Permutation Graphs: Hamiltonian Paths*

Charles Riedesel and Jitender S. Deogun
Department of Computer Science and Engineering
University of Nebraska-Lincoln
Lincoln, NE 68588-0115, U.S.A.

ABSTRACT. Permutation graphs, a well known class of perfect graphs, has attracted the attention of numerous researchers. There are two noteworthy representations of permutation graphs. Permutation diagrams have been widely employed in theoretical and application research. The 2-dimensional Euclidean representation suggested by Ore is relatively unknown and unexplored. In this paper we demonstrate the utility of the latter representation in the investigation of the Hamiltonian Path problem in permutation graphs.

1 Permutation Graph Representations

Let π be a permutation of the first n natural numbers, and let π^{-1} and π^R denote the inverse and reverse permutations respectively. For example, if

$$\begin{aligned} \pi &= [11,2,1,12,5,9,7,6,8,3,4,10], \text{ then} \\ \pi^{-1} &= [3,2,10,11,5,8,7,9,6,12,1,4] \text{ and} \\ \pi^R &= [10,4,3,8,6,7,9,5,12,1,2,11], \text{ so that} \\ \pi_1 &= 11, \pi_{11}^{-1} = 1, \text{ and } \pi^R = 10. \end{aligned}$$

The graph $G(\pi)$ is constructed with vertex set $V = \{1, 2, \ldots, n\}$ and edge set $E = \{(i,j) | (i-j)(\pi_i^{-1} - \pi_j^{-1}) < 0\}$.

An undirected graph G is called a permutation graph if there exists a permutation π and a vertex labeling V called a permutation labeling such that $G = G(\pi)$ [1]. Unless otherwise indicated, a vertex will be referred

*This research was supported in part by the Office of Naval Research under Grant No. N0014-91-J-1693 to by its permutation labeling and its position in π by π_v^{-1}. A graph $G(\pi) = G(V, E)$ has a transitive orientation $F_G = \{ij | i > j, (i, j) \in E\}$. The complement graph $G^C(\pi) = G(\pi^R)$ is also a permutation graph. This means permutation graphs are comparable and cocomparable. Thus there exist partial orderings of V associated with F_G for $G(\pi)$ and F_{G^C} for $G^C(\pi)$, so both graphs $G(\pi)$ and $G^C(\pi)$ can be represented by Hasse diagrams.

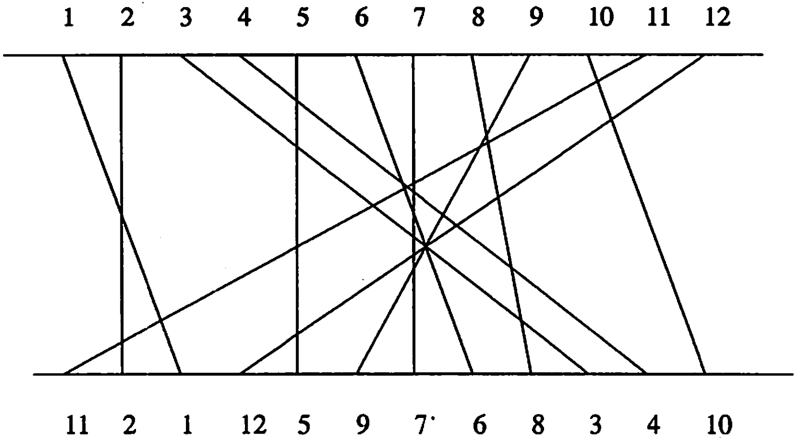

Figure 1: A Permutation Diagram for $G(\pi)$

The common representation for a permutation graph $G(V, E) = G(\pi)$ is the permutation diagram in which the ordered sequence V and the permutation π are listed opposite each other on two parallel lines. Straight line segments are drawn connecting all pairs of matching elements. The intersection of two line segments indicates the existence of an edge between the corresponding vertices. In this way a permutation graph can be thought of as an intersection graph. An intersection occurs exactly when the labels are reversed in the orderings V and π.

An alternate representation was inspired by Ore [3]. He showed that an n-dimensional partial order $P = (S, >)$ can be represented by points in a quadrant of an n-dimensional Euclidean space. The axes correspond to the n linear extensions in a realizer for the partial order. Each coordinate for plotting an element s of S is the position $p_{s,r}$ of s along the linear extension r. Members of P can be identified as those ordered pairs $s_i s_j$ such that for each coordinate r, $p_{s_i,r} > p_{s_j,r}$.

A 2-dimensional partial order $PG = (V, >)$ corresponding to the permutation graph $G = (V, E)$ can be defined using the transitive orientation F_{G^C}. Let the two linear extensions of a realizer be the natural ordered sequence V and the permutation π. Now $P_G = \{ij | ij \in F_{G^C}\}$. For convenience we use the fourth quadrant of an $x - y$ coordinate Euclidean space to plot the vertices of G, and let y increase downward. In this quadrant, members $v_i v_j$ of P identify exactly those pairs of vertices v_i and v_j lying on negatively sloped lines. The edges of G will then be identified by the pairs of vertices lying on positively sloped lines. Some features useful for investigating the Hamiltonian Path problem become much more visible this way.

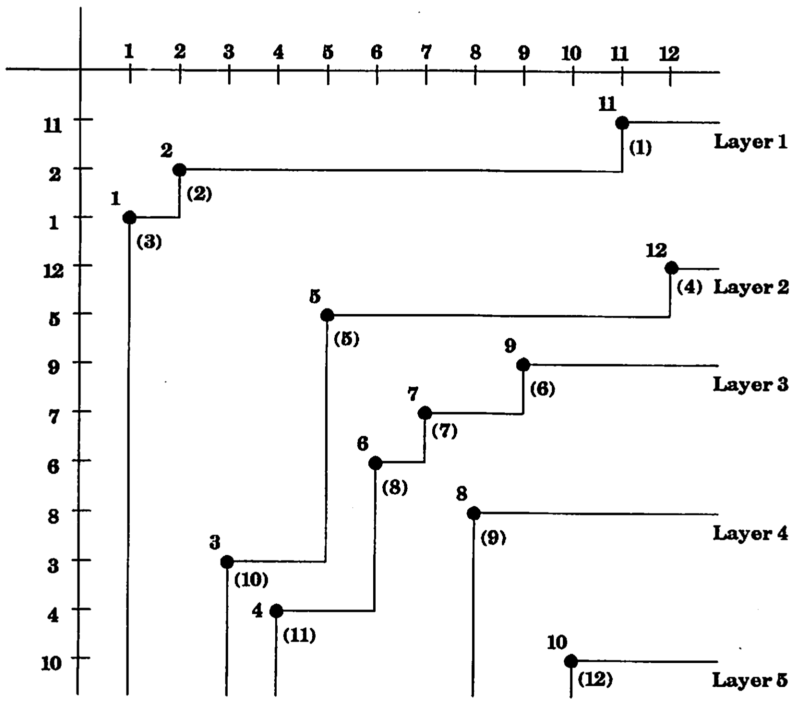

Figure 2: Euclidean Representation of Permutation Graph $G(\pi)$

1.1 Layering in the Euclidean Representation

Figure 2 shows the Euclidean representation of graph G for the example permutation π with layers drawn in. The plotted points are labeled with V, the permutation labeling (upper left), and position π^{-1} (lower right), so for instance, the lowest rightmost vertex is $v = 10$ at position $\pi_{10}^{-1} = 12$. An edge exists exactly when the slope between a pair of vertices is positive, indicating labels being reversed in the permutation, or equivalently nonmembers of the partial order P_G (without regard to the ordering of the vertices).

Each vertex v is assigned to a layer $l(v)$ numbered $1, 2, \ldots, \lambda$ from upper left to lower right. Let the number of vertices in $l(v)$ be denoted by $C_{l(v)}$. For example, $l(10) = 5$ and $C_{l(10)} = C_5 = 1$. The layers are drawn as non-intersecting leftward descending staircases using vertices as the outer tips of the steps. The process of drawing the layers involves no backtracking when going in sequence starting with layer 1, proceeding from either end through the whole layer. The labeling V decreases along each layer from upper right to lower left, while the positions π^{-1} increase.

There is an edge (v_i, v_j), $l(v_i) < l(v_j)$ when either $v_i > v_j$ as with vertices 9 and 8 of layers 3 and 4 respectively, or when $\pi_{v_i}^{-1} > \pi_{v_j}^{-1}$ as with vertices 4 and 8 ($\pi_4^{-1} = 11 > \pi_8^{-1} = 9$), also of layers 3 and 4. This stronger statement is justified because given $l(v_i) < l(v_j)$, at least one coordinate of v_i must be less than that of v_j. Edges also exist between all pairs of vertices of the same layer.

Let $L_{l,p}$ be the pth vertex on layer l from the upper right, and $R_{l,p}$ be the pth vertex from the lower left.. In the example $L_{3,1} = 9$ and $R_{3,1} = 4$. The vertex ordering within each layer can be expressed as $R_{l,i} < R_{l,j}$ and $\pi_{L_{l,i}}^{-1} < \pi_{L_{l,j}}^{-1}$ for $i < j$.

The direct consequence of the relations in the preceding two paragraphs is this observation:

Lemma 1. *If there exists an edge with one vertex in layer i and the other in layer j, then at least one of the following is true:*

- $(L_{i,1}, R_{i,1})$ is an edge.
- $(R_{i,1}, L_{j,1})$ is an edge.

A generalization to this observation can also be made:

Lemma 2. *Given vertex v in layer i and the non-empty set of edge $E' = \{(v, v_j) | (v, v_j) \in E, v_j \in$ layer $1\}$, then there exist m and/or n such that $E' = \{L_{l,1}, L_{l,2}, \ldots, L_{l,m}, R_{l,1}, R_{l,2} \ldots, R_{l,n}\}$.*

Generating the layers for a permutation graph of order n, given a defining permutation π, is straight forward and can be done in $O(n\lambda)$ time, where $\lambda \leq n$. Each layer beginning with the top layer is formed by searching and extracting the maximal length decreasing sequence in the permutation starting with the first (left-most) vertex remaining in π.

Algorithm 1 (Layering)

```
input:
  permutation $\pi$
  number of vertices $n$
output:
  2-D array of vertices by layer $L$
  array of number of vertices per layer $C$
  number of layers $\lambda$
$\lambda \leftarrow 0$
$n' \leftarrow n$
While $\pi \neq \emptyset$
  Increment $\lambda$
  $C_\lambda \leftarrow 1$
  $j \leftarrow 1$
  $L_{\lambda, C_\lambda} \leftarrow \pi_j$
  Delete $\pi_j$ from $\pi$
  Increment $j$
  While $j \leq n'$
      If $L_{\lambda, C_\lambda} > \pi_j$ Then
         Increment $C_\lambda$
         $L_{\lambda, C_\lambda} \leftarrow \pi_j$
         Delete $\pi_j$ from $\pi$
         Decrement $n'$
      Increment $j$
  End While
  $n \leftarrow n'$
End While
```

2 Hamiltonian Path Algorithm

A Hasse diagram partitions the ground set S of a partial order $P = (S, >)$ into levels. In particular there is a partitioning $\mathcal{L}(P)$ in which the Hasse diagram is arranged so that each element $s \in S$ is at the highest possible level. In $\mathcal{L}$ each element s of each level below the top has an upper bound element s_{ub} in the next higher level.

By construction of the Euclidean representation, the edge complement relation for vertices exhibits the same characteristics when 'layer' replaces 'level'. Specifically, for all $v, l(v) > 1$, there exists $v', l(v') = l(v) - 1$, such that $(v, v') \notin E$). The layers do therefore partition V of $G(\pi)$ to be the same as $\mathcal{L}(P_G)$.

It has been shown [2] that if a cocomparability graph is traceable, then there exists a layered Hamiltonian path that traces a route through $\mathcal{L}(P_G)$ from the bottom up. That result was based on the correspondence of the Hamiltonian path problem in cocomparability graphs with the bump number problem in partial orders. Such a path is built by iteratively claiming vertices from the lowest existing level. As a level is exhausted, a lowest vertex to which an edge exists is claimed, even if the vertex must be *pulled* *down* from a higher level. These transitions between the levels can be made by applying greedy choice rules for pulling down vertices from higher levels. Our algorithm utilizes this existence characteristic without the need to generate and store a Hasse diagram. Furthermore the internal ordering of vertices in each layer allows the greedy choice conditions to be met by testing no more than four vertices per layer, specifically $L_{p,1}$, $L_{p,2}$, $R_{p,2}$, and $R_{p,1}$, where p is the lowest layer to which a transition exists (the *pull* layer). At most two vertices, $L_{b,1}$ and $R_{b,1}$, may need to be tested in the layer being exited (the *base* layer). This is an immediate consequence of Lemma 2.

There are three greedy rules for selecting a transition vertex in the pull layer to which an edge exists from an available vertex in the base layer:

1. If either $(L_{b,1}, R_{p,2})$ or $(R_{b,1}, L_{p,2})$ is an edge and $C_p > 2$, choose that edge for the transition vertices. This leaves $L_{p,1}$ and $R_{p,1}$ available to exit the pull layer later.

2. If both $(L_{b,1}, R_{p,1})$ and $(R_{b,1}, L_{p,1})$ are edges, reserve them until a choice is forced later or it is determined that neither vertex result in the path being blocked.

3. Either $(L_{b,1}, R_{p,1})$ or $(R_{b1}, L_{p,1})$ must be an edge because some edge is assumed and lemma 1 states it must be one of these. Choose that edge for the transition vertices.

Input for the algorithm is a permutation π representing the permutation graph $G(\pi)$. The internal representation of the graph may consist of only the permutation and its inverse π^{-1}, thus saving the space and computation time for an adjacency matrix. Many vertices may never need to be checked for edges. A call must first be made to the Layering algorithm to generate the rest of the commonly used parameters.

Note: L and R are always updated to reflect the current positioning of vertices in the layers. In the implementation, L and R are heads at either end of a doubly linked list of elements.

The algorithm returns an array of transitions in *TransitionEdge* and the remaining vertices in L (or equivalently R). These vertices can be inserted arbitrarily between the transitions layer by layer to produce a Hamiltonian path.

Algorithm 2 (Hamiltonian Path)

```
$b \leftarrow \lambda$
While $b > 1$ and not fail
    $p \leftarrow b$
    While $p > 1$ and a pull layer is not found
```

```
        $p \leftarrow p - 1$
        Case
            Greedy rule 1 applies:
                $Pull(b, p, LtoR_2$ or $RtoL_2$ as appropriate)
            Greedy rule 2 applies:
                $Reserve(b, p)$
            Greedy rule 3 applies:
                $Pull(b, p, LtoR$ or $RtoL$ as appropriate)
            Else pull layer not found yet
    End While
    If $p = 1$ and a pull layer is not found
        Then fail
    $b \leftarrow b - 1$
End While

Procedure $Reserve(b, p)$
If $Reserve_p$ already exists
    Then    $Resolve(p, LtoR)$
            $Pull(b, p, RtoL)$
    Else    $Reserve_p \leftarrow b$

Procedure $Resolve(p, EdgeChoice)$
$Pull(Reserve_p, p, EdgeChoice)$
Clear $Reserve_p$

Procedure $Pull(b, p, EdgeChoice)$
Case $EdgeChoice$ of
    $LtoR$:   Extract vertices $L_{b,1}$ and $R_{p,1}$ from $L$
    $RtoL$:   Extract vertices $R_{b,1}$ and $L_{p,1}$ from $L$
    $LtoR_2$: Extract vertices $L_{b,1}$ and $R_{p,2}$ from $L$
    $RtoL_2$: Extract vertices $R_{b,1}$ and $L_{p,2}$ from $L$
$TransitionEdge(b) \leftarrow$ extracted vertices
If $L_{b,1}$ and $R_{b,1}$ were reserved
    Then $Resolve(b$,edge to remaining vertex)
Update $C_p$, $C_b$
```

Reserves may be resolved if either of a couple cases exist:

1. a) If the closest transitions are to $R_{p,1}$ and $L_{p,1}$ only and these are already reserved, then both must be claimed and will not be available as exits later. Any existing reserved vertices at the base layer can be resolved arbitrarily and the reserved vertices at the pull layer can then be resolved using the remaining vertex.

2. b) If some closest transitions include $R_{p,2}$ or $L_{p,2}$, then such a vertex can be pulled without penalty. Any current reserved vertices at the base layer can be resolved in a way compatible with this choice.

Chains of reserves may exist and can be resolved either at the end of the algorithm or as the next higher reserve is resolved, depending on which occurs first.

The base layer moves up one layer with each iteration of the outer loop, so there are $\lambda - 1$ iterations. With no more than a constant number of vertices to check at each iteration, the algorithm runs in $O(\lambda)$ time, where $\lambda \leq n$. The overall time complexity of our algorithm is $O(n\lambda)$ because of the initial layering routine. In the worst case this could be $O(n^2)$. This is an improvement on the $O(n^3)$ complexity of the best existing algorithm for cocomparability graphs.

2.1 Example of Algorithm

Let $\pi = [11, 2, 1, 12, 5, 9, 7, 6, 8, 3, 4, 10]$ so that $\pi^{-1} = [3, 2, 10, 11, 5, 8, 7, 9, 6, 12, 1, 4]$ as in the previous examples.

The vertices at each layer L_i and remaining π as each layer is extracted are as follows:

$$\begin{array}{ll}
L_1 = [1, 2, 11] & \pi = [12, 5, 9, 7, 6, 8, 3, 4, 10] \\
L_2 = [3, 5, 12] & \pi = [9, 7, 6, 8, 4, 10] \\
L_3 = [4, 6, 7, 9] & \pi = [8, 10] \\
L_4 = [8] & \pi = [10] \\
L_5 = [10] & \pi = \emptyset
\end{array}$$

The initial assignments of each layer's end vertices are as follows:

layer	$R_{l,1}$	$R_{l,2}$	$L_{l,2}$	$L_{l,1}$
1	1	2	2	11
2	3	5	5	12
3	4	6	7	9
4	8			8
5	10			10

Starting at layer 5, no edge exists to vertex 8 of layer 4, and again no edge exists to either vertex 4 or 9 of layer 3, but an edge is found to exist to vertex 12 of layer 2. There is no edge to either 3 or 5,so greedy rule 3 applies. Claim 12 as the pull vertex to layer 4 using edge (10,12),so in effect 12 a member of layer 4.

From the only remaining vertex (8) of layer 4 edges exist to only the outside vertices 4 and 9 of layer 3. Greedy rule 2 applies. Reserve these vertices, allowing them to be considered for exiting layer 3.

Looking for edges from layer 3 to layer 2, two are found: (9,3) and (4,5). (Vertex 12 was claimed earlier.) Because 3 and 5 are both outside vertices, greedy rule 2 applies again. Reserve them, allowing either to be used to exit layer 2.

One edge, (3,11), is found from layer 2 to layer 1. Greedy rule 3 applies and vertex 11 must be the pull vertex. Earlier reserved vertices can now be examined.

Vertex 3 is used to exit layer 2, leaving vertex 5 as the pull vertex for layer 3. Now vertex 4 used to exit layer 1, leaving vertex 9 as the pull vertex for layer 4.

The edges for traversing the layers are (10,12), (8,9),(4,5), and (3,11). Inserting the remaining vertices produces a Hamiltonian Path $[10, 12, 8, 9, 6, 7, 4, 5, 3, 11, 1, 2]$.

References

[1] M. C. Golumbic. "Algorithmic Graph Theory and Perfect Graphs". Academic Press. 1980

[2] P. Damaschke, J.S. Deogun, D. Kratsch and George Steiner. Finding Hamiltonian Paths in Cocomparability Graphs Using The Bump Number Algorithm. *Order*, **8**(1992), 383–391.

[3] O. Ore. Theory of Graphs. Section 10.4 *Amer. Math. Soc. Colloq. Publ.* **38** Providence, RI. M R27 #740.