Graph Layout Using Queues

Lenwood S. Heath and Arnold L. Rosenberg

TR 89-45

GRAPH LAYOUT USING QUEUES

Lenwood S. Heath
Department of Computer Science
Virginia Polytechnic Institute
Blacksburg, VA 24061

Arnold L. Rosenberg
Department of Computer and Information Science
University of Massachusetts
Amherst, MA 01003

December 21, 1989

Abstract

We study the problem of laying out the edges of a graph using queues. In a k queue layout, vertices of the graph are placed in some linear order and each edge is assigned to exactly one of the k queues so that the edges assigned to each queue obey a first-in/first-out discipline. This layout problem abstracts a design problem of fault-tolerant processor arrays and a problem of sorting with parallel queues. We relate the queue layout problem to the corresponding stack layout problem using stacks (the book embedding problem) and immediately derive some asymptotic bounds for d-valent graphs. We show that every 1-queue graph is a 2-stack graph and that every 1-stack graph is a 2-queue graph. We characterize the 1-queue graphs (they are almost leveled-planar graphs). We prove that the problem of recognizing 1-queue graphs is NP-complete. We give some queue layouts for specific classes of graphs. Relationships to cutwidth, bandwidth, and bifurcators are presented. We show a tradeoff between queuenumber and stacknumber for a fixed linear order of the vertices of G.

1. Introduction

A recurring theme in computer science is the comparison of the relative powers and properties of queues and stacks in a variety of computational situations. As just one significant and relevant example, Tarjan [T72] and Even and Itai [EI71] study the problem of permuting objects using queues and stacks. In this paper, we study the problem of laying out graphs using queues. We have three motivations for this study. First, queues and stacks are dual data structures, so the queue layout problem is dual to the stack layout problem, more commonly known as the book embedding problem (Bernhart and Kainen [BK79]). Our second motivation is the DIOGENES approach to the problem of designing fault-tolerant arrays of VLSI processors ([Ro83]); either queues or stacks (or both) can be incorporated into a fault-tolerant design. A third motivation is a proposed queuing solution to scheduling problems in parallel operating systems. Throughout, we contrast each result about laying out graphs using queues with the analogous result about laying out graphs using stacks.

In DIOGENES, an array of communicating processors is implemented in a conceptual line and some number of hardware queues and/or stacks pass over the entire line. The queues and/or stacks implement the communication links among processors in such a way that faulty processors are ignored, and all good processors are utilized. If the processors and their connections are represented by an undirected graph, then the DIOGENES layout problem is equivalent to a graph layout problem, where edges are assigned to conceptual queues and/or stacks. The variant of DIOGENES in which only stacks are used is one motivation for the studies of the book embedding problem: Bernhart and Kainen [BK79], Buss and Shor [BS84], Chung, Leighton and Rosenberg [CLR87], Heath [He84,He85], and Yannakakis [Y86,Y89]. This research intends to investigate the same issues for queues as that study does for stacks. In particular, we show parallels between queue and stack layouts in some asymptotic results, but we also find significant points of departure between the two layouts.

A k-queue layout of a graph $G = (V, E)$ has two aspects. The first aspect is a linear order of V (which we will think of as being on a horizontal line). The second aspect is an assignment of each edge in E to one of k queues such that the set of edges assigned to each queue obeys a first-in/first-out discipline. Think of scanning the vertices in

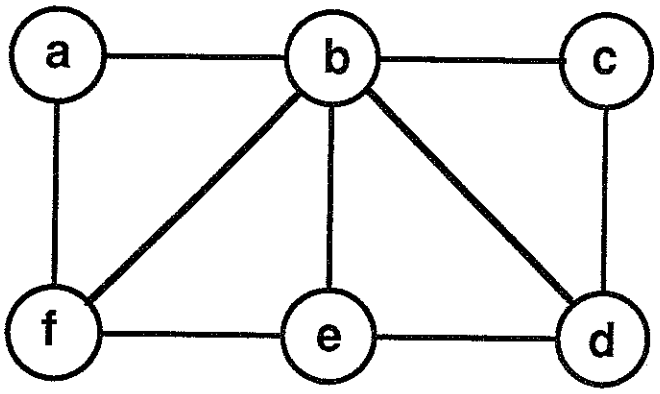

Figure 1.1: Example graph G.

order from left to right. When the left vertex of an edge is encountered, the edge enters its assigned queue (at the back of the queue). When the right vertex of an edge is encountered, the edge exits its assigned queue (and must, therefore, be at the front of the queue). If a queue is examined at any instant, the edges in the queue are in the order of their right vertices, with the leftmost of those right vertices belonging to edges at the head of the queue. The freedom to choose the order of V and the assignment of E so as to optimize some measure of the resulting layout constitutes the essence of the queue layout problem.

As an example of a 1-queue layout, consider the graph G in Figure 1.1. A 1-queue layout of G is shown in Figure 1.2. The linear order of V is a, f, b, e, c, d. The order in which edges pass through the single queue is

$$(a, f), (a, b), (f, b), (f, e), (b, e), (b, c), (b, d), (e, d), (c, d).$$

Note that edges having the same left vertex enter the queue in an order determined by their right vertices. For example, edge (a, f) must enter the queue before edge (a, b) since f is to the left of b.

The queue layout problem also generalizes the problem of permuting a sequence using parallel queues that was studied by Even and Itai [EI71] and Tarjan [T72]. Let

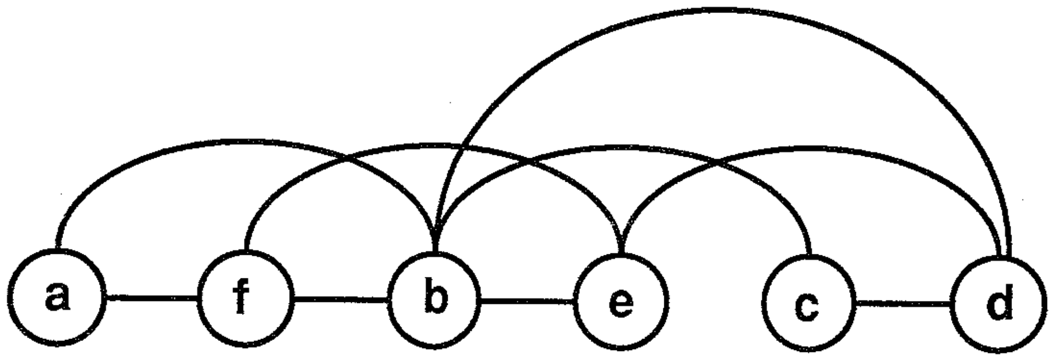

Figure 1.2: 1-queue layout.

π be a permutation defined on $\{1, \ldots, n\}$. Define the bipartite graph G by

$$\begin{aligned} V &= \{a_1, \ldots, a_n, b_1, \ldots, b_n\} \\ E &= \{(a_i, b_i) \mid 1 \leq i \leq n\}. \end{aligned}$$

Then realizing π by k parallel queues is equivalent to laying G out using k queues when V is ordered $a_1, \ldots, a_n, b_{\pi(1)}, \ldots, b_{\pi(n)}$.

A k-stack layout of G also has two aspects. The first aspect is a linear order of V. The second aspect is an assignment of each edge in E to one of k stacks such that the set of edges assigned to each stack obeys a last-in/first-out discipline. Unlike a queue layout, edges do not exit a stack in the same order in which they enter it. As an example, Figure 1.3 shows a 1-stack layout of the graph G in Figure 1.1. The linear order of V is a, b, c, d, e, f. The order in which edges enter the stack is

$$(a, f), (a, b), (b, f), (b, e), (b, d), (b, c), (c, d), (d, e), (e, f).$$

The order in which edges exit the stack is

$$(a, b), (b, c), (c, d), (b, d), (d, e), (b, e), (e, f), (b, f), (a, f).$$

Book embedding is a related problem. A book consists of a spine (a line) and some number of pages (half-spaces having the spine as boundary). A graph is embedded in

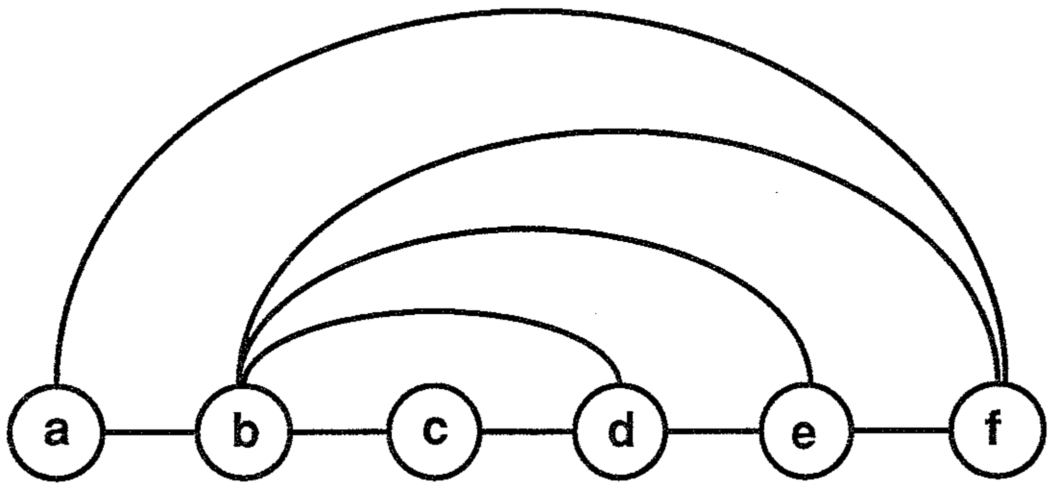

Figure 1.3: 1-stack layout.

a book by placing its vertices along the spine in some order and assigning each edge to a single page such that each edge can be drawn in its assigned page with no two edges in the same page intersecting. Chung, Leighton, and Rosenberg [CLR87] have shown that the book embedding problem is equivalent to the stack layout problem. We will refer to the literature on book embeddings to reveal similarities and differences between queue and stack layouts.

2. Basics

We start with formal definitions of queue and stack layouts and their associated cost measures. We develop results on fixed-order queue and stack layouts, including an optimal and efficient algorithm for fixed-order queue layouts.

2.1. Definitions

A *k-queue layout* QL of an undirected graph $G = (V, E)$ consists of a linear order of V and an assignment of each edge in E to exactly one of k queues, $q_1, \ldots, q_k$. We use

$\sigma = 1, 2, \ldots, n$ to denote the order. Each queue q_j operates as follows. The vertices of V are scanned in left-to-right (ascending) order. When i is encountered, any edges assigned to q_j that have i as their right endpoint must be at the front of the queue and are removed (dequeued). Any edges assigned to q_j that have i as left vertex are placed on the back of the queue (enqueued) in ascending order of their right vertices. k is the *queuenumber* of the layout. The *queuenumber* of G, $\mathrm{QN}(G)$, is the smallest k such that G has a k-queue layout; G is said to be a *k-queue* graph. Let $w(i, q_j)$ be the number of edges in q_j just before i is encountered. Then the *queuewidth* of q_j is $\mathrm{QW}(q_j) = \max_{i \in V} w(i, q_j)$. The *maximum queuewidth* of the layout is $\mathrm{QW}(QL) = \max_i \mathrm{QW}(q_j)$. The *cumulative queuewidth* of the layout is $\mathrm{CQW}(QL) = \sum_j \mathrm{QW}(q_j)$.

A *k-stack layout SL* of an undirected graph consists of a linear order of V and an assignment of each edge in E to exactly one of k stacks, $s_1, \ldots, s_k$. Each stack s_j operates as follows. The vertices of V are scanned in left-to-right (ascending) order. When i is encountered, any edges assigned to s_j that have i as their right endpoint must be on the top of the stack and are removed (popped). Any edges assigned to s_j that have i as left vertex are placed on the top of the stack (pushed) in descending order of their right vertices. k is the *stacknumber* of the layout. The *stacknumber* of G, $\mathrm{SN}(G)$, is the smallest k such that G has a k-stack layout; G is said to be a *k-stack* graph. Let $w(i, s_j)$ be the number of edges in s_j just before i is encountered. Then the *stackwidth* of s_j is $\mathrm{SW}(s_j) = \max_{i \in V} w(i, s_j)$. The *maximum stackwidth* of the layout is $\mathrm{SW}(SL) = \max_i \mathrm{SW}(s_j)$. The *cumulative stackwidth* of the layout is $\mathrm{CSW}(SL) = \sum_j \mathrm{SW}(s_j)$.

2.2. Fixed-Order Layouts

In this subsection, we fix an order $\sigma = 1, 2, \ldots, n$ of V and examine the difficulty of minimizing the number of queue or stacks required to complete σ to a layout. We concentrate on sets of edges that are obstacles to minimizing the number of stacks or queues. A *k-rainbow* is a set of k edges

$$\{e_i = (r_i, s_j),\, 1 \leq i \leq k\}$$

such that

$$r_1 < r_2 < \cdots < r_{k-1} < r_k < s_k < s_{k-1} < \cdots < s_2 < s_1;$$

in other words, a rainbow is a *nested* matching. A *k-twist* is a set of k edges

$$\{e_i = (r_i, s_j)\,,\, 1 \le i \le k\}$$

such that

$$r_1 < r_2 < \cdots < r_{k-1} < r_k < s_1 < s_2 < \cdots < s_{k-1} < s_k;$$

in other words, a twist is a fully intersecting matching.

A rainbow is an obstacle for a queue layout because no two nested edges can be assigned to the same queue.

Proposition 2.1 *Suppose σ has a k-rainbow. Then there is no queue layout of σ with fewer than k queues. There exists a stack layout of σ in which all edges of the k-rainbow are assigned to the same stack.*

A twist is an obstacle for a stack layout because no two intersecting edges can be assigned to the same stack.

Proposition 2.2 *Suppose σ has a k-twist. Then there is no stack layout of σ with fewer than k stacks. There exists a queue layout of σ in which all edges of the k-twist are assigned to the same queue.*

The largest rainbow in σ determines the smallest number of queues needed in a queue layout of σ.

Theorem 2.1 *If σ has no rainbow of more than k edges, then there is a k-queue layout for σ. Such a layout can be found in time $O(|E| \log n)$.*

Proof: We describe an algorithm for assigning edges to queues. Maintain an array R indexed by integers $0..n$. For $1 \le i \le n$, array entry $R[i]$ contains the larger of 0 and the index of the rightmost vertex of any edge that has already been assigned to the queue q_i. The algorithm maintains the invariant that nonzero entries in R are in strictly decreasing order ($R[i] < R[i-1]$, $1 \le i \le n$, if $R[i-1] > 0$). A suitable initialization for R assigns $R[0] = n + 1$ and $R[i] = 0$, otherwise; then R satisfies the

show that every 1-queue graph is a 2-stack graph and that every 1-stack graph is a 2-queue graph.

3.1. Characterizing 1-queue Graphs

The characterization of 1-stack graphs was given in [BK79].

Proposition 3.1 *G is a 1-stack graph if and only if G is outerplanar.*

(An *outerplanar* graph is a planar graph having a planar embedding in which all vertices appear on a common face.) We show that the 1-queue graphs are also planar graphs that have a particular kind of planar embedding.

Consider the normal cartesian (x, y) coordinate system for the plane. For i an integer, let ℓ_i be the vertical line defined by $\ell_i = \{(i, y) \mid y \in \mathbf{R}\}$. A graph $G = (V, E)$ is a *leveled-planar graph* if V can be partitioned into levels $V_1, V_2, \ldots, V_m$ and G can be embedded in the plane such that all vertices of V_i are on the line ℓ_i, each edge in E is embedded as a straight line segment wholly between ℓ_i and ℓ_{i+1} for some i, and the embedding is a valid planar embedding for G (i.e., no edges cross). Figure 3.1 shows a leveled-planar graph having 3 levels. Note that the leveled-planar embedding of a leveled-planar graph is not unique. Henceforth, we assume that an arbitrary leveled-planar embedding is given along with a leveled-planar graph.

A leveled-planar embedding induces an order (the *induced order*) on V as follows. As i takes the values $1, 2, \ldots, m$, scan line ℓ_i from bottom to top. Label the vertices $1, 2, \ldots, n$ as they are encountered. For $1 \leq i \leq m$, let b_i be the (bottom) first vertex in level i, and let t_i be the (top) last. Let s_i be the first vertex in level i that is adjacent to some vertex in level $i + 1$, or, if there are no edges between levels i and $i + 1$, let $s_i = t_i$. Consider augmenting G with new edges. A *level i arch* for G is an edge connecting t_i with j, where $b_i \leq j \leq \min(t_i - 1, s_i)$. A leveled-planar graph G, augmented by any number of arches, can be embedded in the plane by drawing the arches around level 1; because of the leveling, the arches do not cross. See Figure 3.2 where $(3, 5)$ and $(6, 8)$ are arches. A leveled-planar graph augmented by (zero or more) arches is called an *arched leveled-planar graph.* The edges that are not arches are called *leveled edges.* An arched leveled-planar graph that cannot be augmented with further arches or leveled

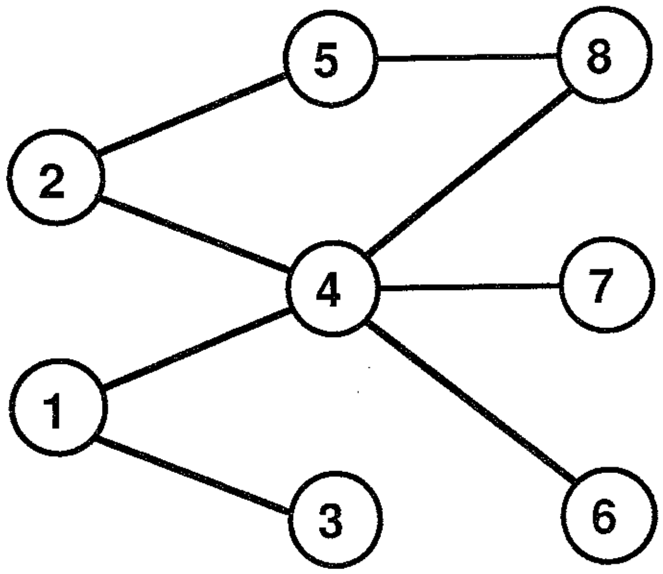

Figure 3.1: A leveled-planar graph.

edges is *maximal.* See Figure 3.3 for an example. The above definitions for b_i, s_i, and t_i will be used throughout the paper to refer to vertices in arched leveled-planar graphs.

We can now state the characterization of 1-queue graphs.

Theorem 3.1 *A graph G is a 1-queue graph if and only if G is an arched level planar graph.*

We develop the proof of the theorem through three lemmas.

Lemma 3.1 *Every leveled-planar graph is a 1-queue graph. The induced order of vertices yields a 1-queue layout.*

Proof: Given a leveled-planar graph $G = (V, E)$ with m levels $V_1, V_2, \ldots, V_m$, order V in the induced order $1, \ldots, n$. We claim that this order yields a 1-queue embedding of G. It suffices to show that no two edges nest. If two edges have a vertex in common, then the edges cannot nest. So consider two edges (p_1, q_1) and (p_2, q_2) such that $p_1 < q_1$, $p_2 < q_2$, $p_1 < p_2$, and $q_1 \neq q_2$. If p_1 and p_2 are in the same level V_i, then q_1 and q_2 are in the same level V_{i+1}, and $q_1 < q_2$ because the edges do not intersect. If p_1 and p_2 are

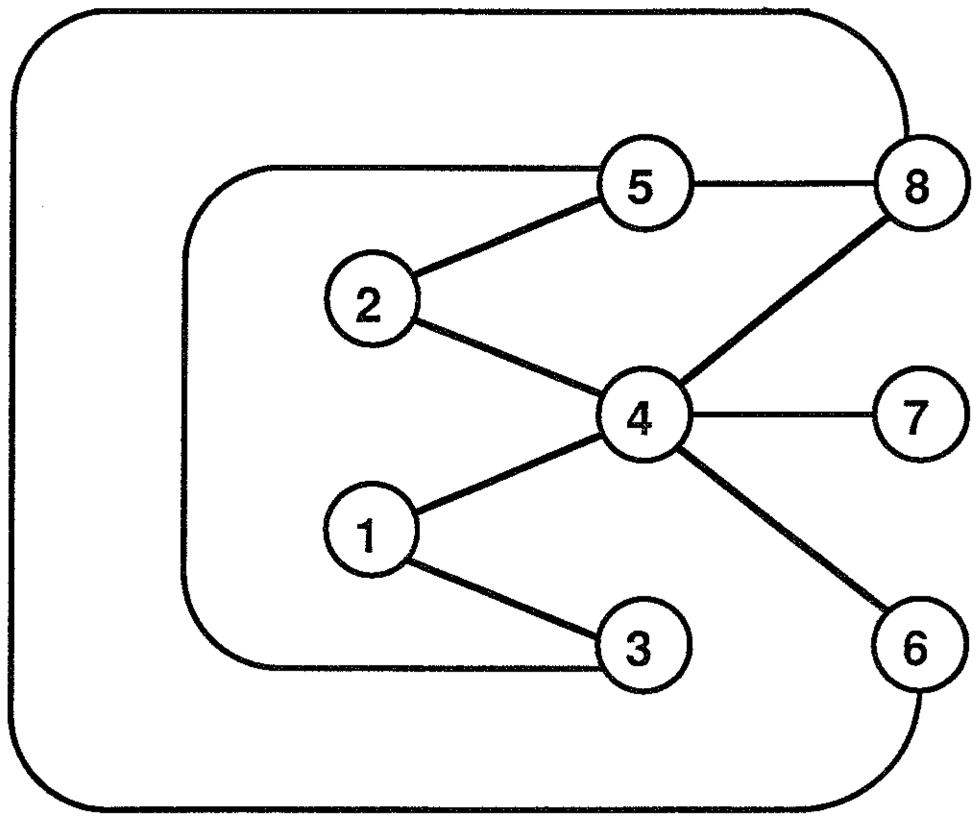

Figure 3.2: Drawing arches.

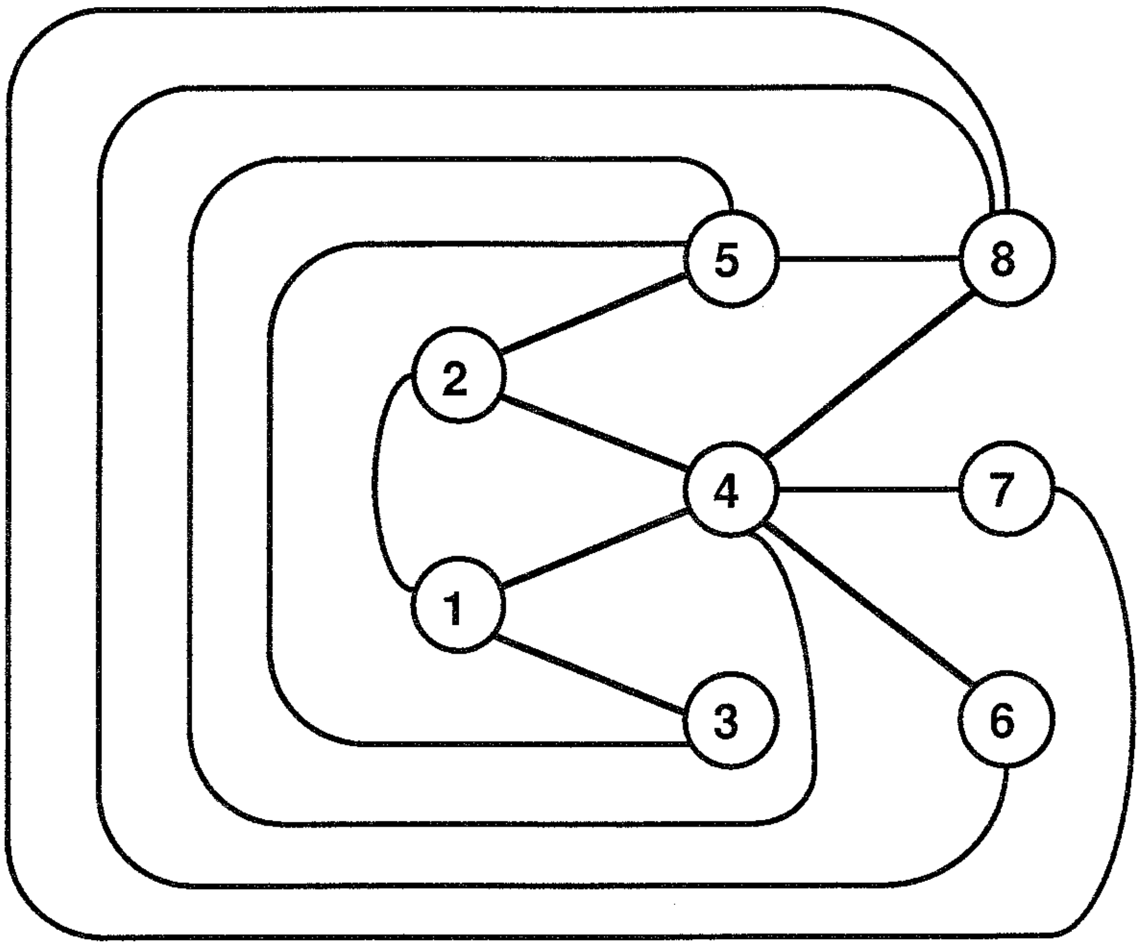

Figure 3.3: A maximal arched leveled-planar graph.

First construct a leveled-planar embedding of $G_\ell = (V, E_\ell)$. Place the vertices of V_i on line ℓ_i in the order $b_i, b_i + 1, \ldots, t_i$ from bottom to top. Draw the edges in E_ℓ as line segments. Two of these line segments might cross only if they end on two adjacent lines ℓ_i and ℓ_{i+1}. Let (p_1, q_1), $p_1 < q_1$ and (p_2, q_2), $p_1 < q_2$ be the edges corresponding to two line segments that cross. Then $p_1 < p_2 < q_2 < q_1$, that is, the edges nest. That is a contradiction to σ being the order of a 1-queue embedding of G.

It remains to show that $E - E_\ell$ contains only arches for G_ℓ. Let $(p_3, q_3) \in E - E_\ell$, where $p_3, q_3 \in V_i$, $p_3 < q_3$. Clearly, $p_3 \leq \min(t_i - 1, s_i)$ since otherwise there is an edge from s_i to some vertex in V_{i+1} that nests over (p_3, q_3). Since t_i is adjacent to some vertex $x \in V_{i-1}$, we must have $q_3 = t_i$, for otherwise (x, t_i) and (p_3, q_3) nest. We conclude that $E - E_\ell$ contains only arches and that we have constructed an arched level planar embedding of G. □

Theorem 3.1 follows from Lemmas 3.2 and 3.3.

It is well known that a maximal outerplanar graph on n vertices contains $2n - 3$ edges. A similar result is now shown for maximal arched leveled-planar graphs.

Theorem 3.2 *Let $G = (V, E)$ be a graph on n vertices having a maximal arched leveled-planar embedding of m levels. Suppose that f of the levels $V_2, \ldots, V_{m-1}$ are singletons. Then G has at most $2n - 1 - |V_1| - f$ edges.*

Proof: Partition E into levels $E_1, \ldots, E_m$, where an edge is in level E_i if its left vertex is in V_i. Then all level i arches are in E_i, and V_m contains only arches. For convenience, let $t_0 = 0$.

By the maximality of the embedding: (a) E_i contains $s_i - t_{i-1}$ arches if $t_i \neq s_i$ and $s_i - t_{i-1} - 1$ arches if $t_i = s_i$; (b) if $t_i \neq b_i$, then there is a leveled edge from $t_i - 1$ to level $i + 1$ and hence $s_i \neq t_i$. Thus E_i contains $s_i - t_{i-1} - 1 = 0$ arches only when V_i is a singleton level.

Each leveled edge in E_i, $1 \leq i \leq m - 1$, has one endpoint among $t_i - s_i + 1$ vertices in level i and one endpoint among $t_{i+1} - t_i$ vertices in level $i + 1$. By planarity, there is a bottom-to-top order on the set of leveled edges in E_i. Scanning these edges in order, in different levels, then q_1 and q_2 are in different levels, and again $q_1 < q_2$. In either case, the two edges do not nest. Hence, the given layout is a 1-queue layout of G. □

Lemma 3.2 *Every arched leveled-planar graph is a 1-queue graph. The induced order of vertices yields a 1-queue layout.*

Proof: Let $G = (V, E)$ be an arched leveled-planar graph. By the previous lemma, it suffices to show that no arch nests with another edge.

Let (p_1, t_i) and (p_2, t_j) be two arches. If $t_i = t_j$, then the arches do not nest since they have a vertex in common. If $t_i \neq t_j$, say $t_i < t_j$, then $p_1 < t_i < p_2 < t_j$, and the arches do not nest.

Now suppose (p_3, q_3), $p_3 < q_3$, is a leveled edge between levels k and $k+1$. Since every arch is between two vertices on the same level, no leveled edge can nest inside an arch. For (p_1, t_i) to nest inside (p_3, q_3), we must have $k = i$ and $p_3 < p_1 \leq s_i$. By the definition of s_i, there are no leveled edges from level i vertices to the left of s_i. Thus, (p_1, t_i) and (p_3, q_3) do not nest. We conclude that we have a 1-queue layout for G. □

Lemma 3.3 *Every 1-queue graph is an arched leveled-planar graph.*

Proof: Let $G = (V, E)$ be a 1-queue graph, and let $\sigma = 1, 2, \ldots, n$ be the order of a 1-queue layout of G. It suffices to describe an arched leveled-planar embedding of G. Without loss of generality, we may assume that G is connected.

Partition V as follows. $V_1 = \{1\}$, and $b_1 = s_1 = t_1 = 1$. For $i > 1$, until each vertex is placed in some set, let t_i be the rightmost vertex incident to some vertex in V_{i-1}. Let $b_i = t_{i-1} + 1$. Let $V_i = \{b_i, \ldots, t_i\}$. Let s_i be the leftmost vertex in V_i that is adjacent to some vertex to the right of t_i; let $s_i = t_i$ if $t_i = n$.

Let the resulting partition be $V_1, V_2, \ldots, V_m$. This partition consists of breaking the sequence σ into m contiguous subsequences that end at $1 = t_1, t_2, \ldots, t_m = n$, respectively. By the construction of V_i, it is clear that there is no edge from a vertex in V_i to a vertex in V_j if $|i - j| \geq 2$. Let E_ℓ be the subset of E consisting of edges that connect vertices at consecutive levels; that is,

$$E_\ell = E \cap \bigcup_{1 \leq i < m} V_i \times V_{i+1}.$$

recognizing 1-queue graphs is NP-complete (see Garey and Johnson [GJ79]). Formally, the recognition problem for 1-queue graphs is the following decision problem.

ARCHED LEVELED-PLANAR

INSTANCE: A graph $G = (V, E)$, represented by adjacency lists.

QUESTION: Does G have an arched leveled-planar embedding?

The next recognition problem is an intermediate destination on the way to the desired NP-completeness result.

LEVELED-PLANAR

INSTANCE: A graph $G = (V, E)$, represented by adjacency lists.

QUESTION: Does G have a leveled-planar embedding?

Notice that it is not immediate that either of these problems reduces to the other.

We describe a decision problem first defined by Lichtenstein [L82]. An instance of 3-SAT [GJ79] is a boolean formula ϕ in conjunctive normal form such that each clause contains at most 3 literals. Let $\{v_1, v_2, \ldots, v_n\}$ be the variables of ϕ, and let $\{c_1, c_2, \ldots, c_m\}$ be the clauses. Each c_j is a set containing at most 3 literals, where each literal is either a variable v_i or the complement $\overline{v_i}$ of a variable; call a clause containing exactly k literals a *k-clause*. The *graph* of ϕ, $G(\phi) = (N, A)$ has vertex set

$$N = \{c_j \mid 1 \leq j \leq m\} \cup \{v_i \mid 1 \leq i \leq n\}$$

and edge set $A = A_1 \cup A_2$ where

$$\begin{aligned} A_1 &= \{(c_j, v_i) \mid v_i \in c_j \text{ or } \overline{v_i} \in c_j\} \\ A_2 &= \{(v_i, v_{i+1}) \mid 1 \leq i \leq n-1\} \cup \{(v_n, v_1)\}. \end{aligned}$$

The edges of A_2 form a cycle called the *variable cycle*. The graph in Figure 3.4 represents the graph of the formula having clauses $c_1 = \{v_1, \overline{v_2}, v_5\}$, $c_2 = \{\overline{v_3}, \overline{v_4}, v_5\}$, $c_3 = \{\overline{v_1}, v_2\}$, $c_4 = \{v_2, v_3, v_4\}$, and $c_5 = \{v_2, v_4, \overline{v_5}\}$.

Lichtenstein shows that the following restricted version of 3-SAT is NP-complete.

PLANAR 3-SAT (P3SAT)

INSTANCE: An instance of 3-SAT ϕ such that $G(\phi)$ is planar.

QUESTION: Is ϕ satisfiable?

the first edge connects two vertices, and each subsequent edge connects a new vertex to a previously encountered vertex. Thus, the number of leveled edges in E_i is

$$(t_{i+1} - t_i) + (t_i - s_i + 1) - 1 = t_{i+1} - s_i.$$

Therefore, for $1 \le i \le m-1$,

$$|E_i| = \begin{cases} t_{i+1} - t_{i-1} & \text{if } |V_i| > 1 \\ t_{i+1} - t_{i-1} - 1 & \text{if } |V_i| = 1 \end{cases}.$$

Also,

$$|E_m| \;=\; t_m - t_{m-1} - 1.$$

The cardinality of E is then

$$\begin{aligned} |E| &= \sum_{i=1}^{m} |E_i| \\ &= t_m - t_{m-1} - 1 - f + \sum_{i=1}^{m-1} (t_{i+1} - t_{i-1}) \\ &= t_m - t_{m-1} - 1 - f + t_m + t_{m-1} - t_1 \\ &= 2t_m - 1 - t_1 - f \\ &= 2n - 1 - t_1 - f \\ &= 2n - 1 - |V_1| - f \\ &\le 2n - 3 \end{aligned}$$

□

Thus the greatest number of edges that can be assigned to a single queue is $2n - 3$. This value can be used to obtain a lower bound on the queue number of a graph.

Corollary 3.1 $\mathrm{QN}(G) \ge \left\lceil \frac{|E|}{2|V|-3} \right\rceil$.

3.2. Recognizing 1-queue Graphs

The 1-stack graphs are exactly the outerplanar graphs and, therefore, can be recognized in linear time (Sysło and Iri [SI79]). In contrast, we show that the problem of

It always suffices to consider only instances such that each clause contains either 2 or 3 literals. From Lemma 1 of [L82], we may assume that $G(\phi)$ has a planar embedding such that, for each v_i, all clauses containing the literal v_i are on one side of the variable cycle, and all clauses containing the literal $\overline{v_i}$ are on the other side. Call this property of the planar embedding of $G(\phi)$ *consistency.* The planar embedding of Figure 3.4 is consistent.

While LEVELED-PLANAR is as simple a recognition problem as one could formulate for queue layouts, we show that it is NP-complete.

Theorem 3.3 *LEVELED-PLANAR is NP-complete.*

Proof: We reduce P3SAT to LEVELED-PLANAR. As LEVELED-PLANAR is easily in NP, this suffices to prove the theorem.

Let $V = \{v_1, \ldots, v_n\}$ and $C = \{c_1, \ldots, c_m\}$ be an instance of P3SAT. Fix a planar embedding of $G(\phi)$ that is consistent. We will construct an instance H of LEVELED-PLANAR, which is a biconnected planar graph.

We need some building blocks. Consider the copy of $K_{2,3}$ in Figure 3.5. Suppose this copy is in a leveled-planar embedding. Then a_1 and a_2 are exactly two levels apart, and b_1, b_2, and b_3 are all on the level in between. Further, only two of b_1, b_2, and b_3 can have any additional edges incident to them. In a leveled-planar embedding, the leveling of any copy of $K_{2,3}$ is forced. If b_1, b_2, and b_3 are made to correspond to a single vertex, a path of length 2 results. In a leveled embedding, $K_{2,3}$ differs from a path of length 2 in the sense that it cannot "bend" in the middle in order to bring the ends together; its two endpoints must appear two levels apart. We think of $K_{2,3}$ as a *rigid path of length 2.* By joining $k-1$ copies of $K_{2,3}$ in the manner illustrated in Figure 3.6 for $k = 3$, rigid paths of any length k can be obtained. In general, we call a rigid path of length k a *k-rod.* (A 1-rod is an edge.) We draw a k-rod as a thick hollow line with intermediate vertices as needed (Figure 3.7). Note that what appears to be a single intermediate vertex is actually two different vertices, one on each side of the rod.

A second building block is called a *semi-rod.* It consists of a 3-rod and a 2-rod connected by 2 edges. See Figure 3.8. A semi-rod has one degree of flexibility that a 5-rod does not have: if x is in level t and y in level $t-1$, then z is either in level $t-5$

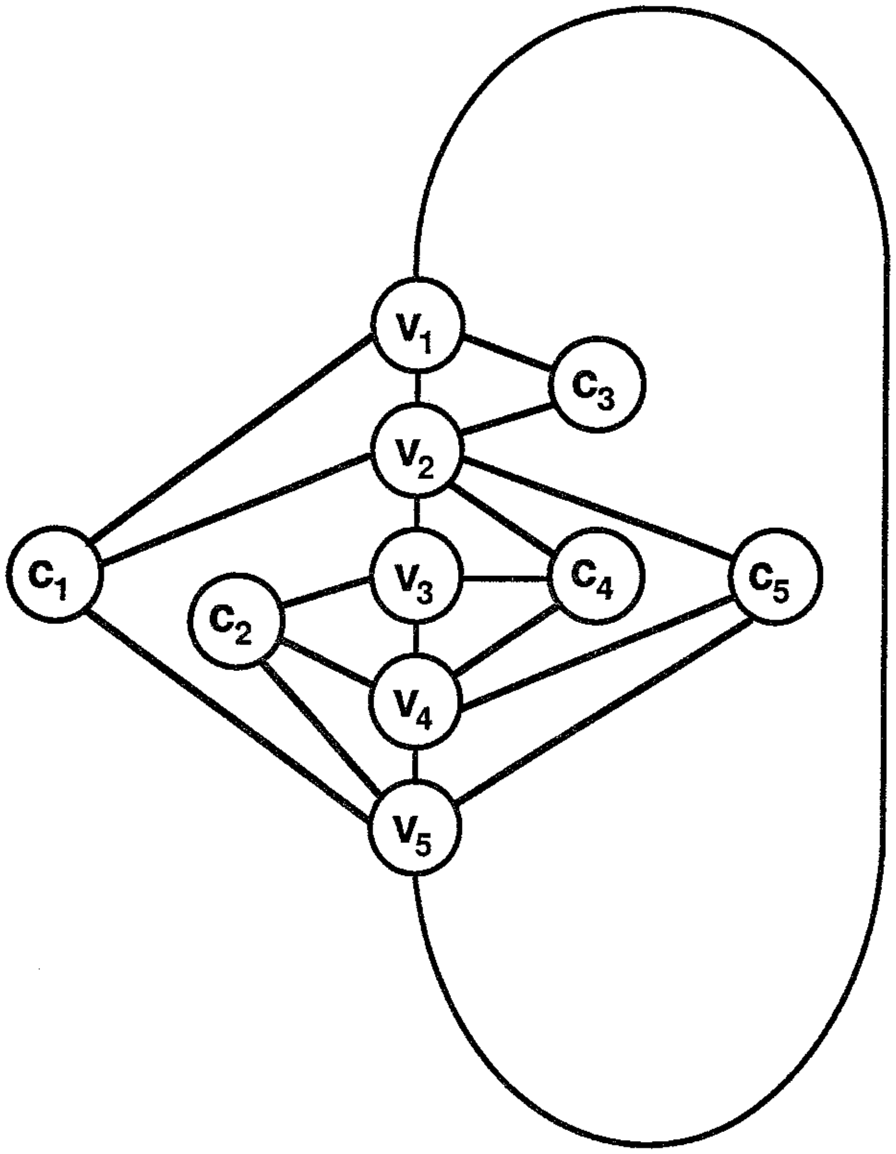

Figure 3.4: Example of Planar 3-SAT.

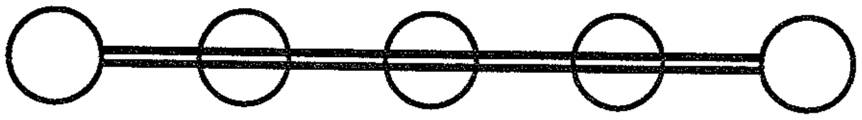

Figure 3.7: Representation of a k-rod, $k = 4$.

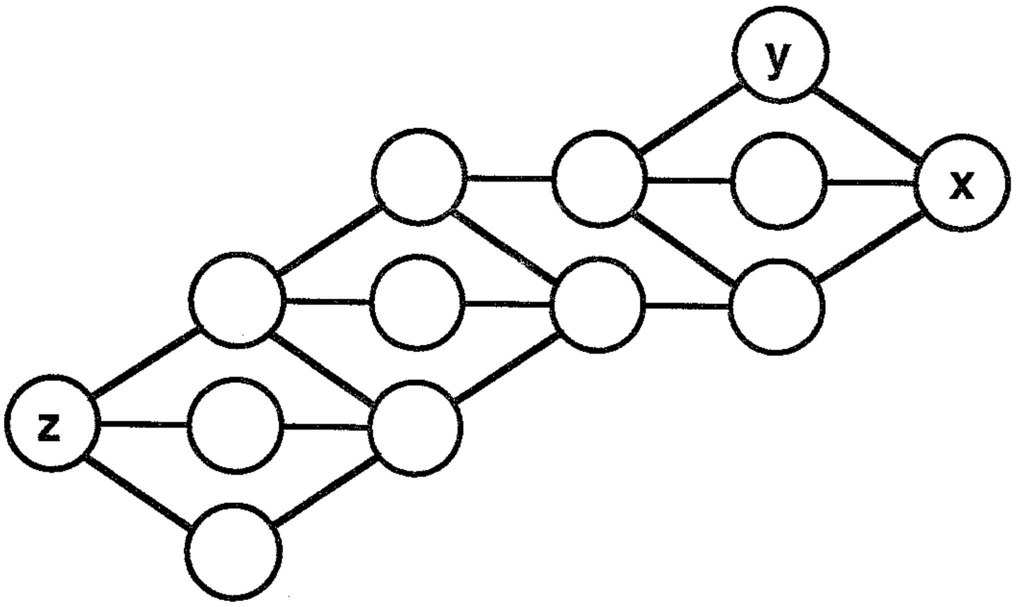

Figure 3.8: A semi-rod.

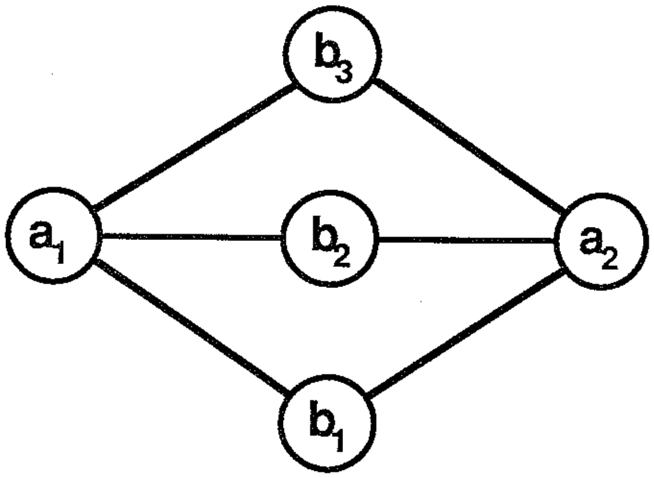

Figure 3.5: A 2-rod.

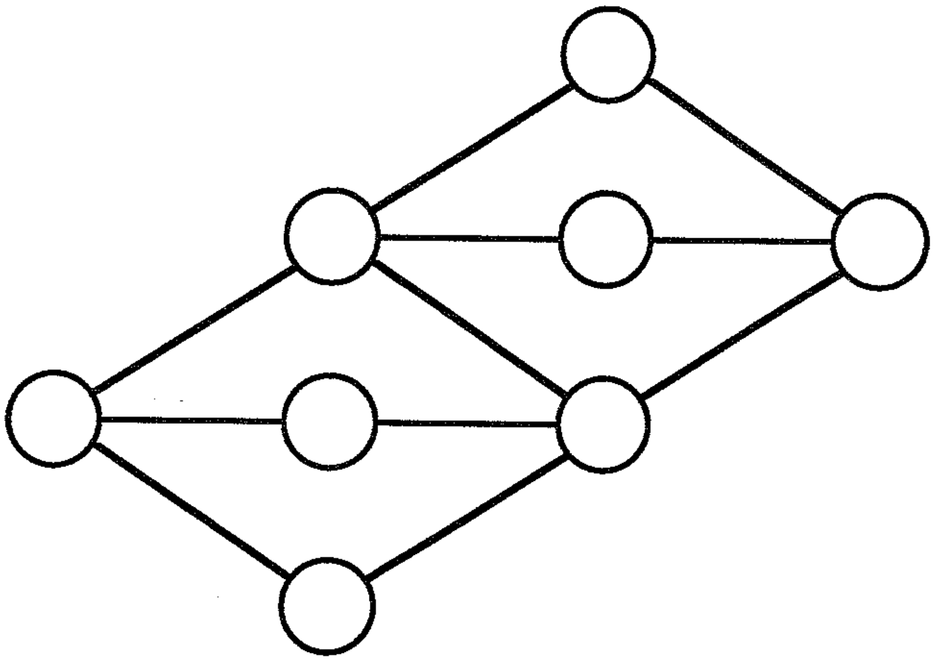

Figure 3.6: A 3-rod.

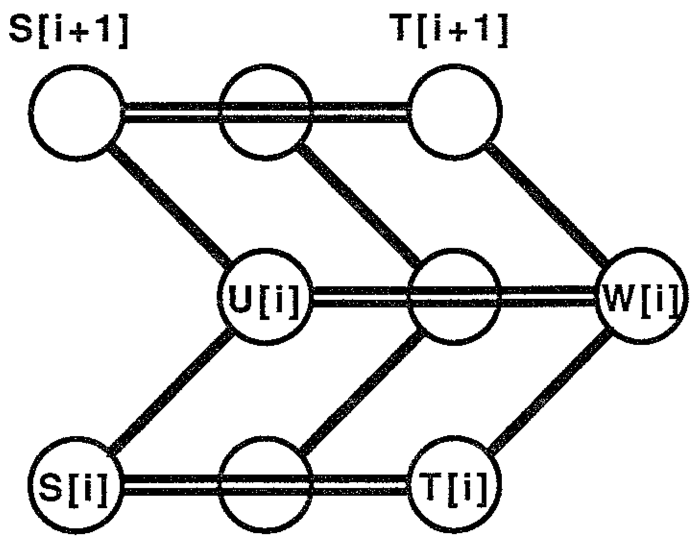

Figure 3.10: Representing the variable path.

and therefore $\mathcal{L}(V[i]) = \mathcal{L}(W[i]) - 2$, $0 \leq i \leq n$. The intention of the construction (not yet realized) is that all $W[i]$'s appear on the same level $\mathcal{L}(W[i]) = \mathcal{L}(W[0])$. For the time being, we use the level $\Lambda = \mathcal{L}(W[0])$ as a relative reference for other $\mathcal{L}$ values. Call the property of all $W[i]$'s appearing on level Λ *line up*.

Because the embedding of $G(\phi)$ is planar, the variable cycle partitions the clause set C into two subsets C_1, C_2 such that the clauses in C_1 nest and the clauses in C_2 nest. We will place the clauses in C_1 to the left of P and the clauses in C_2 to the right. (In Figure 3.4, $C_1 = \{c_1, c_2\}$ and $C_2 = \{c_3, c_4, c_5\}$. c_2 is nested under c_1, and c_4 is nested under c_5.) A clause $c_j \in C_2$ is associated with the 2 or 3 $T[i]$'s that correspond to its literals. (Similarly, a clause in C_1 is associated with 2 or 3 $S[i]$'s). Because the embedding of $G(\phi)$ is consistent, each $T[i]$ can be associated consistently with either v_i or $\overline{v_i}$ (the corresponding $S[i]$ is associated with the complementary literal). If the $W[i]$'s line up, then each $T[i]$ appears either on level $\Lambda + 1$ or $\Lambda - 1$. If $T[i]$ is on level $\Lambda - 1$, then we say that $T[i]$ is *intruded* and $S[i]$ is *extruded;* otherwise, $T[i]$ is extruded and $S[i]$ is intruded. If $T[i]$ (or $S[i]$) is intruded, we will interpret its associated literal to be true; otherwise, its associated literal is false.

We need gadgets for each clause in C. By mirror-image symmetry in P, we consider

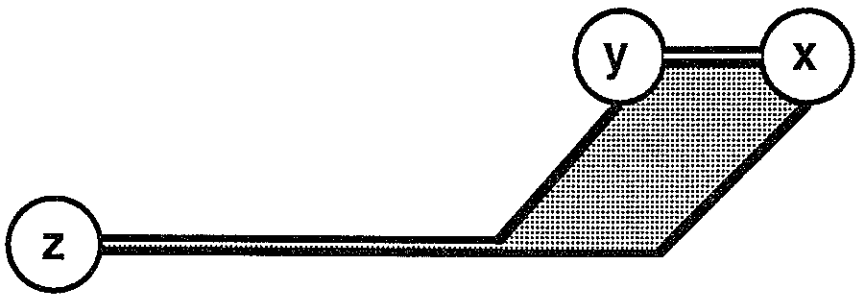

Figure 3.9: Representation of a semi-rod.

or $t-3$; note, however, that z must be at a lower level than y. We draw a semirod as a 5-rod with a textured interior (Figure 3.9).

In the construction of H, some vertices will be called *fixed*; if X is a fixed vertex, we intend that, in any leveled-planar embedding of H, the level containing X is always the same (given that a particular vertex, to be specified later, is on the first level). The level in which X should appear is its *preferred level* $\mathcal{L}(X)$. If we fix the two ends of a rod, then the intermediate vertices of the rod are also fixed, with preferred levels derived in the obvious way. During the construction we designate certain vertices X fixed and give a value to $\mathcal{L}(X)$. We show later that each fixed X indeed must appear on level $\mathcal{L}(X)$.

To begin the construction, represent each variable v_i by a 2-rod $VROD[i]$ having left and right vertices $S[i]$ and $T[i]$. Represent the edge (v_i, v_{i+1}), $1 \leq i \leq n-1$, by a 2-rod $EROD[i]$ having left vertex $V[i]$ and right vertex $W[i]$, as shown in Figure 3.10. Partially represent the edge (v_n, v_1) by a 2-rod $EROD[0]$ connected to $VROD[1]$ and by a 2-rod $EROD[n]$ connected to $VROD[n]$ (the representation of the edge will be completed later). Call the graph constructed so far P. P may be thought of as a path of thickness 3 from $EROD[0]$ to $EROD[n]$. The vertices of each $EROD$ are fixed, and the vertices of each $VROD$ are not fixed. Note that, in any leveled-planar embedding containing P, the levels of $T[i]$ and of $W[i]$ differ by exactly one level. Further, if $W[0]$ is to the right of $V[0]$, then each $W[i]$ is to the right of $V[i]$, and vice versa. By symmetry, we may assume that any leveled-planar embedding has each $W[i]$ to the right of $V[i]$,

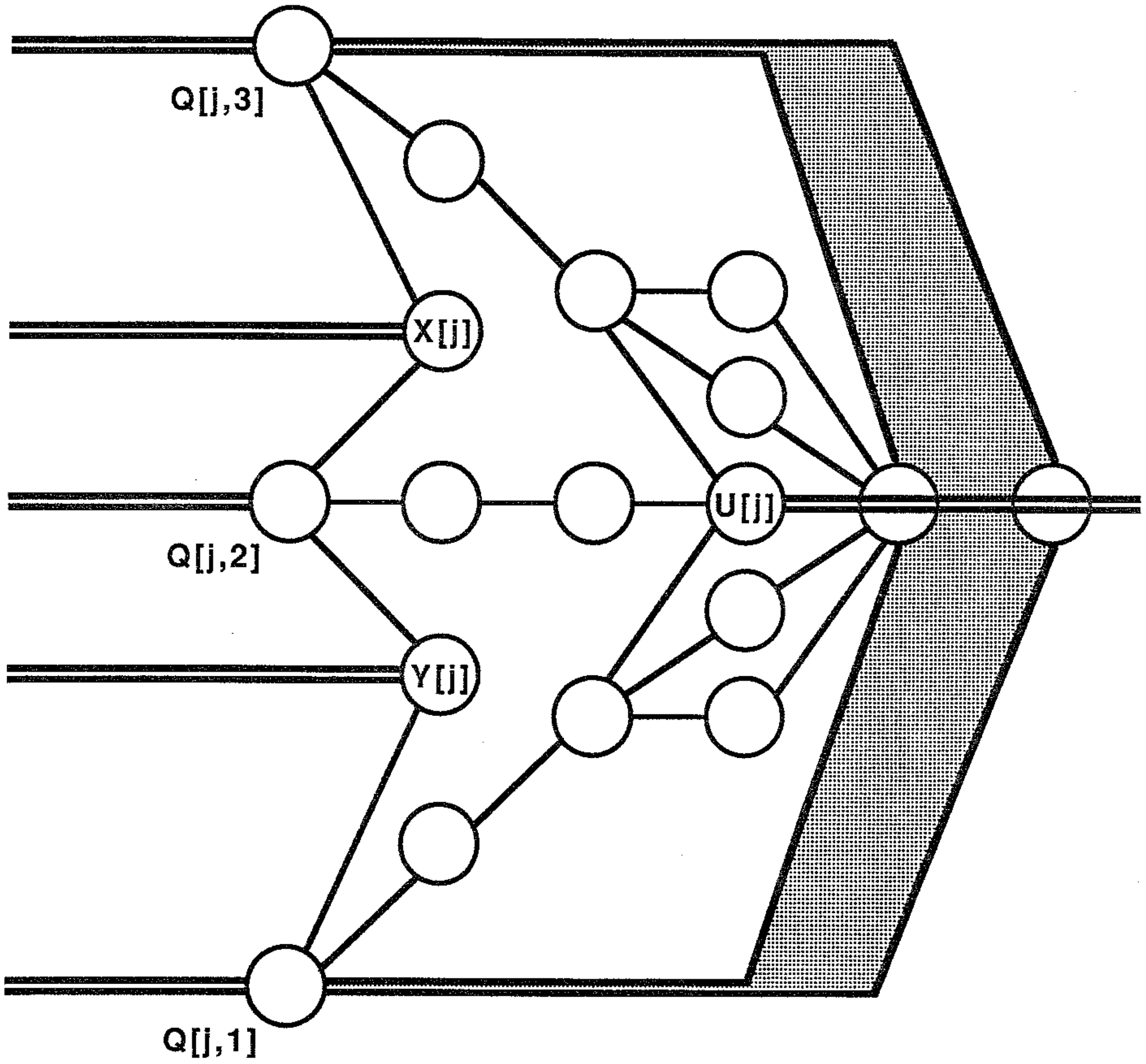

Figure 3.11: The gadget for a 3-clause.

only clauses in C_2 and place their gadgets to the right of P. Construct the gadgets for the clauses in C_2 in any order that is consistent with the nesting of clauses, taking the more deeply nested clauses earlier. We first assume that $c_j \in C_2$ is a 3-clause. Let c_j be associated with $T[i_1], T[i_2], T[i_3]$ in order from bottom to top. By the construction order, the gadgets for any clauses nested inside c_j have already been constructed. The gadget for each $c_s \in C_2$ contains a fixed vertex $U[s]$ that is visible on the right side of the gadget. If no clauses nest under c_j, then $\mathcal{L}(U[j]) = \Lambda + 4$. If there are one or more clauses nested under c_j, let c_s be one that maximizes $\mathcal{L}(U[s])$. Put $\mathcal{L}(U[j]) = \mathcal{L}(U[s]) + 4$. Let $k = \mathcal{L}(U[j]) - \Lambda - 2$. Place a k-rod on each of $T[i_1], T[i_2], T[i_3]$ and connect them to $Q[j,1], Q[j,2], Q[j,3]$, as shown in Figure 3.11. $Q[j,a]$ is intruded (extruded) exactly when $T[i_a]$ is intruded (extruded) . $X[j]$ and $Y[j]$ are fixed with $\mathcal{L}(X[j]) = \mathcal{L}(Y[j]) = \Lambda + k$. There will be $U[s]$'s or $W[i]$'s visible under $X[j]$ (or $Y[j]$). For each, connect a rod of the appropriate length from $X[j]$ (or $Y[j]$) to each $U[s]$ or $W[i]$. For example, between $X[j]$ and $U[s]$, connect a $(\mathcal{L}(X[j]) - \mathcal{L}(U[s]))$-rod.

In the case that c_j is a 2-clause, the gadget is the same. There are only two vertices $T[i_1]$ and $T[i_2]$ associated with c_j. Connect $T[i_1]$ to $Q[j,1]$ and $T[i_2]$ to $Q[j,3]$ with rods as before. There will be at least one fixed vertex visible under $X[j], Y[j], Q[j,2]$. Connect rods of appropriate lengths between one such fixed vertex and $X[j]$, $Y[j]$, and $Q[j,2]$, so that $Q[j,2]$ is always extruded. Then c_j is represented by the gadget in the same manner as a 3-clause in which the second literal is always false. This allows us to treat every clause as though it were a 3-clause.

Once gadgets have been constructed for each clause to the right of P, *cap* the right end of H with a path around the right end. Let $c_s \in C_2$ maximize $\mathcal{L}(U[s])$. Let $k = \mathcal{L}(U[s]) - \Lambda + 3$. Place a k rod at each of $W[0]$ and $W[n]$; identify their free ends. Notice that two edges, one from each rod, are also identified as the edge $(X[m+1], Z[right])$. $X[m+1]$ is a fixed vertex with $\mathcal{L}(X[m+1]) = \Lambda + k$. See Figure 3.12. Some $U[s]$'s or $W[i]$'s will be visible from $X[m+1]$. Connect $X[m+1]$ to each of them with a rod of appropriate length. The cap around the right end may be thought of as a dummy clause containing no literals whose purpose is to provide a rod for any $U[s]$'s and $W[i]$'s that have yet to be connected to a rod.

After the gadgets for the clauses to the left of P are constructed, cap the left end by two rods from $V[0]$ to $X[0]$ and from $V[n]$ to $X[0]$ in a similar manner to the preceding paragraph. This completes the construction of H. Λ is now fixed as 1 plus the distance between $Z[left]$ and $W[0]$. Clearly, the construction can be accomplished in polynomial time. Also, H is a planar graph with a planar embedding that is essentially unique except for some freedom in embedding the intermediate vertices in k-rods.

Every $V[i]$, $W[i]$, and $U[j]$ in H has a rod connecting it to either an $X[j]$ or a $Y[j]$. If H has a leveled-planar embedding, then it has a leveled-planar embedding in which: (1) $Z[left]$ is in level 1; (2) every vertex in the capping cycle

$$Z[left], \ldots, V[n], \ldots, W[n], \ldots, Z[right], \ldots, W[0], \ldots, V[0], \ldots, Z[left]$$

is in its preferred level; (3) $W[n]$ is above $W[0]$ in level Λ. Because the capping cycle has only one leveled-planar embedding satisfying these constraints, all other vertices are forced to be inside the capping cycle. In such an embedding, we want each fixed vertex to be in its preferred level. We show that this must be the case in the following two claims. In Claim 1, we assume that, for a particular clause $c_j \in C_2$, $U[j]$ is in level t, $X[j]$ and $Y[j]$ are in level $t-2$, the rod connected to $U[j]$ goes right, and the rods connected to $Q[j,1], X[j], Q[j,2], Y[j], Q[j,3]$ go left.

Claim 1. The gadget for c_j has such a leveled-planar embedding if and only if at least one of $Q[i,1], Q[i,2], Q[i,3]$ is intruded.

Proof: If $Q[j,2]$ is intruded, Figure 3.13 shows such an embedding. If $Q[j,3]$ (or, by symmetry, $Q[j,1]$) is intruded, Figure 3.14 shows such an embedding. From these figures, it is clear that there is no leveled-planar embedding if all three vertices $Q[j,1]$, $Q[j,2]$, and $Q[j,3]$ are intruded. □

Claim 2. If H has a leveled-planar embedding such that each vertex in the capped cycle is in its preferred level, then each fixed vertex of H is in its preferred level.

Proof: Suppose there is a fixed vertex not in its preferred level. If there is such a vertex in an $EROD$, choose i smallest where $EROD[i]$ contains such a vertex. By left-right symmetry, it suffices to consider the case that $W[i]$ is in a level $t > \Lambda$. Because i is minimum, $t = \Lambda + 2$. $W[i]$ is connected by a rod to either an $X[j]$ or a $Y[j]$. Without loss of generality, suppose the rod is to $Y[j]$. The rod forces $Y[j]$ to be in level $\mathcal{L}(Y[j]) + 2 = \mathcal{L}(U[j])$. We claim that $U[j]$ is in a level higher than $\mathcal{L}(U[j])$.

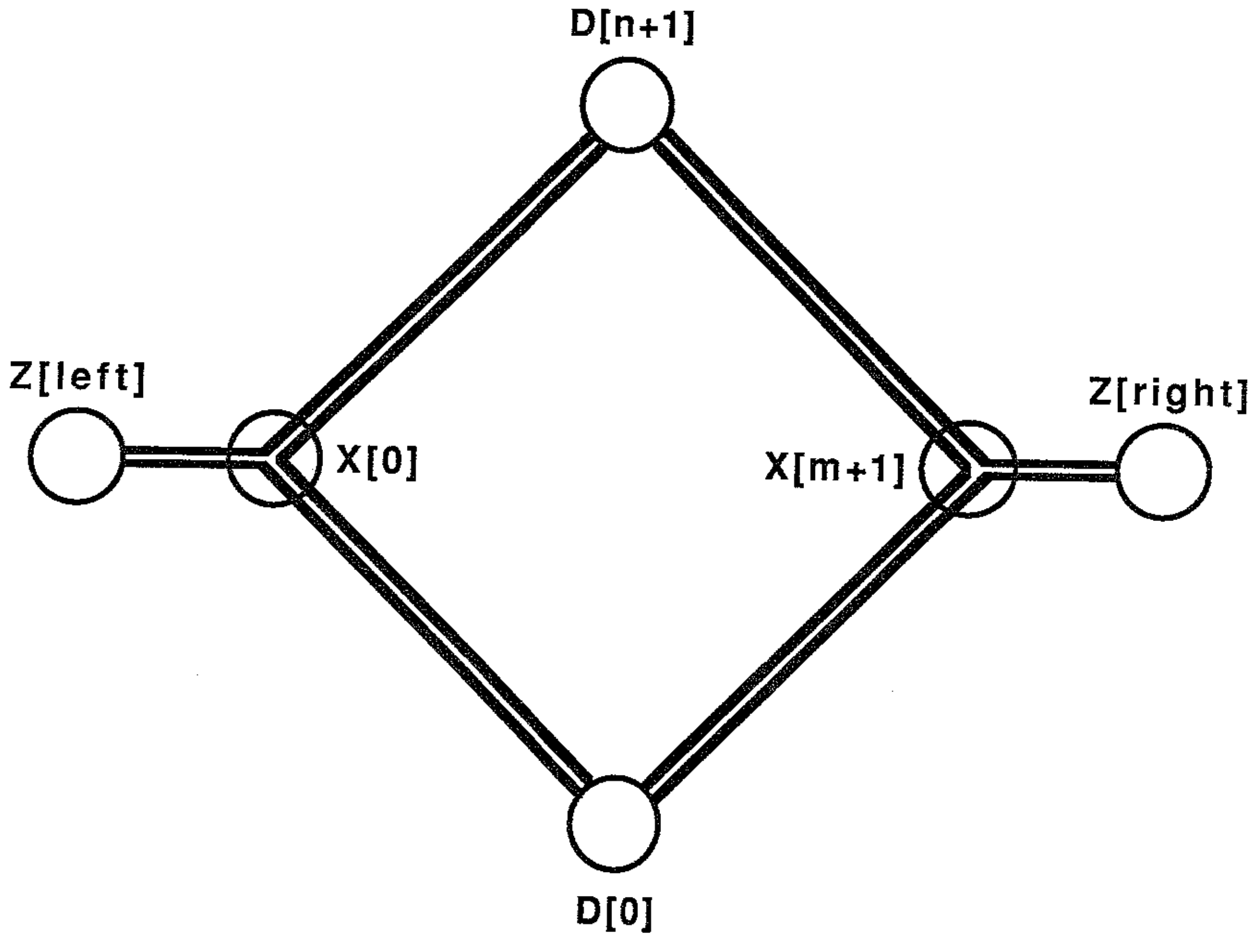

Figure 3.12: Capping the left and right ends.

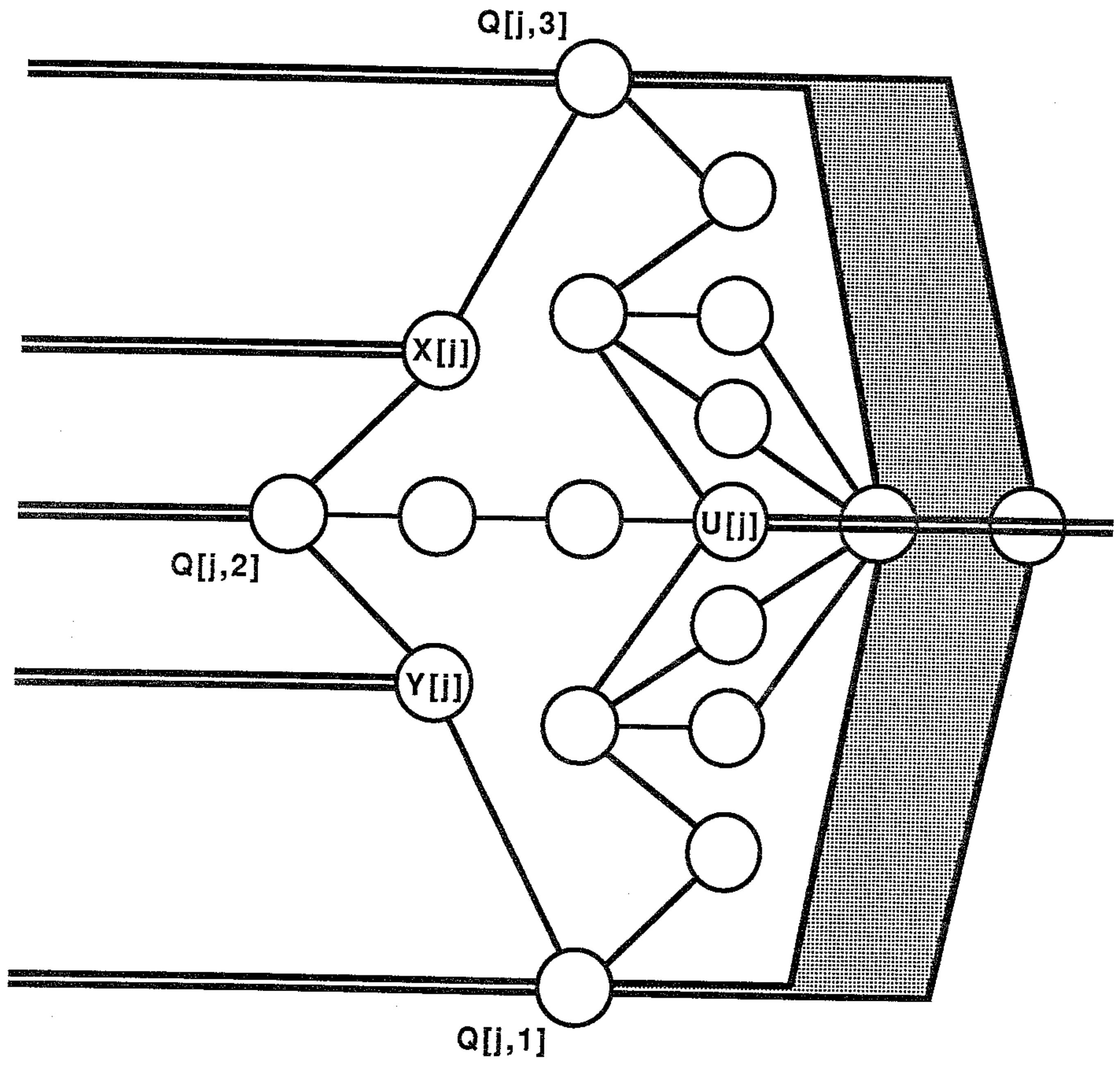

Figure 3.13: $Q[j, 2]$ intruded.

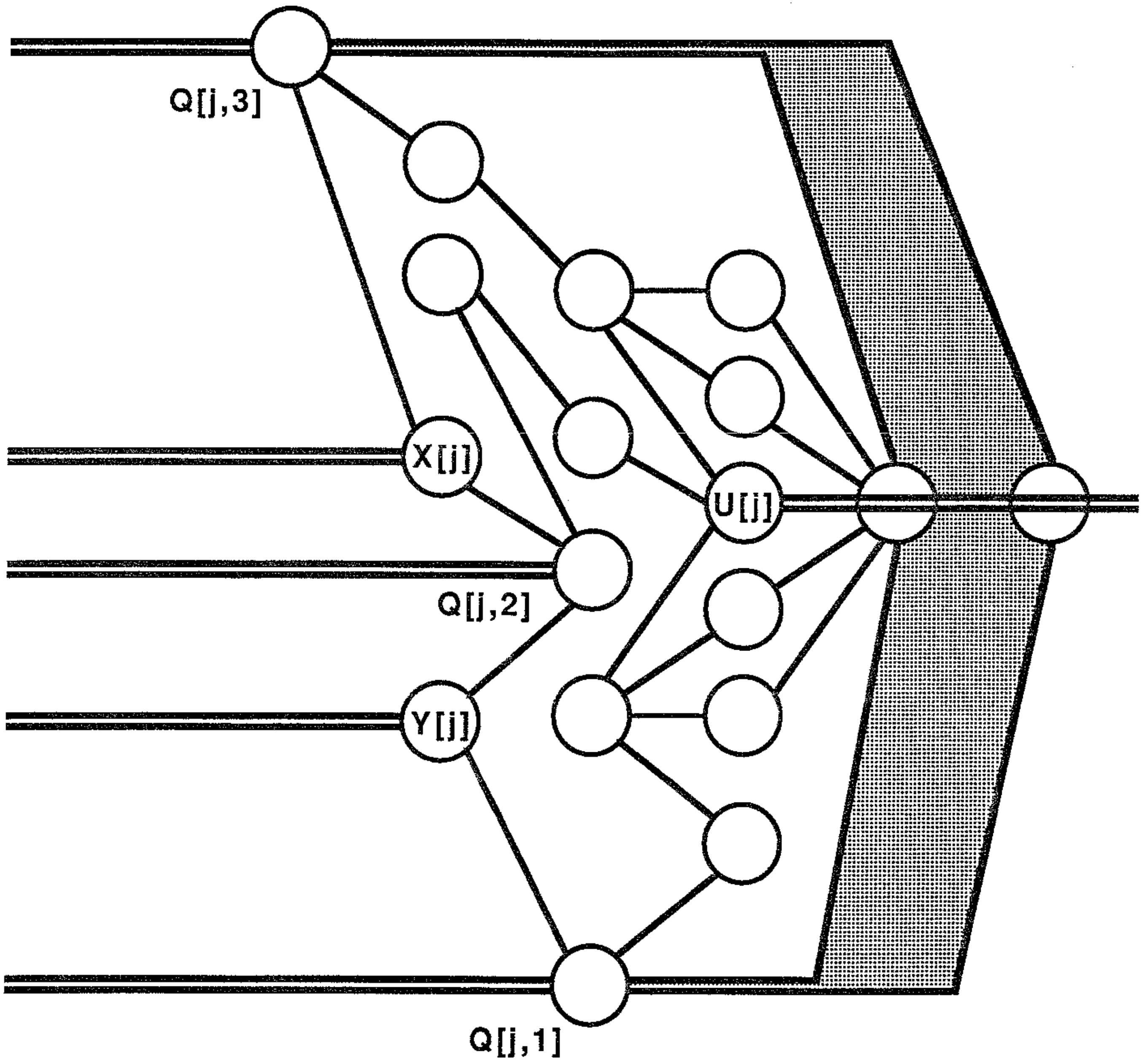

Figure 3.14: $Q[j,3]$ intruded.

Suppose $U[j]$ is in level $\mathcal{L}(U[j])$. The semi-rod to $Q[j,1]$ forces $Q[j,1]$ to be in level $\mathcal{L}(U[j])-1$. The rods of $Y[j]$ and $Q[j,1]$ force $Q[j,1]$ to be below $Y[j]$. Therefore, it is not possible to embed the path from $Q[j,1]$ to $U[j]$.

We conclude that $U[j]$ is in level $\geq \mathcal{L}(U[j])+2$. The above argument repeats with the rod from $U[j]$ connecting to some $X[j']$ or $Y[j']$, shifting $U[j']$ to a level higher than $\mathcal{L}(U[j'])$. Repetition of the argument ends at $X[m+1]$ which must be in level $\mathcal{L}(X[m+1])$, not higher. This contradiction proves that each $W[i]$ is in Λ.

A similar argument shows that any fixed vertex that is not a $W[i]$ must also be in its preferred level. $\square$

We need to show that ϕ is satisfiable if and only if H has a leveled-planar embedding.

Suppose ϕ is satisfiable. Choose a satisfying assignment for ϕ. Embed P first. Place all fixed vertices of P on their assigned levels. If v_i is true, let whichever of $S[i]$ and $T[i]$ corresponds to the literal v_i be intruded. If v_i is false, let whichever of $S[i]$ and $T[i]$ corresponds to the literal $\overline{v_i}$ be intruded. Then each $u[j]$ has at least one intruded $Q[i,j]$ and can be level embedded by Claim 1. Thus H has a leveled-planar embedding.

Now suppose H has a leveled-planar embedding. By Claim 2, we may assume that each fixed vertex F is on level $\mathcal{L}(F)$. Let $Z[i]$ be whichever of $S[i]$ and $T[i]$ corresponds to the literal v_i. If $Z[i]$ is intruded, assign v_i the value true; otherwise, assign v_i the value false. By Claim 1, every $U[j]$ has an intruded $Q[i,j]$. Therefore, each clause c_j contains a literal that is true under this assignment. This truth assignment satisfies ϕ, that is, ϕ is satisfiable.

Thus P3SAT reduces to LEVELED-PLANAR. As P3SAT is NP-complete, we conclude that LEVELED-PLANAR is NP-complete. $\square$

It appears that the graph H is arched leveled-planar if and only if it is leveled-planar. To be certain of this, we modify the construction slightly by adding an *arched cap* on the left and right ends of H. The cap on the right end is shown in Figure 3.15. The rightmost edge of the right cap and the leftmost edge of the left cap must be arches, and no other edges may be arches. With this change to H, H is an arched

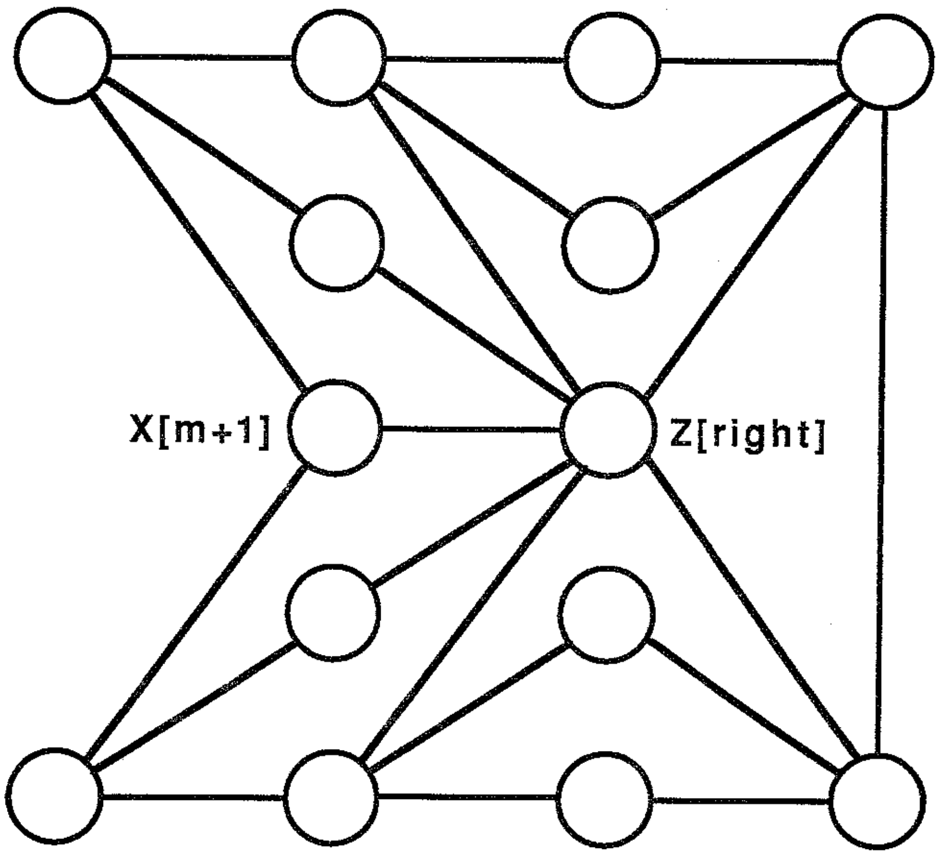

Figure 3.15: An arched cap.

leveled-planar graph if and only if ϕ is satisfiable. This proves the following corollary.

Corollary 3.2 *ARCHED LEVELED-PLANAR is NP-complete.*

3.3. Comparing Queues and Stacks

A graph $G = (V, E)$ is *subhamiltonian* if it is a subgraph of a planar graph that has a hamiltonian cycle. [BK79] provides a characterization of 2-stack graphs.

Proposition 3.2 *A graph G has a 2-stack layout if and only if G is subhamiltonian.*

We can bound the stack number of a 1-queue graph.

Theorem 3.4 *Every 1-queue graph is a 2-stack graph.*

Proof: Let $G = (V, E)$ be a 1-queue graph having $n \geq 3$ vertices. By Lemma 3.3, G has an arched leveled-planar embedding with some leveling of V, say $V_1, \ldots, V_m$. By Proposition 3.2, it suffices to show that G is subhamiltonian.

Because the stacknumber of a graph equals the maximum stacknumber of any of its biconnected components [CLR87], we may assume that G is biconnected, so, in particular, none of the levels $V_2, \ldots, V_{m-1}$ is a singleton. We may also assume that G is a maximal arched leveled-planar graph. For each level i, add the *vertical* edges $(p, p+1)$, $b_i \leq p \leq t_i - 1$, that is, the edges that go along the line ℓ_i, connecting consecutive vertices of V_i. Let the resulting graph be $G' = (V, E')$. Clearly G' is planar; we claim that it is hamiltonian.

Note that when $|V_i| > 2$, the vertical edges on level i together with the arch (b_i, t_i) form a cycle on V_i. Call these edges the *level-i cycle edges.* These cycles on levels are nested in the planar embedding. Our strategy is to connect each pair of consecutive cycles by two leveled edges to obtain a hamiltonian cycle for G'.

By an induction on $m - i \geq 1$, we show that there is a particular kind of spanning cycle for levels $V_i, V_{i+1}, \ldots, V_m$. The inductive hypothesis is that there is a cycle C spanning levels $V_i, \ldots, V_m$ such that all but one of the level-i cycle edges are in C; if

$|V_i| = 2$, then C contains the edge (b_i, t_i) (which is considered to be both a vertical edge and an arch).

For the base case $i = m - 1$, there are three subcases. First, if $|V_m| = 1$, then let C be the spanning cycle

$$m, t_i, b_i, b_i - 1, \ldots, t_i - 1, m;$$

all level-i cycle edges are present except $(t_i - 1, t_i)$. Second, if $|V_m| > 1$ and $|V_i| = 1$, then $i = 1$ (because of our assumption that G is biconnected) and G' is obviously hamiltonian. Third, if $|V_m| > 1$ and $|V_i| > 1$, then choose four vertices $p, p+1, q, q+1$ such that $p, p+1 \in V_i$, $q, q+1 \in V_m$, and $(p, q), (p+1, q+1) \in E$. Because G is maximal and $|V_i| > 1$, $|V_m| > 1$, this choice is always possible. Let C be the spanning cycle

$$p+1, \ldots, t_i, b_i, \ldots, p, q, \ldots, b_m, t_m, \ldots, q+1, p+1.$$

All level-i cycle edges except $(p, p+1)$ are in the spanning cycle; if $|V_i| = 2$, then $p = b_i$, $p + 1 = t_i$, and $(p, p+1)$ is in C.

For purposes of induction, assume there is a spanning cycle C satisfying the inductive hypothesis for $V_{i+1}, \ldots, V_m$. We extend the spanning cycle to a spanning cycle C' for $V_i, \ldots, V_m$. If $i = 1$ and $|V_i| = 1$, then choose some level-2 vertical edge $(p, p+1)$ that is in C. Construct C' from C by deleting $(p, p+1)$ and adding $(1, p)$ and $(1, p+1)$. A hamiltonian cycle for G' results.

Otherwise, $|V_i| > 1$. Let (x, y), $x < y$, be the level-$(i+1)$ vertical edge (if any) that is not in C. We wish to choose four vertices $p, p+1, q, q+1$ with the properties $p, p+1 \in V_i$, $q, q+1 \in V_{i+1}$, and $(p, q), (p+1, q+1) \in E$. If such a choice is possible so that $(q, q+1) \neq (x, y)$, then C can be extended to C' by removing edge $(q, q+1)$ and adding the path

$$q, p, \ldots, b_i, t_i, p+1, q+1.$$

All level-i cycle edges except $(p, p+1)$ are in the spanning cycle; if $|V_i| = 2$, then $p = b_i$, $p + 1 = t_i$, and $(p, p+1)$ is in C.

Suppose the only choices for the four vertices forces $(q, q+1) = (x, y)$. (This implies that either x or y is the only level-$(i+1)$ vertex that is adjacent to more than one level-i vertex. Thus, either $x = b_{i+1}$ or $y = t_{i+1}$. If $x = b_{i+1}$, then every level-$(i+1)$ vertex is adjacent to t_i. If $y = t_{i+1}$, then every level-$(i+1)$ vertex is adjacent to b_i.) Fix one such choice. Because G is maximal, either $(p, q+1) \in E$ or $(p+1, q) \in E$. First suppose $(p+1, q) \in E$. Because the choice of q and $q+1$ was forced, we can conclude that $q = b_{i+1}$ and $p+1 = t_i$. Further, the edges (j, q), $s_i \le j \le p+1 = t_i$ and $(p+1, j)$, $b_{i+1} = q \le j \le t_{i+1}$ are all the leveled edges between V_i and V_{i+1}. (b_{i+1}, t_{i+1}) is an edge in C. Replace it with the path

$$b_{i+1}, p, \ldots, b_i, t_i, t_{i+1},$$

yielding C'. The result is a spanning cycle for $V_i, \ldots, V_m$ satisfying the inductive hypothesis.

Now suppose $(p, q+1) \in E$. We can conclude that $q+1 = t_{i+1}$ and $p = s_i$. Further, the edges $(j, q+1)$, $s = p \le j \le t_i$ and (p, j), $b_{i+1} \le j \le q+1 = t_{i+1}$ are all the level edges between V_i and V_{i+1}. (b_{i+1}, t_{i+1}) is an edge in C. Replace it with the path

$$b_{i+1}, s_i, \ldots, b_i, t_i, t_{i+1},$$

yielding C'. The result is a spanning cycle for $V_i, \ldots, V_m$ satisfying the inductive hypothesis.

By induction, G' has a hamiltonian cycle. The theorem follows. □

This result is best possible in the sense that there are 1-queue graphs that require 2-stacks. For example, the complete bipartite graph $K_{2,3}$ is a leveled-planar, hence 1-queue, graph, but is not outerplanar, hence not a 1-stack graph. Similarly, 1-stack graphs need not be 1-queue graphs, but they never need more than two queues.

Theorem 3.5 *Any 1-stack graph is a 2-queue graph.*

Proof: Let $G = (V, E)$ be a 1-stack graph having $n \ge 3$ vertices. Then G is outerplanar. We may assume that G is a maximal outerplanar graph. Then G has a unique outerplanar embedding such that all its vertices are on the exterior face, and the boundary of that face is the unique hamiltonian cycle C for G.

Level V as follows. Choose any $1 \in V$. For each $v \in V$, let $\delta(v)$ be the length of a shortest path from 1 to v. Let $m = 1 + \max_{v \in V} \delta(v)$. For $1 \le i \le m$, define

$$V_i = \{v \in V \mid \delta(v) = i - 1\}.$$

Then, $V_1, \ldots, V_m$ is a partition of V. In each V_i, order the vertices $b_i, \ldots, t_i$ as they are encountered in a counterclockwise traversal of C, beginning at 1. Ordering V level by level, we obtain a linear order $\sigma = 1, 2, \ldots, n$ for V. We need to show that σ accommodates an assignment of E to 2 queues.

Let $E_\ell \subset E$ be the edges between consecutive levels. We claim that no two edges in E_ℓ nest with respect to σ. Suppose $(p_2, q_2) \in E_\ell$, $p_2 < q_2$, nests inside $(p_1, q_1) \in E_\ell$, $p_1 < q_1$. Then $\delta(p_1) \leq \delta(p_2)$ and $\delta(q_2) \leq \delta(q_1)$. Since $\delta(q_1) = \delta(p_1) + 1$ and $\delta(q_2) = \delta(p_2) + 1$, we must have $\delta(p_1) = \delta(p_2)$ and $\delta(q_1) = \delta(q_2)$. Then the vertices occur in the counterclockwise order

$$1, p_1, p_2, q_2, q_1.$$

But then

$$i + 1 = \delta(q_1) \leq \delta(p_2) = i,$$

a contradiction.

Each edge in $E - E_\ell$ is incident on two vertices in the same level. Clearly, for two edges in $E - E_\ell$ to nest, they must be in the same level. Suppose there are two edges $(p_1, q_1), (p_2, q_2) \in E - E_\ell$ that nest so that $p_1, p_2, q_1, q_2 \in V_i$ and $p_1 < p_2 < q_2 < q_1$. But then the counterclockwise order of the vertices is

$$1, p_1, p_2, q_2, q_1.$$

Since $\delta(p_1) = \delta(q_1) = i$, it is not possible that $\delta(p_2) = \delta(q_2) = i$, a contradiction. Therefore, no two edges of $E - E_\ell$ nest.

We conclude that all the edges in E_ℓ can be assigned to one queue and all edges in $E - E_\ell$ to a second queue. This produces a 2-queue layout for G, as required. □

This result is also best possible in the sense that there are 1-stack graphs that require 2-queues. We will demonstrate this fact in Corollary 4.1.

From the results of Theorems 3.4 and 3.5, one might hope for a result like the following:

Any k-queue (k-stack) graph is a $2k$-stack ($2k$-queue).

However, in both theorems, the transformation from a queue (stack) layout to a stack (queue) layout transforms the vertex order in a way that depends on the original queue (stack). But for a general multi-queue (multi-stack) layout, the order transformations will be different for each queue (stack) and hence not consistent for all queues (stacks). In fact, we conjecture

> **Conjecture** The QN/SN ratio of graphs is not bounded by any constant.
> The SN/QN ratio of graphs is not bounded by any constant.

In harmony with the fact that planar graphs can be laid out in a bounded number of stacks (Yannakakis [Y89]), we conjecture

> **Conjecture** Planar graphs can be laid out in a bounded number of queues.

4. Layouts for Specific Graphs

In this section, we present queue layouts for a variety of specific families of graphs. The intuition that an easily-leveled graph has a good queue layout is illustrated by most of these families. Some details are left to the reader.

4.1. Trees and Meshes

We begin with trees and meshes, two naturally leveled families of graphs.

Proposition 4.1 *A tree is a leveled-planar, hence 1-queue, graph.*

Proof: Choose a root for the tree. Lay the tree out breadth-first from the root. The result is a 1-queue layout of the tree. □

An $m \times n$ *mesh* is a graph with vertices

$$\{v_{ij} \,|\, 1 \le i \le m, 1 \le j \le n\}$$

and edges

$$\{(v_{ij}, v_{i,j+1}) \,|\, 1 \le j \le n-1\} \cup \{(v_{ij}, v_{i+1,j}) \,|\, 1 \le i \le m-1\}.$$

Proposition 4.2 *An $m \times n$ mesh is a leveled-planar, hence 1-queue, graph. There is a 1-queue layout QL of the mesh having queuewidth* $\mathrm{QW}(QL) \leq \min\{m, n\}$.

Proof: An $m \times n$ mesh has a natural embedding in the plane with vertices in m rows and n columns. If this embedding is rotated 45°, vertices line up on $m + n - 1$ vertical lines. The result is a leveled-planar embedding with the stated queuewidth. □

For $m > 2, n > 2$, an $m \times n$ mesh is not an outerplanar graph and, in fact, has stacknumber 2 ([CLR87]). Such a mesh provides another example of a 1-queue graph that fails to be a 1-stack graph.

4.2. Unicyclic Graphs

A *unicyclic* graph is an undirected graph in which each connected component contains at most one cycle. The family of unicyclic graphs includes trees, forests, and cycles of all lengths.

Proposition 4.3 *A unicyclic graph is an arched leveled-planar, hence 1-queue, graph. Each connected component contributes at most one arch.*

Proof: Let $G = (V, E)$ be a unicyclic graph. We may assume that G is connected. By Proposition 4.1, we need only treat the case that G contains a cycle. Let

$$C = u_1, u_2, \ldots, u_k, u_1$$

be that cycle. If k is even, level C into $\frac{k}{2} + 1$ levels

$$U_1 = \{u_1\}, U_2 = \{u_2, u_k\}, \ldots, U_i = \{u_i, u_{k-i+2}\}, \ldots, U_{\frac{k}{2}+1} = \{u_{\frac{k}{2}+1}\};$$

a leveled-planar embedding of C results. If k is odd, level C into $\frac{k+1}{2}$ levels

$$U_1 = \{u_1, u_k\}, U_2 = \{u_2, u_{k-1}\}, \ldots, U_i = \{u_i, u_{k-i+1}\}, \ldots, U_{\frac{k+1}{2}} = \{u_{\frac{k+1}{2}}\};$$

an arched leveled-planar embedding of C results with the single arch (u_1, u_k).

Let G' be G without the edges of C. G' contains one connected component for each u_i. The connected component containing u_i is a tree that we root at u_i. Proceeding as in Proposition 4.1, we obtain a leveled-planar embedding for the tree that begins at the level of u_i. An arched leveled-planar embedding for G results. □

4.3. X-Trees

The *depth-d complete binary tree* $CBT(d)$ has vertex set $\{1, 2, \ldots, 2^{d+1} - 1\}$ and edge set

$$\{(\alpha, 2\alpha), (\alpha, 2\alpha + 1) \,|\, 1 \leq \alpha \leq 2^d - 1\}.$$

The root of $CBT(d)$ is 1, and $CBT(d)$ has $d + 1$ levels in the leveling starting at the root. The *depth-d X-tree* $X(d)$ is the supergraph of $CBT(d)$ that has edges added across each of the levels from left to right. See Figure 4.1.

Every $X(d)$ is a 2-stack graph; when $d \leq 2$, $X(d)$ is a 1-stack graph ([CLR87]). In contrast, even small X-trees require two queues.

Proposition 4.4 *For $d \geq 1$, $X(d)$ admits a 2-queue layout with queuewidths 2^d and 1. For $d \geq 2$, $X(d)$ is not a 1-queue graph.*

Proof: For the upper bound, choose the order $\sigma = 1, 2, \ldots, 2^{d+1} - 1$. The edges of $CBT(d)$ are assigned to one queue and the edges across the levels are assigned to a second queue.

For the lower bound, since $X(2)$ is a subgraph of $X(d)$, $d \geq 2$, it suffices to show that $X(2)$ is not a 1-queue graph.

We exploit an alternate means of constructing $X(2)$. Given any graph G and any edge (x, y) in G, define the operation of *hatting* (x, y) as adding a new vertex (the *peak*) z and new edges (x, z) and (y, z). Start with a cycle of length 4

$$C = u_1, u_2, u_3, u_4, u_1.$$

Choose any three of the four edges of C. Hat each of the chosen edges. The resulting graph is isomorphic to $X(2)$.

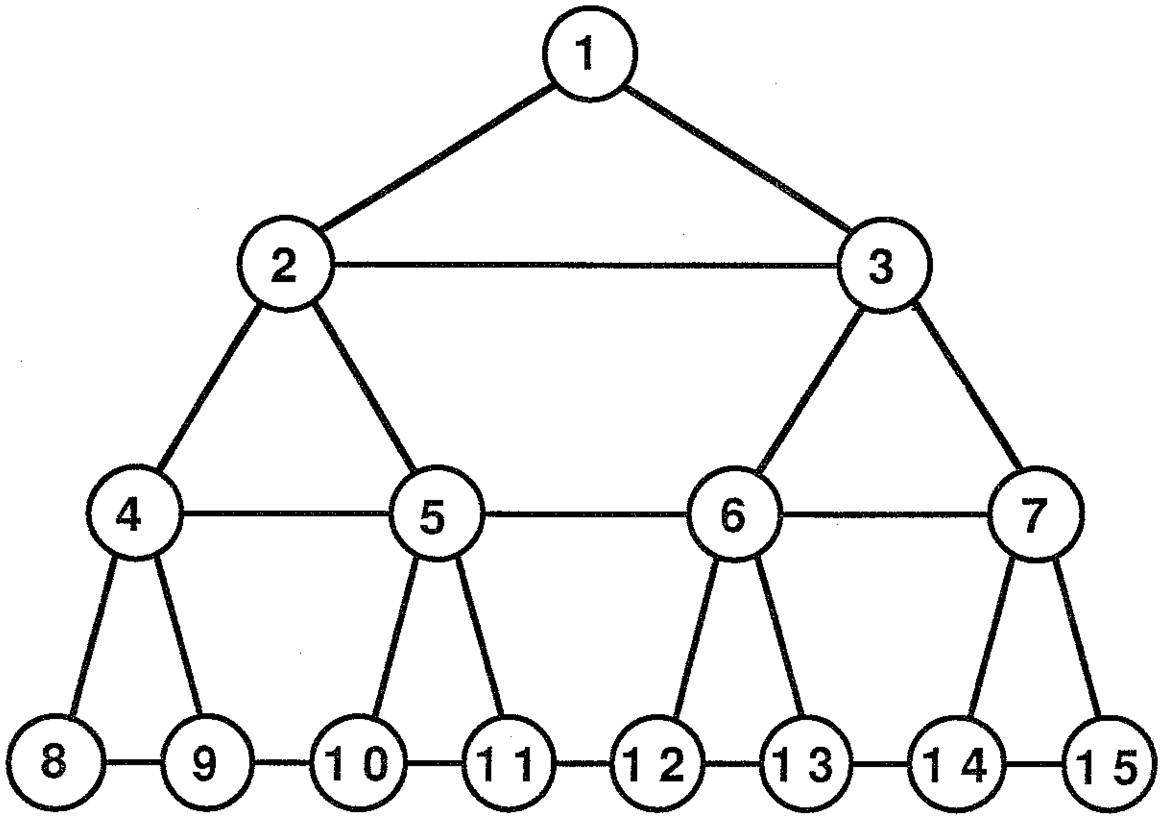

Figure 4.1: X-tree $X(3)$.

To obtain a contradiction, suppose that $X(2)$ has a 1-queue layout. Let σ be the order of the vertices. Without loss of generality, assume that u_1 is the leftmost vertex of C in σ. Neither u_2 nor u_4 can be the rightmost vertex of C in σ, for then two edges of C would nest. By symmetry we may assume that the order of the vertices of C in σ is u_1, u_2, u_4, u_3. Three of the four edges of C must be hatted. In particular, either u_1 or u_3 has both of its incident edges hatted. By symmetry, we may assume that (u_1, u_4) and (u_1, u_2) are hatted. Let w be the peak of (u_1, u_4).

There are 5 possible placements of w within the order u_1, u_2, u_4, u_3. Only placement of w between u_1 and u_2 fails to yield two nested edges. But, with w between u_1 and u_2, there is no placement of the peak of (u_1, u_2) that does not yield two nested edges. This is a contradiction to σ giving a 1-queue layout of $X(2)$. □

Since $X(2)$ is outerplanar, we have the following corollary, which completes a comment made in section 3.3.

Corollary 4.1 $X(2)$ *is a 1-stack graph that is not a 1-queue graph.*

4.4. DeBruijn Graphs

The *order-2^d deBruijn graph* $DB(d)$ has vertex set $\{0, 1, \ldots, 2^d - 1\}$ and edges

$$(x, 2x \pmod{2^d})(x, 2x + 1 \pmod{2^d}).$$

See Figure 4.2. Note that multiple edges and loops are discarded.

Proposition 4.5 $DB(d)$ *admits a 2-queue layout with queuewidths* 2^{d-1}.

Proof: The edges of $DB(d)$ of the forms $(x, 2x)$ and $(x, 2x+1)$, $x \in \{1, 2 \ldots, 2^{d-1} - 1\}$, are the edges of a depth-$(d-1)$ complete binary tree with root 1 and containing all vertices except 0. Similarly, the edges of the forms $(2^{d-1} + x, 2x)$ and $(2^{d-1} + x, 2x + 1)$, $x \in \{0, 1 \ldots, 2^{d-1} - 2\}$, are the edges of a depth-$(d-1)$ complete binary tree with root $2^{d-1} - 2$ and containing all vertices except $2^{d-1} - 1$. Choose the order $\sigma = 0, 1, \ldots, 2^d - 1$. Assign edges of the forms $(x, 2x)$ and $(x, 2x + 1)$ to one queue and edges of the forms

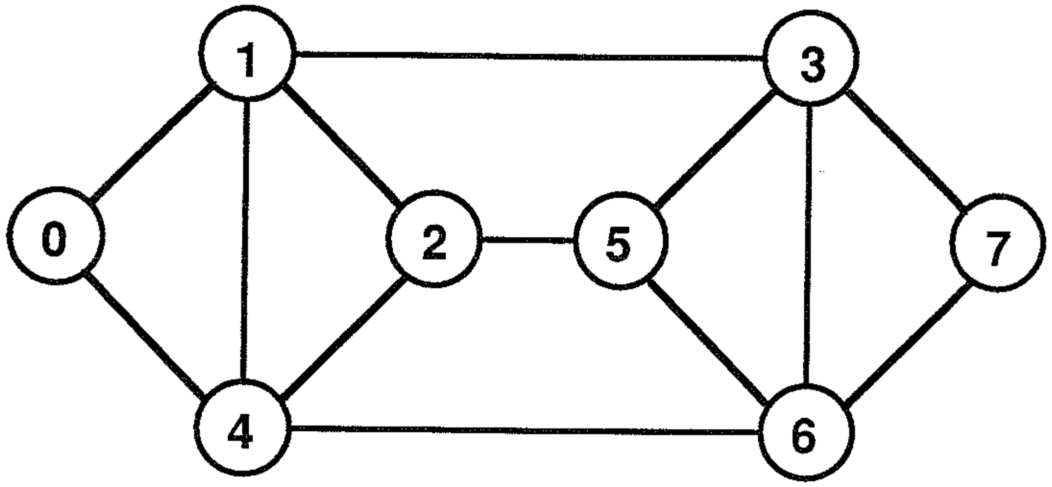

Figure 4.2: The deBruijn graph $DB(3)$.

$(2^{d-1}+x, 2x)$ and $(2^{d-1}+x, 2x+1)$ to a second queue. (When edges are assigned to both queues, break ties arbitrarily.) □

Proposition 4.6 *$DB(d)$, $d \geq 4$, does not admit a 1-queue layout. $DB(3)$ does admits a 1-queue layout.*

Proof: $DB(d)$, $d \geq 4$, is not planar, hence not a 1-queue graph. The order $\sigma = 1, 0, 2, 3, 4, 5, 7, 6$ yields a 1-queue layout of $DB(3)$. □

4.5. FFT and Benes Networks

We now consider two related families of graphs that have importance as computational networks. The *FFT network* represents the computational structure of the Fast Fourier Transform algorithm. The *Benes rearrangeable permutation network* is a switching network capable of realizing at its n outputs any permutation of its n inputs (Benes [B64]).

The n-input Benes network $B(n)$, n a power of 2, is defined inductively as follows.

1. $B(2)$ is the complete bipartite graph $K_{2,2}$ on the two input vertices $I[1,1]$ and $I[1,2]$ and the two output vertices $O[1,1]$ and $O[1,2]$.

2. $B(n)$ is obtained from two copies of $B(n/2)$ together with n new input vertices $I[n,1], I[n,2], \ldots, I[n,n]$ and n new output vertices $O[n,1], O[n,2], \ldots, O[n,n]$. In the second copy of $B(n/2)$, each vertex $I[k,i]$ is relabeled $I[k, i+n/2]$, and each vertex $O[k,i]$ is relabeled $O[k, i+n/2]$; all vertices then have distinct labels. For $1 \leq i \leq n$, add edges to create a copy of $K_{2,2}$ on vertices $I[n,i]$ and $I[n, i+n/2]$ and vertices $I[n/2, i]$ and $I[n/2, i+n/2]$; also, add edges to create a copy of $K_{2,2}$ on vertices $O[n,i]$ and $O[n, i+n/2]$ and vertices $O[n/2, i]$ and $O[n/2, i+n/2]$.

As shown in Figure 4.3, the Benes network has a natural level structure with $2\log n$ levels. The n-input FFT network is the graph consisting of the first $n+1$ levels of $B(n)$.

As the Benes and FFT networks are not planar, the queuenumber of each is at least 2. The level structure (of either network) provides a straightforward 3-queue layout: order the vertices level by level, going up each level; one queue for the "across" edges, one queue for the "upward" edges, and one queue for the "downward" edges suffices. A more complicated 2-queue layout of $B(n)$ is due to Reibman [Re84].

Proposition 4.7 *The Benes network $B(n)$ admits a 2-queue layout with each queue of width n. The layout is optimal in queuenumber and within a factor of 2 of optimal in queuewidth.*

Proof: The layout of $B(n)$ follows its inductive definition. The inductive hypothesis is that $B(n)$ has a 2-queue layout which respects the leveling of $B(n)$; that is, all level i vertices appear before any level $i+1$ vertices, though no restriction is placed on the relative order of vertices within each level.

1. The vertex order for $B(2)$ is $I[1,1], I[1,2], O[1,1], O[1,2]$. The two edges incident to $I[1,1]$ are assigned to one queue and the two edges incident to $I[1,2]$ are assigned to the second queue. The layout satisfies the inductive hypothesis.

2. We assume that $B(n/2)$ has a 2-queue layout satisfying the inductive hypothesis. Let B_1 and B_2 be two copies of $B(n/2)$. Lay each out in the 2-queue order that is guaranteed by the induction. Merge the two layouts level by level so that the level i

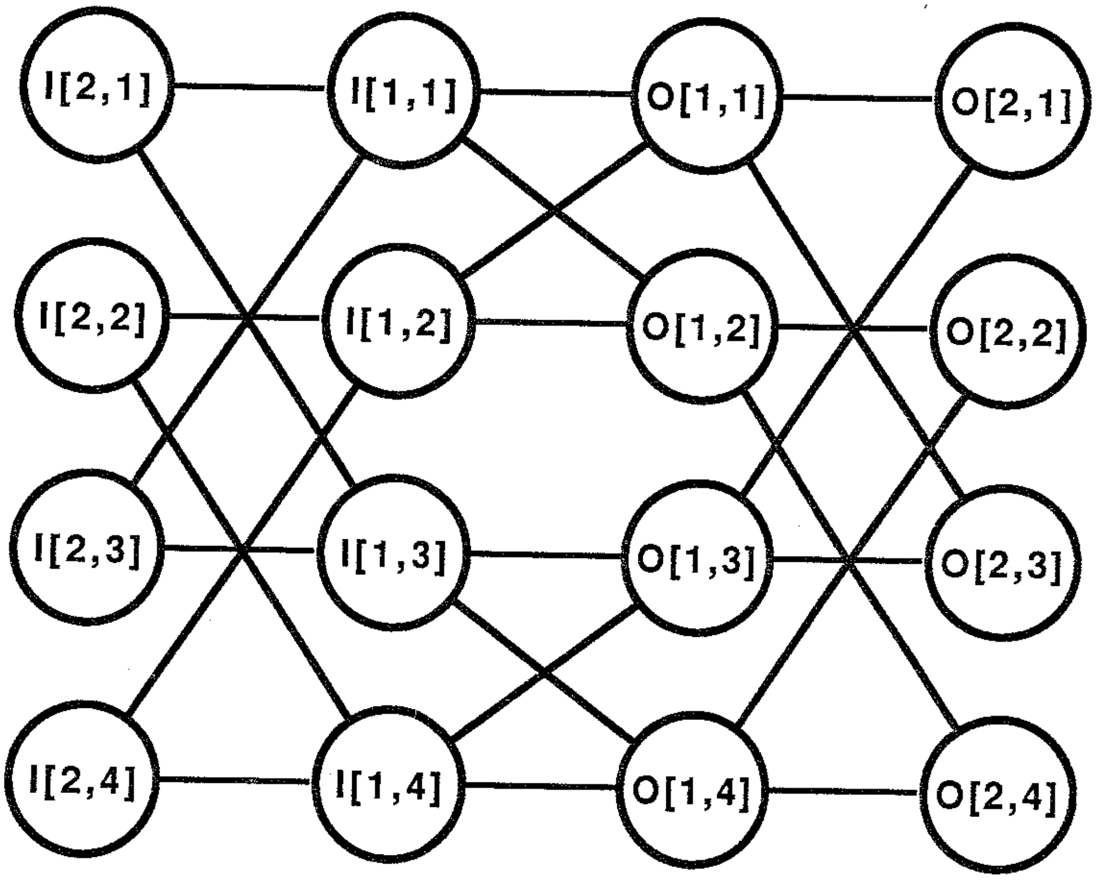

Figure 4.3: The Benes network $B(4)$.

vertices of B_2 always appear immediately to the right of the level i vertices of B_1. In particular, $I[k, i+n/2]$, respectively, $O[k, i+n/2]$, is always $n/2$ vertices to the right of $I[k,i]$, respectively, of $O[k,i]$. Because the leveling of $B(n)$ is honored in the layout, each level-i edge of B_1 crosses every level-i edge of B_2, so no nesting results from the merging; hence, a 2-queue layout of B_1 and B_2 results. Add n new input vertices to the left and n new output vertices to the right of the entire layout. View the n new inputs as consisting of $n/2$ consecutive pairs of vertices. Add edges from the first pair to the first vertices of B_1 and B_2 to form a copy of $K_{2,2}$. In general, add edges from the ith pair to the ith vertices of B_1 and B_2. Assign the added edges incident to B_1 to the first queue and the added edges incident to B_2 to the second. Similarly, connect the n new outputs to the last vertices of B_1 and B_2. The result is a 2-queue layout of $B(n)$.

□

Because the FFT network is a subgraph of the Benes network, it also has a 2-queue layout. This compares favorably with the stacknumber optimal 3-stack layouts of the Benes and FFT networks in Games [G87]. The natural leveling of these networks is a definite advantage in constructing queue layouts that are good, at least in the sense of queuenumber.

4.6. Hypercube

The *d-dimensional hypercube* $H(d)$ has vertex set $\{0,1\}^d$ and edges connecting every pair of vertices that differ in exactly one bit position. The hypercube admits a very regular layout strategy.

Proposition 4.8 *For $d \geq 2$, $H(d)$ admits a $(d-1)$-queue layout with queuewidths*

$$2^{d-1}, 2^{d-2}, \ldots, 2^2, 3.$$

Proof: We lay out $H(d)$ inductively. The order $\sigma = 00, 01, 10, 11$ gives a 1-queue layout of $H(2)$ with queuewidth 3. To obtain a layout for $H(d)$, $d > 2$, inductively lay out two adjacent copies of $H(d-1)$, similarly ordered. The two copies of $H(d-1)$ use $d-2$ queues with queuewidths

$$2^{d-2}, 2^{d-3}, \ldots, 2^2, 3.$$

The 2^{d-1} edges to connect one copy of $H(d-1)$ to the other requires one additional queue of width 2^{d-1}.

□

The queuenumber of the preceding layout is optimal to within a constant factor.

Proposition 4.9 $\mathrm{QN}(H(d)) = \Omega(d)$.

Proof: $H(d)$ has $d2^{d-1}$ edges. By Corollary 3.1,

$$\mathrm{QN}(H(d)) \geq \left\lceil \frac{d2^{d-1}}{2^{d+1}-3} \right\rceil = \Omega(d).$$

□

4.7. Complete Graphs

The *complete graph* K_n has a vertex set of size n and an edge connecting every pair of vertices.

Proposition 4.10 $\mathrm{QN}(K_n) = \lceil n/2 \rceil$.

Proof: Every vertex order for K_n is symmetric, so fix any order $\sigma = 1, 2, \ldots, n$. The maximum size of a set of nesting edges is exactly $\lceil n/2 \rceil$. By Proposition 2.1 and Theorem 2.1, the result follows.

□

An explicit assignment of edges of K_n to queues is easily described. In the fixed order σ, every edge (i, j) has length $|i-j|$. There are edges of every length from 1 to $n-1$. Assign all edges of length $2i-1, 2i$, $1 \leq i \leq \lceil n/2 \rceil$, to queue q_i. No two edges having the same length or having lengths differing by 1 can nest.

4.8. Complete Bipartite Graphs

The *complete bipartite graph* $K_{m,n}$ has vertex set

$$\{a_1, a_2, \ldots, a_m\} \cup \{b_1, b_2, \ldots, b_n\}$$

and edge set

$$\{(a_i, b_j) \,|\, 1 \leq i \leq m, 1 \leq j \leq n\}.$$

Proposition 4.11 $\mathrm{QN}(K_{m,n}) = \min(\lceil m/2 \rceil, \lceil n/2 \rceil)$.

Proof: Without loss of generality, assume that $m \leq n$. We need to show $\mathrm{QN}(K_{m,n}) = \lceil m/2 \rceil$.

Upper Bound. Choose the layout order

$$\sigma = a_1, a_2, \ldots, a_{\lceil m/2 \rceil}, b_1, \ldots, b_n, a_{\lceil m/2 \rceil + 1}, \ldots, a_m.$$

Assign edge (a_i, b_j), $1 \leq i \leq \lceil m/2 \rceil$ to queue q_i. Assign edge $(a_{\lceil m/2 \rceil + i}, b_j)$, $1 \leq i \leq \lceil m/2 \rceil$ to queue q_i. Clearly, no two edges in queue q_i nest, so $\lceil m/2 \rceil$ queues suffice, as required.

Lower Bound. Let σ be an order of the vertices in a $\mathrm{QN}(K_{m,n})$-queue layout. By symmetry, we may assume that the a_i's appear in the order $a_1, a_2, \ldots, a_m$ in σ and that the b_j's appear in the order $b_n, b_{n-1}, \ldots, b_1$ in σ. Because we may reverse σ and still have a $\mathrm{QN}(K_{m,n})$-queue layout, we may assume that $b_{\lceil m/2 \rceil}$ appears after $a_{\lceil m/2 \rceil}$ in σ. Then the set of edges

$$\{(a_i, b_i) \,|\, 1 \leq i \leq \lceil m/2 \rceil\}$$

nest. By Proposition 2.1, $\mathrm{QN}(K_{m,n}) \geq \lceil m/2 \rceil$. □

This straightforward determination of $\mathrm{QN}(K_{m,n})$ contrasts with the status of $\mathrm{SN}(K_{m,n})$ as reported in [MWW88]. Even after much effort, the exact stacknumber of $K_{m,n}$, or even of $K_{n,n}$, has not been determined, though Muder, Weaver and West have obtained nontrivial bounds.

5. Queuenumber and Graph Structure

We now explore some structural properties of graphs that provide bounds on queuenumber. For the valence of a graph, we give some probabilistic worst-case results for queue layouts that are similar to those for stack layouts. Other significant properties that we have identified include bandwidth, bifurcator size, and cutwidth.

5.1. Valence

The *valence* of G is the maximum degree of a vertex of G. G is *regular* if all vertices of G have the same degree. The observation that twists are obstacles for stack layouts has been exploited to obtain upper and lower bounds on stack number for d-valent graphs ([CLR87,M88]). The proofs dualize to queue layouts with rainbows playing the role of twists.

The first theorem contains a probabilistic upper bound on the queuenumber of a graph of bounded valence.

Theorem 5.1 *Let G be an n vertex graph of valence $d \geq 3$. Then, G has an $F(d,n)$-queue layout, where*

$$F(d,n) = \min\left(n/2, O\left(dn^{1/2}\right)\right).$$

Proof: Use the same argument as Theorem 4.7 of [CLR87], except replace permutations having long increasing sequences (large twists) by permutations having long decreasing sequences (large rainbows). □

The next theorem contains a probabilistic lower bound on the queuenumber of a graph of bounded valence that leaves a significant gap with the upper bound of Theorem 5.1. In particular, there are bounded-valence graphs of arbitrarily large queuenumber, though we do not know an example of a sequence of such graphs.

Theorem 5.2 *Most regular d-valent graphs on n vertices have queue number*

$$\Omega\left(\sqrt{d}n^{\frac{1}{2}-\frac{1}{d}}\right).$$

Proof: Use the same argument as Theorem 6.1 of [M88], except replace completely crossing (twist) with completely nested (rainbow). □

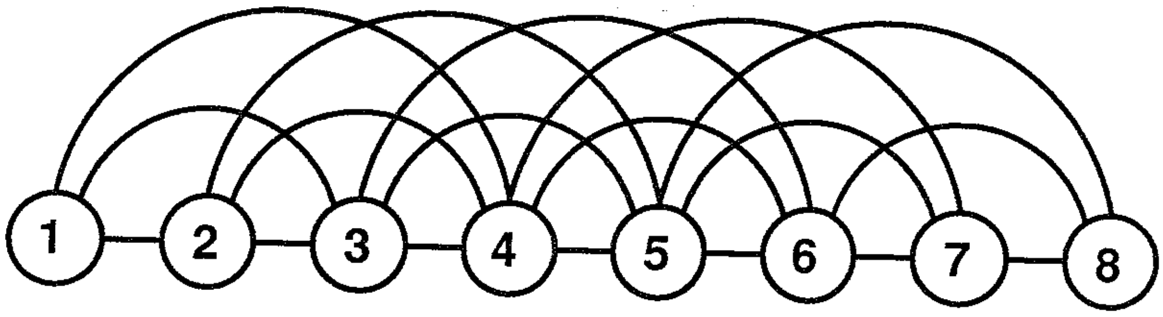

Figure 5.1: Maximal bandwidth-B graph $M(3,8)$.

5.2. Bandwidth

Let $\sigma = 1, 2, \ldots, n$ be any order of the vertices of G. The *bandwidth* of σ is the length of the longest edge; that is,

$$\mathrm{BW}(\sigma) = \max_{(i,j)\in E} |i - j|.$$

The *bandwidth* of G is the minimum bandwidth of any σ; that is,

$$\mathrm{BW}(G) = \min_{\sigma} \mathrm{BW}(\sigma).$$

Assume that $n \geq B + 1$. The *maximal bandwidth-B graph on n vertices* $M(B, n)$ has vertex set $\{1, 2, \ldots, n\}$ and edge set that forms a complete graph on each subset of vertices

$$\{i, i+1, \ldots, i+B\}, \quad 1 \leq i \leq n - B.$$

See Figure 5.1.

The following lemma establishes a relationship between bandwidth and queuenumber.

Lemma 5.1 $\mathrm{QN}(M(B,n)) = \lceil B/2 \rceil$.

Proof:

Upper Bound. Choose the order $\sigma = 1, 2, \ldots, n$ for the vertices of $M(B,n)$. There are edges of every length from 1 to B. Assign all edges of length $2i-1, 2i$, $1 \leq i \leq \lceil B/2 \rceil$, to queue q_i. No two edges having the same length or having lengths differing by 1 can nest. A $\lceil B/2 \rceil$-queue layout of $M(B,n)$ results.

Lower Bound. $M(B,n)$ contains complete graphs on $B+1$ vertices. By Proposition 4.10,

$$\mathrm{QN}(M(B,n)) \geq \left\lfloor \frac{B+1}{2} \right\rfloor = \left\lceil \frac{B}{2} \right\rceil .$$

□

Since every bandwidth B graph is a subgraph of some $M(B,n)$, Lemma 5.1 immediately yields

Theorem 5.3 *If* $\mathrm{BW}(G) = B$, *then* $\mathrm{QN}(G) \leq \lceil B/2 \rceil$.

5.3. Bifurcator Size

Next, we consider a measure of the difficulty of recursively decomposing a graph into two subgraphs of approximately equal size. That measure is the notion of bifurcator in Bhatt and Leighton [BL85]. An n-vertex graph G has an *α-bifurcator of size F*, $\alpha \geq 1$, F a function of n, if G has a *(F, α)-decomposition tree* defined recursively as follows. G is the root of the tree. If G is empty or has fewer than $F(n)$ edges, then G has no children. Otherwise, the children of G are two equal-size graphs that partition G such that the number of edges between them is at most $F(n)$; each of these graphs is the root of a $(F(n)/\alpha, \alpha)$-decomposition tree. $\sqrt{2}$-bifurcators are important in obtaining good VLSI layouts ([BL85]).

For our purposes, we can use a $\sqrt{2}$-bifurcator of a graph to produce a queue layout.

Theorem 5.4 *If G has a $\sqrt{2}$-bifurcator of size $F(n)$, then*

$$\mathrm{QN}(G) = O\left(n/\log(n/F(n))\right).$$

Proof: [BL85] shows that a graph having a $\sqrt{2}$-bifurcator of size F has bandwidth $O\left(n/\log(n/F(n))\right)$. By Theorem 5.3, the result follows. □

5.4. Cutwidth and Valence

Let $\sigma = 1, 2, \ldots, n$ be a fixed order of the vertices of G. Intuitively, the cutwidth of σ is the maximum number of edges cut by any line perpendicular to σ. Formally, define the *cut at vertex i*, $1 \leq i \leq n-1$, to be

$$\mathrm{CUT}(i) = \{(j,k) \mid 1 \leq j \leq i < k \leq n\}.$$

The *cutwidth* of σ is

$$\mathrm{CW}(\sigma) = \max_i |\mathrm{CUT}(i)|.$$

We develop a tradeoff between the queuenumber and the stacknumber of σ using the following result due to Erdös and Szekeres [ES35].

Proposition 5.1 *Suppose P is the sequence $\pi(1), \ldots, \pi(n)$ where π is some permutation of $1, \ldots, n$. Let a be the length of the longest ascending subsequence in P and let d be the length of the longest descending subsequence in P. Then $ad \geq n$.*

The tradeoff is based on finding an interesting matching in the graph.

Theorem 5.5 *Let $\sigma = 1, 2, \ldots, n$ be a fixed order of the vertices of G. Then*

$$\mathrm{SN}(\sigma) \times \mathrm{QN}(\sigma) \geq \mathrm{CW}(\sigma)/\mathrm{valence}(G).$$

Proof: Choose i, $1 \leq i \leq n-1$, such that $|\mathrm{CUT}(i)| = \mathrm{CW}(\sigma)$. $\mathrm{CUT}(i)$ is the edge set of a bipartite graph H with valence$(H) \leq$ valence(G). Select a maximum matching $M \subset \mathrm{CUT}(i)$ in H. The size of M is at least $\mathrm{CW}(\sigma)/\mathrm{valence}(G)$.

The left vertices of M give an order to the edges in M. The right vertices give some permutation π of that order. Let a and d be as required for Proposition 5.1. Then a gives the length of a longest similarly ordered sequence between left and right vertices of M. Therefore M contains an a-twist. By Proposition 2.2, $\mathrm{SN}(\sigma) \geq a$. Similarly, M contains a d-rainbow. By Proposition 2.1, $\mathrm{QN}(\sigma) \geq d$. Finally,

$$\mathrm{SN}(\sigma) \times \mathrm{QN}(\sigma) \geq ad \geq |M| \geq \mathrm{CW}(\sigma)/\mathrm{valence}(G).$$

□

The factor valence(G) is necessary. Consider the *star graph* G with vertex set $\{1, 2, \ldots, n\}$ and edge set $\{(1, i) \mid 2 \leq i \leq n\}$. If $\sigma = 1, 2, \ldots, n$, then $\mathrm{SN}(\sigma) = 1$, $\mathrm{QN}(\sigma) = 1$, $\mathrm{CW}(\sigma) = n - 1$, and valence$(G) = n - 1$.

6. Queuewidth and Graph Structure

In this section, the queuewidth of a 1-queue layout is related to the diameter of the graph. We provide evidence of an apparent tradeoff between queuenumber and queuewidth for queue layouts of complete binary trees.

The *diameter* of a connected graph is the greatest distance between two vertices in the graph. The next theorem suggests a tradeoff between diameter and queuewidth for 1-queue graphs.

Theorem 6.1 *Suppose G is a connected 1-queue graph having diameter D. Let QL be a 1-queue layout of G. Then,*

$$\mathrm{QW}(QL) \geq |E|/(2D + 1).$$

Proof: By Theorem 3.1, layout QL yields an arched level planar embedding of G having levels $V_1, V_2, \ldots, V_m$. Consider the following $2m - 1$ cuts of QL

$$\mathrm{CUT}(t_1 - 1), \mathrm{CUT}(t_1), \ldots, \mathrm{CUT}(t_i - 1), \mathrm{CUT}(t_i), \ldots, \mathrm{CUT}(t_m - 1).$$

Every edge either has some t_i as a right endpoint or passes over some t_i. Thus, every edge is in at least one of these cuts. By the definition of diameter, $m \leq D + 1$. There are at most $2D + 1$ cuts. The result follows. □

For a depth-d complete binary tree T, there are $n = 2^d$ leaves, $|V| = 2n - 1$, $|E| = 2n - 2$ and $D = 2d = 2 \log n$. We have this corollary.

Corollary 6.1 *Any 1-queue layout of a depth-d complete binary tree has queuewidth at least $(2n - 2)/(4d + 1) = \Omega(n/\log n)$.*

The breadth-first layout of T starting at the root has queuewidth n. The lower bound on queuewidth in the corollary is close to this upper bound. We do not know how to achieve this lower bound and doing so appears difficult. These bounds are much larger than the $O(\log n)$ stackwidth of a 1-stack layout of T ([CLR87]). The higher width of queue layouts over stack layouts suggests that stack layouts of trees are preferable to queue layouts. The question arises whether a lower cumulative queuewidth can be achieved by using additional queues. The following theorem answers the question affirmatively.

Theorem 6.2 *A depth-d complete binary tree T with $n = 2^d$ leaves has a k-queue layout QL with* $\mathrm{CQW}(QL) = O(kn^{1/k})$.

Proof: We give the proof for $k = 2$. To simplify the construction, we assume that $d = 2d'$ is even. Let $n' = 2^{d'} = \sqrt{n}$. Let T_p be the upper $d' + 1$ levels of T, and let $1, 2, \ldots, n'$ be the leaves of T_p in canonical order. Each i is the root of a subtree T_i of depth d'. Order the vertices of T_p in the obvious breadth-first order so that $1, \ldots, n'$ appear rightmost in the order. For each $i, 1 \leq i \leq n'$, place the vertices of T_i in breadth-first order immediately to the right of its root i. Assign the edges of T_p to one queue and the edges of $T_1, T_2, \ldots, T_{n'}$ to a second queue. Each queue has queuewidth $n' = \sqrt{n}$. The cumulative queuewidth of the 2-queue layout is $2n' = O(2n^{1/2})$.

The details of the proof for $k > 2$ are left to the reader. □

We have no lower bounds on cumulative queuewidth for multi-queue embeddings of T. Thus we do not know whether there is a real tradeoff between queuenumber and queuewidth here.

7. Future Directions

We have offered two conjectures in Section 3.3. We believe there is hope of resolving the second conjecture, that the queuenumber of planar graphs is bounded. Further comparison of the relative merits of queues and stacks is warranted. In particular, queues appear to be more appropriate than stacks for graphs with a leveled structure;

can this insight be formalized? [CLR87] and [He87] show that there are tradeoffs between stacknumber and stackwidth in the sense that, for certain graphs, devoting more stacks to a layout decreases the cumulative stackwidth. We expect that there are analogous tradeoffs between queuenumber and queuewidth.

The notions of stack and queue layouts may be generalized in several directions. One approach is to define layouts that simultaneously utilize queues and stacks. We conjecture that each planar graph admits a 1-stack, 1-queue layout. Another approach is to utilize deques or more general permutation mechanisms. In the realm of such generality, it becomes necessary to consider relative cost measures for the various mechanisms.

Acknowledgements

Part of this research was conducted while the first author was at the University of North Carolina at Chapel Hill. A portion of the work of the second author was done during a visit to the Department of Applied Mathematics and Informatics of the University of Saarbrucken.

We wish to thank Sandeep Bhatt, Fan Chung, Tom Leighton, and Andrew Reibman for helpful conversations. We also acknowledge the support of the National Science Foundation Grants DCI-87-96236 and CCR-88-12567.

References

[B64] V. E. Benes, "Optimal rearrangeable multistage connecting networks," *Bell System Technical Journal,* Vol. 43, 1964, pp. 1641-1656.

[BK79] F. Bernhart and B. Kainen, "The book thickness of a graph," *Journal of Combinatorial Theory B,* vol. 27, 1979, pp. 320-331.

[BL85] S. Bhatt and F. T. Leighton, "A framework for solving VLSI graph layout problems," *Journal of Computer and System Sciences,* vol. 28, 1984, pp. 300-343.

[BS84] J. Buss and P. Shor, "On the pagenumber of planar graphs," *Proceedings of the 16th ACM Symposium on Theory of Computing,* pp. 98-100.

[CLR87] F. R. K. Chung, F. T. Leighton and A. L. Rosenberg, "Embedding graphs in books: a layout problem with applications to VLSI design," *SIAM Journal on Algebraic and Discrete Methods,* vol. 8, 1987, pp. 33-58.

[ES35] P. Erdös and E. Szekeres, "A combinatorial problem in geometry," *Compositio Math.,* vol. 2, 1935, pp. 463-470.

[EI71] S. Even and A. Itai, "Queues, stacks and graphs," In *Theory of Machines and Computations* (Z. Kohavi and A. Paz eds.) Academic Press, NY, 1971, pp. 71-86.

[GKS89] Z. Galil, R. Kannan, and E. Szemerédi, "On nontrivial separators for k-page graphs and simulations by nondeterministic one-tape Turing machines," *Journal of Computer and System Sciences,* Vol. 38, 1989, pp. 134-149.

[G87] R. Games, "Optimal book embeddings of the FFT, Benes, and barrel shifter networks," *Algorithmica,* Vol. 1, 1986, pp. 233-250.

[GJ79] M. R. Garey and D. S. Johnson, *Computers and Intractability.* W. H. Freeman and Company, New York, 1979.

[GJMP80] M. R. Garey, D. S. Johnson, G. L. Miller and C. H. Papadimitriou, "The complexity of coloring circular arcs and chords," *SIAM Journal on Algebraic and Discrete Methods,* Vol. 1, 1980, pp. 216-227.

[He84] L. S. Heath, "Embedding planar graphs in seven pages," *Proceedings of the 25th Annual IEEE Symposium on Foundations of Computer Science,* 1984, pp. 74-83.

[He85] L. S. Heath, "Algorithms for embedding graphs in books," Ph.D. dissertation, University of North Carolina at Chapel Hill, 1985.

[He87] L. S. Heath, "Embedding outerplanar graphs in small books," *SIAM Journal on Discrete and Algebraic Methods,* Vol. , 1987, pp.

[Hs85] W.-L. Hsu, "Maximum weight clique algorithms for circular-arc graphs and circle graphs," *SIAM Journal on Computing,* Vol. 14, 1985, pp. 224-231.

[L82] D. Lichtenstein, "Planar formulae and their uses," *SIAM Journal on Computing,* Vol. 11, 1982, pp. 329-343.

[M88] S. M. Malitz, "Genus g graphs have pagenumber $O(\sqrt{g})$," *Proceedings of the 29th Annual IEEE Symposium on Foundations of Computer Science,* 1988, pp. 458-468.

[MWW88] D. J. Muder, M. L. Weaver, and D. B. West, "Pagenumber of complete bipartite graphs," *Journal of Graph Theory,* Vol. 12, 1988, pp. 469-489.

[Re84] A. Reibman, "DIOGENES layouts using queues," Typescript, Duke University, 1984.

[Ro83] A. L. Rosenberg, "The DIOGENES approach to testable fault-tolerant arrays of processors," *IEEE Transactions on Computers,* vol. C-32, 1983, pp. 902-910.

[SI79] M. M. Sysło and M. Iri, "Efficient outerplanarity testing," *Fundamenta Informaticae,* Vol. 2, 1979, pp. 261-275.

[T72] R. E. Tarjan, "Sorting using networks of queues and stacks," *Journal of the ACM,* vol. 19, 1972, pp. 341-346.

[Y86] M. Yannakakis, "Four pages are necessary and sufficient for planar graphs," *Proceedings of the 18th Annual ACM Symposium on Theory of Computing,* 1986, pp. 104-108.

[Y89] M. Yannakakis, "Embedding planar graphs in four pages," *Journal of Computer and System Sciences,* vol. 38, 1989, pp. 36-67.