A Simple and Efficient Strategy for Solving Very Large-Scale Generalized Cable-Trench Problems

Francis J. Vasko and Eric Landquist
Department of Mathematics, Kutztown University, Kutztown, Pennsylvania 19530

Gregory Kresge and Adam Tal
Department of Computer Science, Kutztown University, Kutztown, Pennsylvania 19530

Yifeng Jiang and Xenophon Papademetris
Department of Diagnostic Radiology, Yale University School of Medicine, 310 Cedar Street, P.O. Box 208042, New Haven, Connecticut 06520-8042

Vasko et al., Comput Oper Res 29 (2002), 441–458 defined the cable-trench problem (CTP) as a combination of the Shortest Path and Minimum Spanning Tree Problems. Specifically, let $G = (V, E)$ be a connected weighted graph with specified vertex $v_1 \in V$ (referred to as the *root*), length $l(e) \geq 0$ for each $e \in E$, and positive parameters τ and γ. The CTP is the problem of finding a spanning tree T of G such that $\tau l_\tau(T) + \gamma l_\gamma(T)$ is minimized, where $l_\tau(T)$ is the total length of the spanning tree T and $l_\gamma(T)$ is the total path length in T from v_1 to all other vertices of V. Recently, Jiang et al., Proceedings of MICCAI 6893 (2011), 528–536 modeled the vascular network connectivity problem in medical image analysis as an extraordinarily large-scale application of the generalized cable-trench problem (GCTP). They proposed an efficient solution based on a modification of Prim's algorithm (MOD_PRIM), but did not elaborate on it. In this article, we formally define the GCTP, describe MOD_PRIM in detail, and describe two linearly parallelizable metaheuristics which significantly improve the performance of MOD_PRIM. These metaheuristics are capable of finding near-optimal solutions of very large GCTPs in quadratic time in $|V|$. We also give empirical results for graphs with up to 25,001 vertices.

Keywords: cable-trench problem; generalized cable-trench problem; vascular image analysis; mathematical programming formulation; shortest path tree; minimum spanning tree; bicriteria optimization

1. INTRODUCTION

Let $V = \{v_1, \ldots, v_n\}$ be a set of vertices and let $E \subseteq \{(v_i, v_j) | 1 \leq i, j \leq n, i \neq j\}$ be a set of edges. As defined in [11], for a connected weighted graph $G = (V, E)$ with specified root vertex $v_1 \in V$, let $l(e) \geq 0$ be the length of the edge $e \in E$, and let τ and γ denote positive weighting parameters. A solution to the cable-trench problem (CTP) is any spanning tree T of G that minimizes $\tau l_\tau(T) + \gamma l_\gamma(T)$, where $l_\tau(T)$ is the total length of T and $l_\gamma(T)$ is the total path length in T from v_1 to all other $v_k \in V$.

If $\gamma > 0$ and $\tau = 0$, then a solution to the CTP is any shortest path tree (SPT) of G with root vertex v_1. In contrast, if $\tau > 0$ and $\gamma = 0$, then a solution to the CTP is any minimum spanning tree of G. Thus, solutions to these two limiting cases can be found efficiently, that is, in polynomial time, using Dijkstra's algorithm and Prim's algorithm, respectively.

The name "Cable-Trench" comes from the fact that a physical application of this problem is the problem of minimizing the cost to create a campus network in which each building on a campus is connected to a central server with its own dedicated cable. The central server is represented by v_1 and the remaining buildings are represented by the other vertices of G. In the standard CTP, each edge of G corresponds to an allowable route to dig a trench and lay cables between two buildings. As each building must be connected by cable to the central server and no auxiliary edges are allowed in the solution spanning tree, the solution to the standard CTP will not be a Steiner tree. A trench may carry more than one cable once it is dug. The cost of digging the trench is τ per unit length and the cost of the cable required is γ per unit length.

In addition to this classical application, the CTP has been used to model other situations, all of which are rooted in the concepts of laying cables and digging trenches. Nielsen

Received May 2014; accepted March 2015
Correspondence to: F. J. Vasko; e-mail: vasko@kutztown.edu
DOI 10.1002/net.21614
Published online in Wiley Online Library (wileyonlinelibrary.com).

et al. [9] applied the CTP to significantly reduce the cost of upgrading and deploying wired and wireless access networks. Marianov et al. [7] generalized the CTP to forests (disjoint unions of trees) to optimize the construction of logging roads and sawmills for a logging operation and to optimize the construction of canals and wells for irrigation, for example. The CTP formulation has also been used by Girard et al. [3] for an array of radio telescopes.

In this article, we define the Generalized CTP (GCTP) and develop efficient metaheuristics to find nearly optimal solutions. Although other generalizations of the CTP are possible [7], in our formulation of the GCTP, each edge of the graph is assigned two lengths: a cable length and a trench length. (If both lengths of each edge are the same, then the GCTP reduces to the CTP.) The obvious motivation for introducing the GCTP is the situation in which the cost to dig a trench is not always proportional to the cost to lay a cable because of obstacles, soil composition, or overhead costs, for example. Our motivation for defining the GCTP, however, is the nontrivial problem of vasculature reconstruction from images generated by computed tomography (CT) and micro-CT, as described by Jiang et al. [5]. Here, we will briefly show how the vascular imaging problem can be described as a graph-theoretic problem. A CT system takes a large number of cross-sectional images of a subject. When imaging a vascular network, we identify each blood vessel that passes through each cross-section, store the (Cartesian) coordinates of the center of each vessel slice, and represent this point as a vertex of a graph. The radius of each vessel is also determined and stored. In this way, we generate an extraordinarily large set of discrete points V, slice by slice. The objective, then, is to find the edge set E, which represents the paths of the blood vessels, thereby digitally representing a network of blood vessels (vasculature) as a graph $G = (V, E)$. Such a graph assists critical vasculature-related research, including angiogenesis and cancer detection. Section 3 describes how this problem is then modeled as a GCTP.

Vasko et al. [11] proved that the CTP is nondeterministic polynomial (NP)-complete. Hence, the GCTP is also NP-complete. Thus, finding optimal solutions to the CTP or GCTP is expected to require exponential time in $n = |V|$ in the worst case. This prohibits one from finding optimal solutions for large examples, such as those encountered in vascular image analysis. Vasko et al. [11] developed a metaheuristic, which is conjectured to run in polynomial time in n, which finds good solutions to the CTP for all $\rho = \tau/\gamma \in [0, \infty)$. Vasko et al. only solved CTPs with up to 20 vertices. The metaheuristic finds a set of spanning trees, each of which finds good solutions to the CTP over a closed interval of values of ρ. For large CTPs in which τ and γ are known, this metaheuristic is both inefficient and unnecessary, hence the need for the more efficient metaheuristics presented in sections 4 and 5.

In the next section, we will review the mathematical programming formulation given in [11] for the CTP. We will then define the GCTP and extend the CTP mathematical formulation to the GCTP. This will be followed by a brief background on vascular image analysis. Next, we will discuss different metaheuristic methods and compare empirical results for solving large GCTPs (with large CTPs as a special case). The article will close with our conclusions and areas of future work.

2. MATHEMATICAL PROGRAMMING FORMULATIONS FOR THE CTP AND THE GCTP

Before discussing the GCTP, we will review the mathematical programming formulation for the CTP first given in [11].

2.1. A Mathematical Formulation of the CTP (MFCTP)

The following is a zero-one mixed integer linear programming formulation of the CTP.

$$\text{minimize } Z = \gamma \left[\sum_{j=1}^{n} \sum_{i=1}^{n} d_{ij} x_{ij} \right] + \tau \left[\sum_{j=1}^{n} \sum_{i=1}^{n} d_{ij} y_{ij} \right] \tag{1}$$

subject to

$$\sum_{j=2}^{n} x_{1j} = n - 1 \qquad i = 1 \text{ (vertex1)} \tag{2}$$

$$\sum_{j=1}^{n} x_{ij} - \sum_{k=1}^{n} x_{ki} = -1 \qquad \text{for all } 2 \le i \le n \tag{3}$$

$$\sum_{j=2}^{n} \sum_{i=1}^{j-1} y_{ij} = n - 1 \tag{4}$$

$$(n-1) y_{ij} - x_{ij} - x_{ji} \ge 0 \qquad \text{for all } i < j \text{ (for all edges)} \tag{5}$$

$$x_{ij} \ge 0 \qquad \text{for all } i, j \tag{6}$$

$$y_{ij} \in \{0, 1\} \qquad \text{for all } i < j, \tag{7}$$

where x_{ij} is the number of cables from $v_i \in V$ to $v_j \in V$ (no cables flow back to v_1, which is the central server); $y_{ij} = 1$ if a trench is dug between v_i and v_j ($i < j$), and 0 otherwise; and $d_{ij} = l((v_i, v_j))$. The cable cost is γ per unit length and the trench cost is τ per unit length.

Constraint (2) ensures that $n - 1$ cables leave the central server. Constraint set (3) ensures that each of the $n - 1$ buildings is connected by exactly one cable. Constraint (4) ensures that exactly $n - 1$ trenches are dug. Although constraint (4) is not required to ensure that the optimal solution is found, Vasko et al. [11] illustrated that constraint (4) considerably strengthens this formulation by reducing the solution space that needs to be searched. Constraint set (5) ensures that cables are not laid unless a trench is dug. Although the number of cables [constraint set (6)] is only constrained to be nonnegative, these variables will, in fact, be integers because of their relationship to the trench variables which must be either 0 or 1 [constraint set (7)].

2.2. Generalizing the Cable-Trench Problem

In the GCTP, each edge $(v_i, v_j) \in E$ of the graph $G = (V, E)$, has two lengths: s_{ij}, the cable length of (v_i, v_j), and t_{ij}, the trench length of (v_i, v_j). Then, the cable cost from v_i to v_j is γs_{ij} and the trench cost from v_i to v_j is τt_{ij}. This formulation allows the cable costs to be independent of the trench costs. One physical application is a situation in which, even for the same length, the cost of digging a trench is greater for some edges because of soil composition or physical obstacles. Clearly the GCTP is a generalization of the CTP because setting $d_{ij} = s_{ij} = t_{ij}$ reduces the GCTP to the CTP.

A mathematical formulation for the GCTP is identical to MFCTP except for the objective function. Specifically, the objective function for MFGCTP is

$$\text{minimize } Z = \gamma \left[\sum_{j=1}^{n} \sum_{i=1}^{n} s_{ij} x_{ij} \right] + \tau \left[\sum_{j=1}^{n} \sum_{i=1}^{n} t_{ij} y_{ij} \right]. \tag{8}$$

In the next section, we will give a brief background to our motivation of the GCTP, the vascular network reconstruction problem.

3. BACKGROUND ON VASCULAR IMAGE ANALYSIS

As noted in the introduction, discrete locations of blood vessels, along with the vessel radii at these points, can be detected from three-dimensional medical images, such as CT. These points correspond to the vertices V of a complete graph $G = (V, E)$, with edge lengths determined by Euclidean length, vessel volume, or other physiological factors. The problem is to determine the spanning tree of G which represents the actual network of blood vessels. This task currently constitutes a bottleneck in quantitative vascular research [12]. Previously, the best methods computed the minimum spanning tree of G, but their methods required manual correction [2, 6, 10].

Recently, some methods have been proposed to utilize further physiological constraints [1, 4, 5]. In particular, Jiang et al. [5] applied Murray's Minimum Work Principle [8], which states that any vasculature will minimize the total work due to blood flow resistance and metabolic support for the blood volume. We refer the reader to [5] for the technical details of their derivation, but offer an intuitive understanding of the concepts here. Consider a segment of a blood vessel, idealized as a cylinder. In terms of the first factor, narrower vessels require more work to pump a given volume of blood through to overcome resistance. This is consistent with the fact that the work to overcome resistance in the given segment is inversely proportional to the radius of the vessel. We also understand that the work to overcome resistance by the vessel is proportional to its length. As the heart is pumping blood to the tips of capillaries, and needing to overcome resistance along each path, as a graph, we consider the blood volume as a set of "cables" from the root (heart or main artery) to each leaf vertex (capillary tip). Therefore, as a graph, the cable length for an edge $e = (v_i, v_j)$ is $s_{ij} = l(e)/r(e)$, where $l(e)$ is the (Euclidean) length of the (vessel) edge e and $r(e)$ is the radius of the vessel (at v_j). The second factor, work required for metabolic support, is proportional to the volume of blood. In terms of the idealized vessel segment, this factor is proportional to the volume of the segment. Thus, the blood vessels themselves are the "trenches" in our model so that the trench length of (vessel) edge e is $t_{ij} = l(e)r(e)^2$. (We absorb the constant π, along with other proportionality constants, into the trench cost τ, in practice.) This represents the application of Murray's Minimum Work Principle to the vascular imaging problem as a GCTP.

To solve the resulting large GCTP, Jiang et al. [5] proposed using a modified version of a combination of Prim's algorithm and Dijkstra's algorithm. Their initial results demonstrated statistically significant improvements over those of strictly minimum spanning tree (MST)-based methods. However, they do not give the details or running time analysis of their algorithm. In the next section, we will formally introduce this algorithm and present its worst-case running time.

4. THE MODIFIED PRIM ALGORITHM

The metaheuristic proposed in [5], which we call the Modified Prim algorithm MOD_PRIM, adds vertices and edges into the current tree in a greedy manner, like Prim's algorithm. However, the "cost" of a vertex $v_i \in V$ not in, but adjacent to, a vertex, v_j, in the current tree is the minimal weighted sum of the cost to "dig" the trench from v_j to v_i and the cost to "lay" the cable from v_1 to v_i via v_j. This cost is minimal in the sense that v_i may be adjacent to more than one vertex in the current tree. More precisely, we have

$$\text{cost}(v_i) = \min_{\mathcal{P} \in \mathcal{P}(v_1, v_i)} \left(\gamma \left(\sum_{(v_a, v_b) \in \mathcal{P}} s_{ab} \right) + \tau t_{ji} \right),$$

where $\mathcal{P}(v_1, v_i)$ is the set of all paths $\mathcal{P}$ of edges in the current tree from the root v_1 to the vertex v_i in which only the last edge, (v_j, v_i), is not in the current tree. A formal description of MOD_PRIM is introduced below, although we will first define a pair of subroutines to streamline the pseudocode for it and two other metaheuristics that we will describe in section 6. (Subroutine 2 is the key component of MOD_PRIM and its variants.) As can be seen from the pseudocode, MOD_PRIM reduces to Prim's algorithm for finding a minimum spanning tree when $\gamma = 0$. When $\tau = 0$, MOD_PRIM reduces to Dijkstra's shortest path algorithm.

Subroutine 1: Initialize: Initialize variables.
Input: $G = (V, E)$, a doubly-weighted connected graph with vertex set $V = \{v_1, \ldots, v_n\}$, edge set E, cable and trench edge lengths s_{ij} and t_{ij}, respectively, for edge (v_i, v_j); per-unit cable cost γ; and per-unit trench cost τ;

Output: Initialized values of W, a vertex set; F, an edge set; d_{root}, an array of distances from each vertex to v_1; *cost*, an array giving the costs to insert each vertex into the current tree; and *Pre*, an array giving the index of the vertex preceding each vertex on the path from v_1.

```
Initialize (W, F, G, τ, γ, d_root, cost, Pre):
  W := {v_1}                  // The vertex set only
                              // contains the root.
  F := {}                     // The edge set is
                              // initially empty.
  d_root := {0, ∞, ..., ∞}    // Distances from each
                              // vertex to the root.
  cost := {∞, ∞, ..., ∞}      // The added cost to
                              // insert a new vertex.
  Pre := {1, 1, ..., 1}       // The index of the
                              // vertex preceding each
                              // vertex on the path
                              // from v_1.
  For (v_1, v_i) ∈ E          // Initialize the
                              // distance and cost
                              // arrays.
        d_root[i] := s_1i
        cost[i] := τt_1i + γs_1i
  Return
```

Subroutine 2: AppendUpdate: Append a vertex and edge to the current tree and update the arrays d_{root}, *cost*, and *Pre*.
Input: $W, F, G, \tau, \gamma, d_{root}$, *cost*, *Pre* as in Subroutine 1: Initialize; and c, the index of the vertex to append to W.
Output: Updated vertex and edge sets W and F; and updated arrays d_{root}, *cost*, and *Pre*.

```
AppendUpdate (W, F, G, τ, γ, d_root, cost, Pre, c)
    W := W ∪ {v_c}
    F := F ∪ {(v_Pre[c], v_c)}
    For v_i ∈ V \ W  // Revise the distance and
                     // cost arrays.
        If (v_c, v_i) ∈ E
        // If v_c allows for a cheaper path
        // to the root, then
        // update the distance, cost, and
        // predecessor arrays.
            If cost[i] > τt_ci + γ(s_ci + d_root[c])
                d_root[i] := d_root[c] + s_ci
                cost[i] := τt_ci + γd_root[i]
                Pre[i] := c
    Return
```

As we add a new vertex and edge with each iteration, MOD_PRIM will terminate. Moreover, the following is clear from Algorithm 1.

Theorem 1 *Let $n = |V|$, the number of vertices of G. MOD_PRIM requires $O(n^2)$ bits of storage and $O(n^2)$ arithmetic operations in the worst case.*

As the storage requirement of a complete graph is $\Theta(n^2)$, the running time of any heuristic that finds approximate solutions to the GCTP on a complete graph cannot be $o(n^2)$. In this sense, then, MOD_PRIM is within a constant factor of being the fastest possible heuristic for the GCTP on a complete graph.

Algorithm 1 **MOD_PRIM**: Modified Prim's algorithm for the GCTP

```
Input: G, τ, and γ as in Subroutine 1.
Output: T = (W, F), a spanning tree of G.
MOD_PRIM( G, τ, γ ):
    Initialize( W, F, G, τ, γ, d_root, cost, Pre )
    While |W| ≠ |V|
        // Find a vertex with minimal
        // extra additional cost,
        // and add that cheapest vertex
        // and corresponding edge
        // into the current tree.
        c := index( min{cost[i] : v_i ∈ V \ W} )
        AppendUpdate( W, F, G, τ, γ, d_root,
          cost, Pre, c )
    Return T = (W, F)
```

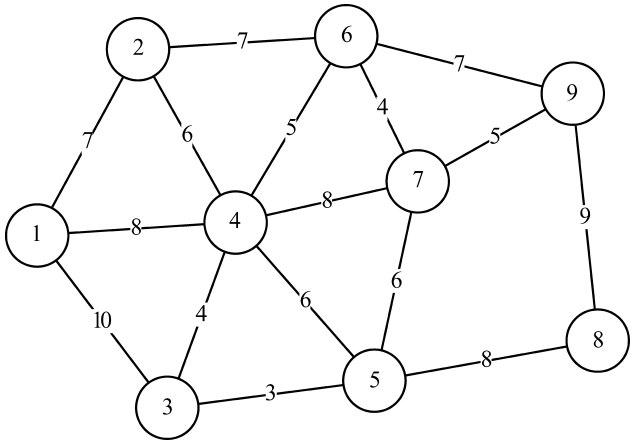

FIG. 1. The nine-vertex graph from Example 4 of [11].

5. EMPIRICAL RESULTS FOR MOD_PRIM

MOD_PRIM was coded in MATLAB® and run on a PC with an Intel I7-3930K 3.2 GHz processor, 16 GB of RAM, running Windows 7 Professional.

Before considering empirical results for large CTPs and GCTPs, we will analyze the solutions generated by MOD_PRIM for the graph given in Figure 1, with root vertex 1.

As shown in [11], the optimal solutions to this CTP are completely characterized by the following five spanning trees, which are illustrated in Figures 2–6. We note that the spanning tree of Figure 6 is the solution tree obtained by running MOD_PRIM on this graph for $\gamma = 1$ and for all $\tau \geq 5$ as well.

In Table 1, we give the optimal and the MOD_PRIM solution values for five CTPs corresponding to $\gamma = 1$ and $\tau = 0.01, 1, 5, 10$, and 100. Recall that for $\gamma = 1$ and small values of τ, the CTP solution should be close to a SPT.

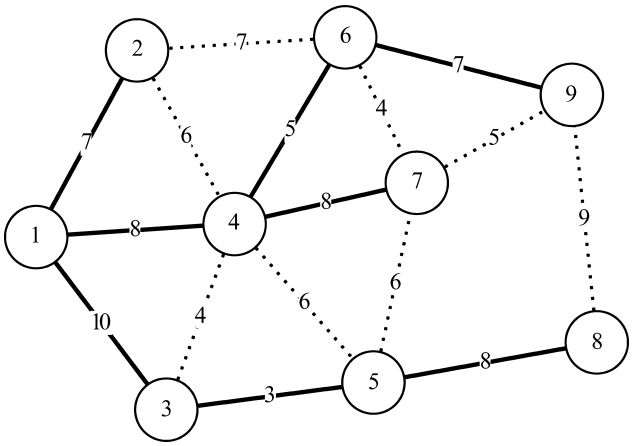

FIG. 2. Optimal Solution for $0 \leq \frac{\tau}{\gamma} \leq \frac{1}{4}$; Total Cost: $56\tau + 108\gamma$ SPT.

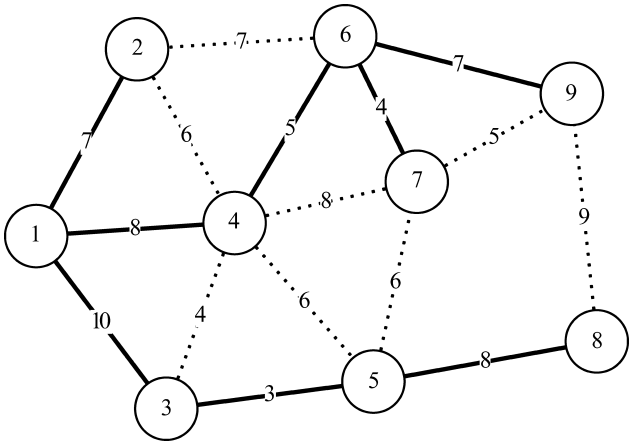

FIG. 3. Optimal Solution for $\frac{1}{4} \leq \frac{\tau}{\gamma} \leq 1$; Total Cost: $52\tau + 109\gamma$.

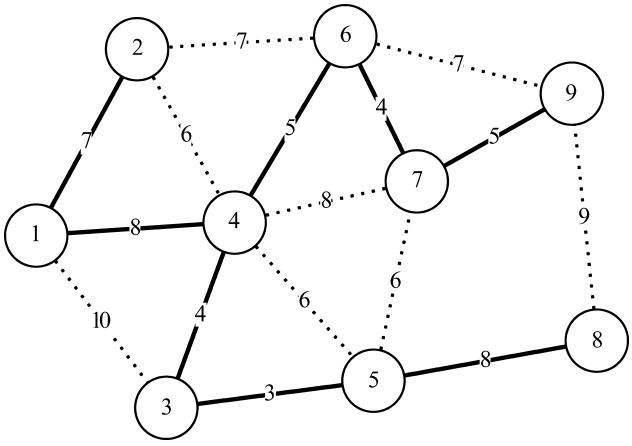

FIG. 4. Optimal Solution for $1 \leq \frac{\tau}{\gamma} \leq 7$; Total Cost: $44\tau + 117\gamma$.

Conversely, for $\gamma = 1$ and large values of τ, the CTP solution should be close to a minimum spanning tree. For these two extremes, as MOD_PRIM is a composite of Dijkstra's SPT algorithm and Prim's minimum spanning tree algorithm, we expect it to perform best when τ/γ is relatively

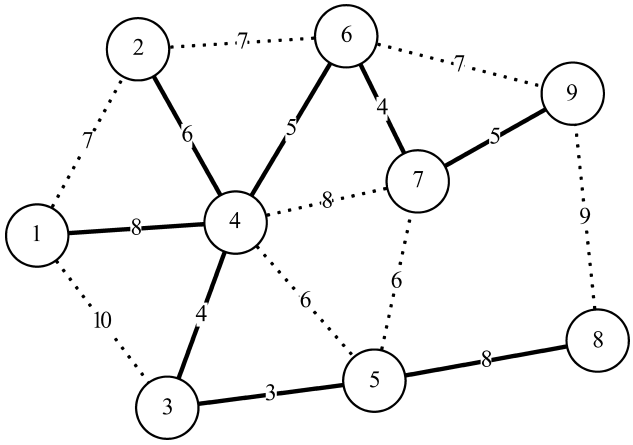

FIG. 5. Optimal Solution for $7 \leq \frac{\tau}{\gamma} \leq 28$; Total Cost: $43\tau + 124\gamma$.

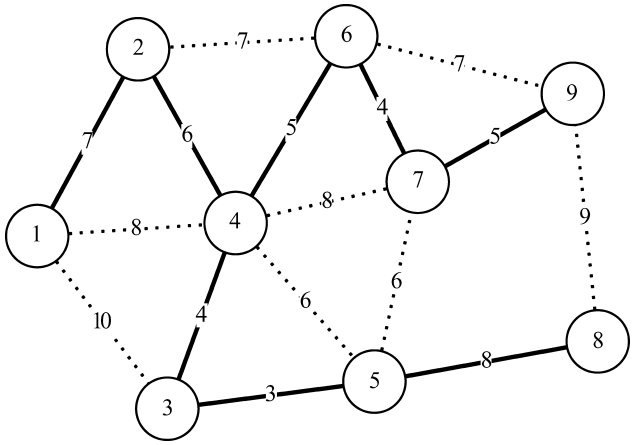

FIG. 6. MOD_PRIM Solution for $\tau = 5, 10$ and Optimal Solution for $28 \leq \frac{\tau}{\gamma}$; Total Cost: $42\tau + 152\gamma$ (MST).

TABLE 1. Optimal and MOD_PRIM results for the nine-vertex graph in Figure 1

τ	Optimum (via MFCTP)	MOD_PRIM	% Deviation
0.01	108.56	108.56	0
1	161	161	0
5	337	362	7.4
10	554	572	3.2
100	4,352	4,352	0

close to 0 or extremely large. This is exactly what happens, with MOD_PRIM finding the optimal solutions when $\tau = 0.01, 1$(close to 0), and 100(relatively large). However, when $\tau = 5$ and 10, the MOD_PRIM solutions deviate 7.4 and 3.2% from the optimal solutions, respectively. Hence, even for this small example, we see that the MOD_PRIM solution can deviate noticeably from the optimal solution.

6. ENHANCEMENTS OF MOD_PRIM: SG_MOD_PRIM, STOC_MOD_PRIM, AND BEST_PRIM

To improve the solutions generated by MOD_PRIM, we made two enhancements to this basic greedy approach—one deterministic and the other partially stochastic. The deterministic enhancement is a multipass, semigreedy modification of MOD_PRIM that we call SG_MOD_PRIM. Specifically, SG_MOD_PRIM constructs solution trees as follows. We compute the combined cable and trench costs for each vertex adjacent to the root and sort these costs. The first GCTP spanning tree generated is the solution tree found by MOD_PRIM. The second spanning tree starts with the second lowest cost vertex adjacent to the root and then chooses the remaining vertices and edges in the strictly greedy manner of MOD_PRIM. In general, the jth lowest cost vertex adjacent to the root starts the jth spanning tree and then the remaining vertices and edges are chosen in the greedy manner. In this way, SG_MOD_PRIM can generate up to $\deg(v_1)$ distinct spanning trees and the solution tree is the one having lowest total cable and trench cost from among these trees. Note that the running time of SG_MOD_PRIM is $O(mn^2)$, where $m \leq \deg(v_1)$ is the number of spanning trees generated. However, this metaheuristic is naturally parallelizable, as only the first edge brought into the spanning tree is done in a semigreedy manner. When parallelized, the speed-up is linear in the number of processors as each processor can begin with a different initial edge. Below is a pseudocode description of SG_MOD_PRIM.

The second metaheuristic approach is a multipass modification of MOD_PRIM, called PSTOC_MOD_PRIM, that generates m spanning trees in a partially stochastic manner, for some choice of m. The tree having the least total cable and trench cost is then chosen as the solution tree. For some choice of s, the first s nonroot vertices of a solution tree are selected stochastically, with the remainder chosen using the greedy approach of MOD_PRIM. Let k be the number of vertices to consider for inclusion into the current tree at any step via a stochastic approach and let $P = [p_1, p_2, \ldots, p_k]$ be a discrete probability distribution. For each of the first s vertices selected for inclusion into a solution tree, the vertex contributing the j th lowest additional cable and trench cost is randomly selected with probability p_j, for $1 \leq j \leq k$. In practice, we let $p_1 \geq p_2 \geq \cdots \geq p_k$ so that the vertex contributing the least total cost at any step is selected with greatest frequency. We determine the j th least costly vertex at each step using a variant of the selection algorithm QUICKSELECT having a worst-case running time of $O(n)$. In this way, the asymptotic running time of PSTOC_MOD_PRIM is $O(m(sn + (n - s)n)) = O(mn^2)$. PSTOC_MOD_PRIM can also be parallelized with linear speed-up in the number of processors.

Finally, for robustness, we combine SG_MOD_PRIM and PSTOC_MOD_PRIM into one metaheuristic which we call BEST_PRIM. BEST_PRIM first executes SG_MOD_PRIM followed by PSTOC_MOD_PRIM. The best overall solution found by SG_MOD_PRIM and PSTOC_MOD_PRIM is the BEST_PRIM solution. In this manner, BEST_PRIM captures the semigreedy characteristic of SG_MOD_PRIM as well as the stochastic characteristic of PSTOC_MOD_PRIM. Note that the asymptotic running time of BEST_PRIM is also $O(mn^2)$ and the process can also be parallelized with linear speed-up in the number of processors.

Algorithm 2 **SG_MOD_PRIM**: Semi-Greedy MOD_PRIM for the GCTP

```
Input: G, τ, and γ as in Subroutine 1; and an integer 0 < m ⩽ deg(v1) for the number of
solution trees to generate.
Output: T = (W, F), a spanning tree of G.
SG_MOD_PRIM( G, γ, τ, m ):
      Initialize( W, F, G, τ, γ , d_root, cost, Pre )
      Store W, F, d_root, cost, and Pre.
      // Scan cost and store the indices of the m cheapest
      // vertices in nondecreasing order, by associated cost,
      // in the array mBest.
      mBest   = Scan( cost )
      T_best = MOD_PRIM( G, τ, γ ) // The first pass is MOD_PRIM.
      For 1 < j ⩽ m // Generate m − 1 more solution trees.
          Retrieve the stored contents of W, F, d_root, cost, and Pre.
          AppendUpdate( W, F, G, τ, γ, d_root, cost, Pre, mBest[j] )
          While |W| ≠ |V|
                c := index( min{cost[i] : v_i ∈ V \ W} )
                AppendUpdate( W, F, G, τ, γ, d_root, cost, Pre, c )
          // If the combined total cable and trench costs of the
          // newly constructed tree is less than that of the
          // previous best tree, then update accordingly.
          If totalCost( (W, F) ) < totalCost( T_best )
                T_best = (W, F)
      Return T_best
```

Algorithm 3 PSTOC_MOD_PRIM: Partially-Stochastic MOD_PRIM

```
Input: G, τ, and γ as in Subroutine 1; m, the number of solution trees to generate;
0 ⩽ s < n, the number of non-root vertices to select stochastically; and P = [p_1, p_2, ..., p_k]
a discrete probability distribution.
Output: T = (W, F), a spanning tree of G.
PSTOC_MOD_PRIM( G, γ, τ, m, s, P ):
     Initialize( W, F, G, τ, γ , d_root, cost, Pre )
     Store W, F, d_root, cost, and Pre.
     Repeat m times // Generate m solution trees.
          Retrieve the stored contents of W, F, d_root, cost, and Pre.
          Repeat s times // Select s vertices stochastically.
               Choose 1 ⩽ j ⩽ k randomly based on P.
               // Find the index of the j^th cheapest vertex,
               // according to associated cost, using a
               // linear-time selection algorithm.
               jthBest = index( Quickselect( cost, j ) )
               Retrieve the stored contents of cost.
               AppendUpdate( W, F, G, τ, γ, d_root, cost, Pre, jthBest )
          While |W| ≠ |V|
               c := index( min{cost[i] : v_i ∈ V \ W} )
               AppendUpdate( W, F, G, τ, γ, d_root, cost, Pre, c )
          // If the combined total cable and trench cost of the
          // newly constructed tree is less than that of the
          // previous best tree, then update accordingly.
          If j == 1 or totalCost( (W, F) ) < totalCost( T_best )
               T_best = (W, F)
Return T_best
```

7. EMPIRICAL RESULTS FOR SG_MOD_PRIM, PSTOC_MOD_PRIM, AND BEST_PRIM

Both SG_MOD_PRIM and PSTOC_MOD_PRIM were coded in MATLAB® and solved on a PC with an Intel I7-3930K 3.2 GHz processor, 16 GB of RAM, running Windows 7 Professional.

In considering the metaheuristic approaches for solving large CTPs or GCTPs that we just described, we used a very large (over 27,000-vertex) vascular connectivity dataset to define several test problems of various sizes. Instead of using a complete graph in each of our examples, the edge set was the set of all edges whose Euclidean length was less than 10% of the diameter of the dataset; this is consistent with attributes of the vascular data in practice. For each graph, one CTP was defined based on Euclidean distances (i.e., $r(e) = 1$ for all $e \in E$) and one GCTP was defined using the lengths s_{ij} and t_{ij} and costs defined in our discussion of vascular image analysis.

Specifically, to test SG_MOD_PRIM, PSTOC_MOD_PRIM, and BEST_PRIM, we generated eight large graphs with 501, 1,001, 2,501, 5,001, 10,001, 15,001, 20,001, and 25,001 vertices. For each graph, CTPs and GCTPs were defined for $\gamma = 1$ and the same five values of τ (0.01, 1, 5, 10, and 100) used in Table 1. Therefore, we defined a total of 40 CTPs and 40 GCTPs. The 40 CTPs can be thought of as "theoretical" problems whereas the 40 GCTPs can be thought of as "application" problems. A total of $m = 30$ spanning trees were generated using SG_MOD_PRIM. For PSTOC_MOD_PRIM, after some empirical experimentation, we used $k = 5$ and the probability distribution [3/9, 2/9, 2/9, 1/9, 1/9], with $s = 0.06(n-1)$ and $s = 0.1(n-1)$, generating $m = 15$ solutions for each choice of s. The ideas behind these choices were that we wanted to choose a relatively small fraction of the vertices stochastically, specifically several of the more critical and "influential" initial edges. We further desired the better edges chosen with greater frequency at each step, hence a decreasing frequency from best candidate edge (with a 1/3 chance of being chosen) to the fifth best candidate edge (with a 1/9 chance of being chosen). For example, for the "smallest" problem ($n = 501$), we chose $s = 0.06(501-1) = 30$ vertices stochastically for the first 15 spanning tree solutions generated and $s = 0.1(501-1) = 50$ vertices stochastically for the last 15 spanning tree solutions generated.

Our results for the three metaheuristics are summarized in Table 2. The table entries give the percent improvement in total cost of each metaheuristic over MOD_PRIM. It is important to remember that the CTP and GCTP are minimization problems and that improvement means that SG_MOD_PRIM, PSTOC_MOD_PRIM, and BEST_PRIM are getting smaller (BETTER) objective function values than MOD_PRIM. Notice that for the 40 "theoretical" CTPs, the percent improvement by BEST_PRIM over MOD_PRIM was 1.44% whereas for the 40 "application" GCTPs, the percent improvement by BEST_PRIM over MOD_PRIM was 3.26%. The average improvement over MOD_PRIM over all 80 problems (40 CTPs and 40 GCTPs) for SG_MOD_PRIM, PSTOC_MOD_PRIM, and BEST_PRIM was 1.66, 2.09, and 2.35%, respectively.

TABLE 2. Percentage improvement over MOD_PRIM for CTPs and GCTPs with up to 25,001 vertices

Vertices-edges	CTP Euclidean distances			GCTP vascular apps		
501 24,675	SG_MOD_PRIM	PSTOC_MOD_PRIM	BEST_PRIM	SG_MOD_PRIM	PSTOC_MOD_PRIM	BEST_PRIM
τ						
0.01	1.3	1.4	1.4	1.5	4.0	4.0
1	1.5	1.3	1.5	1.5	4.0	4.0
5	4.3	3.6	4.3	0.1	4.0	4.0
10	1.0	3.5	3.5	0.3	4.0	4.0
100	0.9	0.6	0.9	1.5	3.7	3.7
AVG	1.8	2.1	2.3	1.0	3.9	3.9
1,001 96,272						
τ						
0.01	0.0	0.2	0.2	4.9	8.7	8.7
1	0.9	0.2	0.9	4.9	8.7	8.7
5	0.8	2.1	2.1	5.5	9.3	9.3
10	1.5	1.8	1.8	5.5	9.5	9.5
100	2.0	0.2	2.0	3.4	7.7	7.7
AVG	1.0	0.9	1.4	4.8	8.8	8.8
2,501 467,698						
τ						
0.01	1.1	1.0	1.1	1.5	2.5	2.5
1	1.1	0.4	1.1	1.4	2.1	2.1
5	0.3	0.6	0.6	1.5	2.3	2.3
10	3.3	3.3	3.3	1.5	2.0	2.0
100	0.4	5.4	5.4	2.4	2.8	2.8
AVG	1.2	2.1	2.3	1.7	2.3	2.3
5,001 1,334,652						
τ						
0.01	3.7	0.9	3.7	1.5	1.3	1.5
1	0.8	0.3	0.8	1.6	1.5	1.6
5	3.3	3.7	3.7	1.7	1.3	1.7
10	1.0	1.7	1.7	1.4	1.3	1.4
100	0.5	0.7	0.7	1.7	2.4	2.4
AVG	1.9	1.5	2.1	1.6	1.6	1.7
10,001 2,639,817						
τ						
0.01	2.0	0.2	2.0	2.8	1.9	2.8
1	0.1	0.2	0.2	3.2	2.3	3.2
5	1.7	1.5	1.7	3.2	2.2	3.2
10	0.9	1.5	1.5	3.1	2.2	3.1
100	0.4	0.6	0.6	2.9	1.8	2.9
AVG	1.0	0.8	1.2	3.0	2.1	3.0
15,001 4,455,521						
τ						
0.01	1.2	0.1	1.2	1.4	1.2	1.4
1	0.1	0.1	0.1	3.0	2.1	3.0
5	1.1	1.0	1.1	3.2	2.6	3.2
10	0.8	1.2	1.2	3.0	2.5	3.0
100	0.4	0.6	0.6	0.2	0.4	0.4
AVG	0.7	0.6	0.8	2.2	1.8	2.2
20,001 7,299,926						
τ						
0.01	0.9	0.1	0.9	1.1	2.1	2.1
1	0.1	0.1	0.1	1.4	1.7	1.7
5	0.8	0.7	0.8	0.6	1.6	1.6
10	0.7	1.1	1.1	5.3	4.0	5.3
100	0.4	0.6	0.6	0.5	1.4	1.4
AVG	0.6	0.5	0.7	1.8	2.2	2.4
25,001 11,164,733						
τ						
0.01	0.7	0.1	0.7	1.1	0.7	1.1
1	0.1	0.1	0.1	1.0	0.5	1.0
5	0.7	0.6	0.7	1.0	0.3	1.0
10	0.6	1.5	1.5	1.4	1.6	1.6
100	0.4	0.6	0.6	3.9	4.5	4.5
AVG	0.5	0.6	0.7	1.7	1.5	1.8
Overall average	1.09	1.14	1.44	2.23	3.03	3.26

TABLE 3. BEST_PRIM percent improvement over MOD_PRIM

τ	0.01	1	5	10	100	Overall
CTPs	1.40	0.60	1.88	1.95	1.43	1.44
GCTPs	3.01	3.16	3.29	3.74	3.23	3.26

From Table 2, it appears that SG_MOD_PRIM gives better results for $\tau \leq 5$, whereas PSTOC_MOD_PRIM gives better results for larger τ. These improvements over MOD_PRIM might not seem large; however, recall that in Table 1, that the MOD_PRIM results for a small strictly theoretical example were on average 2.12% **worse** than the optimal solutions. When the five problems in Table 1 were solved with BEST_PRIM, BEST_PRIM found the optimal solution for all five problems. Therefore, although the primary purpose of this article is to determine simple and efficient metaheuristics that can generate near-optimal results for *very large* GCTPs and CTPs, it should be obvious from their definitions that both SG_MOD_PRIM and PSTOC_MOD_PRIM and especially BEST_PRIM will give excellent results for small and medium-sized graphs with very little computational effort.

As discussed in section 5, when we analyzed Example 4 from [11] (Table 1), we expect MOD_PRIM to perform better for extreme values of τ/γ (either very small or very large). Therefore, we hypothesize that the percent improvement of BEST_PRIM over MOD_PRIM should be greatest for intermediate values of τ (5 and 10). In Table 3, we summarize the percent improvements made by BEST_PRIM over MOD_PRIM for each value of τ. Although these results are not dramatic, as expected, Table 3 shows more improvement (both for CTPs and GCTPs) for $\tau = 5$ and $\tau = 10$ when compared with the overall percent improvement. There is no clear trend with the data, however, as the number of vertices varies.

For the 40 large GCTP problems (vascular imaging applications) that we solved using SG_MOD_PRIM, PSTOC_MOD_PRIM, and BEST_PRIM, it was impractical to generate optimal solutions. Therefore, to best quantify the quality of these solutions, we used the simple lower bound (SLB) defined in [11], which is the weighted average of the total cost of a minimum spanning tree solution and a SPT. Specifically, we generated SLBs for the 40 GCTPs discussed above. On average, the MOD_PRIM solutions were 4.8% higher than the SLBs. However, on average, the BEST_PRIM solution values were *only 1.6% higher* than the SLB. This means, that at most, these results deviate 1.6% from the optimum on average. For the 25,001-vertex GCTP graph in particular, MOD_PRIM results were on average 3.6% higher than the SLB, while the BEST_PRIM solutions were 1.7% higher than the SLB. These results indicate the significant benefit of executing both the SG_MOD_PRIM and the PSTOC_MOD_PRIM algorithms and using the best solution (BEST_PRIM) obtained from these algorithms over just using MOD_PRIM.

To justify the strategy of using both the SG_MOD_PRIM and the PSTOC_MOD_PRIM algorithms, that is, BEST_PRIM, we show in Table 4 how quickly one spanning tree for each problem size can be generated. As each run of SG_MOD_PRIM and PSTOC_MOD_PRIM on a given graph requires the same number of steps, there is negligible deviation in the running times of each metaheuristic on a given graph. Even for a problem with 25,000 vertices and 11 million edges, generating 30 spanning trees with SG_MOD_PRIM and 30 spanning trees with PSTOC_MOD_PRIM only requires 30 min of execution time if executed in a serial manner. If this approach is parallelized, then the execution time is greatly reduced.

TABLE 4. Execution time required to generate one solution tree

Number of vertices	Number of edges	Execution time (s)
501	24,675	0.026
1,001	96,272	0.082
2,501	467,698	0.36
5,001	1,334,652	1.29
10,001	2,639,817	8.34
15,001	4,455,521	10.82
20,001	7,299,926	18.74
25,001	11,164,733	29.32

8. CONCLUSIONS AND FUTURE WORK

In this article, we defined the GCTP to accommodate different edge lengths for the cable cost and trench cost. Furthermore, we developed two highly efficient metaheuristics, SG_MOD_PRIM and PSTOC_MOD_PRIM, that are capable of finding near-optimal solutions to extraordinarily large GCTPs in quadratic time in n. GCTP computations on graphs with 25,000 vertices and 11 million edges took under 30 s to generate one solution tree. We base our claim of near-optimality on the deviation of our results from computed SLBs on the unknown optimal solution values. While MOD_PRIM yielded results that were on average only 4.8% worse than the SLBs for the large GCTP problems, the best results from the modifications to MOD_PRIM yielded results that were only 1.6% worse than the SLBs. This means that our modifications to MOD_PRIM bridged at least 67% of the gap between the MOD_PRIM results and the unknown optimal results on average. For the largest graph we tested, BEST_PRIM bridged at least 53% of the gap between the MOD_PRIM results and the unknown optimal results on average.

Future work will examine the effectiveness of MOD_PRIM on graphs which are sufficiently small to find optimal solutions. Specifically, we will compare SLBs, optimal solutions, and MOD_PRIM solutions for a range of values of τ and n to better understand the effectiveness of MOD_PRIM. We will also apply d-heaps and Fibonacci heaps to MOD_PRIM to improve the running time on sparse graphs.

Our motivation for defining the GCTP and developing efficient metaheuristics for solving large GCTPs is the application to vascular image analysis, where the ultimate goal is to facilitate biomechanical analysis of the vascular network. We gave an intuitive description of the translation of Murray's Minimum Work Principle and vascular analysis into the language of graph theory and the GCTP by viewing a cable as the work needed to overcome friction and a trench as the work needed to maintain the blood volume, or metabolism. With the formulation of the vascular image analysis problem as a GCTP, one can efficiently and algorithmically quantify a blood vessel network more accurately than previous methods. In actual data, there is often missing data, or "gaps," in the scan which have to be corrected manually when using other methods. However, our approach automatically corrects these errors.

Other errors that invariably occur in the imaging process are false positive results in the vessel detection stage: data that does not correspond to a blood vessel. One fact that we will apply in a future paper is that one can determine those vertices that represent the tips of the capillaries. Thus, we will require our solution tree to contain these vertices and will "trim" the solution tree so that these vertices are the only leaf vertices. Other points determined in a scan are optional Steiner vertices. In this way, we will define the Generalized Steiner Cable-Trench Problem (GSCTP), develop efficient metaheuristics for solving very large instances of the GSCTP, and show that our metaheuristics can correct the false positive errors in the scan.

ACKNOWLEDGMENTS

The authors thank Dr. Albert Sinusas and Dr. Zhenwu Zhuang, Yale University School of Medicine, for providing vascular image data for our experiments. We also thank the referees and especially thank the editor, Dr. Douglas Shier, who made numerous suggestions that have significantly improved the quality of this article.

REFERENCES

[1] P. Bruyninckx, D. Loeckx, D. Vandermeulen, and P. Suetens, Segmentation of liver portal veins by global optimization, Proceedings of SPIE7624, Medical Imaging 2010, N. Karssemeijer and R.M. Summers (Editors), SPIE, San Diego, CA, 2010, p. 76241Z.

[2] E. Bullitt, S. Aylward, A. Liu, J. Stone, S. Mukherji, C. Coffey, G. Gerig, and S. Pizer. 3D graph description of the intracerebral vasculature from segmented MRA and tests of accuracy by comparison with X-ray angiograms, Proceedings of Information Processing in Medical Imaging, Lecture Notes in Computer Science Vol. 1613, A. Kuba, M. Sámal, and A. Todd-Pokropek (Editors), Springer, Visegrád, Hungary, 1999, pp. 308–321.

[3] J. Girard, P. Zarka, M. Tagger, L. Denis, D. Charrier, A. Konovalenko, and F. Boone, Antenna design and distribution of the LOFAR super station, C R Phys 13 (2012), 33–37.

[4] Y. Jiang, Z. Zhuang, A. Sinusas, and X. Papademetris, Vascular tree reconstruction by minimizing a physiological functional cost, Conf Comput Vis Pattern Recognit Workshops (2010), 178–185.

[5] Y. Jiang, Z. Zhuang, A. Sinusas, L. Staib, and X. Papademetris, Vessel connectivity using Murray's hypothesis, Proceedings of MICCAI 2011, Lecture Notes in Computer Science Vol. 6893, G. Fichtinger, A.L. Martel, and T.M. Peters (Editors), Springer, Toronto, ON, Canada, 2011, pp. 528–536.

[6] J. Jomier, V. Ledigarcher, and S. Aylward, Automatic vascular tree formation using the Mahalanobis distance, Proceedings of MICCAI 2005, Lecture Notes in Computer Science Vol. 3750, J. Duncan and G. Gerig (Editors), Springer, Palm Springs, CA, 2005, pp. 806–812.

[7] V. Marianov, G. Gutiérrez-Jarpa, C. Obreque, and O. Cornejo, Lagrangean relaxation metaheuristics for the *p*-cable-trench problem, Comput Oper Res 39 (2012), 620–628.

[8] C. Murray, The physiological principle of minimum work, I. The vascular system and the cost of blood volume, Proc Natl Acad Sci USA 12 (1926), 207–214.

[9] R. Nielsen, M. Riaz, J. Pedersen, and O. Madsen, On the potential of using the cable trench problem in planning of ICT access networks, Proceedings of ELMAR-2008, M. Grgic (Editor), IEEE, Zadar, Croatia, 2008, pp. 585–588.

[10] A. Szymczak, A. Stillman, A. Tannenbaum, and K. Mischaikow, Coronary vessel trees from 3D imagery: A topological approach, Med Image Anal 10 (2006), 548–559.

[11] F. Vasko, R. Barbieri, B. Rieksts, K. Reitmeyer, and K. Stott, The cable trench problem: Combining the shortest path and minimum spanning tree problems, Comput Oper Res 29 (2002), 441–458.

[12] L. Zagorchev, P. Oses, Z. Zhuang, K. Moodie, M. Mulligan-Kehoe, M. Simons, and T. Couffinhalt, Micro computed tomography for vascular exploration, J Angiogenes Res 2 (2010), 7.