of a planar set S of n points, where h represents the number of points on the Convex Hull of S, or CH(S) [16]. We know that the perimeter length of CH(S) is less than the perimeter length of TSP polygon for this planar point set due to Isoperimetric Inequality principle. Following Steiner proof of Isoperimetric Inequality, we can "carve out" from CH(S) a triangle on the perimeter of full triangulation with the lowest Triangle Inequality Factor and have high degree of confidence that minimum perimeter polygon is still fully contained in the resulting partial triangulation. We can continue carving out eligible triangles with the lowest Triangle Inequality Measure, until all points are at the perimeter of the partial triangulation. We give priority to removing triangles whose absolute Triangle Inequality, or TI, is not only lowest, but also "optimal". "Optimal" TI on any point is defined as the lowest TI of all triangles containing this point. We consider this method to be the basis of the "apple carving" algorithm. In fact, this method is very similar to the Convex Hull heuristics, through Convex Hull Heuristics does not follow a pre-defined tour building roadmap such as the one provided by the compact triangulation [12]. "Apple carving" algorithm pseudocode is given in Fig. 4.

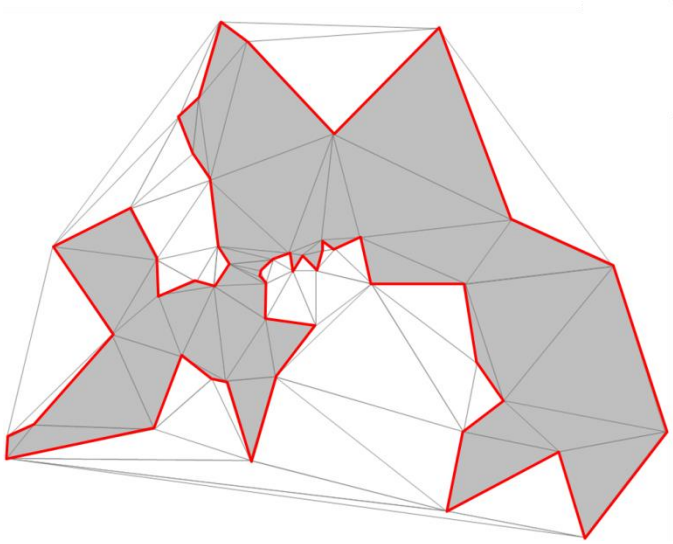

Fig 3. Optimal TSP (Shaded) Fully Contained in GCT for *berlin52* Problem [4].

INPUTS

1. **Planar point set S with n points; S has h points on CH(S).**
2. **iGCT(S) with 2n – h – 2 triangles; each Triangle(a, b, c) and Edge(a, b) in iGCT satisfies a ≤ b ≤ c**

BEGIN Apple Carving Algorithm

1. **Initialize variables**
2. **Import point coordinates**
3. **Initialize iGCT**
    FOR each triangle i in iGCT
        SimpleTriangle(i) := Triangle(a, b, c)
        CountTriangles(a) +=1; CountTriangles(b)+=1; CountTriangles(c) +=1
        IF TIA (a, SimpleTriangle(i)) < min_TI (a) THEN
            min_TI(a) := TIA(a, SimpleTriangle(i))
        ENDIF
        CountEdges(a,b) += 1; CountEdges(a,c) += 1; CountEdges(b,c) += 1
        Apple ← SimpleTriangle(i)
    NEXT i
4. **Initialize Candidate List**
    FOR each Edge(a, b)
        IF CountEdges(a, b) = 1 THEN
            CandidatesList ← Edge(a, b)
            VisitedCities ← a, b
            TourLength += Distance(a, b)
        ENDIF
    NEXT
5. **Carve triangles from Polygon (Apple)**
    change_recorded := 1
    WHILE VisitedCities < n AND change_recorded == 1
        change_recorded := 0
        Let k be the index of a triangle containing the candidate edge Edge(a, b) such that:
            a) CountTriangles(a) > 1 AND CountTriangles (b) > 1 AND CountTriangles(c) > 1,
            b) Min_TI(c) == TIA(c, SimpleTriangle(k))
            c) SimpleTriangle(k) == Triangle(a, b, c) with the min_TI(c) for all triangles satisfying a) and b)
        IF SimpleTriangle(k) doesn't exist THEN
            Let k be the index of a triangle containing any candidate edge Edge(a,b) such that:
                d) CountTriangles(a) > 1 AND CountTriangles(b)>1 AND CountTriangles(c) > 1,
                e) SimpleTriangle(k) = Triangle(a, b, c) with the lowest TIR(c, SimpleTriangle(k)) for all triangles satisfying d)
        ENDIF
        Apple → SimpleTriangle(k)
        CandidatesList ← Edge(a,c), Edge(b,c)
        CandidatesList → Edge(a,b)
        CountTriangles(a) -= 1; CountTriangles(a) -= 1; CountTriangles(a) -= 1
        VisitedCities ← c
        TourLength := TourLength - Distance(a, b) + Distance(a, c) + Distance(b, c)
        VisitedCities += 1; change_recorded := 1
    WHILE END
6. **Correct infeasibility conditions (if any)**
    IF VisitedCities < n THEN
        FOR each point c NOT in VisitedList
            Let a and b be points in S such that
                f) Edge(a,b) is in CandidatesList,
                g) Triangle(a,b,c) has the lowest TIA(c, Triangle(a,b,c)) for any pair of points a and b satisfying f)
            VisitedCities ← c
            CandidatesList ← Edge(a,c), Edge(b,c)
            CandidatesList → Edge(a,b)
            TourLength = TourLength - Distance(a, b) + Distance(a, c) + Distance(b, c)
            VisitedCities += 1
        NEXT c
    ENDIF
7. **Record the polygon tour**
    FOR each Edge(a, b) in CandidatesList
        Predecessor(b) := a
    NEXT

END Apple Carving Algorithm

Fig 4. Apple-Carving Algorithm Pseudocode.

C. Measure of Sub-optimality

We define $\varepsilon_{P''}^{A}(s)$ as the absolute deviation (from the optimal TSP) of the perimeter length of the polygon found via "apple carving" algorithm, and express it mathematically as follows:

$$\varepsilon_{P''}^{A}(s) = \frac{PL(P''(S)))-PL(TSP(S))}{PL(TSP(S))} \times 100\%, \forall S \; in \; R^2 \quad (2)$$

Where S is a given point set, and P'' is the Hamiltonian cycle found by the "apple carving" algorithm.

D. Time Complexity

Theorem 1 The time complexity of the "apple-carving" algorithm is $O(n^2)$.

Proof: Step 2 of the "apple carving" algorithm has the time complexity of O(n), since in this step we initialize arrays of n points. Step 3 of the "apple carving" algorithm has the time complexity of O(n), as we also know that there are O(n) triangles in a full triangulations of a planar point set S of n points [16]. Step 4 of the "apple carving" algorithm loops through no more than n candidate edges, and therefore has time complexity of O(n). Step 5 of the "apple carving" algorithm removes up to n – h triangles from iGCT. In each removal step, we evaluate up to 2n – h – 2 candidate triangles that can be removed. This guarantees time complexity of $O(n^2)$ for Step 5. Step 6 of the "apple carving" algorithm has time complexity of $O(n^2)$. We know this because there are not more than n points that need to be evaluated against up to n candidate edges/triangles. Finally, step 7 of the "apple carving" algorithm assigns predecessors for each of n points in S by looping through not more than n edges in the candidate lists, guaranteeing the time complexity of O(n).

This proves that the time complexity of the "apple carving" algorithm is $4O(n) + 2O(n^2) = O(n^2)$.

Theorem 2 The time complexity of the "apple-carving" algorithm and iGCT algorithm together is $O(n^4)$.

Proof: Time complexity of the stand-alone "apple carving" algorithm is $O(n^2)$. Dodig and Smith proved that the time complexity of the iGCT algorithm is $O(n^4)$ [4].

This proves that the time complexity of the "apple carving" algorithm is $O(n^2) + O(n^4) = O(n^4)$.

E. Space Complexity

Theorem 3 The space complexity of the "apple-carving" algorithm is O(n).

Proof: Number of points in a planar point set S is defined as n. The number of triangles in any full triangulation of S is known to be 2n – h – 2, where h is the number of points belonging to CH(S) [17]. The number of edges in any full triangulation of S is known to be 3n – h – 3, where h is the number of points belonging to CH(S) [17]. This implies that the variables in "apple carving" algorithm tracking both visited cities and candidate edges cannot have the space complexity greater than O(n).

This proves that the space complexity of the "apple carving" algorithm is O(n).

IV. Experimental Methodology

A. Objective

Our experimental objective was to test the validity of the proposed tour construction algorithm experimentally by analyzing how well the length of the resulting Hamiltonian cycle approximates the length of the optimal TSP.

B. Hypothesis

We hypothesize that the "apple carving" algorithm will outperform the traditional Convex Hull algorithm. We further hypothesize that the "apple carving" algorithm will outperform most of the traditional tour construction heuristics.

C. Data Sets

To perform our experiments, we selected 18 problem sets from TSPLIB, a well-known online problem library created to provide researchers with a broad set of test problems from various sources and properties for which the optimal TSP solutions are known [18]. We have chosen 11 problem sets which are given with points in general position (att48, *berlin52*, *ch130*, *eil51*, *eil76*, *eil101*, *gr96*, *gr137*, *rat99*, *rat195*, *rd100*). This was important as point sets in general position do not have 3 or more co-linear points. We have also chosen 7 problem sets with a significant number of co-linear points (*lin105*, *pr76*, *pr107*, *pr124*, *pr136*, *pr144*, *u159*). This was done to test performance of our framework in both point set configurations.

D. Programming

To achieve our experimental objectives we have programmed iGCT Algorithm in VBA for Excel. This algorithm takes a planar point set as an input, and produces a Hamiltonian cycle of S as an output. It also calculates the length of P'' found by "apple carving" algorithm in order to compare to the optimal TSP lengths for each of the problems in our problem set. All of our experiments were performed on Latitude 5490 laptop with Intel Core i5-8250U CPU @ 1.60GHz with 8GB of RAM, running Windows 10 64-bit operating system.

V. Results

Experimental results for 18 given problem sets can be found in Table I.

On average, polygons produced by the "apple carving" algorithm in our test problems are 8.1% longer than optimal TSP solutions. For *gr137* problem, the absolute error is the lowest at 1.9%, and for *pr124* problem, the error is the highest recorded at 15.9%. If we exclude point sets of 3 or more co-linear points, the absolute error drops to the average of 6.1%, with the maximum error recorded for *ch130* problem at 11.2%.

TABLE I. EXPERIMENTAL RESULTS

Set	S	TSP	P''	$\varepsilon_{P''}^{A}$	Co-linear
1	*att48*	108,159	118,702	9.8%	
2	*berlin52*	7,544	7,711	2.2%	
3	*ch130*	6,111	6,793	11.2%	
4	*eil51*	430	453	5.3%	
5	*eil76*	545	578	5.9%	
6	*eil101*	642	701	9.2%	
7	*gr96*	512	534	4.3%	
8	*gr137*	729	743	1.9%	
9	*lin105*	14,383	15,207	5.7%	Yes
10	*pr76*	108,159	112,152	3.7%	Yes
11	*pr107*	44,301	49,653	12.1%	Yes
12	*pr124*	59,030	68,069	15.3%	Yes
13	*pr136*	96,770	108,573	12.2%	Yes
14	*pr144*	58,535	67,867	15.9%	Yes
15	*rat99*	1,219	1,265	3.7%	
16	*rat195*	2,333	2,517	7.9%	
17	*rd100*	7,910	8,426	6.5%	
18	*u159*	42,075	47,354	12.6%	Yes

VI. CONCLUSIONS AND NEXT STEPS

We have introduced a simple algorithm that takes a full triangulation (iGCT) of a planar point set and reduces it to a simple polygon by removing triangles with low Triangle Inequality Measure starting from triangles on the convex hull of this point set. We have proved that the time complexity of the "apple carving" algorithm is $O(n^2)$. We have also shown that the space complexity of the algorithm to be O(n). We have then demonstrated that, on average, polygons produced by this "apple carving" algorithm in our test problems are 8.1% longer than optimal TSP solutions. If we exclude point sets of three or more co-linear points, the absolute error drops to the average of 6.1%, with the maximum error recorded at 11.2%.

Based on these results and our literature review we conclude that "apple carving" algorithm produces better quality of solutions than any other construction heuristics other than match-twice-and-stitch heuristic, as evident in Fig. 5. Here it is important to note that the "apple carving" average results have been adjusted up by 0.8%, since HK lower bound is on average 0.8% lower than the optimal TSP solution [15].

Our initial research hypothesis that the "apple carving" algorithm will produce results superior to that of the classical Convex Hull Algorithm were met (9% average error for "apple carving" versus 12% average error for Convex Hull algorithm). We were also able to demonstrate that the "apple carving" algorithm performs significantly better than all the classical tour construction heuristics and is only slightly outperformed by Match-twice-and-stich heuristic introduced in 2004 [14].

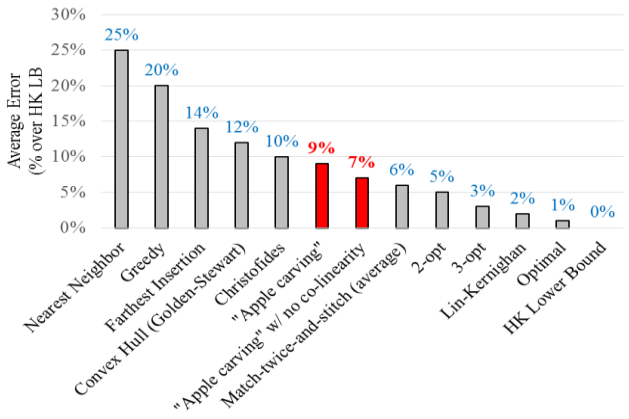

Fig 5. Typical Performance of Cited Heuristics over HK Lower bound (Including "Apple Carving" Algorithm Results).

Limitations in our work lie in the number of TSPLIB instances we used (i.e. 18 problems), as well as in the relatively small problem sizes employed (i.e. maximum of 195 points). To improve quality of our experiments we intend to expand our tests to all named TSPLIB instances, which will also allow us to compare how our algorithm performs on problems of varying size.

Finally, our future work will focus on fine-tuning the "apple carving" algorithm and adding the improvement steps of switching the relevant triangles in and out of the solution polygon depending on whether adding or removing related triangle pairs will result in desired tour improvements. Triangle pairs would be relevant and suitable for "swapping" in and out of the resulting polygon if the share at least one point, and their "swap" would not result in a loss of solution feasibility.

REFERENCES

[1] G. Reinelt, "Fast heuristics for large geometric traveling salesman problems" ORSA Journal on Computing, pp. 206-217, 1992.

[2] W. Stewart, Euclidean traveling salesman problems and Voronoi diagrams. School of Business Administration, College of William and Mary, 1997.

[3] A. N. Letchford, and N. A. Pearson, "Good triangulations yield good tours", Computers and Operations Research, vol. 35(2), 2008, pp. 638-647.

[4] M. Dodig, and M. Smith, "Novel heuristic for approximating minimum weight triangulation of planar point sets", unpublished.

[5] W. Cook, In Pursuit of the Traveling Salesman. Princeton University Press, 2012.

[6] R. Bellman, "Dynamic programming treatment of the travelling salesman problem", Journal of the ACM, vol. 9(1), pp. 61-63, 1962.

[7] R. Garey, D. Johnson, and R. Tarjan, "The Planar Hamiltonian Circuit Problem is NP-Complete", SIAM Journal on Computing, pp. 704-714, 1976.

[8] M. Held, and R. Karp, "A Dynamic Programming Approach to Sequencing Problems", Journal for the Society for Industrial and Applied Mathematics, vol. 10(1), pp. 196-210, 1962.

[9] G. Kizilates, and F. Nuriyeva, "On the nearest neighbor algorithms for the traveling salesman problem", Advances in computational science, engineering, and information technology, vol. 225, pp. 111-118, 2013.

[10] C. Nilsson, Heuristics for the Traveling Salesman Problem. Linköping University, 2003.

[11] D. Rosenkrantz, R. Stearns, and P. Lewis, "Approximate algorithms for the traveling salesperson problem", 15th Annual Symposium on Switching and Automata Theory, pp. 33-42, 1974.

[12] B. Golden, L. Bodin, T. Doyle, and W. Stewart Jr, "Approximate traveling salesman algorithms", Operations Research, vol. 28(3), pp. 694-711, 1980.

[13] N. Christofides, "Worst-case analysis of a new heuristic for the travelling salesman problem (No. RR-388)", Carnegie-Mellon University, Management Sciences Research Group, 1976.

[14] A. Kahng, and S. Reda, "Match twice and stitch: a new TSP tour construction heuristic", Operations Research Letters, pp. 499-509, 2004.

[15] D. Johnson, L. McGeoch, and E. Rothberg, "Asymptotic Experimental Analysis for the Held-Karp Traveling Salesman Bound", Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 341-350, 1996.

[16] R. Osserman, "The Isoperimetric Inequality", Bulletin of the American Mathematical Society, vol. 84(6), pp. 1182-1238, 1978.

[17] T. Vassilev, Optimal Area Triangulations. University of Saskatchewan, 2005.

[18] G. Reinelt, "TSPLIB - a traveling salesman problem library", INFORMS Journal on Computing, pp. 376–384, 1991.