1993

Optimal Greedy Algorithms for Indifference Graphs

Peter J. Looges
Old Dominion University

Stephan Olariu
Old Dominion University

Original Publication Citation

Looges, P. J., & Olariu, S. (1993). Optimal greedy algorithms for indifference graphs. *Computers & Mathematics with Applications, 25*(7), 15-25. doi:10.1016/0898-1221(93)90308-i

Computers Math. Applic. Vol. 25, No. 7, pp. 15–25, 1993
Printed in Great Britain.

0097-4943/93 $6.00 + 0.00

OPTIMAL GREEDY ALGORITHMS FOR INDIFFERENCE GRAPHS

PETER J. LOOGES AND STEPHAN OLARIU†
Department of Computer Science, Old Dominion University
Norfolk, VA 23529-0162, U.S.A.

(Received in revised form October 1992)

Abstract—A fundamental problem in social sciences and management is understanding and predicting decisions made by individuals, various groups, or the society as a whole. In this context, one important concept is the notion of indifference. We characterize the class of indifference graphs, that is, graphs which arise in the process of quantifying indifference relations. In particular, we show that these graphs are characterized by the existence of a special ordering of their vertices. As it turns out, this ordering leads naturally to optimal greedy algorithms for a number of computational problems, including coloring, finding a shortest path between two vertices, computing a maximum matching, the center, and a Hamiltonian path.

1. INTRODUCTION

In trying to understand and predict social phenomena, one is confronted with the problem of quantifying entities which are not as easy to measure as well-known physical variables, such as distance or density occuring in everyday life. It has been recognized that the process of analyzing decisions made by various individuals, groups, or by the society as a whole requires the ability to reason about such things as *preference*, *agreement*, and *indifference* [1]. In the process of decision making, for example, administrators have to take into account opinions and viewpoints expressed by different social groups or organizations. Similarly, in marketing, one is interested in understanding behavior patterns of potential consumers, as expressed by indifference attitudes towards comparable products on the market.

In this work, we propose to investigate the class of indifference graphs that models the notion of indifference relation arising in social sciences and management. Specifically, a graph $G = (V, E)$ is an *indifference* graph [2] if there exists a positive number δ (measuring "closeness" or "indifference") and an assignment of numbers $f(u)$ to the elements of V such that for all $u, v \in V$, uv is an edge in G whenever $|f(u) - f(v)| \leq \delta$. (To be consistent with [1], we ignore loops in indifference graphs, that is, edges of the form uu with $u \in V$.) As it turns out [1], the indifference graphs are a subclass of the well-known class of interval graphs that we discuss next.

Interval graphs are invaluable tools when it comes to modeling real-life situations, especially those involving dependencies that are linear in nature. They find applications to archaeology, biology, psychology, sociology, management, genetics, and many others. The reader is referred to [1] and [3], where many of the above applications are summarized.

More formally, a graph $G = (V, E)$ is termed an *interval graph* if there exists a family $\{I_i\}$ $(1 \leq i \leq n)$ of intervals on the real line such that distinct vertices u, v are adjacent if, and only if, the corresponding intervals overlap. Such a family $\{I_i\}$ $(1 \leq i \leq n)$ of intervals is commonly referred to as the *interval representation* of G.

The interval graphs have been studied intensely from both the theoretical and algorithmic point of view. Early characterizations, in terms of forbidden configurations, appear in [4,5].

†This author was supported, in part, by the NSF Grant CCR-8909996.
The authors would like to thank Professor Rodin and an anonymous referee for a very thorough review that resulted in a better presentation.

Later, Booth and Lueker [6,7] used PQ-trees to investigate the algorithmic properties of interval graphs: they obtain linear-time recognition and isomorphism algorithms.

The purpose of this work is to investigate algorithmic properties of indifference graphs. We first present a new characterization of indifference graphs in terms of a linear order on their sets of vertices and show that this new linear order affords us optimal greedy algorithms to solve problems such as coloring, finding a shortest path between two vertices, a maximum matching, and a Hamiltonian path. The paper is organized as follows. Section 2 presents the basic tools for algorithm development in the form of a number of theoretical results; Section 3 proposes a greedy recognition algorithm for indifference graphs; finally, Section 4 develops a number of greedy algorithms for the computational problems mentioned above. All our greedy algorithms are extremely simple and run in optimal time.

2. BASICS

All the graphs in this work are finite, with no loops nor multiple edges. In addition to standard graph-theoretical terminology compatible with [3], we use some new terms that we are about to define. For a vertex x of a graph G, we let $d(x)$ stand for the number of vertices adjacent to x (adjacency is assumed to be irreflexive, no vertex being adjacent to itself). A graph with vertices a, b, c, d and edges ab, ac, ad is referred to as the *claw* (see Figure 1).

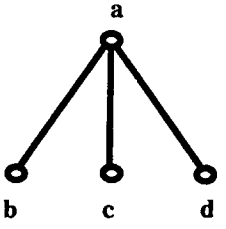

Figure 1. The claw.

Let $G = (V, E)$ be an interval graph and let $\{I_v = [a_v, b_v]\}$ be an interval representation of G; here, a_v and b_v ($a_v \leq b_v$) are referred to as the left and right endpoint of the interval I_v. G is called a *unit* interval graph if all the intervals in the representation have unit length. The family $\{I_V\}_{v \in V}$ is the interval representation of a *proper* interval graph if no interval is properly contained in another. Clearly, unit interval graphs are proper interval graphs. Roberts [1] has proved the following fundamental result that shows that unit interval graphs, proper interval graphs, and indifference graphs are synonyms.

PROPOSITION 1. *See [1]. For a graph G, the following statements are equivalent:*

(i) *G is a unit interval graph;*
(ii) *G is a proper interval graph;*
(iii) *G is an interval graph with no induced claw;*
(iv) *G is an indifference graph.* ∎

We shall rely on the following fundamental results concerning interval graphs.

PROPOSITION 2. *See [5]. A graph G is an interval graph if, and only if, the maximal cliques of G can be linearly ordered in such a way that for every vertex v of G, the maximal cliques containing v occur consecutively.* ∎

PROPOSITION 3. *See [3]. In any interval graph, the sum of the sizes of all maximal cliques is linear in the size of the graph.* ∎

Assume that the maximal cliques of an interval graph G have been enumerated as C_1, $C_2, \ldots, C_m$ as in Proposition 2. If, for every vertex v of G, we let I_v stand for the set $\{C_i \mid v \in C_i\}$, then, by Proposition 2, I_v is an interval. Hence, $\{I_v\}_{v \in V}$ is an interval representation for G. We shall refer to this interval representation as *max-clique*.

The following result shows that interval graphs are characterized by a special ordering of their vertices. This result will be used again and again in the remainder of this work.

PROPOSITION 4. *See [8]. A graph $G = (V, E)$ is an interval graph if, and only if, there exists a linear order $<*$ on V such that for every choice of vertices u, v, w with $u <* v <* w$, $uw \in E$ implies $uv \in E$.* ∎

For an interval graph G the ordering

$$v_1 <* v_2 <* \cdots <* v_n \tag{1}$$

of its vertices with the property specified in Proposition 4 will be referred to as *canonical.*

Our first result is a characterization of indifference graphs in the spirit of the result in [8]. More precisely, we show that just like the interval graphs, the indifference graphs are also characterized by an ordering of their vertices. This linear order not only satisfies (1), but, in fact, satisfies a stronger property which lays the basis for all our optimal greedy algorithms.

THEOREM 1. *A graph $G = (V, E)$ is an indifference graph if, and only if, there exists a linear order $\lessdot$ on V such that for every choice of vertices u, v, w,*

$$u \lessdot v \lessdot w, \text{ and } uw \in E \text{ implies } uv,\ vw \in E. \tag{2}$$

PROOF. First, let $\lessdot$ be a linear order on V with the properties specified in (2). In particular, $\lessdot$ satisfies the condition specified in Proposition 4, and so G is an interval graph.

By Proposition 1, to prove that G is an indifference graph, we need show that G contains no induced claw. For the sake of the argument, suppose that G contains an induced claw with vertices a, b, c, d and edges ab, ac, ad. We propose to show that this assumption leads to a contradiction.

To begin, note that vertex a cannot precede b, c, d in $\lessdot$: otherwise, (2) would imply that b, c, d are pairwise adjacent, a contradiction.

Similarly, vertex a cannot follow b, c, d in the linear order: otherwise, let z stand for the vertex among b, c, d that comes first in the order $\lessdot$. Since $za \in E$, it follows that z is adjacent to the remaining two, contradicting that $\{a, b, c, d\}$ induces a claw.

Now the symmetry of the claw allows us to assume, without loss of generality, that b precedes a, c, d and that d follows a, b, c in $\lessdot$. But now, the ordering $b \lessdot c \lessdot a$ implies $bc \in E$; similarly, the ordering $a \lessdot c \lessdot d$ implies that $cd \in E$. Either case leads to a contradiction.

Conversely, let G be an indifference graph. In particular, G is an interval graph. For every vertex x of G we let $I_x = [a_x, b_x]$ stand for the corresponding interval. Define a linear order $\lessdot$ on V by setting

$$u \lessdot v \text{ whenever } (a_u < a_v) \text{ or } [(a_u = a_v) \text{ and } (b_u \leq b_v)]. \tag{3}$$

Let u, v, w be arbitrary vertices in G satisfying $u \lessdot v \lessdot w$ and assume that $uw \in E$: that is, the intervals I_u and I_w overlap. Now the assumption that $u \lessdot v \lessdot w$, together with (3), implies that

$$a_u \leq a_v \leq a_w.$$

In case $a_u = a_w$, the conclusion is immediate; we shall therefore assume that

$$a_u < a_w.$$

Since I_u and I_w overlap, it must be the case that

$$a_w \leq b_u.$$

But now, $a_v \leq a_w$ guarantees that $a_v \leq b_u$, and so $uv \in E$.

Next, since I_u cannot properly contain I_v, $a_u \leq a_v$ guarantees that $b_u \leq b_v$. It follows that $a_w \leq b_v$, implying that $vw \in E$. This completes the proof of Theorem 1. ∎

Theorem 1 implies the following results.

COROLLARY 1.1. *Let $G = (V, E)$ be an indifference graph and let $\lessdot$ be a linear order on the vertex-set of G satisfying (2). For every choice of subscripts i, j with $(1 \leq i < j \leq n)$ and $v_i v_j \in E$, the vertices $v_i, v_{i+1}, \ldots, v_j$ are pairwise adjacent.*

PROOF. To see this, let v_p and v_q be arbitrary vertices with $i \leq p < q \leq j$. Now the fact that v_i and v_j are adjacent, together with (2), imply that v_p and v_j are adjacent, and so, by (2) again, v_p and v_q must be adjacent, as claimed. ∎

COROLLARY 1.2. *An interval graph is an indifference graph if and only if a canonical ordering $<*$ of the vertices of G satisfies (2).*

PROOF. First, if a canonical ordering $<*$ of G satisfies (2), then by Theorem 1 G is an indifference graph. Conversely, if G is an indifference graph, then by Theorem 1 we find an ordering $\lessdot$ of its vertices satisfying (2). But now, $\lessdot$ also satisfies (1) and the conclusion follows. ∎

Let G be an indifference graph; just as in the case of interval graphs, an ordering $\lessdot$ of the vertices of G satisfying (2) is referred to as *canonical.*

To obtain a characterization of indifference graphs leading to a fast recognition algorithm, consider an interval graph G with a canonical ordering $<*$. For every i $(1 \leq i \leq n)$ define $\text{First}[i] = \min\{i, k\}$ such that $v_i v_k \in E$; similarly, $\text{Last}[i] = \max\{i, k\}$ such that $v_i v_k \in E$.

We are now in a position to prove a result that is central to our recognition algorithm for indifference graphs.

THEOREM 2. *Let G be an interval graph with a canonical ordering $<*$. G is an indifference graph if, and only if, for every v_i $(1 \leq i \leq n)$, $d(v_i) = \text{Last}[i] - \text{First}[i]$.*

PROOF. To begin, let G be an indifference graph. We proceed by induction on the number of vertices in G. If G is disconnected, then the conclusion follows by the induction hypothesis applied to every component of G separately.

We may, therefore, assume that G is connected. By Corollary 1.2, we may assume without loss of generality that $<*$ satisfies (2); it follows, in particular, that every vertex v_i with $(2 \leq i \leq n-1)$ is adjacent to all the vertices $v_{\text{First}[i]}, v_{\text{First}[i]+1}, \ldots, v_{i-1}, v_{i+1}, \ldots, v_{\text{Last}[i]}$, and so $d(v_i) = \text{Last}[i] - \text{First}[i]$.

Further, if $i = 1$, then $\text{First}[v_i] = 1$ and v_i is adjacent to $v_{i+1}, \ldots, v_{\text{Last}[i]}$, confirming that $d(v_i) = \text{Last}[i] - \text{First}[i]$. Finally, if $i = n$, then $\text{Last}[v_i] = n$, and v_i is adjacent to $v_{\text{First}[i]}$, $v_{\text{First}[i]+1}, \ldots, v_{i-1}$, and so $d(v_i) = \text{Last}[i] - \text{First}[i]$.

Conversely, assume that G contains an induced claw with vertices a, b, c, d and edges ab, ac, ad. Symmetry allows us to assume that $b <* c <* d$. But now, in case $c <* a$, $d(b) < \text{Last}[b] - \text{First}[b]$ since b is not adjacent to c and $c <* \text{Last}[b]$; in case $a <* c$, $d(b) < \text{Last}[b] - \text{First}[b]$ since in this case d is not adjacent to c and $\text{First}[d] <* c$. This completes the proof of Theorem 2. ∎

The following result identifies a property of chordless paths joining vertices of an indifference graph G.

THEOREM 3. *Let G be an indifference graph with a canonical ordering $\lessdot$, and let x, y $(x \lessdot y)$ be distinct vertices of G. If $x = u_1, u_2, \ldots, u_p = y$ is an arbitrary chordless path joining x and y, then for all i $(1 \leq i \leq p-1)$, $u_i \lessdot u_i + 1$.*

PROOF. First, note that

$$u_j \lessdot y \qquad \text{for all } j = 1, 2, \ldots, p-1. \tag{4}$$

[Suppose not; let t be the first subscript for which $y \lessdot u_t$. But now, (2) guarantees that u_{t-1} and y are adjacent, contradicting that the path is chordless.]

Next, if the statement is false, then we find a subscript j $(j < p)$ such that

$$u_{j+1} \lessdot u_j.$$

Now (4) implies that $u_{j+1} \lessdot u_j \lessdot y$. Observe that

$$u_j \lessdot u_{p-1},$$

for otherwise condition (2) guarantees that u_j and u_p are adjacent, a contradiction.

Let k be the first subscript larger than $j+1$ for which

$$u_j \lessdot u_k.$$

Since $u_{j+1} \lessdot u_j$, it must be the case that $k \geq j+2$. By our choice of k, $u_{k-1} \lessdot u_j \lessdot u_k$. However, now (2) implies that u_j and u_k are adjacent, a contradiction. This completes the proof of Theorem 3. ∎

COROLLARY 3.1. *Let G be a connected indifference graph with a canonical ordering $\lessdot$. For all subscripts i $(2 \leq i \leq n-1)$, First[i] $< i <$ Last[i].*

PROOF. Let i be an arbitrary subscript with $(2 \leq i \leq n-1)$. Since G is connected, there exists a path in G joining v_1 and v_i. Let

$$v_1 = x_1, x_2, \ldots, x_t = v_i$$

be such a path. By taking t as small as possible, we ensure that the path is chordless. By Theorem 3, it follows that, in particular, $v_{t-1} \lessdot v_t = v_i$, and so First[$i$] $< i$.

The proof that $i <$ Last[i] follows by a mirror argument which is omitted. ∎

3. GREEDY ALGORITHMS I: RECOGNIZING INDIFFERENCE GRAPHS

Let $G = (V, E)$ with $|V| = n$ and $|E| = m$ be an arbitrary graph. The following greedy algorithm determines whether or not G is an indifference graph.

ALGORITHM. Recognize(G);

STEP 1. Invoke the interval graph recognition algorithm of Booth and Lueker [6], running in $O(n+m)$ time. In case G is an interval graph, the algorithm also returns an ordering $C_1, C_2, \ldots, C_m$ of the maximal cliques of G as in Proposition 2.

STEP 2. Using the adjacency information of G, together with the ordering $C_1, C_2, \ldots, C_m$ construct a max-clique interval representation for G. At this stage, we use two arrays $B[1 \ldots m]$ and $H[1 \ldots n]$, initialized to 0: for every vertex u of G, $H(u)$ contains the largest i for which $u \in C_i$; B is used as a set of buckets: specifically, $B[j]$ contains (in a linked list) all the vertices u of G for which $j = \min\{k \mid u \in C_k\}$. The details of this step are spelled out by the following procedure.

PROCEDURE. Compute_Ordering(G);
{Input: an interval graph $G = (V, E)$ and an ordering $C_1, C_2, \ldots, C_m$ of its maximal cliques;
Output: an ordering of the vertices of G as in Proposition 4.}
0. **begin**
1. **for** $j \leftarrow 1$ **to** m **do**
2. **for** all u in C_j **do begin**
3. **if** $H(u) = 0$ **then**
4. add u to bucket $B[j]$; {think of $B[j]$ as a linked list}
5. $H(u) \leftarrow j$;
6. **end**; {for}
7. sort each bucket $B[j]$ in ascending order of $H(v)$;
8. return(B)
9. **end**; {Compute_Ordering}

It is easy to confirm (see [8], for example) that the ordering of the vertices of G returned in line 8 of Compute_Order is a canonical ordering $<*$ for the interval graph G. Furthermore, Proposition 3 guarantees that the overall running time of Step 2 is bounded by $O(m+n)$.

STEP 3. Scanning the adjacency list of every vertex v_i of G once, we compute First[i] and Last[i] for every i $(1 \leq i \leq n)$.

STEP 4. Finally, for every i $(1 \leq i \leq n)$ we check whether $d(v_i) =$ Last[i]$-$First[i]. By Theorem 2, G is an indifference graph if, and only if, this equality holds for i $(1 \leq i \leq n)$. The details are expressed as follows.

```
PROCEDURE. Test_Indifference(G);
{Input: a graph G along with an ordering <* of the vertices as in (4);
 Output: "yes" or "no" depending on whether or not G is an indifference graph} 0. begin
1.   for i ← 1 to n do
2.       compute Last[i] and First[i];
3.   for i ← 1 to n do
4.       if Last[i] − First[i] ≠ d(v_i) then
5.           return("no");
6.   return("yes")
7. end; {Test_Indifference}
```

The following result summarizes our findings in this section.

THEOREM 4. *With a graph G with n vertices and m edges as input, Test_Indifference correctly decides in $O(n+m)$ whether G is an indifference graph.*

PROOF. The correctness follows immediately from Theorem 2. To address the time complexity, it is helpful to imagine that after the ordering $<*$ computed in Step 2 is available, the adjacency structure of G is updated in $O(n+m)$ time to inform every vertex of its position in the canonical ordering. Now computing Last[i] and First[i] is easy: for every i, scan the adjacency list of v_i once retaining the largest and smallest subscript. Clearly, this takes $O(d(v_i))$. Consequently, the overall complexity is bounded by $O(n+m)$, as claimed. ∎

It is important to note that should G turn out to be an indifference graph, the ordering (1) satisfies condition (2) as well.

4. GREEDY ALGORITHMS II: VARIOUS COMPUTATIONAL PROBLEMS

Let $G = (V, E)$ be an arbitrary indifference graph with an ordering $v_1 \lessdot v_2 \lessdot \cdots \lessdot v_n$ satisfying (2). We are now in a position to show how this linear order can be exploited for the purpose of designing very simple, optimal, greedy algorithms to solve a number of computational problems.

First, we present a coloring algorithm for indifference graphs. The only data structure used is a stack; initially, this stack contains the colors $1, 2, \ldots, n$ in reverse order, that is with 1 at the top of the stack. The idea of the algorithm is straightforward: assign vertex v_1 color 1 (equivalently, v_1 is colored by popping the stack).

After v_{i-1} ($i \geq 2$) has been colored, we proceed to color v_i as follows. Consider the set of colors assigned to the vertices

$$v_{\text{First}[i-1]}, v_{\text{First}[i-1]+1}, \cdots, v_{\text{First}[i]-1}.$$

Note that by Corollary 1.1, all these colors must be distinct. Furthermore, none of these vertices are adjacent to v_i, and so we can reuse any of their colors on v_i. With this observation in mind, we first release these colors by pushing them onto the stack and then proceed to color v_i by simply popping the stack (i.e., assigning v_i the color at the top of the stack). The details are spelled out by the following procedure.

```
PROCEDURE. Color(G);
{Input: an indifference graph G, along with a canonical ordering ≤;
 Output: an optimum coloring of G;}
0. begin
1.   color(v_1) ← pop(stack);
2.   for i ← 2 to n do begin
3.       for j ← First[i − 1] to First[i] do
4.           push(color(v_j));
5.       color(v_i) ← pop(stack)
6.   end {for}
7. end; {Color}
```

THEOREM 5. *With an indifference graph $G = (V, E)$ with n vertices and m edges as input, procedure Color correctly returns an optimal coloring of G in $O(n)$ time.*

PROOF. To justify the correctness, we need to show that procedure Color returns a proper coloring of G and that this coloring uses as few colors as possible. First, we claim that

$$\begin{gathered}\text{no vertex } v_j \ (i \le j) \text{ is adjacent to one of the vertices} \\ v_k \text{ such that } \mathrm{First}[i-1] \le k \le \mathrm{First}[i]-1.\end{gathered} \tag{5}$$

[To see that this is the case, note that if some vertex v_j with $i \le j$ is adjacent to a vertex v_k with $\mathrm{First}[i-1] \le k \le \mathrm{First}[i]-1$, then by (2) it must be that v_i and v_k are adjacent, contradicting that $k < \mathrm{First}[i]$.]

Now (5) implies that when a color assignment takes place in line 5, vertex v_i and the vertices that have received the same color prior to v_i are not adjacent. Thus, procedure Color produces a proper coloring.

To argue about the optimality of this coloring, we only need show that if procedure Color uses a total of k colors, then G contains k pairwise adjacent vertices (i.e., no fewer than k colors can possibly be used to properly color G). Consider the first vertex, say v_i, that received color k. Observe that the way we initialized the stack, along with line 5, guarantees that all the first $k-1$ colors $1, 2, \ldots, k-1$ were in use when v_i was about to be colored. Now (5) implies that these colors must have been used on the vertices v_k with $\mathrm{First}[i] \le k \le i-1$. But now, Corollary 1.1 guarantees that G contains a set of k pairwise adjacent vertices, namely $v_{\mathrm{First}[i]}, v_{\mathrm{First}[i]+1}, \ldots, v_i$. This shows that the coloring produced is optimal.

To address the complexity, we note that by (5), the loop in lines 3–4 takes at most $O(n)$ time. Consequently, the running time of the procedure is bounded by $O(n)$, as claimed. ∎

Next, we propose a simple greedy algorithm that computes a shortest path between two given (but otherwise arbitrary) vertices of a connected indifference graph.

PROCEDURE. Shortest_Path(x, y);
{Input: a connected indifference graph G with a canonical ordering $\lessdot$ and two vertices $x \lessdot y$;
Output: a shortest path $x = x_1, x_2, \ldots, x_t = y$ joining x and y}
0. **begin**
1. $t \leftarrow 1$;
2. $x_t \leftarrow x$;
3. **while** x_t and y are not adjacent **do begin**
4. $t \leftarrow t + 1$;
5. $x_t \leftarrow v_{\mathrm{Last}[x_{t-1}]}$;
6. **end**; {while}
7. $t \leftarrow t + 1$;
8. $x_t \leftarrow y$;
9. return$(x_1, x_2, \ldots, x_t)$
10. **end**; {Shortest_Path}

THEOREM 6. *Let G be a connected indifference graph with n vertices and m edges, and let x and y, $x \lessdot y$ be two vertices of G. Procedure Shortest_Path correctly constructs in $O(n)$ time a shortest path between x and y.*

PROOF. The complexity being obvious, we only need address the correctness of this procedure. Let $x = x_1, x_2, \ldots, x_t = y$ be the path returned by Shortest_Path(x, y). Suppose that there exists a path $x = z_1, z_2, \ldots, z_q = y$ joining x and y such that $q < t$.

This, however, implies that for a suitable choice of subscripts i, j $(2 \le i \le t-1;\ 2 \le i \le q-1)$

$$z_i \lessdot x_j \lessdot x_j + 1 \lessdot z_i + 1.$$

Note that (2) in Theorem 1 guarantees that x_j and z_{i+1} are adjacent. But now we contradict the choice of x_{j+1} in line 5 of the procedure. With this, the proof of Theorem 6 is complete. ∎

It is interesting to observe that the greedy procedure above fails to work for interval graphs which are not indifference graphs. Consider the graph H featured in Figure 2. The ordering $<*$ of the vertices of H specified in Proposition 4 is $a <* b <* c <* d <* e <* f$. Now with vertices a and f as input parameters, Shortest_Path will return a, c, e, d, f which is not the shortest path between a and f.

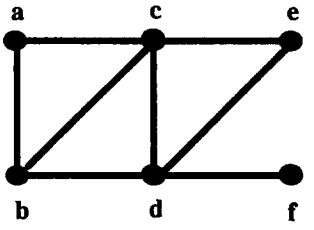

Figure 2. An interval graph with a canonical ordering $a <* b <* c <* d <* e <* f$.

The notion of a *center* in a graph is motivated by a large class of problems collectively referred to as *facility-location* problems. Here, one is interested in identifying a subset of the vertices of the graph at which certain facilities (such as police stations, shopping centers, hospitals, schools, etc.) are to be located in such a way that for every vertex in the graph, the distance to the nearest facility is minimized.

More formally, given a connected graph $G = (V, E)$, the *distance* $d(u, v)$ between vertices u and v is the smallest number of edges in a path joining u and v. The *diameter* and the *radius* of G are defined as

$$\operatorname{diam}(G) = \max_{u,v \in V} d(u, v)$$

$$r(G) = \min_{u \in V} \max_{v \in V} d(u, v).$$

Finally, the *center* of G is defined as

$$C(G) = \{u \in V \mid \max_{v \in V} d(u, v) = r(G)\}.$$

It is both well-known and easy to see that the center of an arbitrary graph $G = (V, E)$ with n vertices and m edges can be computed by the following brute-force approach: perform a breadth-first search of G starting, in turn, at every vertex of G. Clearly, this procedure takes $O(mn)$ time.

Theorems 2 and 5 seem to imply that in a connected indifference graph, the diameter is realized by vertices v_1 and v_n. This is indeed the case as shown by the following result.

COROLLARY 6.1. *In a connected indifference graph G with a canonical ordering $\leqslant$,* $\operatorname{diam}(G) = d(v_1, v_n)$.

PROOF. To settle this claim, we prove that

$$\text{for all vertices } x, y \text{ with } x \leqslant y,\ d(x, y) \leq d(v_1, v_n).$$

Let $v_1 = w_1, w_2, \ldots, w_r = v_n$ be the shortest path between v_1 and v_n. Choose the largest i such that $x \not\leqslant w_i$; similarly, find the smallest j such that $w_j \not\leqslant y$.

Note that by (2),

$$x, w_{i+1}, w_{i+2}, \ldots, w_{j-1}, y$$

is a path in G joining x and y. Since this path contains exactly $j - i$ edges, it follows that

$$d(x, y) \leq j - i \leq r - 1 = d(v_1, v_n).$$

This completes the proof of Corollary 6.1. ∎

Let G be an indifference graph and let

$$v_1 = w_0, w_1, \ldots, w_r = v_n \tag{6}$$

be the path returned by procedure Shortest_Path invoked with parameters v_1 and v_n. For the purpose of computing the center of G, we use an array $D[1 \ldots n]$ that stores, for all i $(1 \leq i \leq n)$ the shortest distance from v_1 to v_i. This is accomplished by the following simple procedure.

PROCEDURE. Find_Distance(G);
{Input: an indifference graph $G = (V, E)$ with $|V| = n$ and $|E| = m$, and a path as in (6);
Output: an array $D[1 \ldots n]$ such that for all i, $D[i] = d(v_1, v_i)$}
0. **begin**
1. **for** $i \leftarrow 2$ **to** n **do**
2. **if** $v_i = w_j$ $(0 \leq j \leq r)$ **then**
3. $D[i] \leftarrow j$
4. **else begin**
5. find the largest subscript j for which $w_j \lessdot v_i$;
6. $D[i] \leftarrow j + 1$
7. **end**
8. return(D)
9. **end**; {Find_Distance}

THEOREM 7. *When procedure Find_Distance terminates, for every i $(1 \leq i \leq n)$, $D[i] = d(v_1, v_i)$. Furthermore, the running time is bounded by $O(n)$.*

PROOF. The correctness follows easily by a trivial inductive argument. To argue for the complexity, we note that by scanning the vertices left to right, we maintain the largest j for which w_j is the last vertex on the path (P) encountered thus far. It follows that the overall complexity is $O(n)$, as claimed. ∎

Once the array D is available, we only need repeat the above procedure starting from v_n: specifically, we invoke the procedure Shortest_Path with parameters v_n and v_1 and compute a shortest path

$$v_n = z_0, z_1, \ldots, z_r = v_1.$$

Next, using an array $D'[1 \ldots n]$, record, by invoking the procedure Find_Distance, the distance from every vertex to v_n. Finally, the center C of G contains exactly those vertices v_i for which $|D[i] - D'[i]| \leq 1$. Hence, we have the following result.

THEOREM 8. *The center of an indifference graph can be computed in $O(n)$ time once a canonical ordering $\lessdot$ of the vertices of G is available.* ∎

A Hamiltonian path in a graph G is a path which contains every vertex of the graph once and only once. The structure of indifference graphs makes it possible to devise an extremely simple greedy algorithm to compute a Hamiltonian path. For this purpose we assume that a *connected* indifference graph G is given along with a canonical ordering $\lessdot$ of its vertices. The details are spelled out by our next procedure.

PROCEDURE. Hamiltonian_Path(G);
{Input: a connected indifference graph G with a canonical ordering $\lessdot$;
Output: a Hamiltonian path P of G}
0. **begin**
1. $P \leftarrow \varnothing$;
2. **for** $i \leftarrow 1$ **to** $n - 1$ **do**
3. add the edge $v_i v_{i+1}$ to P;
4. return(P)
5. **end**; {Hamiltonian_Path}

THEOREM 9. *Let G be a connected n-vertex indifference graph with a canonical ordering $\leqslant$. Procedure Hamiltonian_Path correctly returns a Hamiltonian path P in $O(n)$ time.*

PROOF. To address the correctness, note that by Corollary 3.1, v_i and v_{i+1} are adjacent, for all $i = 1, 2, \ldots, n-1$. Consequently, the set P of edges returned by the procedure is a path in G. To see that P is a Hamiltonian path, note that P contains every vertex once.

The complexity is clearly $O(n)$ once the ordering $\leqslant$ is known. ∎

It is interesting to note that the above procedure fails to return a Hamiltonian path in an interval graph which is not an indifference graph. Put differently, with an ordering $<*$ satisfying (1) (but not (2)), procedure Hamiltonian_Path is not guaranteed to work. To see this, consider again the graph in Figure 2 with the ordering $a <* b <* c <* d <* e <* f$. The procedure returns ab, bc, cd, de, ef which is obviously incorrect since e and f are not even adjacent. (A correct Hamiltonian path on this graph is ba, ac, ce, ed, df.)

A *matching* in a graph is a set of edges with the property that no two of them share a common endpoint. A matching is *maximum* if it is as large as possible. The *matching problem* is to find a maximum matching of a given graph G. The literature on matching is extensive. Matching problems are related to flow problems, covering problems, and scheduling, to name just a few (the interested reader is referred to [10] where many other applications are summarized).

As it turns out, the canonical ordering of an indifference graph can be used to obtain a very simple algorithm to compute a maximum matching. The details are as follows.

```
PROCEDURE. Maximum_Matching(G);
{Input: an indifference graph G with a canonical ordering ⩽;
 Output: a maximum matching M of G}
0. begin
1.    M ← ∅;
2.    for i ← 1 to n − 1 step 2 do
3.       if v_i and v_{i+1} are adjacent then
4.          add the edge v_i v_{i+1} to M;
5.    return(M)
6. end; {Hamiltonian_Path}
```

THEOREM 10. *Let G be a connected n-vertex indifference graph with a canonical ordering $\leqslant$. Procedure Maximum_Matching correctly returns a maximum matching in G in $O(n)$ time.*

PROOF. To address the correctness, consider a connected component H of G and let

$$v_i \leqslant v_{i+1} \leqslant \cdots \leqslant v_t$$

be the restriction of $\leqslant$ to H. Note that by Corollary 1.1, v_j and v_{j+1} are adjacent to all values of $j = i, i+1, \ldots, t-1$. Therefore, the set M_H of edges returned by the procedure is a matching in H. To see that M_H is maximal, note that at most one vertex in H is exposed (i.e., not incident to an edge in M_H). Now the conclusion follows from the observation that a maximum matching in G is the union of maximum matching in every component of G.

The complexity is clearly $O(n)$, and the proof of Theorem 10 is complete. ∎

We note that procedure Maximum_Matching fails to return a maximum matching in an interval graph which is not an indifference graph. To see this, consider the graph in Figure 2 with the ordering $a <* b <* c <* d <* e <* f$. Note that this ordering does not satisfy (2). The procedure returns ab, cd which is clearly not a maximum matching. (A maximum matching is ab, ce, df.)

REFERENCES

1. F.S. Roberts, *Graph Theory and Its Applications to Problems of Society*, SIAM Press, Philadelphia, PA, (1978).
2. F.S. Roberts, Indifference and seriation, *Ann. N. Y. Acad. Sci.* **328**, 173–182 (1979).
3. M.C. Golumbic, *Algorithmic Graph Theory and Perfect Graphs*, Academic Press, New York, (1980).

4. C.G. Lekkerkerker and J.C. Boland, Representations of a finite graph by a set of intervals on the line, *Fund. Math.* **51**, 45–64 (1962).
5. P.C. Gilmore and A.J. Hoffman, A characterization of comparability graphs and interval graphs, *Canad. J. of Math.* **16**, 539–548 (1964).
6. K.S. Booth and G.S. Lueker, Testing for the consecutive ones property, interval graphs, and planarity testing using *PQ*-tree algorithms, *J. Comput. Systems Sci.* **13**, 335–379 (1976).
7. G.S. Lueker and K.S. Booth, A linear time algorithm for deciding interval graph isomorphism, *Journal of the ACM* **26**, 183–195 (1979).
8. S. Olariu, An optimal greedy heuristic to color interval graphs, *Information Processing Letters* **37**, 21–25 (1991).
9. M.M. Syslo, N. Deo and J.S. Kowalik, *Discrete Optimization Algorithms*, Prentice-Hall, (1983).
10. D.R. Luce, Semiorders and the theory of utility discrimination, *Econometrica* **24**, 178–191 (1956).