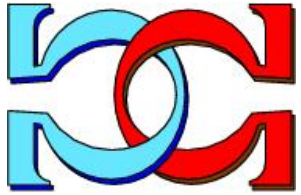
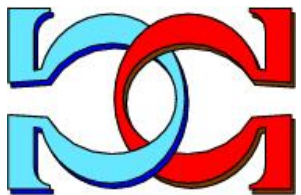
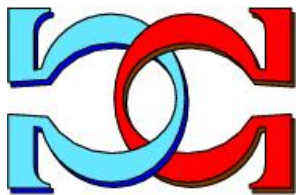


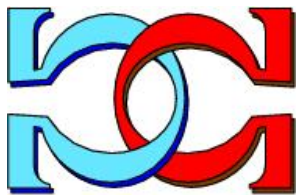
**CDMTCS
Research
Report
Series**



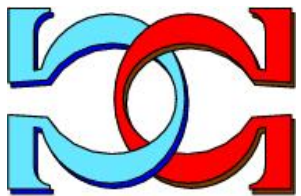
**Hardness of Approximation
and Integer Programming
Frameworks for Searching for
Caterpillar Trees**



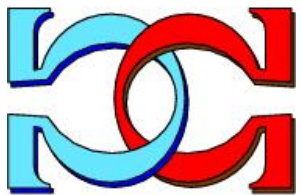
Michael J. Dinneen
and



Masoud Khosravani



Department of Computer Science,
University of Auckland,
Auckland, New Zealand



CDMTCS-394
November 2010

Centre for Discrete Mathematics and
Theoretical Computer Science

Hardness of Approximation and Integer Programming Frameworks for Searching for Caterpillar Trees

Micheal J. Dinneen Masoud Khosravani*

November 14, 2010

Abstract

We consider the problems of finding a caterpillar tree in a graph. We first prove that, unless $P=NP$, there is no approximation algorithms for finding a minimum spanning caterpillar in a graph within a factor of $f(n)$; where $f(n)$ is any polynomial time computable function of n , the order of the graph. Then we present a quadratic integer programming formulation for the problem that can be a base for a branch and cut algorithm. We also show that by using Gomory cuts iteratively, one can obtain a solution for the problem that is close to the optimal value by a factor of $1/\epsilon$, for $0 < \epsilon < 1$. Finally, we show that our formulation is equivalent to a semidefinite programming formulation, which introduces another approach for solving the problem.

Keywords: Caterpillar Trees, Optimization, Approximation Algorithm, Integer Programming, Semidefinite Programming.

1 Introduction

In this paper we study the Minimum Spanning Caterpillar Problem (MSCP) and the Largest Caterpillar Problem (LCP). By a *caterpillar* we mean a tree that reduces to a path by deleting all its leaves. We refer to the remaining path as the *spine* of the caterpillar. The edges of a caterpillar H can be partitioned into two sets, the spine edges, $\mathcal{S}(H)$, and the leaf edges, $\mathcal{L}(H)$. An instance of the MSCP is denoted by a triple (G, s, l) , where $G = (V, E)$ is an undirected graph and $s : E \rightarrow \mathbb{N}$ and $l : E \rightarrow \mathbb{N}$ are two (cost) functions. For each caterpillar H as a subgraph of G we define the cost of H by

$$c(H) := \sum_{e \in \mathcal{S}(H)} s(e) + \sum_{e' \in \mathcal{L}(H)} l(e').$$

In the MSCP one wants to find a caterpillar in a graph with the minimum cost that contains all vertices, as shown in Figure 1. In the LCP the goal is to find a caterpillar with the largest number of vertices. Note that while every connected graph has a spanning tree, there are some graphs that do not have a spanning caterpillar. It is not hard

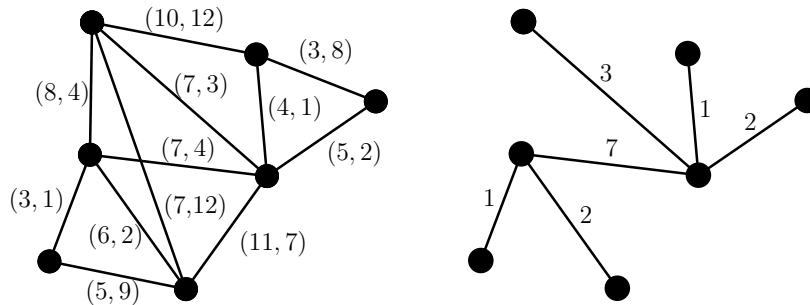


Figure 1: A graph and its minimum spanning caterpillar, where each (s, l) label on an edge represents the spine and leaf costs.

to prove that both problems are NP-hard for general graphs by reductions from the Hamiltonian Path Problem.

In some applications in addition to finding an optimal cost spanning caterpillar, we also need to satisfy some restrictions on the final output. For example one may wish to have a caterpillar whose total spine cost is bounded by a fixed number, or one may wish to have an upper bound on the largest degree of a caterpillar.

Caterpillars appear in many applications. For example, in network design, one may wish to find a cost effective linearly arranged backbone to place the communication routers. We may also consider the MSCP as a facility transportation problem, where we are allowed to divide the task of distributing facilities among one global and costly distributor and some local and cheap ones. Here the global distributor follows the spine path to deliver facilities and the local ones use the leaf edges. The goal is to find a transportation route that has the minimum overall cost.

In chemical graph theory caterpillars are considered as a model for benzenoid hydrocarbon molecules; see [4]. They also appear in bioinformatics in designing algorithms for RNA structure alignment and comparing evolutionary trees; see [2] and [6]. For more applications in combinatorics and mathematics (in general) we refer the reader to [9] and [10]

[8] present an exact non-polynomial time algorithm that finds a minimum spanning caterpillar. Their method is based on an Integer Linear Programming (ILP) formulation by transforming the MSCP to the Minimum Steiner Arborescence Problem. In another paper we present a linear time algorithm for the special case when the input graph is restricted to the bounded treewidth graphs when the associated tree decompositions are given as part of the inputs; see [3].

The organization of this paper is as follows. In the next section we give our hardness result for approximating the minimum spanning caterpillar in a graph. In Section 3 we present our quadratic integer programming formulation for the problems and some restricted versions of them. In Section 4 we present a heuristic algorithm for the MSCP by using the Gomory cutting plane method iteratively. We also show that for every $\epsilon > 0$, our algorithm guarantees to achieve a $1/\epsilon$ factor approximation. Semidefinite programming gives another alternative for solving the problems, which is the theme of Section 5.

2 Hardness of Approximation

In this section we show that by assuming $P \neq NP$ it is not possible to find an approximation algorithm for the MSCP within a factor of a polynomial-time computable function of the number of vertices of the graph. Our proof is based on a reduction from the Hamiltonian Path Problem to an instance of the MSCP. We show if there is such an approximation algorithm for MSCP then there is a polynomial time algorithm for deciding the Hamiltonian Path Problem. Our proof follows the same idea as [7].

Theorem 1 *Let $f(n)$ be a polynomial time computable function. The MSCP has no approximation algorithm within a factor of $f(n)$, unless $P = NP$*

Proof. Let $G = (V, E)$ be an instance of the Hamiltonian Path Problem with $|V| \geq 3$. We reduce it to an instance (G', s, l) of MSCP. Here G' is made from a copy of G with one extra vertex v_{ext} which is connected to all vertices of the copy of G . Then we define two cost functions s and l such that to every edge e that belongs to the copy of G , the function s assigns the value of $s(e) = 1$ and the function l assigns the value of $l(e) = f(n)(n - 1)$, where $n = |V|$ is the number of vertices of G .

To assign cost functions to the edges incident to v_{ext} we choose a fixed vertex v_{fix} in our copy of G . Then both functions assign the value 0 to the edge (v_{fix}, v_{ext}) , and they assign the value of $f(n)(n - 1)$ to the other edges incident to v_{ext} . We add the extra vertex v_{ext} to ensure that graph G' has at least one spanning caterpillar.

Now we show that if one has an $f(n)$ -approximation algorithm for the MSCP then it can be used to decide if G has a Hamiltonian path in polynomial time. We first run the approximation algorithm on G' , then it will return a solution T that has $c(T) \leq f(n)OPT$. Where OPT is the overall cost of a minimum spanning caterpillar in G' . Now if G has a Hamiltonian path then $OPT = n - 1$ and $c(T) \leq f(n)(n - 1)$. Note that in this case v_{ext} is attached to the Hamiltonian path in the copy of G by (v_{fix}, v_{ext}) . If G has no Hamiltonian path then the resulting spanning caterpillar either has at least one leaf edge from the copy of G , or it uses two incident edges of v_{ext} , so we have

$$f(n)(n - 1) < OPT \leq c(T).$$

■

Theorem 1 justifies our effort in the rest of the paper to concentrate on finding heuristic algorithms for solving the MSCP. Since, assuming $P \neq NP$, there is no polynomial-time approximation algorithm for the problem.

3 Integer Programming Formulation

Having an integer linear programming formulation is the first step in applying the branch and cut method for solving an optimization problem. In this section we introduce an integer quadratic programming formulation that can be applied to both problems. As we shall show later, by adding proper constraints, we can use this formulation for solving

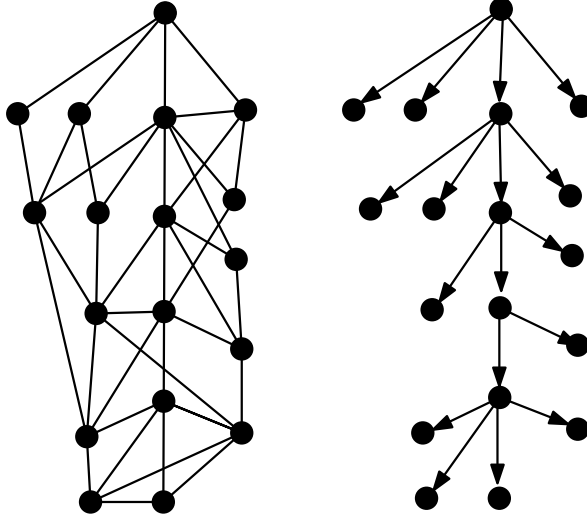


Figure 2: A graph and its rooted spanning caterpillar.

some restricted versions of the problems. Though our basic formulation is not linear, it is known that any integer quadratic programming problem can be transformed to a linear one by adding extra variables and constraints in a mechanical way; see [14]. We first explain the formulation for the MSCP. Note that without loss of generality we can assume that all instances of the MSCP are complete graphs.

Let $G = (V, E)$ be a graph and let (G, s, l) be an instance of the MSCP. We convert G to a directed graph $H = (N, A)$, $N = V$, with replacing each edge $e \in E$ with a pair of anti-parallel arcs, $f, f^- \in A$ (with opposite directions). In this case, we can consider a caterpillar in H as a rooted tree that has a directed path as its spine; see Figure 2. Where each spine vertex has one incoming arc and one or more outgoing arcs and every leaf vertex has one incoming arc and no outgoing arc.

For each vertex $v \in N$, we denote by $in(v)$ and $out(v)$ the incoming and the outgoing arcs of v , respectively. Also for each $f \in A$ we denote by f^- the anti-parallel arc associated with f . The costs of anti-parallel arcs are the same as the cost of their corresponding (undirected) edge, which depend on their role as spine or leaf arcs. For each arc $f \in A$ we denote its spine cost $s(f)$ by s_f and its leaf cost $l(f)$ by l_f .

Here we first present our integer programming formulation for the MSCP with a fixed root r . To each arc $f \in A$ we assign two variables, x_f and y_f . In each solution a variable x_f is 1 if f is chosen as a leaf edge and a variable y_f is 1 if it is chosen as a spine edge, otherwise they are 0. Also to each vertex $v \in N$ we assign variable z_v that represents the level of v in the rooted caterpillar. In particular, we have $z_r = 0$. Our goal is to minimize the objective function

$$\left(\sum_{f \in A} l_f x_f + \sum_{f \in A} s_f y_f \right).$$

In what follows M is a large positive integer, say, a number greater than the order of the graph. We show the integer programming constraints in Figure 3.

$$x_f + y_f + x_{f^-} + y_{f^-} \leq 1, \quad \forall f \in A, \quad (1)$$

$$\sum_{f \in \text{in}(v)} y_f + \sum_{f \in \text{in}(v)} x_f = 1, \quad \forall v \in N \setminus r, \quad (2)$$

$$\sum_{f \in \text{out}(v)} y_f + \sum_{f \in \text{in}(v)} x_f = 1, \quad \forall v \in N, \quad (3)$$

$$\sum_{f \in \text{out}(v)} x_f - M \left(\sum_{f \in \text{in}(v)} y_f + \sum_{f \in \text{out}(v)} y_f \right) \leq 0, \quad \forall v \in N, \quad (4)$$

$$\sum_{f=(u,v) \in \text{in}(v)} y_f(z_v - z_u) + \sum_{f=(u,v) \in \text{in}(v)} x_f(z_v - z_u) = 1, \quad \forall v \in N \setminus r, \quad (5)$$

$$z_r = 0, \quad (6)$$

$$\forall f \in A, \quad (7)$$

$$x_f, y_f \in \{0, 1\}, \quad (8)$$

$$\forall v \in N, \quad (8)$$

$$z_v \in \{0, \dots, n-1\}.$$

Figure 3: The constraints in the integer programming formulation for the MSCP.

The next theorem shows that each integral feasible solution, that satisfies these constraints, represents a spanning caterpillar.

Theorem 2 *Constraints 1-8 make a valid formulation for the MSCP.*

Proof. The first constraint shows that from the pair of anti-parallel arcs that have the same end vertices, only one may be chosen in a feasible solution. The second and the third constraints say that each vertex on the spine has at most one incoming spine arc and one outgoing spine arc while it has no incoming leaf arc. Constraint 4 says that if a vertex is chosen as a leaf then it has no outgoing leaf arc, but if it is chosen as a spine vertex then there is no restriction on the number of its outgoing leaf arcs. Constraint 5 alongside Constraint 6 guarantee that each feasible solution is a connected tree that is rooted at r . Constraints 7 and 8 enforce the integrality of a feasible solution.

On the other hand, let T be a spanning caterpillar of a graph G . Also let e be an edge of T , if e is a spine edge then we assign $y_f = 1$ and $x_f = 0$, if e is a leaf edge then we set $y_f = 0$ and $x_f = 1$. For any other edge f of the graph that does not belong to T , we have $x_f = y_f = 0$. Now it is easily seen that these values make a feasible solution for our formulation. ■

When in our integer programming formulation we replace the Constraints 7 and 8 on the integrality of variables by weaker constraints that for all $f \in A$, $x_f \geq 0$ and $y_f \geq 0$ and also for all $v \in N$, $z_v \geq 0$, then we say that we have a *relaxation* of the

integer programming problem. While solving an integer linear programming problem is NP-hard, its relaxation (a linear programming problem) can be solved in linear time.

Note that with some changes we can use the same formulation for the LCP. First, we need to change the objective to

$$\text{maximize } \left(\sum_{f \in A} (x_f + y_f) \right),$$

then we have to change the equality relations in Constraints 2, 3 to inequalities. Constraint 5 needs more changes to ensure the connectivity and to prevent cycles from appearing in feasible solutions. We leave the details to the reader.

Now we consider more constraints to formulate the restricted versions of the problems. The first one is when we have a restriction on the number of spine edges, as an upper bound U . We can impose this by adding the following constraint

$$\sum_{f \in A} y_f \leq U. \quad (9)$$

If the restriction is on the overall cost of spine then we have

$$\sum_{f \in A} s_f y_f \leq U. \quad (10)$$

Also the following set of inequalities restricts the degree of each vertex to be a value no more than a fixed $\delta > 0$.

$$\sum_{f \in \text{in}(v) \cup \text{out}(v)} (y_f + x_f) \leq \delta, \quad \forall v \in A. \quad (11)$$

In the rest of the paper and with respect to our result on the hardness of approximation of the MSCP, we will concentrate on finding a heuristic algorithm for the problem.

4 An $\left(\frac{1}{\epsilon}\right)$ -Approximation for the MSCP

It is known that each quadratic programming problem can be transformed to a linear programming (LP) formulation in a mechanical way by introducing more variables and constraints. We refer the reader to the text by [14]. So in this section we assume that we have an integer linear programming formulation (ILP) for the problem. By this assumption, we can solve the relaxation by any available method in polynomial time, and then we will apply Gomory cuts iteratively to find an approximation solution to the problem.

A Gomory cutting method adds a linear constraint to the set of constraints of an integer linear programming problem such that it does not exclude any feasible integer solution. The process is repeated until an integer optimal solution is found. It is proven that the Gomory cutting method always terminates with an integer solution that is

optimal. Note that here the number of steps required may be exponential. We use the Gomory cutting method to present a heuristic algorithm for the MSCP. The outline of the algorithm is as follows.

Algorithm: $(\frac{1}{\epsilon})$ -Approximation via Gomory cut

Input: Graph (G, s, l) and $\epsilon > 0$

Output: A Caterpillar T

1. Construct the integer programming formulation.
 2. Find an optimal solution, OPT, for the relaxation.
 3. Find a directed path starting from r such that the y values of the arcs of the path are not less than ϵ , where $0 < \epsilon < 1$. To this end choose an outgoing arc from r with y value at least ϵ . Then follow this process until reaching a vertex that has no outgoing arc with that property.
 4. If all other vertices are attached to this spine with x values that are greater or equal to ϵ , then the resulted caterpillar has a cost less than $(\frac{1}{\epsilon})OPT$ and stop. Else use the Gomory cutting plane method to add one more constraint.
 5. Solve the resulting linear programming problem and go to Step 3.
-

The convergence of Gomory cuts guarantees that the desired approximation is achievable within a finite number of iterations. What follows is our formal justification.

Theorem 3 *There is an algorithm that for any ϵ , $0 < \epsilon < 1$, computes a $1/\epsilon$ factor approximation for the MSCP.*

Proof. Here we prove that by following the algorithm mentioned above, we eventually reach to the desired approximation. Let OPT_A be the cost of the final caterpillar that spans the graph and has arcs with costs at least ϵ .

First, note that if for a vertex $v \in N$ there is an arc $f \in in(v)$, such that $x_f \geq \epsilon > 0$, then we have

$$1 - x_f \leq x_f \left(\frac{1 - \epsilon}{\epsilon} \right).$$

Also if there is an arc f such that $y_f \geq \epsilon > 0$ we have

$$1 - y_f \leq y_f \left(\frac{1 - \epsilon}{\epsilon} \right).$$

Now by using these inequalities for the edges of a caterpillar we have

$$\begin{aligned} \sum_{f \in L} l_f(1 - x_f) + \sum_{f \in S} s_f(1 - y_f) &\leq \\ &\left(\frac{1 - \epsilon}{\epsilon}\right) \left(\sum_{f \in L} l_f x_f + \sum_{f \in S} s_f y_f\right), \end{aligned}$$

where L is the set of leaf edges and S is the set of the spine edges of the caterpillar.

Now by using this information we show that the resulting caterpillar has a cost that is less than $\frac{1}{\epsilon}\text{OPT}$. First of all we have

$$\text{OPT}_A = \sum_{f \in L} l_f + \sum_{f \in S} s_f.$$

By rewriting the right-hand side summations we have

$$\begin{aligned} \sum_{f \in L} l_f + \sum_{f \in S} s_f &= \sum_{f \in L} (x_f + (1 - x_f))l_f + \\ &\quad \sum_{f \in S} (y_f + (1 - y_f))s_f \\ &= \left(\sum_{f \in L} l_f x_f + \sum_{f \in S} s_f y_f\right) + \\ &\quad \left(\sum_{f \in L} l_f(1 - x_f) + \sum_{f \in S} s_f(1 - y_f)\right) \\ &\leq \left(1 + \frac{1 - \epsilon}{\epsilon}\right) \left(\sum_{f \in L} l_f x_f + \sum_{f \in S} s_f y_f\right) \\ &\leq \frac{1}{\epsilon}\text{OPT}. \end{aligned}$$

■

There are many software tools for solving LP problems and this is an advantage for our method. On the other side, using Gomory cuts has its own drawbacks. Since there is no guarantee on the rate of convergence to the optimal (integral) solution. Also, converting the quadratic integer programming formulation to an LP one, introduces many new variables and constraints that increases the size of the input.

In the next section we show that the MSCP can be considered as a semidefinite programming problem. So we do not need to introduce many new variables and constrains. This point of view gives another alternative for obtaining a heuristic algorithm.

5 Semidefinite Programming Transformation

A semidefinite programming problem is the problem of optimization of a linear function of a symmetric and positive semidefinite matrix subject to linear equality constraints;

$$\mathbf{v}_f^x \cdot \mathbf{v}_t + \mathbf{v}_f^y \cdot \mathbf{v}_t + \mathbf{v}_{f-}^x \cdot \mathbf{v}_t + \mathbf{v}_{f-}^y \cdot \mathbf{v}_t \leq 1, \quad \forall f \in A, \quad (12)$$

$$\sum_{f \in \text{in}(v)} \mathbf{v}_f^y \cdot \mathbf{v}_t \leq 1, \quad \forall v \in N, \quad (13)$$

$$\sum_{f \in \text{out}(v)} \mathbf{v}_f^y \cdot \mathbf{v}_t \leq 1, \quad \forall v \in N, \quad (14)$$

$$\sum_{f \in \text{in}(v)} \mathbf{v}_f^x \cdot \mathbf{v}_t + \sum_{f \in \text{in}(v)} \mathbf{v}_f^y \cdot \mathbf{v}_t \leq 1, \quad \forall v \in N, \quad (15)$$

$$\sum_{f \in \text{in}(v)} \mathbf{v}_f^x \cdot \mathbf{v}_t + \sum_{f \in \text{out}(v)} \mathbf{v}_f^y \cdot \mathbf{v}_t \leq 1, \quad \forall v \in N, \quad (16)$$

$$\sum_{f \in \text{out}(v)} \mathbf{v}_f^x \cdot \mathbf{v}_t - M \left(\sum_{f \in \text{in}(v)} \mathbf{v}_f^y \cdot \mathbf{v}_t + \sum_{f \in \text{out}(v)} \mathbf{v}_f^y \cdot \mathbf{v}_t \right) \leq 0, \quad \forall v \in N, \quad (17)$$

$$\sum_{f=(u,v) \in \text{in}(v)} \mathbf{v}_f^y \cdot (\mathbf{v}_v^z - \mathbf{u}_v^z) + \sum_{f=(u,v) \in \text{in}(v)} \mathbf{v}_f^x \cdot (\mathbf{v}_u^z - \mathbf{u}_v^z) = 1, \quad \forall v \in N \setminus r \quad (18)$$

$$\mathbf{v}_r^z \cdot \mathbf{v}_t = 0. \quad (19)$$

Figure 4: The constraints in the vector programming formulation of the MSCP.

for exact definition see [1]. One can solve a semidefinite programming problem within a constant approximation factor.

Semidefinite programming has been applied to solve some problems in combinatorial optimization. For example [5] obtained a 0.878 factor randomized approximation algorithm for the Maximum Cut Problem by using this method.

As we show in the former section the MSCP can be formulated as a quadratic integer program. Now we convert the problem to a vector program. In a vector program the objective function and all constraints are represented as linear combinations of inner products of vectors. It is known that each vector program is equivalent to a semidefinite program; see [5] and [13].

Theorem 4 *The quadratic formulation of the MSCP has a semi-definite equivalent formulation.*

Proof. We assign vectors $\mathbf{v}_e^x = (x_e, 0, \dots, 0)$, $\mathbf{v}_e^y = (y_e, 0, \dots, 0)$, and $\mathbf{v}_u^z = (z_u, 0, \dots, 0)$ to variables x_e, y_e , and z_e , respectively. We also have an extra vector variable $\mathbf{v}_t = (1, \dots, 1, t)$. Now we write the objective function and all constraints in the former section by linear combinations of inner products of these vector variables.

Finally we want to minimize

$$\left(\sum_{f \in A} l_f (\mathbf{v}_f^x \cdot \mathbf{v}_t) + \sum_{f \in A} s_f (\mathbf{v}_f^y \cdot \mathbf{v}_t) \right),$$

subject to the constraints given in Figure 4.

The validity of the assertion follows from the fact that any vector programming problem is equivalent to a semidefinite programming problem and vice versa. ■

6 Conclusion and Open Problems

We introduced the Minimum Spanning Caterpillar Problem. We showed that the problem has no approximation algorithm with a polynomial time computable function as an approximation factor, unless $P = NP$. By this result we end the search for any *proper* approximation algorithm. Then we introduced our quadratic integer programming formulation for the problem and we gave some arguments on its strength and weakness for finding heuristic algorithms, by using Gomory cuts or semidefinite programming.

There are some natural open problems left, such as:

1. We know that the Traveling Salesman Problem (TSP) also has the same property with respect to having no approximation algorithm within a polynomial time computable factor, while its metric version has a constant factor approximation. Now the question that arises is: “Can one find a constant factor approximation algorithm for a metric version of the MSCP?”
2. When compared to our method of using Gomory cuts, does a branch and cut algorithm behave better in practice?
3. Is the LCP (Largest Caterpillar Problem) has a better approximation algorithm? We refer the reader to the relation between the TSP and the longest path problem.
4. It is known that the longest path problem has polynomial time solutions for some classes of graphs like block graphs: [11], and bipartite permutation graphs: [12]. We are interested in knowing the computation complexity of the MSCP and the LCP on those classes of graphs.

References

- [1] Farid Alizadeh. Interior point methods in semidefinite programming with applications to combinatorial optimization. *SIAM J. Optim.*, 5(1):13–51, 1995.
- [2] Miklos Csuros, J. Andrew Holey, and Igor B. Rogozin. In search of lost introns. *Bioinformatics*, 23(13):i87–96, 2007.
- [3] Michael J. Dinneen and Masoud Khosravani. A linear time algorithm for the minimum spanning caterpillar problem for bounded treewidth graphs. In Boaz Patt-Shamir and Tinaz Ekim, editors, *Structural Information and Communication Complexity*, volume 6058 of *Lecture Notes in Computer Science*, pages 237–246. Springer Berlin / Heidelberg, 2010.

- [4] Sherif El-Basil. Caterpillar (gutman) trees in chemical graph theory. In Ivan Gutman and Sven Cyvin, editors, *Advances in the Theory of Benzenoid Hydrocarbons*, volume 153 of *Topics in Current Chemistry*, pages 273–289. Springer Berlin / Heidelberg, 1990.
- [5] Michel X. Goemans and David P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. Assoc. Comput. Mach.*, 42(6):1115–1145, 1995.
- [6] Antoni Lozano, Ron Y. Pinter, Oleg Rokhlenko, Gabriel Valiente, and Michal Ziv-Ukelson. Seeded tree alignment. *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, 5(4):503–513, 2008.
- [7] Sartaj Sahni and Teofilo Gonzalez. P -complete approximation problems. *J. Assoc. Comput. Mach.*, 23(3):555–565, 1976.
- [8] L. Simonetti, Yuri Frota, and Cid C. de Souza. An exact method for the minimum caterpillar spanning problem. In Sonia Cafieri, Antonio Mucherino, Giacomo Nannicini, Fabien Tarissan, and Leo Liberti, editors, *Proceedings of the 8th Cologne-Twente Workshop on Graphs and Combinatorial Optimization, CTW 2009, Paris, France, June 2-4 2009*, pages 48–51, 2009.
- [9] László A. Székely and Hua Wang. On subtrees of trees. *Advances in Applied Mathematics*, 34(1):138–155, 2005.
- [10] Jinsong Tan and Louxin Zhang. The consecutive ones submatrix problem for sparse matrices. *Algorithmica*, 48(3):287–299, 2007.
- [11] Ryuhei Uehara and Yushi Uno. Efficient algorithms for the longest path problem. In *Algorithms and computation*, volume 3341 of *Lecture Notes in Comput. Sci.*, pages 871–883. Springer, Berlin, 2004.
- [12] Ryuhei Uehara and Gabriel Valiente. Linear structure of bipartite permutation graphs and the longest path problem. *Inform. Process. Lett.*, 103(2), 2007.
- [13] Vijay V. Vazirani. *Approximation algorithms*. Springer-Verlag, Berlin, 2001.
- [14] Stanisław Walukiewicz. *Integer programming*, volume 46 of *Mathematics and its Applications (East European Series)*. Kluwer Academic Publishers Group, Dordrecht, 1991.