The Color-balanced spanning tree problem

Štefan BEREŽNÝ[a],
and
Vladimír LACKO[b]

[a] Department of Mathematics,
Technical University, Boženy Němcovej 32, 040 01 Košice, Slovakia
[b] Institute of Mathematics,
P. J Šafárik University, Jesenná 5, 041 54 Košice, Slovakia

June 15, 2004

Abstract

Suppose a graph $G = (V, E)$ whose edges are partitioned into p disjoint categories (colors) is given. In the color-balanced spanning tree problem a spanning tree is wanted that minimizes the variability in the number of edges from different categories.

We show that polynomiality of this problem depends on the number p of categories and present some polynomial algorithm.

Keywords

spanning tree, matroids, algorithms, NP-completeness

1 Introduction

Suppose a graph $G = (V, E)$ with nonnegative edge weights $w(e)$ for $e \in E$ is given and suppose its edges are partitioned into disjoint categories $S_1, \ldots, S_p$. Denote by $\mathcal{T}(G)$ the family of all spanning trees of graph G. Now consider the following objective function:

$$f(T) = \max_{1 \le i \le p} \Big(\sum_{e \in S_i \cap T} w(e) \Big) - \min_{1 \le i \le p} \Big(\sum_{e \in S_i \cap T} w(e) \Big)$$

and optimization problem

$$\begin{array}{c} f(T) \longrightarrow \min \\ T \in \mathcal{T}(G) \end{array} \tag{1}$$

In the definition of function f it is supposed that maximum over the empty set is 0.

In [2] it was shown, that problem (1) is NP-complete even if the number of categories p is equal to 2 and the underlying graph G is outerplanar. It was showed there that the spanning tree, matching and path problems considered with the L_3 objective function (this function is in fact the same as objective function f of this paper) are NP-complete already on bipartite outerplanar graphs even for two categories, similarly the L_3-travelling salesman problem is NP-complete on Halin graphs even for two categories. Some other optimization problems (e.g. matchings, Hamilton circuits etc.) with objective functions similar to f were treated in [3]. For most of functions, they were shown polynomial, if the number of categories p is fixed and NP-complete in general case. Some other problems with categorization of edges and with objective functions using the operators min, max and $\sum$ were treated in [1, 7, 8, 2, 3, 4]

More general review of color-balanced problems can be found in [5]. In this paper we show a reduction of problem (1) to problem 1-CCOP which is a special case of problem K-CCOP treated in [5].

In this paper we deal with the following special case of problem (1): we let all weights of edges be equal, w.l.o.g. $(\forall e \in E)\ w(e) = 1$ and we restrict the number of categories to $p = 2$ (section 2) or let p be constant (section 3). We show, that the problem with constant weights belongs to the class P, in contrast to the original problem (1) with arbitrary weights which is NP-complete.

2 Color-balanced spanning tree problem

Let us consider the special case of problem (1) where $p = 2$ and $(\forall e \in E)\ w(e) = 1$, i.e. $f(T) = \max\{|S_1 \cap T|, |S_2 \cap T|\} - \min\{|S_1 \cap T|, |S_2 \cap T|\} = ||S_1 \cap T| - |S_2 \cap T||$ and problem (1) in this special case can be written as:

$$\begin{array}{c} ||S_1 \cap T| - |S_2 \cap T|| \longrightarrow \min \\ T \in \mathcal{T}(G) \end{array} \qquad (2)$$

For the sake of simplicity assume for the time being that graph G is connected. Disconnected graphs will be dealt with later. Under our assumption, since T is a spanning tree of G, $|T| = |V| - 1$ and thus let $|T| = k$. The objective function $f(T)$ attains its minimum possible value if $|S_1 \cap T|$ and $|S_2 \cap T|$ are as close to each other as possible, which occurs if one of them is equal to $\lceil \frac{k}{2} \rceil$ and the other to $\lfloor \frac{k}{2} \rfloor$. Minimum value of $f(T)$ is then either 0 if k is even or 1 otherwise. On the other hand, if one of $|S_1 \cap T|$ and $|S_2 \cap T|$ is equal to k and the other is 0, $f(T)$ attains its maximum, $f(T) = k$. The range of possible optimum values for given graph is then limited to set $\{0, 1, \dots, k\}$. If we are able to check for each $l \in \{0, 1, \dots, k\}$, whether there exists a spanning tree T with $f(T) = l$, as a consequence we will immediately have the desired optimum spanning tree of problem (2).

The test we need to perform, even in more specific form, is described in the following lemma:

Lemma 2.1 *(Check(i,j)) Given a graph $G = (V, E)$, a partition of E to S_1, S_2 and $i, j \in N$, s.t. $i + j = |V| - 1 = k$, it is possible to find a spanning tree T_{ij} of G with $T \cap S_1 = i$ and $T \cap S_2 = j$ or to determine that such a spanning tree does not exist. In the latter case it is possible to find a maximum cardinality forest T_{ij} of G satisfying $T \cap S_1 \leq i$ and $T \cap S_2 \leq j$. This can be done in polynomial time.*

Proof. Let $M_1 = (E, \mathcal{F}_1)$ be the matroid with the base set E (edges of the graph G) and independent sets $\mathcal{F}_1$ being families of edge sets of all acyclic subgraphs of G.

Matroid M_1 is therefore the graph matroid of graph G. Let $M_2(i,j) = (E, \mathcal{F}_2)$ be another matroid defined on the same base set E with independent sets $\mathcal{F}_2$ being defined as follows: $X \in \mathcal{F}_2 \Leftrightarrow X \subseteq E,\ X \cap S_1 \leq i,\ X \cap S_2 \leq j$. Matroid M_2 is thus the partition matroid over partition $S_1,\ S_2$ with limits i and j respectively.

Using the Cardinality Intersection Algorithm (CI-algorithm) described e.g. in [6] it is possible to determine the maximum cardinality intersection T_{ij} of matroids M_1 and $M_2(i,j)$. The intersection T_{ij} is, from its definition, independent in both matroids, i.e. it is an acyclic subgraph of G having $T_{ij} \cap S_1 \leq i$ and $T_{ij} \cap S_2 \leq j$. CI-algorithm runs in $O(m^2R + mRc(m))$ time (see [6]), where $m = |E|$, R is the cardinality of the resulting intersection and $c(m)$ is the complexity of independence tests in both matroids. Clearly R is at most $|V|-1$ and independence tests in both M_1 and M_2 can be performed in $O(m)$ time giving $O(m^2R + mRc(m)) = O(m^2|V|)$ for the total complexity of CI-algorithm in this case.

Acyclic subgraph T_{ij} of G is a spanning tree of G if and only if $|T_{ij}| = |V|-1$, otherwise it is just a maximum cardinality forest for which $T \cap S_1 \leq i$ and $T \cap S_2 \leq j$ holds. Since matroids M_1 and M_2 can be constructed in $O(m)$ time, the lemma follows. □

Now we can write down the algorithm for solving the problem (2):

Algorithm f-SpanningTree

Input : Graph $G = (V, E)$, partition of E to S_1 and S_2.
Output : f-optimal spanning tree T^{opt}.
K0 : $T^{opt} := \emptyset,\ L^{opt} := \infty$
K1 : **for each** i, j, s.t. $i + j = |V| - 1$ **do**
begin
K2 : $T_{ij} = Check(i,j)$
K3 : **if** $|T_{ij}| = |V| - 1$ & $|i - j| < L^{opt}$ **then**
K4 : $T^{opt} := T_{ij},\ L^{opt} = |i - j|$
end

Lemma 2.2 *Algorithm f-SpanningTree runs in* $O(m^2|V|^2)$ *time.*

Proof. There are exactly $|V|$ possibilities for expressing $|V|-1$ as a sum of two integers $k = |V|-1 = i+j$ in step $K1$ of the algorithm, namely $[k,0],\ [k-1,1], \ldots, [\lfloor\frac{k}{2}\rfloor, \lceil\frac{k}{2}\rceil], \ldots,\ [0,k]$, thus there are k invocations of $Check(i,j)$ in step $K2$. The total complexity is then $(|V|-1).O(m^2|V|) = O(m^2|V|^2)$. □

3 Constant number of categories greater than 2

Let us consider an even more relaxed case of problem (1) where edge weights are still uniform (w.l.o.g. $(\forall e \in E)\ w(e) = 1$). The number of categories p is, however, no more restricted to $p = 2$, but it must be constant, i.e. p does not depend on G.

The problem (1) in this special case can be written as:

$$f(T) = \max_{i=1,\ldots,p} \{|S_i \cap T|\} - \min_{i=1,\ldots,p} \{|S_i \cap T|\} \longrightarrow \min \qquad (3)$$
$$T \in \mathcal{T}(G)$$

Problem (3) can be solved using a similar approach as in section 2. At first, let us show the p-partition analogue of $Check(i,j)$:

Lemma 3.1 *(Check($i_1, \ldots, i_p$)) Given a graph $G = (V, E)$, a partition E to $S_1, \ldots, S_p$ and $i_1, \ldots, i_p \in N$, s.t. $\sum_{j=1}^{p} i_j = |V| - 1$, it is possible to find a spanning tree T of G s.t. $(\forall j)|T \cap S_j| = i_j$ or to determine, that such a spanning tree does not exist. This decision can be done and T can be found in polynomial time.*

Proof. Let M_1 be the graphic matroid defined as in Lemma 2.1 and let $M_2(i_1, \ldots, i_p) = (E, \mathcal{F}_2)$ be the partition matroid over the partition $S_1, \ldots, S_p$ with limits $i_1, \ldots, i_p$ respectively.

Let T be a maximum cardinality intersection of matroids M_1 and M_2 determined using the CI-algorithm [6]. T is an acyclic subgraph of G satisfying $(\forall j) T \cap S_j \leq i_j$. Using the similar arguments as in Lemma 2.1 the proof of this lemma follows. □

The algorithm for solving the problem (3) is thus straightforward:

Algorithm f-SpanningTree(p)

Input :	Graph $G = (V, E)$, partition of E to $S_1, \ldots, S_p$
Output :	f-optimal spanning tree T^{opt}.
K0 :	$T^{opt} := \emptyset,\ c^{opt} := \infty$
K1 :	**for each** $i_1, \ldots, i_p$, s.t. $\sum_{j=1}^{p} i_j = \lvert V\rvert - 1$ **do**
	begin
K2 :	$T = Check(i_1, \ldots, i_p)$
K3 :	**if** $\lvert T\rvert = \lvert V\rvert - 1$ & $f(T) < c^{opt}$ **then**
K4 :	$T^{opt} := T,\ c^{opt} = f(T)$
	end

Lemma 3.2 *Algorithm f-SpanningTree(p) runs in $O(m^2|V|^p)$ time.*

Proof. There are $\binom{|V| - 1 + p - 1}{p - 1} = O(|V|^{p-1})$ possibilities for expressing $|V| - 1$ in the form of sum of p integers $|V| - 1 = \sum_{j=1}^{p} i_j$ in step $K1$ of the algorithm (see e.g. [9]), thus there are $O(|V|^{p-1})$ invocations of $Check(i_1, \ldots, i_p)$ in step $K2$. The total complexity is then $O(|V|^{p-1}).O(m^2|V|) = O(m^2|V|^p)$. □

Remark 3.1 *The range of i_j in step $K1$ of the previous algorithm is limited to interval $[0, |S_j|]$. However, as a special case, cardinality of all sets S_j could be as close to $\frac{|E|}{p}$ as possible and thus for $p \leq \frac{|E|}{|V|-1}$ we have $|S_j| \geq |V| - 1$. Therefore $i_j \leq |S_j|$ is of no use in this special case and the number of iterations in step $K1$ remains $O(|V|^{p-1})$.*

4 A computational complexity improvement in case $p = 2$

Let us look closer at the complexity of determining the f-optimal spanning tree. The maximum cardinality matroid intersection, which can be performed in $O(R(m^2 + mRc(m)))$ steps, consists of $R \leq |V| - 1$ iterations, of complexity $O(m^2 + mRc(m))$ each. According to [6], each iteration consists of two steps:

Step 1: construction of the so-called Border Graph (BG) with complexity $O(mRc(m))$

Step 2: determining the so-called Augmenting Path in BG with complexity $O(m^2)$.

Each iteration increases the cardinality of matroid intersection I by 1. In *Step 1* for a given independent set I with $|I| \leq |R|$ and for each $e \in E - I$ independence of $I \cup \{e\}$ is determined and the unique cycle (in the sense of matroid theory) in $I \cup \{e\}$ is found, if it exists. This needs in case of graphic and partition matroids only $O(|V|)$ operations, provided that $|I| \leq |V| - 1$. In *Step 2* a search for Augmenting Path is performed in bipartite graph BG. Vertices of BG are exactly elements of base set E and edges are only between vertices of I and vertices of $E-I$, thus at most $|I|.|E-I| \leq (|V|-1).m$ edges are present in BG. Consequently, the search for Augmenting Path in BG can be performed in $O(|V|m)$ time.

The previous discussion summs up to the following lemma:

Lemma 4.1 *One iteration of CI-algorithm for graphic and partition matroid (as defined in lemma 2.1) can be done in $O(|V|m)$ time giving the overall complexity of the algorithm $O(|V|^2m)$.*

If we take a closer look at Lemma 2.1 and compare two checks, namely $Check(i,j)$ and $Check(i+1, j-1)$, we see, that they both operate on the same graphic matroid M_1 and two very similar partition matroids $M_2(i,j)$ and $M_2(i+1, j-1)$. Therefore it is immediate to try to use the result of $Check(i,j)$ in the computation of $Check(i-1, j+1)$. The complexity improvement that can be obtained in this way is described in the next lemma:

Lemma 4.2 *Let T_{ij} be the result of $Check(i,j)$ (as defined in Lemma 2.1). Then $Check(i-1, j+1)$ can be performed and its result $T_{i-1,j+1}$ can be found using at most 2 iterations of the maximum cardinality matroid intersection algorithm.*

Proof. Let us denote $T^*_{i-1,j+1} = T_{ij}$ in case $|T_{ij} \cap S_1| \leq i-1$. Otherwise $|T_{ij} \cap S_1 = i$ and there exists $e \in T_{ij} \cap S_1$; in this case let $T^*_{i-1,j+1} = T_{ij} - \{e\}$. Such set $T^*_{i-1,j+1}$ clearly belongs to the intersection of matroids M_1 and $M_2(i,j)$. Thus, let us start the intersection algorithm in $Check(i-1, j+1)$ with the initial intersection $T^*_{i-1,j+1}$. The cardinality of $T^*_{i-1,j+1}$ is at least $|T_{ij}|-1$ and the cardinality of $T_{i-1,j+1}$ is at most $|T_{ij}|+1$ from which it is immediate, that we need at most two iterations of intersection algorithm in $Check(i-1, j+1)$. □

As we can see, the algorithm f-spanning tree can be made faster by suitable ordering of $Check(i,j)$ calls and by reusing the result of previous $Check(i,j)$ calls. If we denote by $Check(i,j,T)$ the $Check(i,j)$ call where maximum matroid cardinality intersection starts with intersection T, we could formalize the faster version of the algorithm:

Algorithm f-SpanningTree(+)

Input :	Graph $G = (V, E)$, partition of E to S_1 and S_2.
Output :	f-optimal spanning tree T^{opt}.
K0 :	$T^{opt} := \emptyset,\ L^{opt} := \infty,\ T_{left} := \emptyset,\ T_{right} := \emptyset$
K1 :	**for** i **from** $\lfloor \frac{\vert V\vert-1}{2} \rfloor$ **to** 0 **do**
	begin
K2 :	$T_{left} := Check(i, \vert V\vert - 1 - i, T_{left})$
K3 :	**if** $\vert T_{left}\vert = \vert V\vert - 1$ **then**
K4 :	$T^{opt} := T_{left},\ L^{opt} = \vert V\vert - 1 - 2 * i$, STOP
K5 :	$T_{right} := Check(\vert V\vert - 1 - i, i, T_{right})$
K6 :	**if** $\vert T_{right}\vert = \vert V\vert - 1$ **then**
K7 :	$T^{opt} := T_{right},\ L^{opt} = \vert V\vert - 1 - 2 * i$, STOP
	end

Steps $K2$ and $K5$ are performed $\lfloor \frac{|V|-1}{2} \rfloor + 1$ times each. The first time they are performed they need $O(|V|^2 m)$ time (see Lemma 4.1) to compute the result of $Check(i, j, T)$, since $T = \emptyset$ in this case. However all subsequent calls of $Check(i, j, T)$ in steps $K2$ and $K5$ use the precomputed sets T_{left} and T_{right} and thus require just $O(|V|m)$ time (see Lemma 4.2). To sum up, algorithm f-SpanningTree(+) needs $O(|V|^2 m) + 2 * \left(\lfloor \frac{|V|-1}{2} \rfloor + 1\right) * O(|V|m)$ time for steps $K2$ and $K5$. The remaining steps are trivial, thus overall complexity of algorithm f-SpanningTree(+) is $O(|V|^2 m)$.

5 Further improvement

It might look promising to use some kind of binary search in step K1 of Algorithm f-SpanningTree to determine optimal i, j pair instead of invoking $Check(i, j)$ on all possible i, j pairs. However, this approach is of no use for finding the optimum spanning tree: after invocation of $Check(i, j)$ for some values of i and j exactly one of the following is true:

1. We have found a spanning tree T_{ij}. Thus L^{opt} is at most $|i - j|$
2. There is no spanning tree T_{ij} s.t. $|T_{ij}| = |i - j|$, implying $L_{opt} \neq |i - j|$. However, it is easy to see that for L^{opt} we may have $L^{opt} < |i - j|$ as well as $L^{opt} > |i - j|$.

The latter case makes binary search unapplicable.

Let us now look closer at the structure of (i, j) pairs for which a spanning tree T_{ij} exists. Let $i_{max} = \max\{i : \ T_{ij} \text{ is a spanning tree}\}$ and $j_{max} = \max\{j : \ T_{ij} \text{ is a spanning tree}\}$. To determine value of i_{max}, it is enough to determine the maximum forest F^1 of $G^1 = (V, S_1)$; Since G was assumed to be connected, the forest F^1, if not itself being a spanning tree of G, must be extendable by edges of S_2 to some spanning tree of G. i_{max} then equals to the number of edges of F^1 and corresponds to a spanning tree $T_{i_{max}, k-i_{max}}$. The value of j_{max} can be determined in the same way.

The following lemma shows that spanning trees $T_{i_{max}, k-i_{max}}$ and $T_{k-j_{max}, j_{max}}$ are sufficient to describe the structure of feasible (i, j) pairs:

Lemma 5.1 *Let $k - j \le i$ and $T_{i,k-i}$ and $T_{k-j,j}$ are spanning trees of G having $|T_{i,k-i} \cap S_1| = i$ and $|T_{k-j,j} \cap S_2| = j$. Then for each $l : \ k - j \le l \le i$ there exists a spanning tree $T_{l,k-l}$ of G having $|T_{l,k-l} \cap S_1| = l$.*

Proof. The statement trivially holds if $k - j = i$. Otherwise let e be any edge from $T_{k-j,j} - T_{i,k-i}$. $T_{i,k-i} \cup \{e\}$ contains unique cycle C_e and let f be any edge from $C_e -$

$T_{k-j,j}$. Then $T^{(1)} = T_{i,k-i} \cup \{e\} - \{f\}$ is also a spanning tree which has more edges in common with $T_{k-j,i}$ than $T_{i,k-i}$, more precisely $|T^{(1)} \cap T_{k-j,j}| = |T_{i,k-i} \cap T_{k-j,j}| + 1$. By repeating this construction we get a sequence Seq of spanning trees $Seq = \{T^{(0)} = T_{i,k-i}, T^{(1)}, T^{(2)}, \ldots, T^{(i-(k-j))} = T_{k-j,i}\}$. If we look at two consecutive spanning trees $T^{(x)}$ and $T^{(x+1)}$, cardinalities of $T^{(x)} \cap S_1$ and $T^{(x+1)} \cap S_1$ are either equal or differ by 1. Thus sequence Seq contains for each $l: \ k-j \le l \le i$ a spanning tree $T_{l,k-l}$ of G having $|T_{l,k-l} \cap S_1| = l$. □

Using the previous results we know that (i,j) pairs for which a spanning tree T_{ij} exists are exactly pairs $\{(l, k-l);\ k - j_{max} \le l \le i_{max}\}$. From that point it requires only a constant amount of time to determine the optimum pair (i^{opt}, j^{opt}) and the optimum value $|i^{opt} - j^{opt}|$ of problem (2). But, even if we know the optimum pair (i^{opt}, j^{opt}), to determine the optimum spanning tree $T_{i^{opt},j^{opt}}$ we need to call $Check(i^{opt}, j^{opt})$ once. The complexity of determining the optimum spanning tree is then $O(|V|^2 m)$, the same as of algorithm f-SpanningTree(+).

The algorithm we present finally is better than algorithm f-SpanningTree(+) in the sense that it determines the optimum value of problem (2) in $O(m+n)$ time and needs only one call of $Check(i,j)$ to determine the optimum spanning tree.

Algorithm f-SpanningTree(++)

Input : Graph $G = (V, E)$, partition of E to S_1 and S_2.
Output : optimum value c^{opt} and f-optimal spanning tree T^{opt}.
K0 : $k := |V|$
K1 : Find the maximum forest F^1 of $G^1 = (V, S_1)$; $i_{max} := |F^1|$
K2 : Find the maximum forest F^2 of $G^2 = (V, S_2)$; $j_{max} := |F^2|$
K3 : **if** $(i_{max} - (k - i_{max}))((k - j_{max}) - j_{max}) \le 0$ **then**
  $(i^*, j^*) := (\lfloor \frac{k}{2} \rfloor, \lceil \frac{k}{2} \rceil)$
 else if $|i_{max} - (k - i_{max})| \le |(k - j_{max}) - j_{max}|$ **then**
  $(i^*, j^*) := (i_{max}, k - i_{max})$
 else
  $(i^*, j^*) := (k - j_{max}, j_{max})$
K5 : $c^{opt} := |i^* - j^*|$, OUTPUT c^{opt}
K6 : $T^{opt} := Check(i^*, j^*)$ STOP.

We have postponed dealing with disconnected graphs until now. Disconnected case only requires small changes in presented algorithms: we are dealing with spanning forests instead of spanning trees. The cardinality of spanning forests is $|V| - c(G)$, where $c(G)$ is the number of connected components of graph G. Lemmas 2.1 and 3.1 are not influenced by disconnectedness since graphical matroid is defined in the same way on disconnected graphs. As a result, all complexity results stated before hold also in the disconnected case.

References

[1] I. Averbakh and O. Berman: *Categorized bottleneck/minisum path problems on networks*. Operation Research Letters 16, 1994, 291-297

[2] Š. Berežný, K. Cechlárová and V. Lacko: *Optimization problems on graphs with categorization of edges*. Proc. SOR 2001, Eds. V. Rupnik, L. Zadnik-Stirn, S. Drobne, Preddvor, Slovenia, 2001, 171-176

[3] Š. Berežný, K. Cechlárová and V. Lacko: *A polynomially solvable case of optimization problems on graphs with categorization of edges*. Proc. of MME'1999, Jindřichúv Hradec, 1999, 25-31

[4] Š. Berežný and V. Lacko : *Balanced problems on graphs with categorization of edges*. Discussiones Mathematicae Graph Theory 23, 2003, 5-21

[5] H. W. Hamacher and F. Rendl: *Color constrained combinatorial optimization problems*. Holt, Operation Research Letters 10, 1991, 211-219

[6] E. L. Lawler: *Combinatorial optimization: Networks and matroids*. Holt, Rinehart and Winston, New York, 1976

[7] A. P. Punnen: *Traveling salesman problem under categorization*. Operation Research Letters 12, 1992, 89-95

[8] M. B. Richey and A. P. Punnen: *Minimum weight perfect bipartite matchings and spanning trees under categorizations*. Discrete Appl. Math 39, 1992, 147-153

[9] K. H. Rosen, J. G. Michaels, K. Rosen: *Handbook of Discrete and Combinatorial Mathematics*. CRC Press, 1997