Vertex heaviest paths and cycles in quasi-transitive digraphs

Jørgen Bang-Jensen
Gregory Gutin*
Department of Mathematics and Computer Science
Odense University, Denmark

Abstract

A digraph D is called a quasi-transitive digraph (QTD) if for any triple x, y, z of distinct vertices of D such that (x, y) and (y, z) are arcs of D there is at least one arc from x to z or from z to x. Solving a conjecture by J. Bang-Jensen and J. Huang (*J. Graph Theory*, to appear), G. Gutin (*Australas. J. Combin.*, to appear) described polynomial algorithms for finding a Hamiltonian cycle and a Hamiltonian path (if it exists) in a QTD. The approach taken in that paper cannot be used to find a longest path or cycle in polynomial time. We present a principally new approach that leads to polynomial algorithms for finding vertex heaviest paths and cycles in QTD's with non-negative weights on the vertices. This, in particular, provides an answer to a question by N. Alon on longest paths and cycles in QTD's.

1 Introduction

A digraph D is called *quasi-transitive* if for any triple x, y, z of distinct vertices of D such that (x, y) and (y, z) are arcs of D there is at least one arc from x to z or from z to x. A digraph D is *semicomplete* if, for every pair of vertices of D, there is at least one arc between them.

Quasi-transitive digraphs (QTD's) were introduced by Ghouilà-Houri [4] and have been studied in [1, 7, 8, 9]. Bang-Jensen and Huang [1] characterized those quasi-transitive digraphs that have a Hamiltonian cycle (Hamiltonian path, respectively). They noted that their theorems do not seem to

*This work was supported by the Danish Research Council under grant no. 11-0534-1. The support is gratefully acknowledged.

imply polynomial algorithms and conjectured that there exist polynomial algorithms for solving the Hamiltonian path and cycle problems for QTD's on n vertices. A positive answer to this conjecture was given in [7], where $O(n^4)$-time algorithms were described.

The approach taken in [7] cannot be used to find a longest path or cycle in polynomial time. N. Alon (personal communication, 1993) posed the following question: do there exist polynomial algorithms for finding a longest path and a longest cycle in a quasi-transitive digraph? In this paper, we present a principally new approach that leads to $O(n^5)$-time algorithms for finding vertex heaviest paths and cycles in QTD's with non-negative weights on the vertices (here the weight of a path or cycle is the sum of the weights of its vertices). In particular, one can solve the longest path and cycle problems as well as the X-cyclicity problem (find a cycle through a given set X of vertices, if one exists) for QTD's in time $O(n^5)$.

2 Totally Φ-decomposable digraphs

The terminology is rather standard, generally following [3]. By a *cycle (path)* we mean a simple directed cycle (path, respectively). A *trivial digraph* is a digraph without arcs. Let R be a digraph on r vertices $v_1, ..., v_r$ and let $H_1, ..., H_r$ be a disjoint collection of digraphs. Then $D = R[H_1, ..., H_r]$ is the new digraph obtained from R by replacing each vertex v_i of R by H_i and adding an arc from every vertex of H_i to every vertex of H_j if and only if (v_i, v_j) is an arc of R $(1 \leq i \neq j \leq r)$. If each of $H_1, ..., H_r$ is trivial, D is called an *extension* of R. In particular, a digraph D is an *extended semicomplete digraph* (ESD, for short) if there exists a semicomplete digraph R such that $D = R[H_1, \ldots, H_r]$, where each H_i is an independent set of vertices (possibly of size 1).

Let Φ be a set of digraphs containing the trivial digraph with one vertex. Then Φ^{ext} denotes the set of all extensions of digraphs in Φ. A digraph D is called totally Φ-*decomposable* if either D has only one vertex, or there is a decomposition $D = R[H_1, ..., H_r]$, $r \geq 2$ so that $R \in \Phi$ and each of $H_1, ..., H_r$ is totally Φ-*decomposable*. In this case, the decomposition $D = R[H_1, ..., H_r]$, appropriate decompositions $H_i = R_i[H_{i1}, ..., H_{ir_i}]$ of all H_i except trivial ones on one vertex, appropriate decompositions of all H_{ij} except trivial ones of order 1, and so on, form a *total* Φ-*decomposition* of D.

Let Ψ be the union of all acyclic and all semicomplete digraphs. One of the basic results we use is the following weakening of a decomposition

theorem from [1].

Theorem 2.1 *Every QTD D on n vertices is totally Ψ-decomposable. One can find a total Ψ-decomposition of D in time $O(n^3)$.*

From now on, assume that every digraph D we consider has non-negative weights $w(.)$ on the vertices. The weight $w(H)$ of a subgraph of D is the sum of the weights of its vertices. A *k-path subgraph* of D is a collection of k vertex disjoint paths of D. For a positive integer k, the symbol $w_k(D)$ denotes the weight of a heaviest k-path subgraph of D, i.e. one with the maximum weight among all k-path subgraphs. For convenience we define $w_0(D) = 0$. We consider the following problem which we call *the HPS problem*: Given a digraph D on n vertices, find a heaviest k-path subgraph of D for every $k = 1, 2, ..., n$.

Theorem 2.2 *Let Φ be a set of digraphs including the trivial digraph on one vertex. Suppose that $\Phi = \Phi^{ext}$ and, for every $D \in \Phi$ on n vertices,*

$$w_{k+1}(D) - w_k(D) \leq w_k(D) - w_{k-1}(D), \tag{1}$$

where $k = 1, 2, ..., n-1$. If there is a constant $s \geq 2$ so that, for every $L \in \Phi$, the HPS problem can be solved in time $O(|V(L)|^s)$, then, for every totally Φ-decomposable digraph D, the HPS problem can be solved in time $O(|V(D)|^{s+1})$, provided we are given a total Φ-decomposition of D.

Proof: Let $D = R[H_1, ..., H_r]$ be a decomposition of D, where $R \in \Phi$ and H_i is totally Φ-decomposable and has n_i vertices ($i = 1, ..., r$). Set $D_0 = R[E_1, ..., E_r]$, where E_i is the trivial digraph on n_i vertices. Assign new weights to the vertices of D_0 as follows. The i'th vertex of E_j has the weight

$$\widetilde{w}_{ij} = w_i(H_j) - w_{i-1}(H_j), \quad j = 1, ..., r; \ i = 1, ..., n_j.$$

We show that, given solutions of the HPS problem for $H_1, ..., H_r$ and D_0, one can easily construct a solution of the HPS problem for D. This will lead to a recursive algorithm as desired.

Let F_k be a heaviest k-path subgraph of D_0 and let F_k contain m_j vertices of E_j ($j = 1, ..., r$). By (1), $\widetilde{w}_{ij} \geq \widetilde{w}_{qj}$ whenever $q > i$. Therefore, we can always change the vertices of F_k so that F_k contains precisely the first m_j vertices of E_j for each $j = 1, ..., r$. Assume now that this is the case. Now, for each $j = 1, ..., r$, replace the vertices of E_j in F_k by a heaviest m_j-path

subgraph of H_j. This replacement provides a k-path subgraph T_k of D. It is easy to check that

$$\widetilde{w}(F_k) = \sum_{j=1}^{r} \sum_{i=1}^{m_j} \widetilde{w}_{ij} = \sum_{j=1}^{r} w_{m_j}(H_j) = w(T_k) \leq w_k(D).$$

So, the weight of a heaviest k-path subgraph of D_0 is at most $w_k(D)$. Analogously, starting with a heaviest k-path subgraph of D, one can prove that the weight of a heaviest k-path subgraph of D_0 is at least $w_k(D)$. Therefore, T_k is a heaviest k-path subgraph of D.

The arguments above lead to the following recursive algorithm called $\mathcal{GA}$.

1. Find $D = R[H_1, ..., H_r]$ using the total Φ-decomposition of D.
2. Solve the HPS problem for $H_1, ..., H_r$ using $\mathcal{GA}$.
3. Form D_0 (with the weights $\widetilde{w}_{ij}$) and solve the HPS problem for D_0 using an $O(|V(D)|^s)$-time algorithm. Change the solutions F_k (if it is necessary) so that each of F_k contains the first vertices of E_j without 'blanks', for each $j = 1, ..., r$.
4. Using the solutions obtained in Step 2, transform every F_k into a k-path subgraph T_k of D as in the discussion above.

It is easy to check that the complexity of Algorithm $\mathcal{GA}$ is $O(|V(D)|^{s+1})$. □.

3 Main results

Theorem 3.1 *For a QTD D on n vertices, the following two problems can be solved in time $O(n^5)$:*

1. *For every $k = 1, 2, ..., n$, find a heaviest k-path subgraph of D.*
2. *Find a heaviest cycle of D.*

By Theorems 2.1 and 2.2, we can prove the first part of Theorem 3.1 by showing that every digraph $D \in \Psi^{ext}$ satisfies the conditions of Theorem 2.2 with $s = 4$. This will be done later using some results on both extended semicomplete digraphs and flows in networks. We start with theorems on

flows in networks. We use standard terminology on the topic, the undefined terms can be found in [10].

We consider a network $N = (V, A, s, t, b)$ with source s, sink t and a capacity function b on the arc set A. A flow f on N is called *integral* if, for every arc $a \in A$, $f(a)$ is a non-negative integer. A *circulation* is a flow with value 0. A circulation is a *cycle flow* if the digraph induced by the arcs with non-zero flow is just a (directed) cycle. The following observation can be easily proved analogously to Theorem 7.2 in [2].

Proposition 3.2 *Every integral flow of value $k \geq 0$ can be decomposed into k flows of value 1 along (s,t)-paths and a number of cycle flows. Such a decomposition of f can be found in time $O(\sum_{a \in A} f(a))$, where $f(a)$ is the number of units of f along an arc a.*

A flow f in N is *feasible* if $0 \leq f(a) \leq b(a)$ for every arc $a \in A$. Below we always assume that we are dealing with a feasible flow in N.

Assume now that, besides the network N, we have a non-negative real valued cost function c on A. The *cost* $c(f)$ of a flow f is defined by the relationship $c(f) = \sum_{a \in A)} c(a)f(a)$. A flow f of value k is a *minimum cost* (*maximum cost*) flow of value k if its cost is minimum (maximum) among all flows of value k in N. Below we present a few results on maximum cost flows. They can all be derived in an analogous way to their counterparts for minimum cost flow (see e.g. pages 109-110 in [11]). We recall the definition of the *residual network* $N(f) = (V, A(f), s, t, \kappa)$. Here $A(f)$ consists of the following arcs:

(a) If $(u, v) \in A$ and $f(u, v) < b(u, v)$, then $(u, v) \in A(f)$ and $\kappa(u, v) = b(u, v) - f(u, v)$.

(b) If $(u, v) \in A$ and $f(u, v) > 0$, then $(v, u) \in A(f)$ and $\kappa(v, u) = f(u, v)$.

Proposition 3.3 *A flow f with value k in a network N has maximum cost among the flows with value k if and only if the residual network $N(f)$ has no cycles of positive cost.*

Corollary 3.4 *Let N be a network in which the value of a maximum flow from s to t is m and let f_i denote a maximum cost flow of value i in N, $i = 1, 2, \ldots, m$. For all $k = 1, 2, \ldots, m-1$ we have*

$$c(f_{k+1}) - c(f_k) \leq c(f_k) - c(f_{k-1}). \tag{2}$$

For a flow f in a network N and a (t,s)-path $P = v_1v_2\dots v_p$, $v_1 = t$, $v_p = s$, in $N(f)$, we denote by ϵ_P the function defined on $A(N)$ as follows:

$$\epsilon_P(a) = \begin{cases} 1, & \text{if there is an } i \in \{1,...,p-1\} \text{ such that } a = (v_i, v_{i+1}) \text{ and } a \in N, \\ -1, & \text{if there is an } i \in \{1,...,p-1\} \text{ such that } a = (v_{i+1}, v_i) \in N \text{ but } (v_i, v_{i+1}) \notin N, \\ 0, & \text{otherwise.} \end{cases}$$

Proposition 3.5 *Let f be a maximum cost flow with positive value k in a network N and let P be a (t,s)-path of maximum cost in the residual network $N(f)$. Then $f + \epsilon_P$ (here we mean arc-wise addition) is a maximum cost flow with value $k-1$ in N.*

Proposition 3.5 can be proved analogously to Theorem 8.12 in [11].

In our applications, we consider networks with costs and capacities only on the vertices. We can easily transform any such a network N' to a network N'' with costs and capacities on the arcs: The vertex and arcs sets of N'' are $\{v^-, v^+ : v \in V(N')\}$ and $\{(u^+, w^-) : (u,w) \in A(N')\} \cup \{(v^-, v^+) : v \in V(N')\}$. All arcs of the kind $(u^+, w^-), u \neq w$ have cost zero and infinite capacity in N''. The arcs (v^-, v^+) have cost and capacity equal to the cost and capacity of v in N'. It is easy to see that the problem of finding a maximum cost flow in N' is equivalent to the problem of finding a maximum cost flow in N''.

Let D be a digraph with vertex set $\{v_1,...,v_n\}$, m arcs and with a given nonnegative cost $c(v_i) = w(v_i)$ for each vertex v_i. Construct a transport network N_D as follows. Add a zero-cost source s and a zero-cost sink t to D. For each vertex v_i of D, we add the arcs (s, v_i) and (v_i, t). Finally, we assign capacity one to each vertex of D and capacity n to s and t. Using Propositions 3.3 and 3.5 we can easily construct an $O(n^2m)$-time algorithm (which will be called $\mathcal{FA}$) for finding maximum cost flows $f_0,...,f_n$ of values $0,...,n$ in N_D:

Let f_n be the flow with value n, obtained by sending one flow unit along each of the paths sv_it, $i = 1,...,n$. Obviously, f_n is a maximum cost flow with value n. We start with this flow. Using Proposition 3.5 we can find $f_{n-1},...,f_0$. To find f_k we construct a maximum cost path from t to s in $N(f_{k+1})$. Such a path can be obtained in time $O(nm)$ using an implementation of Dijkstra's algorithm described in [11], p. 93 (we just change the sign of the costs and find a minimum cost path in a digraph without negative cost cycles).

Now we consider some properties of paths and cycles in extended semicomplete digraphs which were obtained in [5, 6]. A collection F of vertex disjoint paths and cycles of a digraph D is called a *k-path-cycle subgraph* of D if F has exactly k paths. For a non-negative integer k and a given weight function w on $V(D)$, $w'_k(D)$ denotes the weight of a heaviest k-path-cycle subgraph of D with respect to w.

Theorem 3.6 *Let D be an extended semicomplete digraph on n vertices.*

1. *For every 1-path-cycle subgraph F of D, there exists a path P so that $V(P) = V(F)$. Given F, one can construct P in time $O(n^2)$.*

2. *If D is strong, then, for every 0-path-cycle subgraph Z of D, there is a cycle C so that $V(C) = V(Z)$. Given Z, one can construct C in time $O(n^2)$.*

Proof of the first part of Theorem 3.1: Consider a digraph $D \in \Psi^{ext}$ on n vertices. We show that D satisfies the conditions of Theorem 2.2 with $s = 4$.

For every $k = n, ..., 1$, construct a maximum cost flow f_k of value k in the network N_D in time $O(n^4)$ (using Algorithm $\mathcal{FA}$). By Proposition 3.2, every f_k is the sum of k flows of value 1 along paths from the source to the sink and a number of cycle flows. Hence, f_k provides a collection of k paths and a number of cycles such that the paths and the cycles have no common vertices, except the source and the sink of the network. Moreover, by the definition of N_D, none of the cycles contain the source or the sink. Deleting the source and the sink from N_D with the flow f_k, we get k paths and some cycles in D which are vertex disjoint and have total cost $c(f_k)$. It follows from the definition of N_D and the fact f_k is a maximum cost flow in N_D that the paths and the cycles in D form a heaviest k-path-cycle subgraph L_k in D. In particular, $c(f_k) = w'_k(D)$ for every $k = 1, ..., n$.

If D is an ESD then, by the first part of Theorem 3.6, for every $k = 1, ..., n$, we can construct a k-path subgraph Q_k so that $V(Q_k) = V(L_k)$. If D is acyclic then just let $Q_k = L_k$. Obviously, Q_k is a heaviest k-path subgraph of D. Note that $Q_1, ..., Q_n$ can be found in time $O(n^4)$.

Since $w_k(D) = w'_k(D) = c(f_k)$, it follows from (2) in Corollary 3.4 that (1) holds. □.

Proof of the second part of Theorem 3.1: Let D be a strong QTD on $n \geq 2$ vertices and let $D = R[H_1, ..., H_r]$, where R is semicomplete, $H_1, ..., H_r$ are QTD's and $r \geq 2$. (If D is not strong, then we consider the strong components of D one by one.) We claim that D has a heaviest cycle C containing vertices from more than one of the digraphs $H_1, ..., H_r$. Indeed, let C' be a heaviest cycle of D completely contained in a H_i. Since D is strong, there is a path in D, of length at least 2, starting at a vertex x of C', terminating at a vertex y of C' and containing no other vertices from H_i. Hence, by the definition of $R[H_1, ..., H_r]$, there is a path of length at least 2, starting at x, terminating at the succesor x' of x (in C') and containing no other vertices from H_i. Clearly, the last path and C' minus the arc (x, x') form a cycle as desired. Now it is easy to see the correctness of the following algorithm for finding a heaviest cycle of D. Note that our approach finds a heaviest cycle C which contains vertices from at least two H_i's. By the remark above this is also a heaviest cycle of D.

1. Solve the HPS problem for $H_1, ..., H_r$ using Algorithm $\mathcal{GA}$.
2. Form D_0 with the weights $\widetilde{w}_{ij}$, as in the proof of Theorem 2.2, and the network N_{D_0}.
3. Using Algorithm $\mathcal{FA}$, construct a maximum cost circulation f_0 in N_{D_0}. Deleting the source and sink of N_{D_0}, form a heaviest 0-path-cycle subgraph Z of D_0.
4. Using an $O(n^2)$-time algorithm of the second part of Theorem 3.6, construct a heaviest cycle C of D_0.
5. Using the solutions of Step 1 and the cycle C, form a heaviest cycle of D (analogously to what we did in the proof of Theorem 2.2).

Clearly, the complexity of this algorithm is $O(n^4)$. □.

Theorem 3.1 implies

Corollary 3.7 *For a QTD D on n vertices, the following problems can be solved in time $O(n^5)$.*

1. *Find a longest path of D.*
2. *Find a longest cycle of D.*

3. *For a set $X \subset V(D)$, check if D contains a cycle through X and construct one (if it exists).*

□.

References

[1] J. Bang-Jensen and J. Huang, Quasi-transitive digraphs. *J. Graph Theory* **20** (1995) 141-161.

[2] R.G. Busacker and T.L. Saaty, *Finite Graphs and Networks.* (McGraw Hill, N.Y.,1965).

[3] J. A. Bondy and U.R.S. Murty *Graph Theory with Applications*, (North Holland, N. Y., 1976).

[4] A. Ghouilà-Houri, Caractérisation des graphes non orientès dont on peut orienter les arrêtes de maniere à obtenir le graphe d'un relation d'ordre, *C.R.Acad. Sci.Paris* **254** (1962) 1370-1371.

[5] G. Gutin, Finding a longest path in a complete multipartite digraph, *SIAM J. Discrete Math.*, **6** (1993) 270-273.

[6] G. Gutin *Paths and cycles in digraphs.* Ph.D thesis, Tel Aviv Univ., 1993.

[7] G. Gutin, Polynomial algorithms for finding Hamiltonian paths and cycles in quasi-transitive digraphs. *Australas. J. Combin.* **10** (1994) 231-236.

[8] P. Hell and J. Huang, Lexicographic orientation and representation algorithms for comparability graphs, proper circular graphs, and proper interval graphs, submitted.

[9] J. Huang, *Tournament-like oriented graphs.* Ph.D. thesis, Simon Fraser Univ., 1992.

[10] C.H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity* (Prentice-Hall, N.J., 1982).

[11] R.E. Tarjan, *Data Structures and Network Algorithms*, (SIAM, Phil., 1983).