Bundling n-Stars in Polygonal Maps

Victor Parque*† and Tomoyuki Miyashita*
*Department of Modern Mechanical Engineering, Waseda University, Japan
{parque@aoni.waseda.jp, tomo.miyashita@waseda.jp}
†Department of Mechatronics and Robotics, Egypt - Japan University of Science and Technology, Egypt
{victor.parque@ejust.edu.eg}

***Abstract*—This paper aims at computing minimal-length tree layouts given an n-star graph in a polygonal map. This problem is strongly related to the edge bundling problem, which consists of compounding the edges of an input graph to obtain topologically compact graph layouts being free of clutter and easy to visualize. Computational experiments using a diverse set of polygonal maps and number of edges in the input graph shows the feasibility, efficiency and robustness of our approach.**

I. Introduction

Aiming towards the realization of efficient routing of resources in the integration, communication, and coordination of large-scale networked systems, our interest is to tackle the *cherry bundling* problem which consists of computing minimal-length tree layouts given an n-star graph, or *cherry* [1], [2], in a polygonal map.

Generally speaking, an instance of the cherry bundling problem consists of the following elements:

- a source $r \in \mathbb{R}^2$,
- a finite non-empty set $S \subseteq \mathbb{R}^2$ of terminals, where a sense of *direction* from $r \rightarrow \forall s \in S$ is implicit.
- a polygonal map $P = \{p_1, p_2, ..., p_k\}$ with k polygons denoting *obstacles* in the environment.

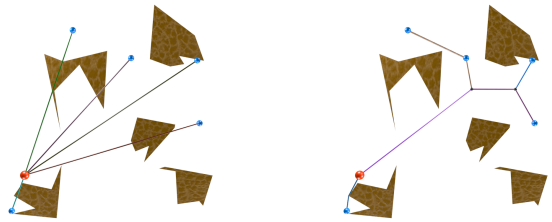
And a feasible solution of the above instance is

- a rooted tree $T = (V, E)$ with $V = \{r\} \cup S \cup I$ in which $I \subseteq \mathbb{R}^2$ is the set of intermediate vertices of T, and the elements of S are either intermediate or terminal vertices of T.

To give a glimpse of the above instance, Fig. 1 shows a basic example of the cherry bundling problem. In this example, Fig. 1(a) shows the input elements: source r is depicted by a red-colored node, the elements of $s \in S$ are depicted by blue-colored nodes, and the elements of P are depicted by brown-colored polygons.

In line of the above, the feasible solution is depicted by Fig. 1(b), in which a minimal-length tree avoiding obstacles is computed. Note that new intermediate nodes are inserted to minimize tree-length, and nodes $s \in S$ are either intermediate or terminals of T.

A closely related line of work in the existing literature is the more general and widely-studied *edge bundling* problem, which consists in compounding multiple edges of a graph into *overlapped* and *colinear* edges to obtain holistically compact topologies. The edge bundling concept has recently attracted the attention of researchers due to its potential to realize



(a) Input Graph (b) Tree

Fig. 1. Basic Concept

clutter-free visualization of complex networks. That is, given a graph G, bundling can be thought as a mapping $\beta : G \rightarrow G'$, where G' is a compact structure whose edges represent spatial density of the edges of G. Well-known bundling schemes include the modeling of edges as flexible springs being *co-attractable* with respect to compatibility metrics based on angle, scale, position and visibility [3], and the inclusion of edge directions, connectivity patterns and edge weights into the attraction-based approach [4]. Also, the approach of forcing edges to pass through points in a control mesh [5]; and the strategy of using kernel functions as a graph operator [6]. Furthermore, there exists the optimization of ink usage through multilevel agglomeration of edges [7]; the routing of edges along tree layouts by using B-splines [8]; the convex optimization of internal trunks in hierarchical layouts that enable modular bundles [9], and the nature-inspired optimization bundling in non-hierarchical bipartite networks [10], [11].

Although the above studies have rendered graph bundles being clutter-free, topologically compact and easy to visualize; the study of edge bundling of n-star trees under the presence of arbitrary obstacles and optimality considerations has received little examination in the existing literatures. In this study, we aim at contributing to the field by proposing the first algorithm for edge bundling of n-stars considering optimality, as well as obstacle avoidance.

Another closely related line of work is the *Steiner tree* problem, which consists of finding a minimal-length graph (tree) connecting n nodes. The problem first appeared in the 30's [12], and was later popularized by Robbins and Courant in the 40's [13]. The generalized version, the Steiner tree with

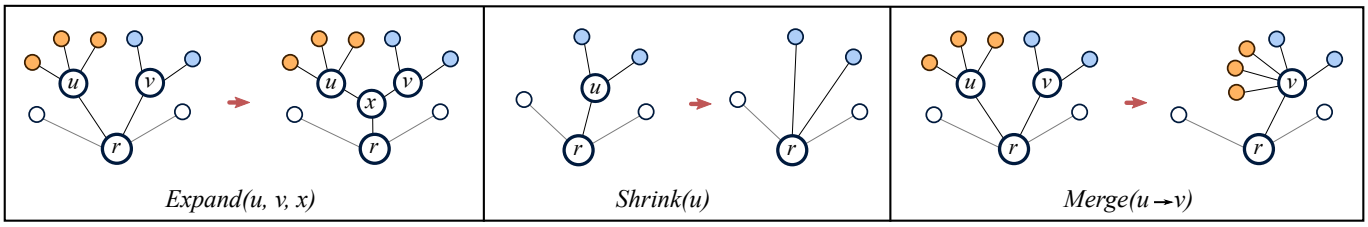


Fig. 2. Basic operations in tree $T = (r, \{..., u, ..., v, ...\})$. These fundamental operators are used due to the fact of allowing not only the growth, but also and shortening in both depth and breadth.

obstacles in a polygonal map, has received recent attention due to its direct implications to design optimal VLSI systems. In [14], a two step approach was proposed: first, the authors generate a routing graph (an extended full Steiner tree) and then shortest paths are generated by a search heuristic that considers slew constraints. In [15], a linear time algorithm for Steiner tree using layouts up to four terminals was proposed. In [16], a three-phase approach was proposed: first, the authors generate connected components (hypergraphs), then trees are computed by connecting terminals in each hypergraph (sub-trees), and finally the sub-trees are connected to span all terminals. In [17], the obstacle-avoiding steiner minimal tree considering visibility graphs was proposed. In [18], a parallel approach to generate multiple routes simultaneously was considered; and in [19], an $O(nlog^2n)$ approximation scheme was proposed (n is number of terminals plus obstacle vertices). In summary, the above studies have rendered algorithms which approximate solutions to the (Rectilinear) Steiner tree in the presence of obstacles and given n nodes in the plane. In contrast to the above studies, having received no attention, finding minimal-trees given an n-star imposes connectivity constraints which implies rendering topologically-constrained Steiner trees. In this study, the focal point of our contribution lies in how to compute minimal trees given an n-star in a polygonal map.

Thus, our contributions is a nature-inspired algorithm for searching optimal tree bundles while preserving connectivity of the input graph while avoiding obstacles in a polygonal domain.

In the rest of this paper, section 2 describes our proposed algorithm, section 3 describes and discusses our computational experiments, and section 4 concludes our paper..

II. Proposed Method

In this section we describe the basic preliminaries, as well as the algorithmic concepts in our proposed approach.

A. Preliminaries

We tackle the *cherry bundling* problem, which assumes the existence of the following:

- a source $r \in \mathbb{R}^2$.
- a finite non-empty set $S \subseteq \mathbb{R}^2$ of *terminals*, where a sense of *direction* from source $r \rightarrow \forall s \in S$ is implicit (bipartite directed graph).
- a polygonal map $P = \{p_1, p_2, ..., p_k\}$ with k polygons denoting *obstacles* in the environment.

Here, from a logistics perspective, the source represents the central (emitter) hub whereas the terminal units represent the decentralized receptor units.

Then, we aim at bundling (compounding) the source-destination pairs $r \rightarrow S$ to obtain minimal-length connectivity from the source r towards the terminal set S while avoiding elements (obstacles) in the polygonal map P. The expected result is a *rooted tree* $T = (V, E) = (r, L)$, in which $V = \{r\} \cup S \cup I$, $I \subseteq \mathbb{R}^2$ is a set of intermediate vertices of T, r is the root of T, and $L = \{..., u, ..., v, ...\}$ is the set of forest leaves of T, in which u, v are trees, by recursive definition. Note that elements of S can be either intermediate or leaf vertices of T.

Furthermore, let the following be:

Definition II.1. $id(T)$
is the unique identification of T, in which $id(T) \in \mathbb{N}$. The identification is used for bundling (compounding) purposes.

Definition II.2. $leaves(T)$
is the set of leaves of T. Let $u \in leaves(T)$, then by recursive definition u is a tree $u = (u_r, u_L)$ rooted at u_r and having leaves $u_L = leaves(u)$.

Definition II.3. $leaf(T, k)$
is the leaf of T having k as its unique identification, that is $id(u) = k$, for $u \in leaves(T)$ and $k \in \mathbb{N}$.

Definition II.4. $end(T)$
is the set of terminal nodes of T. Here, elements of $end(T)$ have no leaves, that is $|leaves(u)| = 0$ for $u \in end(T)$.

The above definitions are the building blocks in our proposed approach; in the subsequent section, we describe the fundamental operators used in our scheme.

B. Tree Operations

We allow the initial n-star graph (given as input) to *evolve* through a finite number of tree operators, as follows:

Definition II.5. $Expand(u, v, x)$
Let $u, v \in leaves(T)$ and x be a node in the polygonal map; the expand operator inserts x into the set $leaves(T)$, and moves the leaves u, v into the set of $leaves(x)$ (refer to Fig.2, left).

Definition II.6. $Shrink(u)$
Let $u \in leaves(T)$; the shrink operator deletes leaf u and moves the set $leaves(u)$ into the set $leaves(T)$ (refer to Fig.2, middle).

Definition II.7. *Merge*$(u \to v)$
Let $u, v \in leaves(T)$; the merge operator moves the set $leaves(u)$ into the set $leaves(v)$. After this procedure, the leaf u is deleted from the set $leaves(T)$ (refer to Fig.2, right).

The above operations are inspired by natural phenomena in trees. For example, the expand operator resembles the growth of the trunk in natural trees, while the shrink operator resembles the survival strategy of ancient and large trees. The merge operator resembles inosculation, which is a natural phenomenon where trunks grow together. The above operators are used due to the fact of allowing not only growth, but also shortening of the tree in both depth and breadth. The study of a richer set of operators which allows finer degree to tree manipulation is in our agenda. In the following subsection, we describe the basic concept of our proposed algorithm.

C. Basic Algorithm

The basic concept of our proposed approach is shown in Algorithm 1. In the following points we describe the dynamics and the main points:

The input

The *input* in our algorithm is the tree $T_o \leftarrow (r, S)$, representing the initial guess of the minimal tree structure; and the output is the tree aiming at minimizing the total length while preserving the root at source r, preserving the connectivity towards the nodes in the terminal set S, and avoiding the obstacles of the polygonal map P.

Clustering Routes

The subscript ρ denotes the set of *shortest routes* from the source r to each terminal node of T. At initial iteration, $|\rho| = |S|$, since all leaves of T_o are terminal nodes, while at later iterations $|\rho| < |S|$ since some terminals become intermediate nodes due to merging, expansion and shrinkage.

The hierarchical clustering of the set ρ computes the *dendogram* $\mathbf{Z} = [z_{ij}] \in \mathbb{R}^{|\rho|\times 2}$, in which the number of terminals of $\mathbf{Z}$ is equivalent to $|\rho|$. The rows of $[z_{ij}]$ are ordered in a way that the first (last) rows indicate higher (lower) degree of similarity between columns (which indicate the index of the route). In other words, let a distance metric $d : \rho_i, \rho_j \to \mathbb{R}$ map the distance between routes ρ_i and ρ_j, and let $k \in [|\rho|]$, then the matrix $\mathbf{Z} = [z_{ij}] \in \mathbb{R}^{|\rho|\times 2}$ is ordered such as the following condition is fulfilled $d(\rho_{z_{k,1}}, \rho_{z_{k,2}}) < d(\rho_{z_{k+1,1}}, \rho_{z_{k+1,2}})$.

For clustering and similarity computation, the distance between two routes is computed by the following metric:

$$d(\rho_i, \rho_j) = \left(\sum_{k=1}^{SP} ||\rho_i^k - \rho_j^k||^2 \right) . \left(cos^{-1}\left(\frac{\mathbf{a_i}.\mathbf{a_j}}{|\mathbf{a_i}||\mathbf{a_j}|} \right) \right) \quad (1)$$

, where $\rho_i \in \rho$, ρ_i^k is the k-th sampled point along the route ρ_i, SP is the total number of 1-D equally-separated interpolated points along the route ρ_i, and $\mathbf{a_i} = \rho_i^{end} - \rho_i^{init}$ in which $\rho_i^{init}, \rho_i^{end} \in \mathbb{R}^2$ are the *starting* and the *end* coordinates of the root ρ_i, respectively. The main rationale of using the above distance metric is due to its key benefit of measuring

Algorithm 1 Cherry Bundling

procedure BUNDLE(T_o)
  Input $T_o = (r, S)$ ▷ Source and Terminals
  Output $T = (r, \{..., u, ..., v, ...\})$ ▷ Tree
  $T \leftarrow T_o$ ▷ Initial solution (r, S)
  if $|leaves(T)| > 1$ **then**
    $\rho \leftarrow$ Shortest paths from r to $end(T)$
    $\mathbf{Z} \leftarrow$ **Hierarchical Clustering** of ρ
    for $k \leftarrow 1$ **to** $|\rho|$ **do**
      $u \leftarrow leaf(T, z_{k,1})$
      $v \leftarrow leaf(T, z_{k,2})$
      $x \leftarrow \underset{x}{\textbf{Minimize}}\ J(T, u, v, x)$
      if x is *far* from r **then**
        if x is *close* to u or v **then**
          $Merge(farthest \to closest)$
          $id(closest) \leftarrow k$
        else
          $Expand(u, v, x)$
          $id(x) \leftarrow k$
        end if
      else
        if x is *close* to u or v **then**
          $Shrink(closest)$
        end if
        $id(T) \leftarrow k$
      end if
    end for
  else
    $Shrink(leaves(T))$
  end if
  for each $u \in leaves(T)$ **do**
    BUNDLE(u)
  end for
  return T
end procedure

procedure STRAIGHTEN(T)
  for each $u \in leaves(T)$ **do**
    if u is an *intermediate node* **then**
      $x \leftarrow \underset{x}{\textbf{Minimize}}\ H(T, u, x)$
      $u_r \leftarrow x_r$
    end if
  end for
  for each $u \in leaves(T)$ **do**
    STRAIGHTEN(u)
  end for
end procedure

not only piecewise gaps (due to difference in topology of the routes), but also orientation gaps (due to arbitrariness of end nodes in the map). Furthermore, note that the above distance metric is able to be performed under parallelization schemes, bringing benefits in scalability for large-scale applications.

Searching Anchoring Points The dendogram *dendogram* $\mathbf{Z}$ is useful to identify which routes are similar and are promising to be *bundled* (since bundling similar routes will render compact structures). Thus, it is desirable to find anchoring nodes for routes which are similar from the point of view of piecewise and orientation gaps. Specifically, the anchoring node x is found by minimizing the following cost function:

$$J(T,u,v,x) = \sum_{w \in \{r,u_r,v_r\}} \ell(x,w) \tag{2}$$

, where r is the root of T, the leaf $u,v \in leaves(T)$ satisfy $id(u) = z_{k,1}$, and $id(v) = z_{k,2}$ respectively, $\ell(x,w)$ is the length of the *shortest route* from node x to node w along the polygonal map P. The above cost function has the role to measure the length of an intermediate node connecting r, u_r and v_r, in which u_r and v_r are the roots of trees u and v respectively. Then, by minimizing the above cost function, we aim at finding a minimal 3-star which links the root r of T and the leaves u,v. Furthermore, since the cost function J is a non-convex function due to the presence of the polygonal map P, we use a non-linear optimization based on the interior-point algorithm [20]. In order to ease the sampling of feasible points that avoid the obstacles in the polygonal map, we represent the coordinates of the root of x by the following 3-tuple:

$$x_r = (i, \lambda_1, \lambda_2) \tag{3}$$

, where $i \in [|\tau|]$ and $\lambda_1, \lambda_2 \in [0,1]$/ Here, τ is the set of triangles of the Delaunay Triangulation of the convex hull that spans node x, r, u and v. In the above representation, i is the index of the i-th triangle $\tau_i \in \tau$, and λ_1, λ_2 are real numbers in the interval $[0,1]$. The unique feature of the above representation lies in the ability to encode arbitrary points which guarantee to be inside the *(free) navigable space*. The reader may note tat the following relation holds: $x_r \in \mathbb{N}^{[|\tau|]} \times \mathbb{R}^{[0,1]} \times \mathbb{R}^{[0,1]}$. The equivalent 2-dimensional *cartesian* coordinates can be easily computed as follows [21]:

$$x_r^C = (1-\lambda_1)A_i + \sqrt{\lambda_1}(1-\lambda_2)B_i + \sqrt{\lambda_1}\lambda_2 C_i \tag{4}$$

where x_r^C is the cartesian coordinate of x_r, and A_i, B_i, C_i are the 2-dimensional *cartesian* coordinates of the vertices of the i-th triangle $\tau_i \in \tau$.

Tree Operations

Merging occurs when the anchoring node x is far from r, yet close to either u or v. Farness of node x to r, and closeness of x to u,v is computed by $\ell(x,r) > \delta_1$ and $D(r,u,v) < \delta_2$, respectively, where:

$$D(r,u,v) = min(\ell(r,u), \ell(r,v)) \tag{5}$$

The role of using the user-defined thresholds δ_1 and δ_1 is to allow flexibility and granularity when designing and generating minimal trees: smaller (larger) values of δ_1, δ_2 creates more (less) intermediate nodes x, thus the global tree length is expected to be small (large). Tackling the optimal trade-off in tree granularity and length is out of the scope of this paper, and left for future work.

Then the *farthest* leaf (u or v) is merged to the *closest* one (u or v), in which the *closest* leaf is computed by the following metric:

$$closest = \begin{cases} u, & \text{for } \ell(x,u) < \ell(x,v) \\ v, & \text{for } \ell(x,u) > \ell(x,v) \end{cases} \tag{6}$$

As a natural consequence, the *farthest* node is the opposite of the above. After *merge* operation, the *id* of the closest node is set to k in order to continue with the ordered sequence of bundling.

Expansion occurs when the anchoring node x is *far* from the root r, and *far* from leaves u,v. After *expand* operation, the *id* of the leaf x is set to k in order to continue with the ordered sequence of bundling.

Shrinkage occurs either when the tree has a single leaf, or when the anchoring node x is *close* to the root r and *close* to either u or v. In the latter, after shrink operation, the *id* of T is set to k in order to continue with the ordered sequence of bundling.

The tree operations are guided by the *dendogram* $\mathbf{Z}$; in which some of the edges of the T are compounded and some intermediate nodes are inserted due to the *expand* operation. Note that tree operations are performed recursively.

Updating Intermediate Points

Once the bundle algorithm has performed the tree operations, the straighten operator is performed. This operator has the role of adjusting the intermediate nodes in order to aim for global minimal metric. Thus, the root of the intermediate nodes are found by minimizing the following cost function:

$$H(T,u,x) = \ell(x,r) + \sum_{w \in leaves(u)} \ell(x,w) \tag{7}$$

, where $u \in leaves(T)$. The above function has the role of finding a minimal q-star (a cherry having root at u and whose leaves are r and $leaves(u)$). The q-star has the role of linking the root r of T and the leaves of u, for which the following is fulfilled $q = 1 + |leaves(u)|$. Since the above cost function is non-convex, a non-linear optimization based on the interior-point algorithm [20] is used. To realize an efficient computation of obstacle-free sampling points, the representation proposed in Eq. 3 is also used.

In summary, our proposed approach consists of two phases. In the first phase, the bundling algorithm *evolves* the initial input graph by using not only the dendogram from the hierarchical clustering of the shortest paths from roots to terminals, but also the tree operations which are guided by the dendogram and closess/farness metrics. In the second

phase, the straightening algorithm updates the locations of the intermediate nodes in order to minimize a global tree metric.

In the next section, we describe our computational experiments and discuss our obtained results.

III. Computational Experiments

In order to evaluate the performance of our proposed bundling algorithm, we performed exhaustive computational experiments using diverse topologies of polygonal networks and polygonal maps. This section describes our experimental conditions, results and insights.

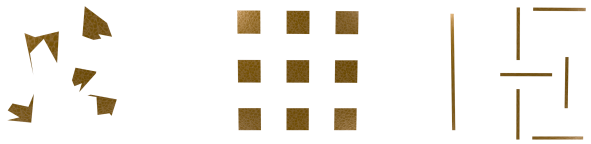
A. Settings

Our computing environment was an Intel i7-4930K @ 3.4GHz, and experiments were performed using Matlab 2016a. In order to enable a meaningful evaluation of our proposed approach, we consider the following environmental settings:

- No. of polygonal maps: 3 types as shown by Fig. 3, where each $P \subset \mathbb{R}^{[0,50]} \times \mathbb{R}^{[0,50]}$.
- No. of edges in the input graph, $m = |E| = \{5, 10, 20\}$,
- For each combination of the above, 30 independent runs were performed,
- In each independent run, the source r and the terminal set S is initialized randomly and independently.
- In all experiments, the following parameters in Algorithm 1 were used: $SP = 10$, $\delta_1 = \delta_2 = 3$, as well as *complete-linkage* agglomerative clustering, and minimization of function J with convergence rate 10^{-6} or 3,000 function evaluations limit were used.

The main reason of using the polygonal maps depicted in Fig. 3 and the number of edges $|E|$ up to 20 is due to our interest in evaluating the performance close to the number of transport/communication needs in robotic indoor environments, where the complexity of the environment is controlled not only by the convexity of the obstacle geometry, but also by the disposition and the number of obstacles in the polygonal map. Complex polygonal environments induce in large number of triangles (since our representation shown in Eq. 3 is based on triangulation of the free-space), thus representing a challenging search space for any search algorithm. Our future work aims at using configurations considering large scenarios and being close to outdoor environments.

Also, we use the values $SP = 10$ in order to allow a reasonably fast computation of distance metrics between roots (Eq. 1) in the domain $\mathbb{R}^{[0,50]} \times \mathbb{R}^{[0,50]}$. Larger values of SP would are recommended for large scale graphs or polygonal environments. Also, $\delta_1 = \delta_2 = 3$ are used in order to allow the growth of intermediate nodes within a reasonable distance metric (these values represent 6% of the total dimension of the map). It is possible to generate diverse minimal trees using other values of δ_1 and δ_2. Furthermore, the use of the *complete-linkage* hierarchical algorithm is based on various preliminary tests confirming its effectiveness. And the main rationale of using 10^{-6} or 3000 function evaluations is to evaluate the efficiency of our algorithm under a restrictive computational budget. Finally, the use of 30 independent runs enables to evaluate our proposed approach under different and arbitrary initialization conditions, avoiding any bias in random luckiness during the interior-point optimization algorithm. As a result, 270 experimental conditions were evaluated.



(a) Map 1 (b) Map 2 (c) Map 3

Fig. 3. Basic Concept

B. Results and Discussion

In order to show the kind of tree structures obtained in the bundling process, as well as the convergence characteristics of the optimization algorithm, Fig. 4 - Fig. 7 show the obtained bundles of an arbitrary subset of results from the total number of 270 experiments; and Fig. 8 shows the complexity behaviour of our algorithm. The reader may note that our results are arranged in a way that the x-axis shows the input graph as well as the bundled tree, and the y-axis shows the number of edges $|E| = \{5, 10, 20\}$ in the input graph.

In regards to the obtained route bundles, Fig. 4 - Fig. 7 present obstacles as dark-brown polygons, the edges of the input graphs and of the trees are presented as straight lines, and nodes are presented by colored spheres. Here, the root node of the input graph is depicted by a red color, whereas the terminal nodes are depicted by blue spheres. The intermediate nodes are shown by black-colored and smaller spheres. In all experimental cases, and exemplified by Fig. 4 - Fig. 7, we confirmed the following facts:

- Regardless of the configuration of the polygonal map and the structure of the input graph, it is possible to generate obstacle-avoiding tree structures representing the bundled routes from source r to the terminal set S which aim at minimizing the global distance metric while preserving connectivity.
- Intermediate angles having degree 3 (which means 3 leaves), have edges separating each other at 120 degrees, which is in line with the property of optimal Euclidean Steiner trees with no obstacles. Note that this property has not been explicitly considered in our approach, but rather it is a consequence of the convergence of our algorithm.
- The topology of the bundled tree has a terminal set which differs from the terminal set of the input graphs. This is due to the fact that a number of nodes of the input graph become intermediate in the bundled tree due to *merging*, *expansion* and *shrinking* operations. How to predict the number of terminal nodes in the bundled tree is out of the scope of our paper.
- Our approach is able to preserve the connectivity from root r towards the terminal set in S, since our bundling

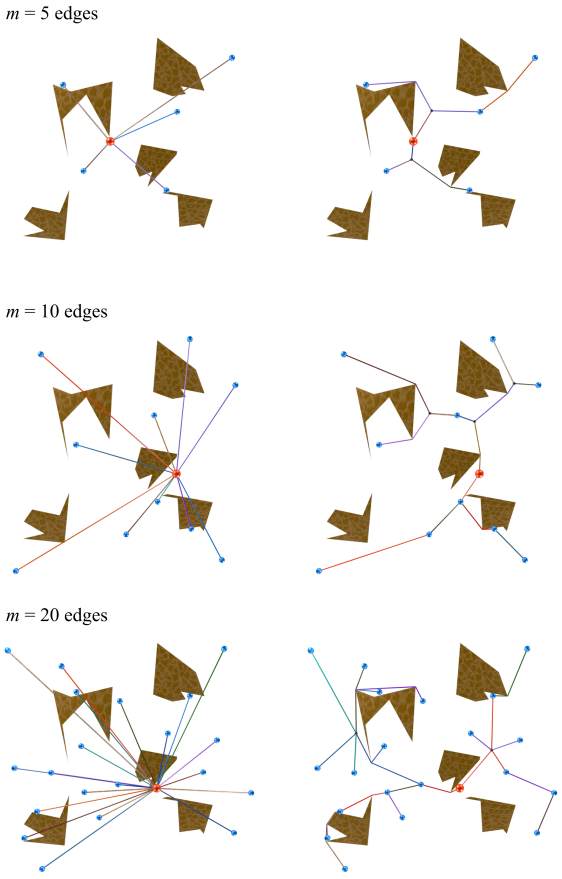


Fig. 4. Performance in Map 1.

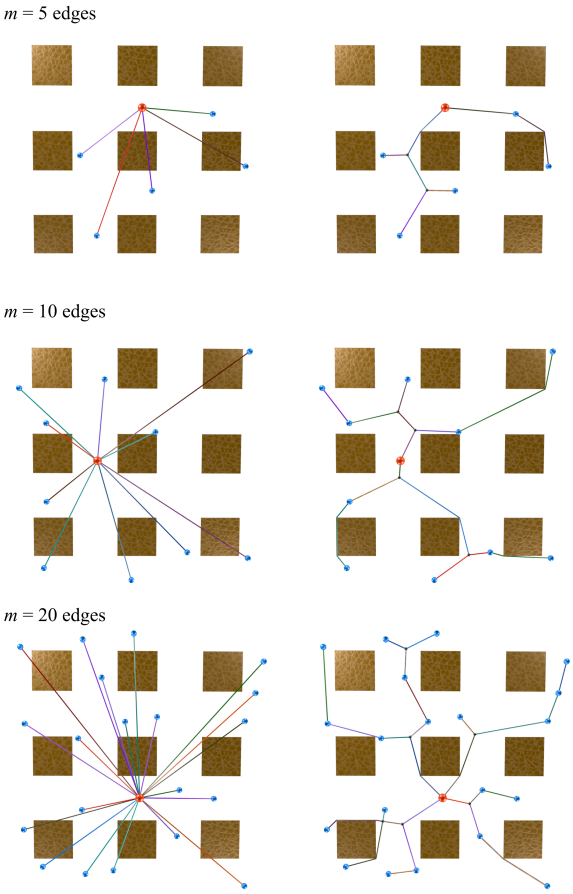


Fig. 5. Performance in Map 2.

algorithm is based on compounding shortest routes from roots to terminal leaves in a recursive manner. An example of this fact is shown by Fig. 7, where the source r is in a non-central location of the map. This figure exemplifies the different approach when compared to the existing literatures on obstacle-avoiding Steiner trees [15]–[19], [22], in which all nodes are considered homogeneous and preserving the connectivity from root r towards the terminal set in S is unfeasible.

The above observations have important implications to extend our proposed method in the following ways: (1) instead of sampling the search over the free-space using arbitrary initial conditions, it may be possible to compute the 3-star graphs connecting sources and their leaves by using the Steiner property of triangles, or by computing the fermat Point, and (2) it may be possible to use pre-computed routes between the anchoring points as initial solutions whenever the polygonal map is expected to changes, since the routes are expected to be structurally similar for small changes in the polygonal map. The above are foundational insights to enable even faster convergence to the optimal solutions in bundling edges in large and dynamic environments.

Furthermore, in regards to the complexity behaviour of the propose bundling algorithm, Fig. 8 shows the time to compute the bundled trees over 30 independent runs. Here, x-axis shows the number of edges, while y-axis shows the time to compute the bundled tree. The blue line indicates the average over 30 independent cases. By observing Fig. 8, we confirm the following facts:

- Regardless of the configuration of the polygonal maps and the structure of input graphs, the average complexity function shows a linear tendency. However, the reader may note a high variance among all cases (lines with alpha-transparency). The high variance occurs due to having independent runs under arbitrary initialization conditions: the source and the terminal are initialized so that path planning becomes harder when needed to go through many obstacles, thus time is expected to change for harsh conditions. Investigating the complexity as a function of the number sides of each polygonal map is left for future work, as well as the evaluation of the time complexity in outdoor environments.
- In all cases, increasing the number of edges in the input graph has a natural direct effect on increasing the time by some small constant above the linear factor. This observation is in line with our above insights regarding

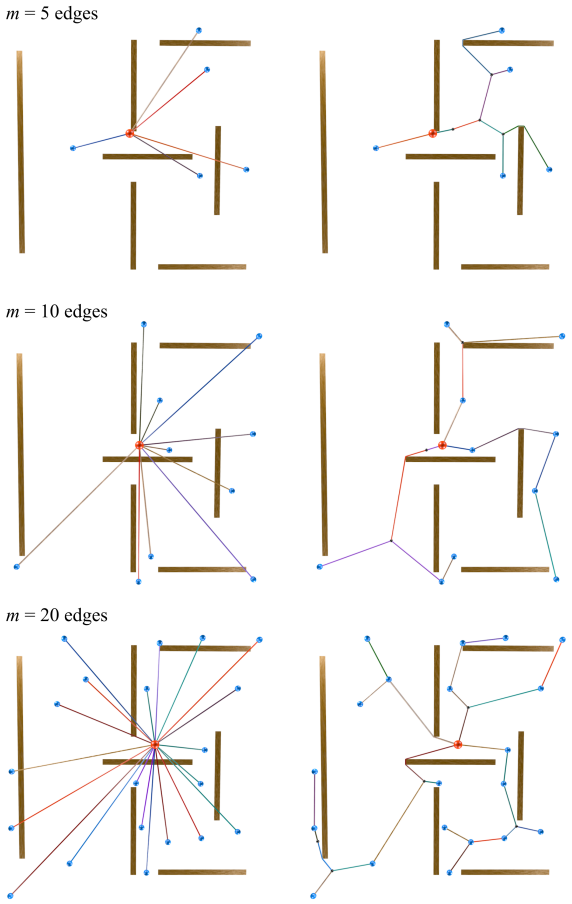


Fig. 6. Performance in Map 3.

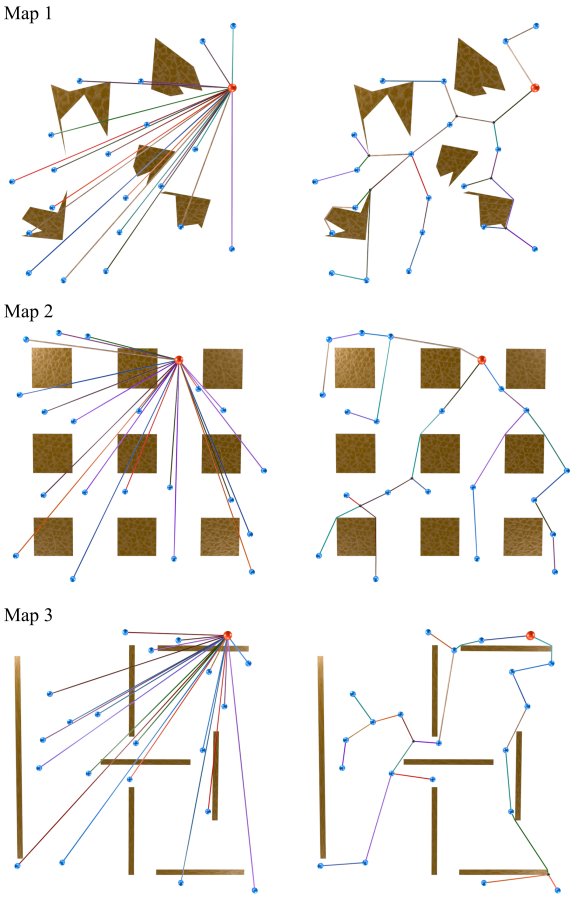
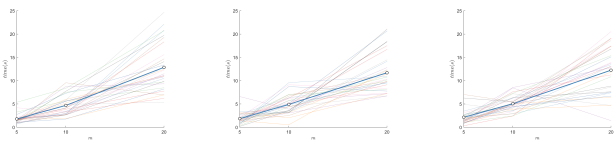


Fig. 7. Performance in non-centralized cases.

the linear tendency.

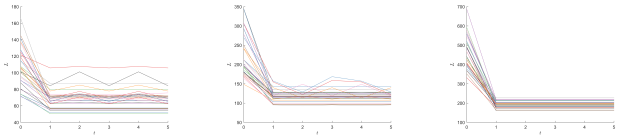
- In order to investigate the optimality of the converged solutions, we also evaluated Algorithm 1 during 5 iterations (executed 5 times consecutively). Fig. 9-10 shows the evolution of length vs. number of iterations. Here, in x-axis, we show the number of iterations, and in y-axis we show the total length of the tree. By these figures, we can confirm that the tree length converges in the first iteration, which implies that Algorithm 1 is able to compute minimal trees by using only bundling and straighten operations. The reader may also note that the bundling operator has an explorative effect compared to the corrective role of the straighten operator. This is due to the fact that the bundling function allows for exploration over the search space by allowing to insert intermediary nodes among trees. We can also observe that the above described explorative behaviour is able to be self-enhanced in further iterations.

The above results show the feasibility, efficiency and robustness to obtain minimal trees in polygonal maps with both convex/non-convex obstacles. Computational experiments with large number of edges and obstacles reminiscent of outdoor environments are in our agenda. Also, in our agenda remains the use of directed [23] and undirected graphs [24], and modularity concepts [25] to allow combinatorial groupings of nodes/edges (to further improve the scalability). Furthermore, designing the optimal networking of modular and adaptive systems is a potential venue of research, e.g. to enable networks with synchrony [**?**] in obstacle-constrained environments; to enable adaptive networks with changing structures [**?**]. Last but no least, the extension to generate smooth, curved and collision-free navigation bundles in cluttered environments is left for future work. Our results in these paper are building blocks to further advancement towards developing global network optimization with convex, flexible and scalable representations. We believe the minimal length tree approach may find use in several applications.



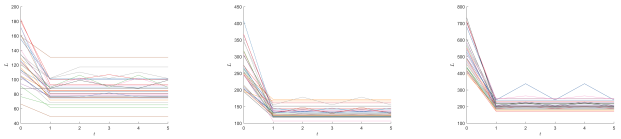
(a) Map 1 (b) Map 2 (c) Map 3

Fig. 8. Time Complexity



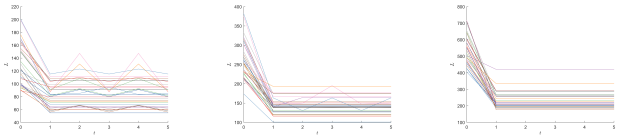
(a) m = 5 edges (b) m = 10 edges (c) m = 20 edges

Fig. 9. Length vs. Iterations in Map 1



(a) m = 5 edges (b) m = 10 edges (c) m = 20 edges

Fig. 10. Length vs. Iterations in Map 2



(a) m = 5 (b) m = 10 (c) m = 20

Fig. 11. Length vs. Iterations in Map 3

IV. Conclusion

In this paper, we have proposed a method for searching optimal tree bundles based on nature-inspired tree operations. The basic idea of our approach is to let trees evolve while sampling the free-space over a triangulated and convex search space. Then, it becomes possible the rendering of feasible and minimal tree bundles efficiently since: (1) absence of overlaps with obstacles is guaranteed, and (2) computation of point inside polygon is explicitly avoided. Computational experiments using a diverse class of polygonal maps and number of edges in the input graphs show that (1) it is possible to obtain bundled trees with an optimized global distance metric, and (2) the time complexity for convergence is possible over independent runs and has a linear tendency, in average. In our future work, we aim at using polygonal environments reminiscent of outdoor configurations, as well as exploring the generalization ability in dynamic environments, where both the input graph and the polygonal obstacles are allowed to change. Potential applications of our include: (1) design of transportation networks, (2) design of resource distribution systems (e.g. water, oil, gas, energy, information), (3) design of integration of mechanical and electrical systems in buildings, (4) design of the optimal wiring in natural and artificial systems, and (5) design and identification of cellular pathways. We believe our approach opens new frontiers to further develop compounded and global compounded path planning algorithms via nature-inspired graph operations and canonical representations of the search space.

References

[1] P. Erdös and Rényi, *Asymmetric Graphs*. Acta Math. Acad. Sci. Hungar. Vol. 14, 295-315, 1963.

[2] F. Harary, *Graph Theory*. Reading, MA: Addison-Wesley, 1994.

[3] D. Holten and J. J. van Wijk, *Force-Directed Edge Bundling for Graph Visualization*. Eurographics, Symp. on Visualization, 2009.

[4] D. Selassie, B. Heller, and J. Heer, *Divided Edge Bundling for Directional Network Data*. IEEE Transactions on Visualization and Computer Graphics, Vol. 17, No. 12, pp. 2354 - 2363, 2011.

[5] W. Cui, H. Zhou, P. C. W. H. Qu, and X. Li, *Geometry-based edge clustering for graph visualization*. IEEE Transactions on Visualization and Computer Graphics, Vol. 14, pp. 1277 - 1284, 2008.

[6] C. Hurter, O. Ersoy, and A. Telea, "Graph bundling by kernel density estimation," in *Eurographics*, 2011, pp. pp. 865–874.

[7] E. R. Gansner, S. N. Y. Hu, and C. Scheidegger, *Multilevel agglomerative edge bundling for visualizing large graphs*. IEEE Pacific Visualization Symposium, pp. 187 - 194, 2011.

[8] D. Holten, *Hirerarchical Edge Bundles: Visualization of Adjacency Relations in Hierarchical Data*. IEEE Pacific Visualization Symposium, pp. 187 - 194, 2006.

[9] V. Parque, M. Kobayashi, and M. Higashi, *Optimisation of Bundled Routes*. 16th Int. Conf. on Geometry and Graphics, pp. 893-902, 2014.

[10] V. Parque, S. Miura, and T. Miyashita, *Optimization of Route Bundling via Differential Evolution with a Convex Representation*. IEEE Int. Conf. on Real-time Computing and Robotics, Okinawa, Japan, 2017.

[11] ——, *Computing Path Bundles in Bipartite Networks*. 7th Int. Conf. on Simulation and Modelling Methodologies, Technologies and Applications, pp. 422-427, Madrid, Spain, 2017.

[12] J. Vojtěch and M. Kössler, *O minimálních grafech, obsahujících n daných bodů*. Časopis pro pěstování matematiky a fysiky 063.8, pp. 223-235, http://eudml.org/doc/25703, 1934.

[13] H. Robbins and R. Courant, *What is Mathematics?* Oxford University Press, 1941.

[14] H. Zhang, D. Ye, and W. Guo, *A heuristic for constructing a rectilinear Steiner tree by reusing routing resources over obstacles*. INTEGRATION, the VLSI journal, Vol. 55, pp. 162-175, 2016.

[15] P. Winter, M. Zachariasen, and J. Nielsen, *Short trees in Polygons*. Discrete Applied Mathematics, Vol. 118, pp. 55-72, 2002.

[16] T. T. Jing, Y. Hu, Z. Feng, X. Hong, X. Hu, and G. Yan, *A full-scale solution to the rectilinear obstacle-avoiding Steiner problem*. INTEGRATION, the VLSI journal, Vol. 41, pp. 413-425, 2008.

[17] P. Winter, *Euclidean Steiner minimal trees with obstacles and Steiner visibility graphs*. Discrete Applied Mathematics, Vol. 47, pp. 187-206, 1993.

[18] W. Chow, L. Li, E. Young, and C. Sham, *Obstacle-avoiding rectilinear Steiner tree construction in sequential and parallel approach*. INTEGRATION, the VLSI journal, Vol. 47, pp. 105-114, 2014.

[19] M. Müller-Hannemann, , and S. Tazari, *A near linear time approximation scheme for Steiner tree among obstacles in the plane*. Computational Geometry: Theory and Applications, Vol. 43, pp. 395-409, 2010.

[20] R. Byrd, , J. Gilbert, and J. Nocedal., *A Trust Region Method Based on Interior Point Techniques for Nonlinear Programming*. Mathematical Programming, Vol 89, No. 1, pp. 149185, 2000.

[21] R. Osada, T. Funkhouser, B. Chazelle, and D. Dobki, *Shape Distributions*. Euro-graphics, ACM Trans. on Graphics, Vol. 21, No. 4, pp. 807-832, 2002.

[22] M. Zachariasen and P. Winter, *Obstacle-avoiding Euclidean Steiner trees in the plane: an exact algorithm*. Algorithm Engineering and Experimentation, Vol. 1619, pp. 282-295, 1999.

[23] V. Parque and T. Miyashita, *On Succinct Representation of Directed Graphs*. IEEE Int. Conf. on Big Data and Smart Computing, pp. 199-205, 2017.

[24] V. Parque, M. Kobayashi, and M. Higashi, *Bijections for the numeric representation of labeled graphs*. IEEE Int. Conf. on Systems, Man and Cybernetics, pp. 447-452, 2014.

[25] ——, *Searching for Machine Modularity using Explorit*. IEEE Int. Conf. on Systems, Man and Cybernetics, pp. 2599-2604, 2014.