A geometric heuristic for Uncapacitated Vehicle Routing Problem

Nodari Vakhania[a],
Jose Alberto Hernandez[b], Federico Alonso-Pecina[b], Crispin Zavala[b]
(a) Centro de Investigación en Ciencias, UAEMor, Mexico.
(b) Facultad de Contaduría, Administración e Informática UAEMor, Mexico.

***Abstract*—We propose a two-phase construction heuristic for the solution of the classical Euclidean (uncapacitated) vehicle routing problem in which the minimum cost k distinct vehicle tours are to be formed for the given n customer locations. At the first phase we construct a polygon in the 2-dimensional Euclidean space that girds all the given points (customer locations and the depot). The second phase combines the clustering and the routing stages which are performed in an alternate fashion. Iteratively, if the current clustering doesn't bring us to k distinct tours then it is modified and a new routing attempt is made until k distinct tours are formed.**

Key words: vehicle routing problem, heuristic algorithm, Euclidean space, weighted graph

I. Introduction

One of the most practical and also complex combinatorial optimization problems is the Vehicle Routing Problem (VRP) proposed by Dantzig and Ramser in early 1959. The basic (uncapacitated) version of this problem can be stated as follows. We are given an undirected weighted (complete) graph $G = (V, E)$ with edge weights w_e, for all $e \in E$, a distinguished node v_d from set V (called *depot*) and a positive integer number k. $(i, j) \in E$ is the edge connecting node i with node j. For any $Y \subseteq V$ containing node v_d, a *tour* T_Y defined by set Y is a directed cycle that starts at that node, visits every node in Y exactly once and returns to the same node v_d; in other words, $T_Y = (i_1, i_2, \ldots, i_l, 1)$, where $(i_2, \ldots, \imath_l)$ is an enumeration of the nodes in set Y not including node v_d and $i_1 = v_d$. The *cost* of tour T_Y, $c(T_Y)$ is the sum of the weights of the edges on this cycle, i.e., $c(T_Y) = w(i_1, i_2) + w(i_2, i_3) + \ldots + w(i_l, 1)$. VRP aims to find the partition of nodes from set $V \setminus \{v_d\}$ into k subsets $V_1, \ldots, V_k$ with the minimal possible total cost; that is, with the minimal $\sum_{i=1,2,\ldots,k} c(V_i)$.

VRP is a generalization of a well-known Traveling Salesman's Problem (TSP): VRP with $k = 1$ becomes TSP. Multiple TSP, a generalization of TSP with k-tours, k-TSP, is a VRP with k vehicles.

We deal with geometric two-dimensional version of the problem when edge weights represent Euclidean distances between the nodes, considering the nodes themselves as points (cities or customers) in the two-dimensional Euclidean space. The corresponding problem with already 1 vehicle, i.e., the corresponding 1-TSP is already NP-hard Papadimitriou [10]. Therefore, we do not pretend to solve our VRP optimally but rather suggest an efficient heuristic for its solution.

Giving a practical interpretation to VRP, we assume we have k identical distinct resources or vehicles (one for each of the subsets V_i) that can travel in between the cities. The weight $w(i, j)$ is the distance between nodes i and j. We aim to minimize the total travel distance (time) of all the vehicles. There are a number of extensions of VRP, the most common of which is the capacitated version in which every vehicle has a given capacity that cannot be exceeded during its tour.

The vehicle routing problems have been extensively studied, the most of the solutions methods in the literature being heuristic (see, for example, Laporte and Semet [7], Gendreau et al. [4] and Mester and Braysy [9]). There are a few enumerative algorithms as well that work on small instances relatively good (see, for example Lysgaard et al. [8] and Fukasawa et al. [3]). Here we do not pretend to cover all the enormous related work, we rather refer the interested reader to the book edited by Toth and Vigo [11], a newer overview book edited by Golden et al. [5], review papers by Christofides et al. [1], Laporte [6] and Cordeau et al. [2].

In this paper, we propose a two-phase construction heuristic for the the classical Euclidean (uncapacitated) version of the problem. At the first phase we construct a polygon in the 2-dimensional Euclidean space that girds all the given points (customer locations and the depot). The second phase combines the clustering and the routing stages which are performed in an alternate fashion. The initial clustering is formed based on the most "dance" k vectors that bring us from the depot to a vertex of the polygon. This vector defines a tour in which all the points close-by this vector are included. If all the formed k tours cover all the nodes, then our heuristic halts with the created solution. Otherwise, it updates the formed k tours by including in each of them some more points. Iteratively, if the current clustering doesn't bring us to k distinct tours covering all the given n nodes, then it is modified and a new roting attempt is made until k distinct tours including all the n nodes are formed.

To the best of our knowledge, no similar approach has been earlier proposed for the solution of VRP. The computational experiments are still at an early development stage, the heuristic has not been yet tested for large benchmark instances. This is an ongoing work and we hope to have some results in a close future.

II. Phase 1: The construction of the girding polygon

Without loss of generality and for the commodity, we assume that the given n points from set V have non-negative coordinates (otherwise, we can shift them uniformly without changing the distances between them).

Our task at phase 1 is to determine a special kind of a (closed) convex polygon that contains all the nodes from set V girding in this way the whole tour area. No node from set V may be left outside the area of such a polygon, and, of course, there are many such polygons. Though, we are interested in the minimal such convex polygon with its edges containing the maximal possible number of nodes from set V. We shall refer to this particular polygon as the *girding polygon* for set V and denote it by $P = P(V)$.

It follows that all vertices of polygon $P(V)$ are nodes from set V. Besides these vertices, polygon P may contain nodes from set V as the interior points of its edges, whereas the rest of the nodes from set V are within the internal area of the polygon. In Fig. 1 we illustrate polygon P for a problem instance of VRP with 30 customers and one depot.

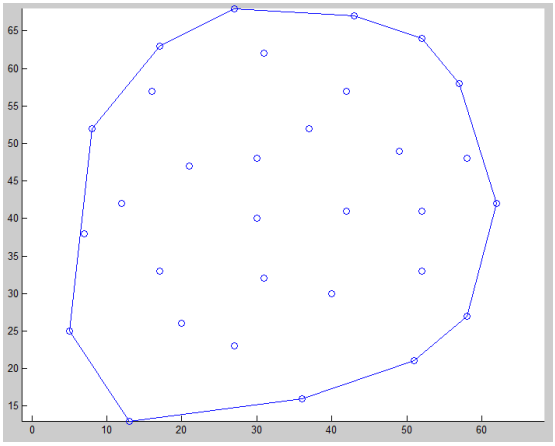

Fig. 1. Polygon with 12 vertices and 19 nodes from its internal area

Before we describe our procedure for the construction of polygon $P(V)$, we define special types of nodes from set V that pertain to polygon $P(V)$. These are the uppermost, lowermost, leftmost and rightmost points from V. Formally, we call a point in set V with the maximum (minimum, respectively) y-coordinate an *uppermost* (a *lowermost*, respectively) node. Likewise, we call a point in set V with the maximum (minimum, respectively) x-coordinate a *rightmost* (a *leftmost*, respectively) node.

We shall refer to these four types of nodes as *extreme* points in set V. From all the extreme points of the same type (if there are two or more such nodes), we call *exterior* nodes the two nodes with the maximum and minimum co-coordinate, and the rest of the nodes the *interior* nodes of that type. The latter two nodes are the endpoints of the corresponding edge on polygon P. In general, we may have more than two nodes from set V lying on the same edge of polygon P, two exterior ones of which are endpoints of that edge.

The next observation is straightforward.

Observation 1: All extreme points belong to polygon P, whereas all extreme points of the same type belong to the same edge of that polygon. In particular, the two edges containing all the uppermost and all the lowermost nodes are parallel to the x-axes, and the two edges containing all the rightmost and all the leftmost nodes are parallel to the y-axes.

We use the following notation for the four distinguished extreme points. (1) v_1 is the right exterior uppermost node (i.e., among all the uppermost nodes v_1 has the maximum x-coordinate); (2) v_l is the lowest exterior leftmost node (i.e., among all the leftmost nodes v_l has the minimum y-coordinate); (3) v_o the right exterior lowermost node (i.e., among all the lowermost nodes, v_o has the minimum x-coordinate); (4) v_r is the lowest exterior rightmost node (i.e., among all the rightmost nodes v_r has the minimum y-coordinate).

The procedure POLYGON that forms the polygon P, first determines all the extreme points verifying the corresponding coordinates in the straightforward way. It declares node v_1 as the first vertex of polygon P. Starting from vertex v_1, the construction of polygon P goes on the following four stages (which, in principal, are independent and can be carried our in parallel).

At stage 1 the construction proceeds in the "right-to-left" downward fashion that moves from vertex v_1 towards vertex v_l completing the upper left border of polygon P. At stage 2 the construction proceeds also in the "right-to-left" but upward fashion that moves from vertex v_o towards vertex v_l completing the lower left border of polygon P. At stage 3 the construction proceeds in the "left-to-right" upward fashion that moves from vertex v_o towards vertex v_r completing the lower right border of polygon P. At stage 4 the construction moves also in the "left-to-right" but downward fashion that moves from vertex v_1 towards vertex v_r completing the upper right border of polygon P. Below we describe these stages in more details.

At stage 1 ("right-to-left" downward) we add points to the left of the latest added so far point to polygon P, initially, to the left of vertex v_1. We first determine the next to v_1 vertex v_2 to the left of vertex v_1 (i.e., $x_2 < x_1$) on the projected border of polygon P. v_2 is the uppermost closest to v_1 vertex on its left hand side. In other words, among all nodes in set V with no-larger than y_1 y-coordinate and no-larger than x_1 x-coordinate, v_2 has the maximum y-coordinate (note that by the definition of the initial node v_1, no other node in set V may have a larger than y_1 y-coordinate). If it turns out that $y_2 = y_1$, the corresponding edge of polygon P (one containing nodes v_1 and v_2) is parallel to x-axes. The next point v_3 on the border of polygon is similarly defined, where we restart now from node v_2 (replacing of node v_1), which is now the next node in set V with the maximum y-coordinate. If $y_3 = y_2$ then all three nodes v_1, v_2, v_3 lie on the same edge of polygon P and node v_2 turns out to be an intermediate

point on that edge, i.e., it is not a vertex of polygon P, but it belongs to its border. Observe that v_1 is a vertex of polygon P, whereas whether v_3 is a vertex of polygon P or not, depends on whether for the next point v_4, y_4 is equal to or is less than y_3 (it cannot be more).

The "right-to-left" downwards pass of stage 1 ends by adding the leftmost vertex (verices) (ones with the minimum x-coordinate) to polygon P; if there are several such nodes in set V, polygon P possesses an edge parallel to the y-axes containing all these nodes (Observation 1), which is the leftmost edge of the polygon. All of these nodes are successively added and stage 1 ends by adding the lowermost such node (one with the smallest y-coordinate), which we denoted by v_l.

Stage 4 works as stage 1 with the only difference that the construction here moves in the "left-to-right" (instead of "right-to-left") downward fashion from vertex v_1 now to vertex v_r. In the description of stage 1 above, we merely replace "left" with "right" and vertex v_l with vertex v_r.

At stage 2 ("right-to-left" upward) we add points to the left of the latest added so far point to polygon P, starting from vertex v_o. We determine the next to v_o vertex v_ω to the left of vertex v_o on the projected border of polygon P. v_ω is now the lowermost closest to v_1 vertex on its left hand side. In other words, among all nodes in set V with no-smaller than y_o y-coordinate and no-larger than x_o x-coordinate, v_ω has the minimum y-coordinate. As at stage 1, it may again turn out that $y_\omega = y_o$, in which case we have an edge in polygon P parallel to x-axes. The next point on the projected border of polygon P is similarly defined, where we restart now from node v_ω (replacing of node v_o), which is now the next node in set V with the minimum y-coordinate. Proceeding in this way, the construction at stage 2 ends by matching the latest added vertex with vertex v_l (by the construction, this event will clearly take the place).

Stage 3 works as stage 2 with the only difference that the construction here moves in the "left-to-right" (instead of "right-to-left") upward fashion from vertex v_o now to vertex v_r. In the description of stage 2 above, we merely replace "left" with "right" and vertex v_l with vertex v_r.

This completes the description of procedure POLYGON. It straightforwardly follows from the construction that the obtained polygon is convex and contains all the nodes from set V either on its border or within it, and among all such polygons it is minimal. Hence, it is the girding polygon $P(V)$.

The brutal sequential time complexity of procedure POLYGON is $O(n^2)$. Indeed, initially, the selection of each of the extreme nodes v_1, v_l, v_o, v_r takes time $O(n)$. At all stages, the determination of every next added vertex takes also time $O(n)$, and since there are no more than n added points to the constructed polygon, we get a brutal sequential overall time of $O(n^2)$.

Thus we have proved the following result.

Theorem 1: Procedure POLYGON creates the girding polygon $P(V)$ in brutal sequential time $O(n^2)$.

III. Phase 2: The partition and routing

In this section we aim to form the k vehicle tours using the girding polygon $P(V)$ constructed at phase 1. For the convenience, assume for now that the depot v_d is within the internal area of polygon P (it normally shares the central area in between the rest of the nodes).

Let m be the number of the *border* nodes, i.e., ones on the border of polygon P. We tie every such border node v with the depot v_d associating with it an *auxiliary* edge (v_d, v) incident with these two nodes. As a result, we cerate m auxiliary edges of this type. They partition the interior area of polygon into the m corresponding triangle areas (see Fig. 2); every triangle is uniquely defined by two neighboring border nodes (not necessarily the endpoint of the corresponding edge of P) and the depot v_d (which share all the triangles). We associate with every edge the corresponding vector in the 2-dimensional Euclidean space and will refer to both, an edge and the corresponding vector interchangeably.

We define the weight of every auxiliary edge (v_d, v) as the length of the corresponding vector in the 2-dimensional plane.

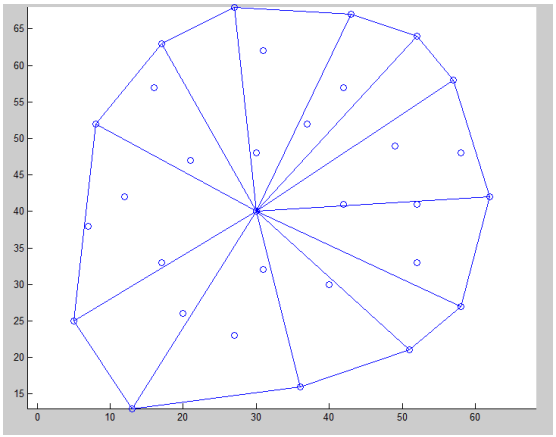

Fig. 2. Polygon of Fig 1 with auxiliary edges

Lemma 1: Twice the minimal auxiliary edge weight plus the length of the border of polygon $P(V)$ is a lower bound $L(V)$ on the optimal solution.

Proof. Immediately follows from an easily seen observation that the above magnitude is the optimal tour length for the case $k = 1$ and for the subset of V containing only the nodes of polygon P and the depot. □

It is not difficult to see that the total length of a tour including a border node v is at least $2w(v_d, v)$. Our algorithm employs an intuitively clear consideration that it is reasonable to include in such a tour also the nodes from the interior of polygon P which lie "across" edge (v_d, v), ones "close by" trajectory of this "local tour" (i.e., the edge (v_d, v)) and perhaps the border node(s) of polygon P which are also "close-by" node v (if in total i neighboring border nodes are included, nodes from the corresponding i triangle areas might be joined together forming one of the destiny k subsets of set V). Part of the nodes can be visited on the way from depot

to node v, whereas the rest of them might be visited on the way back to the depot. In this way, the straight-line tour from the depot to a border node and back from a border node to the depot is substituted by a zigzag-like tour that includes the neighboring nodes on the way. We describe this construction in more details below.

First, we note that the notion of the closeness of two nodes is relative and is dependent on several factors. One such a factor is the number k: the more is k, the freer we are to separate different triangle areas, whereas for "small" k-s, we will be obliged to join together more triangle areas. Other relevant parameters that we may take into account are the length of edge (v_d, v), $w(v_d, v)$, its "relative length" which we let to be the ratio of the average auxiliary edge length (i.e., the sum of weights of all the m edges divided by m) and $w(v_d, v)$, and the lower bound $L(V)$ from Lemma 1 which is a magnitude, slightly more than the perimeter of polygon P.

Let the magnitude b be our closeness measure that, in one way or the other, takes into account the above parameters (in practice, we try different kinds of combinations of these parameters for calculating magnitude b).

For a given auxiliary edge e, the *b-close* set of nodes is formed by all the nodes from set V which are within the distance b from vector e. We denote the b-close set for an edge e by $B(e)$.

Next, we define the *density factor* of an auxiliary edge e, $\delta(e)$ as the number of nodes in set $B(e)$ divided by the length of the corresponding vector. We tend to direct our tours across the auxiliary edges with higher density factors.

Our algorithm determines the auxiliary edges with the $2k$ (or less) greatest density factors, and then partitions the nodes according to these auxiliary edges. Two neighboring auxiliary edges straightforwardly determine one or more triangle areas: if there are i intermediate auxiliary edges in between these two selected auxiliary edges then $i + 1$ triangle areas, are determined by the above pair of auxiliary edges. In this way, the selected $2k$ auxiliary edges with the highest density factors partition the interior area of polygon P into $2k$ triangle areas. Next comes the clustering step which is set up while constructing the k tours determined by the selected k auxiliary edges.

Let $e_1, e_2, \ldots, e_2k$ be an enumeration of the selected auxiliary edges in the non-increasing order of their density factors, $\delta(e_1) \geq \delta(e_2) \geq \ldots \geq \delta(e_{2k})$. Our heuristic generates the k tours for the above $2k$ edges in this order. We describe now how this works for an edge $e = (v_d, v)$ from this enumeration. For the commodity in this presentation, we assume that vector e is on the y-axes of the coordinate system and node v_d coincides with the origin $(0, 0)$ (note that the rotating the whole polygon area by the necessary angle and shifting it will not change any problem data).

We need to create a tour that includes nodes v_d and v and all the rest of the (intermediate) nodes from set $B(e)$. This tour, as earlier noted, has a zigzag type trajectory and consists of a number of "slices". Every slice defines a local tour on the left or right side of vector e and is restricted by a fixed magnitude β called the *thickness factor*, we let $\beta = \alpha b$, for some real $\alpha > 0$ (we normally let $\alpha < 1$). Roughly, the thickness factor determines the amplitude within which the nodes will be included in the generated local tour. We define below such local tours.

Let y_1 be the furthest point from node v_d in set $B(e)$ whose y-coordinate is at most β more than that of node v_d. If there is no such a point, i.e., the minimum y-coordinate of a job from set $B(s) \setminus \{v_d\}$ is more than β more than that of node v_d, we (repeatedly) replace point v_d with the point $(0, i\beta)$ on the y-axes, for the minimum integer i, as long as that point remains within the area of polygon P (i.e., it is on vector e) until point y_1 is determined (or point $(0, i\beta)$ turns out to be outside of the the area of polygon P). Denote the determined in this way point by v'. Without loss of generality and for the purpose of this description, assume node y_1 has a positive x-coordinate, i.e., it is on the right-hand side of vector e.

The vector (v', y_1) forms the skeleton of the local tour from point v' to node y_1, i.e., it is across that vector. Let $\beta(v', y_1)$ denote the set of nodes from $B(e)$ which are within the distance β from vector (v', y_1) on the same side of vector e. In a local partial tour defined by the former vector, all the nodes from set $\beta(v', y_1)$ are visited in the order of the closeness from node v'. In other words, if we let $i_1, i_2, \ldots, i_l$, $i_l = y_1$, be an enumeration of nodes in set $\beta(v', y_1)$ in the non-decreasing order of their distances from node v', then the node i_1 is visited from node v_d, then node i_2 is visited from node i_1, and so on, node $y_1 = i_l$ is visited from node i_{l-1}. Note that the latest visited node in this local tour is y_1 and that it included all the nodes in set $B(e)$ located below vector (v', y_1).

Now we replace point v' with the next point v'' of the same form $(0, i\beta)$ and defined similarly as point v', and carry out a similar construction. I.e., we determine the next furthest point y_2 now from point v'' with the y-coordinate no more than $i\beta$ (for the newly determined value of i), form vector $((0, i\beta), y_2)$ and the next local tour (on the right or the left hand side of vector e, depending on the potion of point y_2).

Consider the stage in the above tour when the border node v is reached, and let z be the next to v border node, the border endpoint of the next to e neighboring auxiliary edge with the next largest density factor. Similarly to the local tour from node v_d to node v, we create another local tour from node v to node z including in it all the "close-by" nodes from the corresponding edge(s) of polygon P (edge v, z if there is no intermediate edge node, otherwise all the corresponding intermediate edges between nodes v and z). Once we reach node z, we form similarly the last portion of the tour that goes from node z to the depot.

The above generated tour covers the triangle area defined by the auxiliary edges (v_d, v) and (v_d, z) and part of the two neighboring triangle areas, one adjacent to edge (v_d, v) and the other adjacent to edge (v_d, z).

Once all the $2k$ auxiliary edges are processed as above described, if all the nodes from set V are included in one

of the k generated tours then our heuristic completes with the created solution. Otherwise, some of the formed tours are completed with the remained nodes at the successive iterations with modified values for parameters b and β. Due to the page limitation, some housekeeping details and figures are omitted in this description (will be included in the extended version of the paper).

IV. Concluding remarks

We have presented a novel heuristic algorithm for the classical uncapacitated version of the vehicle routing problem. Our heuristic has some degree of flexibility due to the possibility of defining the parameters of the algorithm differently. These parameters can be calculated in different ways adapting to the nature of the input problem instances. As to the future work, we plan to carry out extensive computational experiments testing the behavior of the heuristic, and also to extend the heuristic for the capacitated version of the problem.

References

[1] N. Christofides, A. Mingozzi, and P. Toth. The vehicle routing problem. In N. Christofides, A. Mingozzi, P. Toth, and C. Sandi, editors, *Combinatorial Optimization*, chapter 11, pages 315 338. John Wiley & Sons (1979).

[2] J.-F. Cordeau, M. Gendreau, A. Hertz, G. Laporte, and J.-S. Sormany. New heuristics for the vehicle routing problem. In *Logistics Systems: Design and Optimization*, GERAD (Chapter) pp 279-297, Springer (2005)

[3] R. Fukasawa, J. Lysgaard, M. P. de Aragão, M. Reis, E. Uchoa, and R. F. Werneck. Robust branchand-cut-and-price for the capacitated vehicle routing problem. In *Proceedings of IPCO X. Columbia University* (2004)

[4] M. Gendreau, G. Laporte, and J.-Y. Potvin. Metaheuristics for the capacitated vrp. In P. Toth and D. Vigo, editors, *The Vehicle Routing Problem*, volume 9 of SIAM Monographs on Discrete Mathematics and Applications, chapter 6, pages 129154, SIAM, Philidelphia (2002)

[5] Golden, Bruce L., Raghavan, S., Wasil, Edward A. (Eds.) The Vehicle Routing Problem: Latest Advances and New Challenges. Springer (www.springer.com/us/book/9780387777771) (2008)

[6] Laporte G. The vehicle routing problem: An overview of exact and approximate algorithms. *European Journal of Operational Research* 59(3) 345-358 (1992).

[7] G. Laporte and F. Semet. Classical heuristics for the capacitated vrp. In P. Toth and D. Vigo, editors, The Vehicle Routing Problem, volume 9 of *SIAM Monographs on Discrete Mathematics and Applications*, chapter 5, pages 109128. SIAM, Philidelphia (2002)

[8] Lysgaard, A. N. Letchford, and R. W. Eglese. A new branch-and-cut algorithm for the capacitated vehicle routing problem. *Mathematical Programming*, Ser. A, 100:423445 (2004)

[9] D. Mester and O. Braysy. Active guided evolution strategies for large-scale vehicle routing problems with time windows. *Computers and Operations Research* 32, pages 1593 - 1614 (2005)

[10] Papadimitriou, C. H. The Euclidean traveling salesman problem is NP-complete. *Theoretical Computer Science* 4, 237-244 (1977).

[11] Toth P., Vigo D. (editors) Vehicle routing problem *SIAM monographs on discrete mathematics and applications* 386 SIAM, Philidelphia (2002).