Note

On the $k$-path partition of graphs

George Steiner∗

Management Science and Information Systems Area, McMaster University, 1280 Main Street West, L8S 4K1 Hamilton, Ont., Canada

Received 23 October 2001; received in revised form 19 April 2002; accepted 17 June 2002
Communicated by G. Ausiello

Abstract

The $k$-path partition problem is to partition a graph into the minimum number of paths, so that none of them has length more than $k$, for a given positive integer $k$. The problem is a generalization of the Hamiltonian path problem and the problem of partitioning a graph into the minimum number of paths. The $k$-path partition problem remains NP-complete on the class of chordal bipartite graphs if $k$ is part of the input, and we show that it is NP-complete on the class of comparability graphs even for $k = 3$. On the positive side, we present a polynomial-time solution for the problem, with any $k$, on bipartite permutation graphs, which form a subclass of chordal bipartite graphs.

1. Introduction

Consider a simple graph $G = (V, E)$. A path partition of $G$ is a collection of vertex-disjoint paths $P_i = (V_i, E_i), \ldots, P_r = (V_r, E_r)$ in $G$ whose union is $V$, i.e., $V_i \cap V_j = \emptyset$ for $i \neq j$ and $\bigcup_{i=1}^{r} V_i = V$. The path partition problem is to determine the minimum number of paths in a path partition of $G$, denoted by $p(G)$. Naturally, $p(G) = 1$ if and only if $G$ has a Hamiltonian path. Thus, finding $p(G)$ is NP-complete on all graph classes on which the Hamiltonian path problem is NP-complete, including planar graphs, bipartite graphs and chordal graphs [8]. On the other hand, there are many known special graph classes on which the path partition problem is solvable in polynomial time [1,2,5,6,9–11,14,15,18].

© 2002 Elsevier Science B.V. All rights reserved.
We study the following natural generalization of the path partition problem. A path partition is called a \(k\)-path partition if each path has at most \(k\) vertices, for a given integer \(k\). The \(k\)-path partition problem is to determine the minimum number of paths in a \(k\)-path partition of a graph \(G\), denoted by \(p_k(G)\). It is a natural graph problem, which has applications in broadcasting in computer and communications networks [17]. It is also applicable in vehicle routing problems in which the vertices represent customers, each path corresponds to a route for a vehicle and \(k\) is the maximum number of customers a single vehicle can service. Then \(p_k(G)\) is the minimum number of vehicles/routes needed to service all customers. It is easy to see that for \(k=2\) the problem is equivalent to finding a maximum matching of \(G\). For \(k=3\), however, the problem becomes NP-complete [7] on general graphs. As it is customary for such problems, researchers study special classes of graphs for which the problem becomes polynomially solvable. There are only a few such classes known however. Yan et al. [17] gave a polynomial-time algorithm for finding \(p_k(G)\) if \(G\) is a tree. Steiner [16] showed that the problem is NP-complete even on cographs if \(k\) is considered to be part of the input, but it is polynomially solvable if \(k\) is fixed.

The paper is organized as follows. In Section 2 we discuss how the \(k\)-path partition problem is NP-complete on chordal bipartite graphs if \(k\) is considered to be part of the input, and we prove that the problem remains NP-complete for comparability graphs even for \(k=3\). In contrast, in Section 3 we present a polynomial-time solution for the problem on bipartite permutation graphs, a subclass of chordal bipartite graphs. The last section contains our concluding remarks.

2. Complexity

A bipartite graph is chordal bipartite if it has no chordless cycle of length greater than 4. In spite of their well-accepted name, chordal bipartite graphs are not chordal graphs, since they may contain a chordless 4-cycle. Müller [12] proved that the Hamiltonian path problem remains NP-complete on chordal bipartite graphs. Since this is a special case of the \(k\)-path partition problem, when \(k=n\), the \(k\)-path partition problem is also NP-complete on chordal bipartite graphs. We note that it is possible to prove that finding \(p_k(G)\) for a chordal bipartite \(G\) is also NP-complete for \(k<n\), but we omit the details.

The following result shows that \(k\)-path partition is hard on comparability graphs even for \(k=3\). Comparability graphs are graphs which have a transitive orientation [3].

**Theorem 1.** The \(k\)-path partition problem is NP-complete on comparability graphs for \(k=3\).

**Proof.** To prove the theorem, we use a modified construction of Shum and Trotter [13], where they showed that it is NP-complete to determine the minimum-cardinality partition of a partially ordered set into chains of length 3. We reduce the NP-complete problem EXACT COVER BY 3-SETS (X3C) to our problem.
EXACT COVER BY 3-SETS (X3C)

Instance: A set $S'$ with $3p'$ elements, $p' \in \mathbb{Z}^+$, a collection $C = \{c_1, c_2, \ldots, c_{q'}\}$ of 3-element subsets of $S'$, $q' \in \mathbb{Z}^+$.

Question: Can $S'$ be partitioned into members of $C$?

We define a graph $G = (V, E)$ to construct an instance of the $k$-path partition problem with $k = 3$. Let $c_i = \{u_i, v_i, w_i\}$ for $1 \leq i \leq q'$ and associate the subgraph $G_i = (V_i, E_i)$ shown in Fig. 1 with $c_i$. Each $G_i$ has 12 vertices: the original elements $u_i, v_i, w_i$ and nine auxiliary vertices denoted by $z_{ij}$ for $1 \leq j \leq 9$. $G$ is the union of these subgraphs, i.e., $V = V_1 \cup V_2 \cup \cdots \cup V_{q'}$ and $E = E_1 \cup E_2 \cup \cdots \cup E_{q'}$. Thus $|V| = 3p' + 9q'$. Note that $G$ has a transitive orientation: Direct all edges incident with any element of $\{z_{i2}, z_{i5}, z_{i8}\}$ towards this element and the remaining edges incident with $z_{i6}$ towards this vertex in each $G_i$. Thus $G$ is a comparability graph.

We claim that $S'$ can be partitioned into members of $C$ if and only if $G$ can be partitioned into $p = p' + 3q'$ paths of length at most 3.

Since $|V| = 3p$, if $G$ can be partitioned into $p$ paths of length at most 3, then each of these paths must have exactly 3 vertices on it. Observe also that none of these paths can contain more than one of the original elements of $S'$. Furthermore, if a path contained auxiliary vertices from a $G_i$ and a $G_l$, $i \neq l$, then it would have to contain one of $\{z_{i2}, z_{i5}, z_{i8}\}$ as an endpoint. This however would leave one of the pendant vertices $\{z_{i1}, z_{i4}, z_{i7}\}$ disconnected from the rest of the graph after the removal of this endpoint, contradicting the assumption that even these pendant vertices must be on paths of length 3. In summary, this means that each $G_i$ must be partitioned into 3-paths in one of the two ways shown in Figs. 2 and 3.

If a $G_i$ is partitioned as in Fig. 2, i.e., into the paths $u_i z_{i2} z_{i4}, u_i z_{i5} z_{i7}, u_i z_{i8} z_{i7}, z_{i3} z_{i6} z_{i9}$, then let $c_i$ be included in the corresponding 3-cover, and if $G_i$ is partitioned as in Fig. 3, then do not include $c_i$ in the corresponding 3-cover. Since the path partitioning covers all of $G$, the thus selected $c_i$-s will form an exact 3-cover of $S'$.

Conversely, if $S$ has an exact 3-cover, then for each $c_i$ in the cover, partition the corresponding $G_i$ as in Fig. 2, and partition every other $G_i$ as in Fig. 3 to obtain a 3-path partition of $G$. □
3. Bipartite permutation graphs

A bipartite graph $G = (X, Y; E)$ is a bipartite permutation graph if it is both a comparability graph and the complement of a comparability graph, i.e., a cocomparability graph [3]. An alternative and very useful characterization uses the concept of strong ordering of the two parts: A strong ordering of the vertices of a bipartite graph $G = (X, Y; E)$ is an ordering $\{x_1, x_2, \ldots, x_r\}$ of the vertices in $X$ and an ordering $\{y_1, y_2, \ldots, y_s\}$ of the vertices in $Y$ such that whenever $x_i y_l, x_j y_m \in E$ with $i < j$ and $l > m$ then we also have $x_i y_m, x_j y_l \in E$. Spinrad et al. [4] have shown that a bipartite graph $G = (X, Y; E)$ is a bipartite permutation graph if and only if it has a strong ordering of its vertices and gave a linear-time recognition algorithm, which also finds a strong ordering. We will assume for the remainder of the paper that a bipartite permutation graph $G = (X, Y; E)$ is always given together with $X = \{x_1, x_2, \ldots, x_r\}$ and $Y = \{y_1, y_2, \ldots, y_s\}$, a strong ordering of its vertices. Any edges $x_i y_l, x_j y_m \in E$ with $i < j$ and $l > m$ will be called crossing edges. Two paths $P_1$ and $P_2$ are called crossing.
if they contain crossing edges. The strong ordering defines a left-to-right ‘direction’ for every permutation graph. It also makes the concepts of leftmost and rightmost edge well-defined in the natural sense. For the remainder of this section we automatically assume that when we talk about a path we traverse its vertices in a left-to-right direction consistent with the strong ordering. In some cases, however, we will need to orient a path \( P \) in the reverse direction which we denote by \( \overrightarrow{P} \). We use the notation \( uPv \) or \( v \overrightarrow{P} u \) to denote the subpath between the vertices \( u \) and \( v \) on the path \( P \) and \( \overrightarrow{P} \), respectively. When one of these vertices is missing it means that we take the entire path in that direction, e.g., \( uP \) refers to the subpath from \( u \) to the end of \( P \).

The next lemma plays a crucial role in our polynomial-time solution for the \( k \)-path partition problem in bipartite permutation graphs.

**Lemma 2.** If \( P_1 \) and \( P_2 \) are two crossing paths in a \( k \)-path partition of a bipartite permutation graph \( G = (X, Y; E) \), then there exist two vertex-disjoint, crossing-free paths \( P'_1 \) and \( P'_2 \) which cover the same set of vertices in \( G \) and whose length is at most \( k \).

**Proof.** Let \( V(P_1) = \{X_1, Y_1\} \) and \( V(P_2) = \{X_2, Y_2\} \) be the set of vertices on \( P_1 \) and \( P_2 \), respectively. The strong ordering of \( X \cup Y \) induces a strong ordering on \( X_1 \cup X_2 \cup Y_1 \cup Y_2 \). Let \( a_1b_1 \) and \( a_2b_2 \) be the leftmost edges of \( P_1 \) and \( P_2 \), respectively, which cross each other. Similarly, let \( c_1d_1 \) and \( c_2d_2 \) be the rightmost edges of \( P_1 \) and \( P_2 \), respectively, which cross each other. By the strong-ordering property we must have \( a_1 \neq c_1, d_2 \neq b_2 \).

This however implies that \( C = a_2a_1d_1d_2 \) is a cycle. Let \( X_C \cup Y_C \) denote the set of vertices spanned by this cycle. Then \( C \) is a Hamiltonian cycle in the induced subgraph \( G[XC \cup YC] \). Brandstädter et al. [4] showed that a bipartite permutation graph \( G = (X, Y; E) \) has a Hamiltonian cycle if and only if \( x_iy_ix_{i+1}y_{i+1} \) is a four-cycle for \( 1 \leq i \leq |X| = |Y| \). If \( X_C = \{x_1, x_2, \ldots, x_{l+1}\} \) and \( Y_C = \{y_1, y_2, \ldots, y_{l+1}\} \) in the strong ordering induced on \( X_C \cup Y_C \), then by the result of Brandstädter et al. \( x_i, y_i, x_{i+1}, y_{i+1} \) is a cycle of length four for \( 1 \leq i \leq j \leq l \). By the definition of \( a_1b_1 \) and \( a_2b_2 \), only one of \( P_1 \) and \( P_2 \) can have any edge to the left of these two edges. Let this path (if any) be denoted by \( P_r \). Similarly, only one of \( P_1 \) and \( P_2 \) can have any edge to the right of the edges \( c_1d_1 \) and \( c_2d_2 \). Denote this path (if any) by \( P_s \). Assume without the loss of generality that \( P_r \) enters the set \( X_C \cup Y_C \) through vertex \( x_i \) and it has \( p \) points before \( x_i \). Similarly, assume without the loss of generality that \( P_s \) leaves the set \( X_C \cup Y_C \) through vertex \( y_j \). Let \( P'_1 = P_rx_1y_1x_2y_2 \cdots x_{l−p+2}y_{l−p+2} \) if \( k − p \) is even and \( P'_1 = P_rx_1y_1x_2y_2 \cdots x_{l−p+1}y_{l−p+1} \) if \( k − p \) is odd. We cover the remaining vertices by \( P'_2 = P_s_1y_1x_1y_{i−1}x_{j−1} \cdots y_{i−p+2}x_{j−p+2} \) when \( k − p \) is even and by \( P'_2 = P_s_1y_1x_1y_{i−1}x_{j−1} \cdots y_{i−p+2}x_{j−p+2} \) when \( k − p \) is odd. It is clear that these two paths satisfy the claims of the lemma. \( \square \)

Lemma 2 leads to a crucial simplification in our search for a minimum \( k \)-path partition in a bipartite permutation graph. By the repeated application of the lemma, we obtain the following.
Corollary 3. Let $\mathcal{P} = \{P_1, P_2, \ldots, P_t\}$ be a minimum partition of $G$ into vertex-disjoint $k$-paths, then there is a crossing-free partition $\mathcal{P}' = \{P'_1, P'_2, \ldots, P'_t\}$ of $G$ into vertex-disjoint $k$-paths.

Let us call a path contiguous if it is of the form $u_iv_{i+1}v_ju_{i+1} \ldots$, where the $u$–s are consecutive vertices in the strong ordering of $X$ or $Y$ and the $v$–s are consecutive vertices from the other part.

Lemma 4. Let $\mathcal{P} = \{P_1, P_2, \ldots, P_t\}$ be a minimum crossing-free partition of $G$ into vertex-disjoint $k$-paths, then there is a partition $\mathcal{P}' = \{P'_1, P'_2, \ldots, P'_t\}$ of $G$ into contiguous vertex-disjoint $k$-paths.

Proof. Consider a non-contiguous path $P_i$. This means that $P_i$ must contain a segment $u_iv_ju_f$, where $f > i+1$. However, the missed vertex $u_{i+1}$ must be covered by a $P_g \in \mathcal{P}$. But since $P_g$ does not cross $P_i$ by assumption, $P_g$ must consist of the single point $u_{i+1}$. Since $G$ is connected, $u_{i+1}$ must be connected to some $v$ in the other part. The edge $u_{i+1}v$ must either cross one of $\{u_iv_j, v_ju_f\}$ or $v = v_j$. In either case, we have $u_{i+1}v_j \in E$. Replace $P_i$ and $P_g$ in $\mathcal{P}$ by $P'_i = P_iuv_{i+1}$ and $P'_g = u_{i+1}P_i$. (The situation can be demonstrated in Fig. 4 with $k = 5$, $P_1 = P_1x_1y_1x_2y_2x_4$, $P_2 = P_2x_3$ and $P_3 = y_3x_5y_4x_6y_5$. Here $u_i = x_2$, $u_f = x_4$, $v_j = y_2$ and $P_i$ misses $u_{i+1} = x_3$, but defining $P'_i = P_iv_{i+1} = x_1y_1x_2y_2x_4$ and $P'_g = u_{i+1}P_1 = x_3$ yields a contiguous set of paths.) By the repeated application of this argument, we obtain the desired path partition $\mathcal{P}'$ of the lemma.

If we have a collection of vertex-disjoint contiguous paths, then we will assume without loss of generality that they are ordered from left to right consistent with the strong ordering of the vertices in them. This way the last vertex of the collection is well-defined, simply meaning the last vertex on the last path. It was shown in [15] that a minimum path partition—without the upper bound $k$ on the path lengths—of a bipartite permutation graph can be obtained by starting from both $x_1$ and $y_1$, choosing a longest contiguous path, deleting its vertices and repeating this greedy choice in the remaining graph until the whole graph is covered. This is no longer true for the $k$-path partition problem, as the example in Fig. 4 shows: If $k = 6$, then always choosing the longest contiguous path with length less than or equal to 6 yields $P_1 = x_1y_1x_2y_2x_3y_3$, $P_2 = x_4$ and $P_3 = x_5y_4x_6y_5$. On the other hand, $P'_1 = x_1y_1x_2y_2x_3$ and $P'_2 = x_4y_3x_5y_4x_6y_5$ is a valid partition into two paths. Thus, how far we can extend a partial contiguous
and it may be benecial to save the endpoint of a path—\(y_3\) in the example—to serve as a connector between the next vertex on the other side—\(x_4\) in the example—which otherwise may become isolated from the rest of the graph. This of course makes a difference only if the vertex which would become isolated cannot be included in the preceding path because this would make that path too long (in the example \(x_4\) could not be added to \(P_1\) because this would make it longer than 6).

Let \(N(x_i, y_i)\) represent the number of paths in a minimum contiguous \(k\)-path partition of the induced subgraph \(G[\{x_1, x_2, \ldots, x_i\} \cup \{y_1, y_2, \ldots, y_i\}]\) in which the last vertex covered is \(y_i\) for \(1 \leq i \leq r\) and \(1 \leq l \leq s\). Similarly, let \(N(y_i, x_i)\) represent the number of paths in a minimum contiguous \(k\)-path partition of \(G[\{x_1, x_2, \ldots, x_i\} \cup \{y_1, y_2, \ldots, y_i\}]\) in which the last vertex covered is \(x_i\) for \(1 \leq i \leq r\) and \(1 \leq l \leq s\). We call a contiguous path starting at a vertex \(v\) maximum if it is the longest contiguous path starting at \(v\) whose length is not more than \(k\). Let us denote this path by \(P(v)\), its length by \(|P(v)|\) and its last vertex by \(lastP(v)\). We use \((lastP(v))^−\) to denote the vertex preceding \(lastP(v)\) on \(P(v)\) and \((lastP(v))^−2\) to denote the predecessor of \((lastP(v))^−\). Then we have the following algorithm.

**Algorithm 1**

1. Let \(Reached = \{(x_0, y_0), (y_0, x_0)\}\), where \(x_0, y_0\) are articial vertices added to \(X\) and \(Y\), respectively, in the leftmost positions with \(x_0y_1, y_0x_1 \in E\), \(N(x_0, y_0) = 0\), \(N(y_0, x_0) = 0\), \(P(x_0, y_0) = \emptyset\), \(P(y_0, x_0) = \emptyset\). /*The introduction of \(x_0, y_0\) facilitates a uniform presentation for the initial and general iterations of the algorithm.

   Let \(N(u_i, v_j) = \infty\) for all pairs \((u_i, v_j) \neq (x_0, y_0)\). /*\((u_i, v_j)\) is generic notation for a pair with \(u_i \in X\) and \(v_j \in Y\) or \(u_i \in Y\) and \(v_j \in X\).

2. Let \((u_i, v_j)\) be a leftmost pair in the list \(Reached\) and delete this pair from the list.

   If \((u_i, v_j) = (x_r, y_2)\) or \((y_2, x_r)\) then go to 4. /*We have reached the end of \(G\).

   If \(u_i = x_r\) or \(y_2\) then go to 3. /*We have reached the last vertex of \(G\) at the top or bottom, but there are some uncovered vertices left on one side.

   Find \(P(u_{i+1})\). /*Grow a new maximum path from \(u_{i+1}\).

   (2.1) If \(P(u_{i+1})\) has only one vertex and \(N(v_j, u_{i+1}) > N(u_i, v_j) + 1\) then let \(N(v_j, u_{i+1}) = N(u_i, v_j) + 1\), \(P(v_j, u_{i+1}) = P(u_i, v_j) \cup P(u_{i+1})\), add \((v_j, u_{i+1})\) to the list \(Reached\). Repeat 2.

   (2.2) If \(|P(u_{i+1})| > 1\) and \(N((lastP(u_{i+1}))^−, lastP(u_{i+1})) > N(u_i, v_j) + 1\) then let \(N((lastP(u_{i+1}))^−, lastP(u_{i+1})) = N(u_i, v_j) + 1\), \(P((lastP(u_{i+1}))^−, lastP(u_{i+1})) = P(u_i, v_j) \cup P(u_{i+1})\) and add \((lastP(u_{i+1}))^−, lastP(u_{i+1}))\) to the list \(Reached\);

   (2.3) If \(|P(u_{i+1})| = k\) and \(N((lastP(u_{i+1}))^−2, (lastP(u_{i+1}))^−) > N(u_i, v_j) + 1\) then let \(N((lastP(u_{i+1}))^−2, (lastP(u_{i+1}))^−) = N(u_i, v_j) + 1\), \(P((lastP(u_{i+1}))^−2, (lastP(u_{i+1}))^−) = P(u_i, v_j) \cup (P(u_{i+1}) \setminus u_{i+1})\), add \((lastP(u_{i+1}))^−2, (lastP(u_{i+1}))^−\) to the list \(Reached\). Repeat 2. /*This operation saves \(u_{i+1}\) as a possible connector between the next not-yet-covered vertex on the other side and the rest of the graph as in the example of Fig. 4.
(3) Suppose there are \( l \) vertices left on one side which are not covered by \( P(u_i, v_j) \). Add each of these as one-vertex paths to \( P(u_i, v_j) \). The resulting set of paths becomes \( P(x_r, y_r) \) with \( N(x_r, y_r) = N(u_i, v_j) + l \) if \( u_i = x_r \) and \( N(x_r, y_r) > N(u_i, v_j) + l \) or it becomes \( P(y_r, x_r) \) with \( N(y_r, x_r) = N(u_i, v_j) + l \) if \( u_i = y_r \) and \( N(y_r, x_r) > N(u_i, v_j) + l \). Add \((x_r, y_r)\) or \((y_r, x_r)\) to Reached, whichever is applicable, and repeat 2.

(4) Let \( p_k(G) = \min\{N(x_r, y_r), N(y_r, x_r)\} \) and output \( p_k(G) \) and the collection of paths where this minimum is obtained.

**Theorem 5.** Algorithm 1 finds \( p_k(G) \) and a minimum \( k \)-path partition for a bipartite permutation graph \( G = (X, Y, E) \) in \( O(n^2) \) time and space, where \( n = |X| + |Y| \).

**Proof.** The algorithm computes \( N(u_i, v_j) \) by dynamic programming, traversing \( G \) in the left-to-right direction following the strong ordering. Pairs \((u_i, v_j)\) are added to the array \( \text{Reached} \) whenever \( N(u_i, v_j) \) is updated by an improved value. The corresponding collection of \( k \)-paths is stored in \( P(u_i, v_j) \). Pair \((u_i, v_j)\) is deleted from \( \text{Reached} \) when it becomes a leftmost pair in it, and we extend \( P(u_i, v_j) \) by adding the maximum path \( P(v_j) \) to it. The correctness of the algorithm follows from the preceding lemmas, corollary and discussion. Since there are at most \(|X| \cdot |Y| \leq n^2 \) pairs \((u_i, v_j)\), and each step of the algorithm requires \( O(1) \) time for a pair \((u_i, v_j)\), the claimed complexity follows too.

**4. Concluding remarks**

We have proved that the \( k \)-path partition problem is NP-hard on comparability graphs even with fixed \( k = 3 \). We have also sharpened the demarcation line between polynomially solvable and NP-hard cases of the \( k \)-path partition problem by showing that its decision version is polynomially solvable on the class of bipartite permutation graphs while it is NP-complete on chordal bipartite graphs, a class properly containing bipartite permutation graphs. There are several classes of bipartite graphs between these two classes for which the status of the problem remains open. One such class is the class of convex bipartite graphs which properly contains bipartite permutation graphs and is properly contained in the class of chordal bipartite graphs [3]. The problem may be
polynomially solvable on convex bipartite graphs, but the solution will have to be substantially different from the algorithm we presented above. This is due to the fact that the crucial Lemma 2 fails to be true for convex bipartite graphs. This is demonstrated by the graph shown in Fig. 5. If $k=8$, then $P_1 = 1, 8, 3, 10, 5, 11, 6, 12$ and $P_2 = 7, 2, 9, 4$ give a minimum partition into two crossing paths and it is easy to see that there is no optimal $k$-path partition into two non-crossing paths.

Acknowledgements

The author would like to thank Lorna Stewart for useful discussions and anonymous referees for their comments.

References