Journal of Graph Algorithms and Applications
http://jgaa.info/ vol. 10, no. 2, pp. 387–420 (2006)

A New Algorithm for Finding Minimal Cycle-Breaking Sets of Turns in a Graph

Lev Levitin *Mark Karpovsky* *Mehmet Mustafa*
Department of Electrical and Computer Engineering
Boston University
http://www.bu.edu/
levitin@bu.edu markkar@bu.edu mmustafa@bu.edu

Lev Zakrevsky
Department of Electrical and Computer Science
New Jersey Institute of Technology
http://www-ec.njit.edu/
lev.a.zakrevski@njit.edu

Abstract

We consider the problem of constructing a minimal cycle-breaking set of turns for a given undirected graph. This problem is important for deadlock-free wormhole routing in computer and communication networks, such as Networks of Workstations. The proposed Cycle Breaking algorithm, or CB algorithm, guarantees that the constructed set of prohibited turns is minimal and that the fraction of the prohibited turns does not exceed 1/3 for any graph. The computational complexity of the proposed algorithm is $O(N^2\Delta)$, where N is the number of vertices, and Δ is the maximum node degree. The memory complexity of the algorithm is $O(N\Delta)$.

We provide lower bounds on the minimum size of cycle-breaking sets for connected graphs. Further, we construct minimal cycle-breaking sets and establish bounds on the minimum fraction of prohibited turns for two important classes of graphs, namely, t-partite graphs and graphs with small degrees. The upper bounds are tight and demonstrate the optimality of the CB algorithm for certain classes of graphs. Results of computer simulations illustrate the superiority of the proposed CB algorithm as compared to the well-known and the widely used Up/Down technique.

Article Type	Communicated by	Submitted	Revised
regular paper	Khuller	July 2005	April 2006

This work was supported by the NSF under Grant MIP 9630096.

1 Introduction

Recently, Networks of Workstations (NOWs) [11, 16, 19, 24, 25], have emerged as an inexpensive alternative to massively parallel multiprocessors [10, 19]. NOWs comprise a collection of routing switches, communication links and workstations interconnected in an ad hoc manner resulting in a graph of irregular topology. In order to minimize network latency and achieve high-bandwidth communications, recent experimental and commercial switches for NOWs implement wormhole routing [19, 15]. In wormhole routing, each message or packet is sent in chunks referred to as flits (message flow control units) [6, 10]. Each flit is transferred in parallel between adjacent nodes. The header flit (the first one of each message) contains the destination address. When a router at a node receives a header flit, it forwards the flit through an available communication channel towards the destination without having to wait for the rest of the message. If no available channel is found due to congestion, the header flit is blocked and is forced to wait until a channel becomes available.

However, because packets are allowed to hold many resources (channels) while requesting others, wormhole routing is very susceptible to deadlocks [9, 11, 19]. Figure 1 depicts a section of a network in which no measures are taken to prevent deadlock. (The rest of the network where four deadlocked messages have been originated is not shown.) The figure shows four-port routers with their local processors presented as circles. Assume that each message Mi is destined for node i. We show a scenario where communication channels have been occupied by the messages shown juxtaposed next to them. The rest of the messages occupy a number of other channels in the network. It is seen that four messages, M1, M2, M3, and M4 are blocking each other, so that no one can move forward. For example, message M2 has acquired ownership of the vertical communication channel south of node 4, within node 4, and north of node 4 but is waiting for the channel between nodes 1 and 2 which has already been committed to M3. Thus, deadlock prevention has become an important problem in wormhole communication networks.

In [17, 22], authors implemented a layered shortest path routing algorithm, LASH, where a number of virtual paths were used to break cycles in their simulated NOW clusters. In their model, layers contain only unidirectional paths where path assignment is done in a way to prevent cycle formation. Authors conclude that the state-of-the-art Infiniband^{TM} switches could be used to implement their protocols. In contrast, the IP routing is a store-and-forward technology, in which the entire message is stored by a router before it is forwarded to one of the adjacent routers. Furthermore, in networks employing IP technology the physical layer is predominantly Ethernet based in which a spanning tree protocol is used to break all cycles [7].

It has been proven in [9] that the absence of cycles in the channel dependency graph is a sufficient condition for deadlock-free routing. Later it has been shown [21] that this is also a necessary condition for deadlock-free coherent routing algorithms. The elimination of cycles in the channel dependency graph is equivalent to elimination of all cycles in the sense of **Definition 3** (see Section

2, below) in the graph of original communication network. This can be accomplished by prohibition of a carefully selected set of turns in the graph. A turn in a graph G is a three-tuple of nodes, (a, b, c), and (a, b) and (b, c) are edges in G. In order to model existing switch-based networks we assume that G is undirected. Several routing methods using turn prohibition currently exist for regular topologies, such as 2-dimensional meshes, tori or hypercubes [10, 12, 19]. Recently, a number of publications [5, 9, 10, 11, 13, 16, 18, 19, 23, 27] have been devoted to the problem.

It was shown in [12] for meshes and tori and in [18, 24, 25] for irregular topologies that reduction in the number of prohibited turns results in a decrease of average path lengths and in a reduction of average message delivery time, thereby increasing the throughput. The experimental data in Figure 2 show that for each percentage point reduction in the fraction of prohibited turns, there is a considerable gain (7.77) in the maximum sustainable throughput in the network which is referred to as the saturation point, beyond which the message delivery latency tends to infinity [1].

For a general topology, most of the existing routing strategies are based on the spanning tree approach [16]. According to this strategy, a spanning tree is constructed which is subsequently used for communication, thus guaranteeing deadlock freedom. The main shortcomings of this approach are long message paths and congestion on the edges near the root node [16]. This approach is also very inefficient since a large number of links are not used. This method can be improved by allowing shortcuts using cross-edges that do not belong to the spanning tree. For example, for the widely used Up/Down routing [16], after a spanning tree is constructed for G, nodes are labeled preserving the partial order defined by the tree with the root having the label 1. If we denote the label of a node a as ℓ_a, then turn (ℓ_a, ℓ_b, ℓ_c) is prohibited iff $\ell_b > \ell_a$ and $\ell_b > \ell_c$.

For the Up/Down approach [16, 26], given a network topology, the fraction of prohibited turns for deadlock-free routing, depends not only on the selection

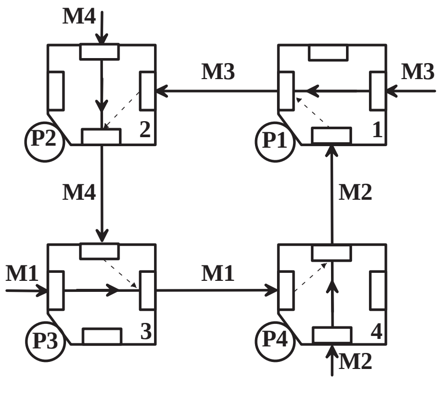

Figure 1: Formation of deadlock in a section of a network in which all turns are permitted. Message Mi is destined for processor Pi at node i, i=1,. . . ,4

of a spanning tree but also on the root of the spanning tree, and could be very close to one [23]. The problem of construction of an optimal spanning tree is NP-hard.

The general problem of finding the minimum set of turns such that it intersects with the set of turns in each cycle in a given graph (the "cycle-breaking problem") has been first introduced in [18, 23, 24, 27], as motivated by its importance for deadlock-free routing in communication networks. It should be pointed out that the cycle-breaking problem is a sub-problem of the general "set covering problem" [4, 8]. A number of similar (or even simpler looking) sub-problems of the set covering problem, such as vertex cover problem, code covering radius problem, etc., are NP-hard. We expect that cycle-breaking problem to be NP-hard as well, though it is not proven yet. Therefore, it is a challenge to develop an algorithm of small complexity that would provide a minimal solution of the problem.

In [18, 23, 24, 27], earlier versions of the turn prohibition algorithm were suggested which did not guarantee minimality and might become inefficient for certain topologies. Attempting to construct a minimal set from a given non-minimal set of prohibited turns is impractical since for a graph $G(V,E)$, the complexity of such an approach is $O(\Delta^2 N^3)$, where $\Delta = \max d_i$, $i = 1, 2, \ldots, N$,

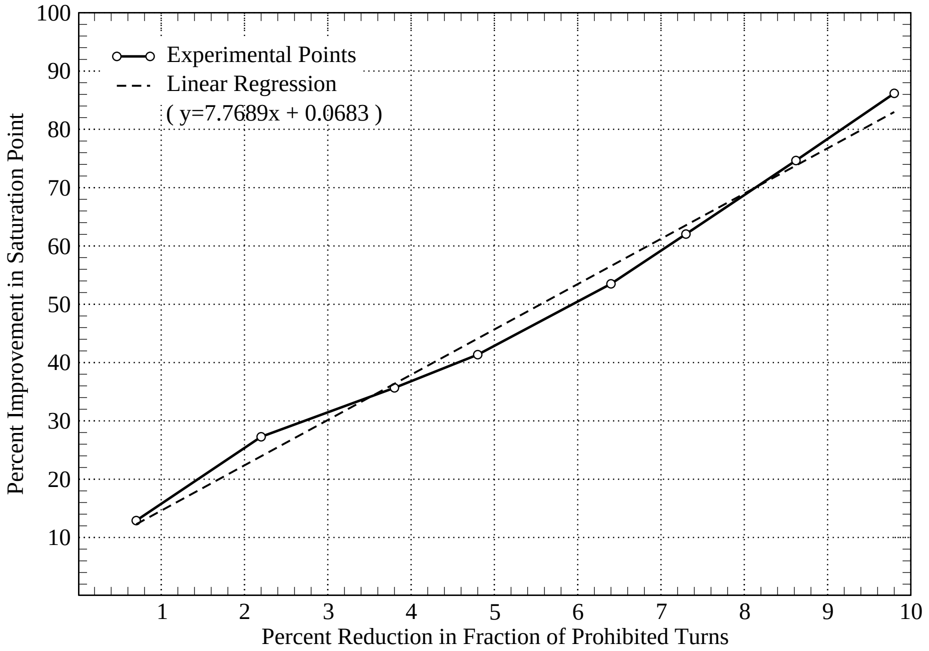

Figure 2: Effect of reducing the fraction of prohibited turns on the saturation point. The dotted line (with a slope of 7.77) shows the linear regression least squares fitting to the experimental points with a fit quality of $R^2 = 0.99$.

and d_i is the degree of node $v_i \in V$, $N = |V|$ is the number of nodes in G. Paper [23] was devoted to application of network calculus in which a turn prohibition (TP) algorithm was presented briefly, and its properties were stated without rigorous proofs.

The present paper formulates an algorithm (the CB algorithm) for construction of a minimal set of prohibited turns, presents rigorous analysis of its properties and performance, establishes lower bounds on the size of the solution, and proves that, for certain classes of graphs (t-partite graphs, graphs with small degree nodes) the CB algorithm is optimal or, in some cases, at least asymptotically optimal. The time complexity of the developed algorithm is $O(N^2\Delta)$ and the required memory complexity is $O(N\Delta)$. The approach developed in this paper provides a viable alternative to other methods of deadlock resolution, e.g. [15, 16, 20]. Furthermore, its results make it possible to increase the maximum load in the network without reaching saturation. Indeed, the fraction of prohibited turns is reduced by at least 15% compared to the Up/Down algorithm (the best known up to now) which corresponds to more than 116% increase of the saturation point.

The rest of the paper is organized as follows. In Section 2, we introduce the mathematical model, followed by establishing lower bounds on the fraction of prohibited turns in Section 3. Section 4 describes the CB algorithm for construction of minimal (irreducible) sets of prohibited turns with the fraction of prohibited turns not exceeding 1/3 for any graph. The main properties of the CB algorithm are formulated and proved in Section 5. Sections 6 and 7 contain an analysis and evaluation of the upper and lower bounds on fractions of prohibited turns for complete bipartite and the t-partite graphs and for graphs with small degrees, respectively. The bounds are shown to be tight for particular classes of graphs. Finally, we present experimental results for randomly generated topologies and offer our conclusions. Our simulation results (Section 8) for topologies with 64 nodes show that the proposed CB algorithm reduces the number of prohibited turns significantly when compared with the Up/Down approach.

2 Mathematical Model

Let us consider an undirected graph $G(V, E)$, with $N = |V|$ vertices or nodes, denoted by $a, b, \ldots$, and $M = |E|$ edges, denoted by (a, b), etc. We assume that graph is connected, i.e. there is a path between any two nodes in G. If this is not the case, we consider individual components separately.

Definition 1 *A **turn** in a graph G is a 3-tuple of nodes (a, b, c) if (a, b) and (b, c) are edges in G and $a \neq c$.*

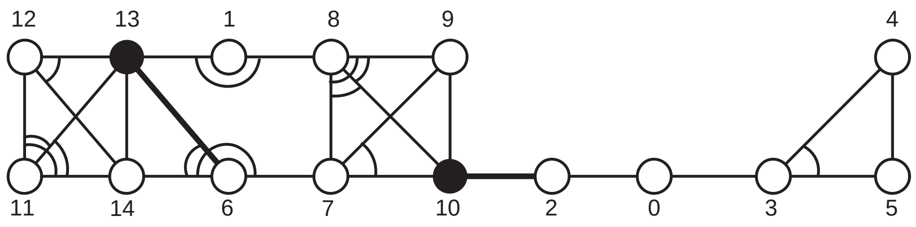

Figure 3: Construction of prohibited turns in a simple graph with cut-nodes. Special edges are shown as thick lines and delayed nodes are shown as solid circles. A prohibited turn is shown as an arc between the two edges defining the turn. For example turns $(4, 3, 5)$ and $(7, 6, 14)$ are prohibited and turn $(3, 4, 5)$ is permitted.

We note that the turn (a, b, c) denotes the same turn as (c, b, a). If the degree of node j is denoted as d_j, and the total number of turns in G is $T(G)$, we have

$$T(G) = \sum_{j=1}^{N} \binom{d_j}{2} = \sum_{j=1}^{N} \frac{d_j(d_j - 1)}{2}. \tag{1}$$

Definition 2 *A* ***path*** *$P = (v_0, v_1, \ldots, v_{L-1}, v_L)$ from node a to node b in G is a sequence of nodes $v_i \in V$ such that, $v_0 = a$ and $v_L = b$, every two consecutive nodes are connected by an edge, and $v_i \neq v_{i+2}$*

Nodes and edges in the path are not necessarily all different but sequences such as $\ldots, a, b, a, \ldots$ are not permitted.

Definition 3 *Path $P = (v_0, v_1, \ldots, v_{k-1}, v_k = v_0, v_1)$ in G is called a* ***cycle*** *of length k, if any ordered pair of nodes (v_i, v_{i+1}), appears at most once in P, except (v_0, v_1) that appears exactly twice.*

If no proper subset of nodes of the cycle P forms a cycle, we call P a ***simple cycle***.

Examples of cycles which are broken by the prohibited turns as shown for the graph in Figure 3 are:

Simple cycle:	(14,13,6,14,13)
Spectacles-type cycle:	(14,13,6,7,9,10,7,6,14,13)
8-type cycle:	(11,12,13,1,8,7,6,13,11,12)
Non-simple cycle:	(11,13,6,14,11,13)

The cycle $(11, 13, 6, 14, 11, 13)$ is not simple because the set of nodes $\{11, 13, 14\} \subset \{11, 13, 6, 14, 11, 13\}$. The nodes $\{11, 13, 14\}$ form a simple cycle $(11, 13, 14, 11, 13)$.

Note that our definitions of a path and a cycle are somewhat different from the conventional definitions [2, 3, 8, 14]. It can be said that we consider "cycles of ordered pairs of adjacent nodes", rather than "cycles of nodes". The reason for this definition is that cycles of ordered pairs of adjacent nodes result in deadlocks in networks of workstations, where computing nodes correspond to the vertices and communication links correspond to the edges of the graph G. Breaking all such cycles in G is necessary and sufficient for preventing deadlocks in the corresponding network [9].

Definition 4 *If edges (a, b) and (b, c) are adjacent and belong to path $P = (v_0, v_1, \ldots, v_L)$ such that, $a = v_{i-1}$, $b = v_i$, $c = v_{i+1}$, $i \in 1, 2, \ldots, L-1$, then turn (a, b, c) is said to* ***cover*** *P, i.e., $(a, b, c) \in P$.*

Definition 5 *A set $W(G)$ of turns in G is called* ***cycle-breaking*** *if every cycle in G is covered by at least one turn from $W(G)$. Elements of $W(G)$ are called* ***prohibited turns****.*

The set $A(G) = \Gamma(G) \setminus W(G)$ is called the set of permitted turns, where $\Gamma(G)$is the set of all turns of the graph G. A path P in G is called permitted if all turns covering P belong to $A(G)$, otherwise the path is ***prohibited****.*

We say that the cycle-breaking set $W(G)$ of prohibited turns ***preserves connectivity*** *if for any two nodes $a, b \in V$, there exists at least one permitted path from a to b.*

For the topology in Figure 4, cycle $(13, 1, 8, 7, 6, 13, 1)$ is covered by the turn $(13, 1, 8)$. As an example, one cycle-breaking set of prohibited turns for the same topology is

$$W(G) = \left\{ \begin{array}{llll} (8,1,13), & (10,8,9), & (8,7,10), & (8,7,9), \\ (9,7,10), & (7,6,13), & (13,6,14), & (13,12,14), \\ (12,11,14), & (12,11,13), & (13,11,14), & (4,3,5) \end{array} \right\}.$$

Given graph G representing a connected network topology, we shall consider in Section 4 the problem of finding a minimal cycle-breaking set of turns for G, which preserves connectivity of the graph. This problem was first formulated and solved for meshes by Glass and Ni [13].

Definition 6 *Path $P = (v_0, v_1, \ldots, v_0)$ in G is called a* ***halfloop*** *if it is not a cycle (see* ***Definition*** *3) and it is permitted under a given set of prohibited turns $W(G)$.*

The number of turns in a minimum cycle-breaking set is denoted by $Z(G) = \min |W(G)|$.

Definition 7 *Cycle-breaking set of turns $W(G)$ is* ***minimal (irreducible)****, if there are no cycle-breaking proper subsets of $W(G)$. In other words, deletion of any turn from a minimal set of prohibited turns will introduce at least one cycle in the graph.*

(We note that a minimal cycle-breaking set is not necessarily a minimum cycle-breaking set.)

3 Lower Bound on Minimal Cycle-Breaking Sets of Turns

In this section we present lower bounds for the fraction $z(G) = Z(G)/T(G)$ of prohibited turns to break all cycles without loss of connectivity in any connected graph $G(V, E)$ where $N = |V|$ and $M = |E|$.

Theorem 1 *If C is a set of cycles in a graph G with N nodes and M edges, $|C| = R$ and r is the maximum number of cycles in C covered by the same turn, then the fraction of prohibited turns $z(G) = Z(G)/T(G)$ satisfies the following inequalities:*

$$z(G) \geq M - N + 1, \tag{2}$$

and

$$z(G) \geq \frac{R}{rT(G)}. \tag{3}$$

Proof: The lower bound (2) follows from the fact that any cycle-breaking set of edges contains at least $\beta = M - N + 1$ elements, where β is the cyclomatic number for G [14], and each cycle-breaking set of turns (a, b, c) generates a cycle-breaking set of edges (a, b) with a smaller or equal number of elements. Lower bound (3) follows from the fact that R cycles should be covered by at least R/r turns. □

For example, for complete graphs K_N with N nodes, we have $M = N(N-1)/2$ and $T(K_N) = N(N-1)(N-2)/2$. If we enumerate the number of turn-disjoint triangular cycles in K_N we obtain $R = N(N-1)(N-2)/6$. Since in turn-disjoint cycles case we have $r = 1$ and by (3), we obtain $z(K_N) \geq 1/3$.

Lemma 1 *Consider a connected graph G with a minimum cycle-breaking set of turns $W(G) = \{(a_i, b_i, c_i)|i = 1, 2, \ldots, Z(G)\}$. If there exists an edge (a, b) such that all turns (a, b, c) are prohibited and $|\{(a, b, c)\}| = s$, then*

$$Z(G) \geq M - N + s. \tag{4}$$

Proof: Since the edge (a, b) is involved in s prohibited turns, we are assured that it will not be part of any cycle and therefore it can be deleted from the graph. After deleting the edge (a, b) we obtain a graph G' consisting of one or two components with total number of edges $M' = M - 1$ and $N' = N$ nodes. By **Theorem** 1, the number of turns to be prohibited in G' to break all cycles is

$$Z(G') \geq M' - N' + 1 = (M - 1) - N + 1 = M - N.$$

Hence,

$$Z(G) \geq M - N + s\,.$$

□

Using **Lemma** 1, we can improve lower bound (2) if in a given graph, in addition to the number of nodes N and the number of edges M, we also know the minimum node degree δ.

Theorem 2 *Let $G(V,E)$ be a connected graph with minimum degree $\delta = \min d_i$, $i = 1, 2, \ldots, N$ where d_1 is the degree of node $v_i \in V$ and $\delta > 2$. Then*

$$Z(G) \geq M - N + \binom{\delta - 1}{2} + 1. \tag{5}$$

Proof: Proof is by induction. Assume that, in graph G with non-empty cycle-breaking set of turns $W(G)$, there is no edge (a,b) such that all turns (a,b,c) are prohibited. This means that, arriving to a node b along the edge (a,b), one can always find an edge (b,c) to leave the node. In other words, there exists paths of unlimited lengths in G. Since the number of edges in G is finite, the same edge in the same direction will be repeated in a path, thereby forming a cycle. This contradiction proves that there should exist an edge (a,b) with all of the turns (a,b,c) prohibited. The number of such turns is at least $\delta - 1$. By Lemma 1, we obtain $Z(G) \geq M - N + (\delta - 1)$. Thus, for $\delta = 3$, the lower bound (5) is valid:

$$\begin{aligned} Z(G) &\geq M - N + 2 \\ &\geq M - N + \binom{2}{2} + 1 \,. \end{aligned}$$

Assume that the lower bound (5) is valid for all graphs with minimum degree $\delta - 1$. Now, consider a connected graph G with minimum degree δ. After removing edge (a,b) involved in all $\delta - 1$ prohibited turns (a,b,c) with adjacent nodes, we obtain a graph G' with minimum degree at least $\delta - 1$, number of nodes N, and the number of edges $M' = M - 1$. By assumption,

$$Z(G') \geq (M-1) - N + \binom{\delta - 2}{2} + 1 \,.$$

Hence,

$$\begin{aligned} Z(G) &\geq (\delta - 1) + Z(G') \\ &\geq (M-1) - N + \binom{\delta - 2}{2} + 1 + (\delta - 1), \end{aligned}$$

or,

$$Z(G) \geq M - N + \binom{\delta - 1}{2} + 1 \,.$$

□

As shown below in **Corollary** 6, bound (5) is attained for $\delta = 3$. However, we believe that a stronger lower bound is valid.

Conjecture 1 $Z(G) \geq M - N + \binom{\delta}{3} + 1.$

Theorem 3 *If G is a connected graph and G^* is a homeomorphic graph to G [14] obtained by adding a node of degree 2 in the middle of one of the edges in G, then*

$$Z(G) \geq Z(G^*). \tag{6}$$

Proof: Any set of turns that breaks all cycles in G is obviously a cycle-breaking set in G^* as well, which proves the theorem. □

Corollary 1 *For any connected graph G with M edges and N nodes there exists a homeomorphic graph $\widetilde{G}$ such that*

$$Z(\widetilde{G}) = M - N + 1. \tag{7}$$

Proof: Consider a spanning tree in G. There are $M - N + 1$ edges that do not belong to the spanning tree. By adding a node of degree 2 at each one of these edges and prohibiting turns at these nodes, all cycles will be broken, which proves Corollary 1. □

4 CB Algorithm for Constructing Irreducible Sets of Prohibited Turns

In this section we describe the Cycle Breaking or CB algorithm. Given a connected graph $G(V, E)$ with $N(G) = |V|$ nodes, the CB algorithm constructs a minimal set of prohibited turns $W(G)$, breaking all cycles and preserving the connectivity of G. Furthermore, the CB algorithm guarantees that the fraction of prohibited turns will not exceed 1/3. As far as we know, this is the first algorithm providing minimal set of prohibited turns and a nontrivial upper bound for the fraction of prohibited turns breaking all cycles.

The algorithm is recursive. At each recursive call of the algorithm one node is selected and each turn at the selected node is either permitted or prohibited and the selected node is then deleted. For example, if, after deleting a node a with degree d_a and all its edges from G, the remaining graph $G - a$ is still connected, then we prohibit all $d_a(d_a - 1)/2$ turns (c, a, b) and permit all turns (a, b, c). By prohibiting all turns (c, a, b) at node a, we break all cycles that include node a. Since the node a cannot participate in any cycle formation, it can be ignored from further consideration. This is accomplished by deleting node a from the graph. Then, the CB algorithm is invoked recursively and applied to the smaller graph $G - a$. At every recursive call of the algorithm labeling of a node to be deleted is done by using the smallest natural number that has not been used at the previous recursive calls as its label. We label a node by assigning a natural number to the node that indicates the order in which the node has been selected. Initially, all nodes are unlabeled. In the course of the algorithm, a node can also be marked as *forcing* or *delayed*. Nodes that have not been marked *forcing* or *delayed* are called *ordinary* nodes. As described in the algorithm, an edge can be marked as *special* if the selected node is a cut-node. The variable called *HALFLOOP* is initially cleared by assigning a value of 0 to it. When it is set, its value becomes 1 and remains set as 1. When set, this flag indicates that nodes attached to the *special* edges become *delayed* nodes. This is done to prevent spectacle type cycle formation.

4.1 Formal Description of CB Algorithm, $CB(G)$

We assume that we are given a connected graph $G(V, E)$ with $N(G) = |V|$ nodes. Before the algorithm starts, we initialize the sets for prohibited and permitted turns, $W(G) := \emptyset$, $A(G) := \emptyset$, and the variable $HALFLOOP := 0$, mark all nodes and edges as *ordinary* and all nodes as unlabeled.

1. If $N(G) = 1$, label the node and **RETURN**
2. If there exists a *forcing* node in G, select the *forcing* node and label it. Otherwise, select an *ordinary* node with minimum degree δ_a. If there are non-cut-nodes in the set V_m of all nodes of minimum degree, we select one node from the set V_m. Label the selected node.
3. Connected components of graph $G - a$, obtained by deleting the selected node and all its edges, are indexed as $G_1, G_2, \ldots, G_k$ using the following criteria:
   a. If there is a *delayed* or a *forcing* node in G, it should be in G_1.
   b. Otherwise, component G_i connected to the selected node a with smaller number of edges should have a larger index i.
4. For $i = 2, \ldots, k$, one edge that connects component G_i to a is marked *special.*
5. All turns (b, a, c) in which (a, b) is *special* **and** $b \in G_i$, $c \in G_j$ with $i > j$ belong to the permitted set $A(G)$, $A(G) := A(G) \cup \{(b, a, c)\}$. Otherwise, they belong to the prohibited set $W(G)$, $W(G) := W(G) \cup \{(b, a, c)\}$. All turns starting with the selected node (a, p, q), where $p, q \in G$, are permitted and belong to set $A(G)$, $A(G) := A(G) \cup \{(a, p, q)\}$.
6. If $HALFLOOP = 1$ then node $x \in G_1$ connected to a is marked *forcing,* provided that x is of degree 1 in G_1 or a cut-node (articulation point) of degree 2 in G_1; otherwise, (if x is of degree 2 but is not a cut-node or if it has a degree larger than 2) x is marked *delayed.*
7. If G_1 has a delayed node b, and, after the deletion of the selected node a, the node b has degree 1 or is a cut-node of degree 2, then node b becomes a *forcing* node.
8. $CB(G_1)$
9. For $i = 2, \ldots, k$

a. If $HALFLOOP = 0$ **and**, after the CB algorithm has been applied to G_{i-1}, there exists a halfloop $(a, x_1, \ldots, a)$, where $x_1, x_2, \ldots, x_k \in G_{i-1}$ with $x_1, x_2, \ldots, x_k$ not necessarily all distinct then $HALFLOOP$:= 1.

b. If $HALFLOOP = 1$ then node x in G_i connected to a with special edge is marked *forcing*, provided that it is of degree 1, or a cut-node of degree 2. Otherwise it is marked *delayed*.

c. $CB(G_i)$

10. **RETURN**.

4.2 Informal Discussion

The CB algorithm is invoked by a call $CB(G)$ where the argument is the graph for which we seek to construct a minimal set of prohibited turns. Prior to the invoking the algorithm, two sets $W(G)$ and $A(G)$ are initialized to be empty, the $HALFLOOP$ flag is cleared to 0, all nodes are marked unlabeled, and all nodes and edges of the graph are marked *ordinary.* Steps $1-7$, 9a, and 9b comprise one recursive call of the algorithm. Thus, at each recursive call, exactly one node a is selected, and this recursive call can be numbered by the label of the node ℓ_a. At Step 1, the algorithm tests for completion. If there is just one node left, then node is labeled and algorithm returns with the sets $W(G)$ and $A(G)$ containing the set of prohibited and permitted turns respectively. At Step 2, if there exists a *forcing* node in G, we select the *forcing* node and label it. This recursive call is called a forced call. The motivation for selecting forcing nodes first is that no turns are prohibited at this recursive call of the algorithm. Note that according to the algorithm, there exists either at most one *forcing* node, or at most one *delayed* node, but not both of them simultaneously in each connected component at each recursive call. At next Step 3, we delete the selected node and its edges and index the connected components. In particular, if there is a *delayed* node it must belong to component G_1. We then index the remaining components based on the number of edges connecting them to the selected node. Component with a smaller number of edges connecting to the selected node has a larger index. This order of indexing minimizes the number of turns to be prohibited at Step 5. At Step 4, one edge connecting the selected node to each component, excluding G_1, is marked *special.* (If there are multiple edges, it is beneficial to choose *special* edge that ends at the node of largest degree). At Step 5 of the algorithm, we identify all turns at the selected node that will be prohibited and make them members of the prohibited set $W(G)$. If the selected node is not a cut-node, all turns at the node are prohibited. Similarly, all permitted turns are made members of the permitted set $A(G)$. The Step 6 is executed only when a is a forcing node. Then the node $x \in G_1$ connected to a (there exists exactly one such node) is marked either *forcing* (if it is of degree 1, or a cut-node of degree 2 in G), or *delayed.* At Step 7, if the *delayed* node, after deletion of the selected node a, turns out to be of degree 1, or a cut-node of degree 2, it is transformed into a *forcing* node. Thus, sooner or later, any *delayed* node becomes a *forcing* node. Then, the algorithm recurses by invoking itself with the component G_1 at Step 8. The remaining Step 9 is executed $k-1$ times whenever there are k connected components in $G-a$. If the flag $HALFLOOP = 0$, we determine if there is any *halfloop* in each component. When a *halfloop* is detected involving the selected node and nodes in the component G_{i-1}, $HALFLOOP$ is set to 1. Once the $HALFLOOP$ flag is set, it remains set until the completion of the algorithm. Therefore, after it has been set $HALFLOOP = 1$, there is no need to execute Step 9a (in fact, $k-1$ steps) in all following recursive calls, and Step 9b can be executed immediately after Step 7. The algorithm then checks again if $HALFLOOP$ flag is set, and if so, one node in each component G_i (i is larger than the index of the component where a *halfloop* has been found) connecting to the selected node is marked as either *forcing* or *delayed*. Note that *forcing* nodes have the smallest labels in their components. Subsequently, we invoke the CB algorithm for each component G_i, $i \geq 2$. Note that each node is selected exactly once in the course of the algorithm. Note also that no turns are ever prohibited at *delayed* or *forcing* nodes.

We note that the CB algorithm labels nodes such that:

- All nodes have different labels.
- All nodes of component G_j will have larger labels than nodes of component G_i, if $j > i$.
- If node b in component G_i is *forcing*, then $\ell_b < \ell_a$ for all other nodes $a \in G_i$.
- If node b in G_i component is *delayed*, then $\ell_b > \ell_a$ for all $a \in G_i$ that have been selected prior to b becoming a *forcing* node.

It can be shown [4] that by using the depth first search algorithm one can identify all cut-nodes and the connected components of a graph in $O(M)$ time. It follows that, for a graph with a maximum node degree Δ, the time complexity of the CB algorithm is $O(N^2\Delta)$ and required memory is $O(N\Delta)$.

4.3 Example

We now illustrate the operation of the CB algorithm with reference to the graph in 4. In the figure, we show node labels in parentheses after the node numbers. Labels show the order in which nodes are selected by the CB algorithm. At the first recursive call, we select the ordinary node 1 of degree 2, delete it and its edges $(1, 13)$, $(1, 8)$, and prohibit one turn. Note that all turns such as $(1, 13, 12)$ starting with node 1 are permitted. We next select node 2, which is a cut-node. We prohibit no turns at node 2, and node 2 and its edges are deleted, two connected components are created. The first component G_1 includes three nodes 0, 3, 4, and 5. We mark the edge $(10, 2)$, which connects the selected node 2 to G_2 as *special*, shown as thick line in the figure. We then apply the CB algorithm to G_1. Node 0 is selected without prohibiting any turns. After deleting node 0 and its edges, we are left with a subgraph in which all three nodes are non-cut-nodes of minimum degree 2. Arbitrarily, the CB algorithm selects node 3, prohibits the turn $(4, 3, 5)$, and deletes edges $(3, 4)$ and $(3, 5)$. Nodes 4 and 5 are then selected with no prohibited turns completing the handling of G_1. When we consider G_2, we discover a *halfloop* $2, 0, 3, 4, 5, 3, 0, 2$ in G_1, set the $HALFLOOP$, and mark node 10 as *delayed*. We are now ready to handle subgraph G_2. When we apply the CB to the subgraph G_2, we find out that there is no *forcing* node in G_2 and therefore select node 6, which satisfies the selection criterion. Since this is a cut-node we prohibit only the two turns as shown, thus maintaining the connectivity between the two components. After deleting node 6 and its edges, we discover two new components. The first one includes nodes 7, 8, 9, and a delayed node 10. The other component includes nodes 11, 12, 13, and 14. As indicated in Step 3a of the CB algorithm, the component with a delayed node becomes the new component G_1, which is handled first. We see that, since node 10 is delayed, its selection is deferred until it is promoted to be a *forcing* node in the component. In this component, node 8 is selected and all three turns are prohibited as shown. Then node 7 is selected prohibiting one turn. After node 7 and its edges are deleted, subgraph has only two nodes both of which are of degree 1. Since node 10 is a delayed node of degree 1, it is promoted to be a *forcing* node. Next recursive run selected the *forcing* node 10 as indicated by its label.

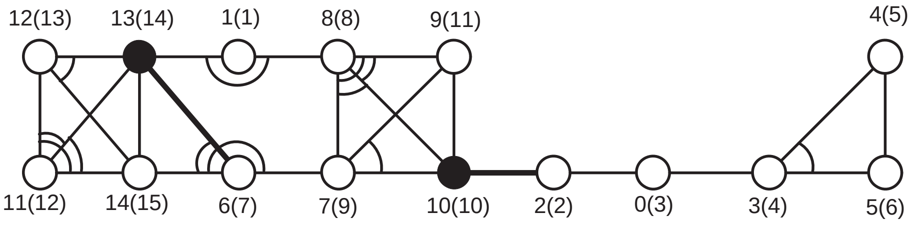

Figure 4: Prohibited turns generated by the CB algorithm showing delayed nodes as solid circles and special edges with thick lines

After a total of 15 recursive calls, a minimal set $W(G)$ of prohibited turns for G is constructed. In this case, $Z(G) = |W(G)| = 12$ turns are prohibited out of $T(G) = 50$. We note that if the initial selection order were $0, 4, 3, 5$, or $0, 4, 5, 3$, or $0, 5, 3, 4$, or $0, 5, 4, 3$, the $HALFLOOP$ flag would not have been set at the label 4 step. All of these alternative selection orders would not have created any *halfloop* in the connected component G_1. In the following table we demonstrate the step-by-step operation of the algorithm, showing the status of the nodes and any related edges, their labels, and the prohibited turns at every step of the algorithm. Note that when the $HALFLOOP$ flag is set when node 5 is selected, it remains set for the duration of the algorithm.

Table 1: Step-by-Step Operation of the CB algorithm for graph in Figure 4. Note that nodes 10 and 13 which were *delayed*, were subsequently promoted to *forcing*.

Selected Node	Node Label	Node Attribute	Special Edge	Delayed Node	*HALFLOOP*	Prohibited Turns
1	1	ordinary	none	none	0	{(13,1,8)}
2	2	cut	(10,2)	none	0	∅
0	3	ordinary	none	none	0	∅
3	4	ordinary	none	none	0	{(4,3,5)}
4	5	ordinary	none	none	0	∅
5	6	ordinary	none	10	1	∅
6	7	cut	(6,13)	none	1	{(13,6,14), (7,6,14)}
8	8	ordinary	none	none	1	{(7,8,9), (7,8,10), (9,8,10)}
7	9	ordinary	none	none	1	{(9,7,10)}
10	10	forcing	none	none	1	∅
9	11	ordinary	none	none	1	∅
11	12	ordinary	none	13	1	{(12,11,13), (12,11,14), (13,11,14)}
12	13	ordinary	none	none	1	{(13,12,14)}
13	14	forcing	none	none	1	∅
14	15	ordinary	none	none	1	∅

5 Main Properties of CB algorithm

Theorem 4 *CB algorithm has the following four properties.*

Property 1. Any cycle in G contains at least one turn from $W(G)$.
Property 2. For any two nodes a and b, if there exists a path between a and b in G, then there exists a path between a and b, with no turns from $W(G)$ along the path, after the CB algorithm is applied.
Property 3. For any graph G, $Z(G) \leq T(G)/3$, where $T(G)$ is the total number of turns in the graph.
Property 4. The set $W(G)$ of prohibited turns generated by CB algorithm is *minimal* (*irreducible*).

Proof of Property 1. First, we will prove the following lemma.

Lemma 2 *If x is a forcing or delayed node in a connected component G, then, after application of the CB algorithm to G, there is no permitted (not containing any turns from $W(G)$) closed path (halfloop) $P = (x, x_1, \ldots, x_k, x)$ in G, where x_i, $i = 1, \ldots, k$ are not necessarily distinct, but are different from x.*

We will prove the lemma by induction. For $N(G) \leq 3$ the lemma is obviously true. Assume that the lemma is valid for any graph G with $N(G) \leq N$. Then consider a graph with $N(G) = N + 1$, and let P be a closed path in G. If x is a *forcing* node then, after this node is selected and deleted, in each connected component of graph $G-x$ there is a *forcing* or a *delayed* node connected to x. Let x_1 be such a node belonging to P. Then P has a form $P = (x, x_1, x_2, \ldots, x_l, x_1, x)$ where $x_1, x_2, \ldots, x_l$ belong to the connected component G_1 of $G - x$. Hence, in G_1, there must be a closed path $P_1 = (x_1, x_2, \ldots, x_l, x_1)$. However, since $N(G_1) \leq N$, such a permitted path does not exist. Therefore, P is not permitted either, which proves the lemma.

Consider now the case when x is a *delayed* node. Let $x_i \in P$ be the node with the smallest label $\ell_{x_i} = \min \ell_{x_j}$, $x_j \in P$. At the recursive call of the algorithm when x_i is selected, the entire path belongs to the same connected component that includes the *delayed* node x. Two cases are possible.

1. After deleting node x_i, the remaining part of P belongs to the same connected component. Then the turn at x_i that covers P must be prohibited, thereby prohibiting path P.

2. After deleting node x_i, path P breaks into at least two parts, P_1 that includes x, and P_2, the parts belonging to different connected components. If P_2 is connected to x_i with at least two edges, then at least one of the turns at x_i that covers P, namely, the turn to a non-special edge, must be prohibited. If P_2 is connected to x_i with just one edge, then this edge is *special*, and the node x_{i+1} connected to x_i with this edge is either *forcing*, or *delayed*. Thus, by inductive assumption, there is no permitted path $P_2 = (x_{i+1}, \ldots, x_{i+1})$, and therefore, there is not permitted path P in G, which proves the lemma.

Return now to the proof of **Property 1**. We will also use induction over the number of nodes in G. For $N(G) < 3$, **Property 1** is trivial. Assume the property is true for all $N(G) \leq N$, and consider a graph G with $N(G) = N+1$. Let $a \in G$ be the node selected at the first recursive call, $\ell_a = 1$. First, consider cycles in G that include nodes from only one of the connected components of $G - a$. Since all turns at a between edges connecting to the same component are prohibited, all such cycles that include a are also prohibited. All cycles in one of the components that do not include a are prohibited by the inductive assumption.

Consider now cycles that include nodes from different connected components, G_i and G_j, where $i > j$. According to the CB algorithm only turns to the special edge, connecting a to G_i are permitted. Therefore, a cycle that includes nodes from G_i and G_j must include the edge (a, x) twice, where $x \in G_i$ is the end point of the special edge. To form a cycle, there should be a closed path (*halfloop*) $p_j = (a, y, \ldots, z, a)$, where $y, \ldots, z \in G_j$, and a path $P_i = (x, x_1, \ldots, x_k, x) \in G_i$. However, if P_j is permitted, then the node x is either *forcing* or *delayed*, and no permitted path P_i exists. Thus, **Property 1** is proved. $\square$

Proof of Property 2. We use induction over the number of nodes $N(G)$ in G. For $N(G) = 3$ the property is trivial. Let the property be true for all $N(G) \leq N$. Consider a graph with $N(G) = N + 1$ nodes. Select a node a and perform steps of the algorithm and delete the node and its edges. We obtain one or more components G_i, $i = 1, 2, \ldots, k$. Any two nodes in the same component are connected. If one node belongs to one component and the other one is either node a, or belongs to another component, they are connected with special edges, since all turns between special edges are permitted, as well as turns between edges connecting node a with G_1 and special edges. Hence, this run of the algorithm does not affect connectivity. Since $N(G_i) \leq N$ for all $i = 1, 2, \ldots, k$, the property is proved. □

Proof of Property 3. At recursive call ℓ_a of CB algorithm we prohibit a subset of the set of all turns (c, a, b) with $\ell_c > \ell_a$, $\ell_b > \ell_a$ and permit all turns (a, b, c) with $\ell_b > \ell_a$, $\ell_c > \ell_a$. The number of prohibited turns at run ℓ_a is $T_a \leq d_a(d_a - 1)/2$ and the number of permitted turns (a, b, c) is $D_a \geq \sum_{i \in nbors}(d_i - 1)$, where summation is made over all nodes i adjacent to node a. If node a has a minimal degree in the remaining graph at recursive call ℓ_a or if it is not connected with a *delayed* node, which has a degree smaller than d_a, then, since $d_a \leq d_i$ for all neighbors of a, $D_a \geq d_a(d_a - 1)$. The only remaining case is when all ordinary nodes of minimal (among ordinary nodes) degree are connected with a delayed node of degree $d' < d_a$. Then, at node a, at most $d' - 1$ edges end at nodes of degree d_a, while at least $(d_a - 1) - (d' - 1) = d_a - d'$ edges end at nodes with degrees at least $d_a + 1$. Thus, the number of permitted turns is

$$\begin{aligned} D_a &\geq (d' - 1)(d_a - 1) + (d_a - d')d_a + (d' - 1) \\ &\geq d_a(d_a - 1). \end{aligned}$$

Hence, in all cases, the number of permitted turns is larger than the number of prohibited turns by at least a factor of two. Since this is true for each run of the algorithm, it follows that $Z(G) \leq T(G)/3$. □

Note that the only graph with $Z(G) = T(G)/3$ is the complete graph K_n, with an edge between every two nodes.

Proof of Property 4. It should be noted that this property was shown not to be satisfied by the original TP algorithm [23]. Our proof here uses induction over the number of nodes $N = N(G)$. For $N \leq 3$ the property is trivial. Assume that the property is true for $N(G) \leq N$. Consider a graph G with $N(G) = N + 1$. Let a be the first selected node, $\ell_a = 1$. It is sufficient to prove that deleting a prohibited turn (b, a, c) from $W(G)$ creates a cycle.

If nodes b and c belong to the same connected component G_i, then, by **Property 1**, after completion of the CB algorithm, there exists a permitted path from b to c that belongs to component G_i, and, therefore, permitting the turn (b, a, c) creates a cycle $(a, b, \ldots, c, a, b)$.

Now, let $b \in G_i$ and $c \in G_j$ with $i > j$. Then the edge (a, b) is *ordinary* or *non-special*, there exists a *special* edge (a, d), $d \in G_i$. Hence, permitting turn (b, a, c) creates a cycle $(b, a, c, \ldots, e, a, d, \ldots, b, a)$. Note that turn (d, a, e), between a special edge connecting node a to G_i and an ordinary edge connecting a to G_j, $i > j$, is permitted by the CB algorithm, and therefore **Property 4** is proved. □

Now, we shall demonstrate the operation of the CB algorithm by applying it to an important class of graphs, namely to full bipartite graphs $K_{n,m}$ [14]. For $K_{n,m}$ class graphs the set of nodes consists of two disjoint subsets $\{a_1, \ldots, a_n\}$, $\{b_1, \ldots, b_m\}$ and set of edges $E = \{(a_i, b_j)|i = 1, \ldots, n, j = 1, \ldots, m\}$. Thus, $N = n + m$, $M = |E| = nm$, and $T(K_{n,m}) = \binom{n}{2} + \binom{m}{2}$.

For the bipartite graph $K_{3,3}$ in Figure 5, $N = 6$, $Z(K_{3,3}) = 5$, and an irreducible set of prohibited turns is $W(K_{3,3}) = \{(2, 1, 4), (2, 1, 6), (4, 1, 6), (3, 2, 5), (4, 3, 6)\}$. By (5), Theorem 2, this is optimal cycle-breaking set of turns for

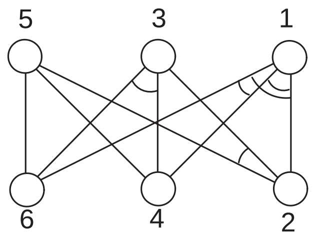

Figure 5: Prohibited turns generated by the CB for the bipartite graph $K_{3,3}$

$K_{3,3}$. For $K_{4,4}$, CB algorithm results in a cycle-breaking set of 14 turns out of a total of 48 turns, which is minimal number of prohibited turns for $K_{4,4}$. For arbitrary n, minimum degree nodes will be selected alternately from two parts of the bipartite graph. Denoting the required number of prohibited turns as $Z(K_{n,n})$, we obtain the following recursive equation

$$\begin{aligned} Z(K_{n,n}) &= Z(K_{n-1,n-1}) + \binom{n}{2} + \binom{n-1}{2} \\ &= Z(K_{n-1,n-1}) + (n-1)^2. \end{aligned}$$

Hence

$$\begin{aligned} Z(K_{n,n}) &= \sum_{k=1}^{n-1} k^2 \\ &= \frac{n(n-1)(2n-1)}{6}. \end{aligned} \tag{8}$$

If $m > n$, at the first stage $(m - n)\binom{n}{2}$ turns around $m - n$ nodes of the larger part containing m nodes will be prohibited, thus

$$Z(K_{n,m}) = \frac{n(n-1)(2n-1)}{6} + \frac{n(n-1)(m-n)}{2}$$

$$= \frac{n(n-1)(3m-n-1)}{6}. \tag{9}$$

6 Minimal Cycle-Breaking Sets of Turns for t-Partite Graphs

In this section we shall use the CB algorithm for determining the upper bounds on minimal cycle-breaking sets for bipartite and t-partite graphs. For the complete bipartite topology $K_{n,m}$ and the symmetric bipartite topology $K_{n,n}$, we obtain a lower bound for the fraction of prohibited turns by means of the following theorem and its corollaries.

Theorem 5 *For complete bipartite graphs $K_{n,n}$ with $n \le m$*

$$\frac{n-1}{2(m+n-2)} \le z(K_{n,m}) \le \frac{(n-1)(3m-n-1)}{3m(m+n-2)}\,. \tag{10}$$

Proof: To prove the upper bound, we use the set of prohibited turns constructed by the CB algorithm with $K_{n,m}$ given by (9). The total number of turns in $K_{n,m}$ is equal to

$$T(K_{n,m}) = n\binom{m}{2} + m\binom{n}{2} = nm(m+n-2)/2\,.$$

Hence,

$$z(K_{n,m}) \le \frac{(n-1)(3m-n-1)}{3m(m+n-2}\,.$$

To prove the lower bound, consider the bound given by (3). If the set of cycles C is taken to contain only cycles of length four, then there are $R = \binom{n}{2}\binom{m}{2}$ of such cycles and each turn can cover no more than $r = m-1$ cycles. Hence,

$$z(K_{n,m}) \ge \frac{R}{rT(K_{n,m})} = \frac{n(n-1)m(m-1)}{4(m-1)(mn(m+n-2)/2)} = \frac{n-1}{2(m+n-2)}\,.$$

□

Corollary 2 *For bipartite graphs $K_{n,n}$, bounds for the fraction of prohibited turns is given by*

$$\frac{(n-1)(n+2)}{4n^2} \le z(K_{n,n}) \le \frac{2n-1}{6n}\,. \tag{11}$$

Proof: The upper bound follows directly from (10).

For the lower bound, note that for $K_{n,n}$ we have $T(K_{n,n}) = 2n\binom{n}{2} = n^2(n-1)$ and no more than $\binom{n}{2} + \binom{n-1}{2} = (n-1)^2$ turns can be selected in such a way that no two or more turns cover the same cycles of length four. These turns cover $(n-1)^3$ cycles. All other turns cover at most $(n-2)$ additional cycles. Thus, we have for the total number of prohibited turns

$$Z(K_{n,n}) \geq (n-1)^3 + \frac{\binom{n}{2}^2 - (n-1)^3}{n-2} = \frac{(n-1)^2(n+2)}{4}\,.$$

□

Corollary 3 *For bipartite graphs $K_{n,n}$, an alternate lower bound for the fraction of prohibited turns is given by*

$$z(K_{n,n}) \geq 1 - \frac{1}{2n} - \frac{1}{2}\sqrt{2 - \frac{2}{n} + \frac{1}{n^2}}\,. \tag{12}$$

Proof: Note that for $K_{n,n}$ we have $T(K_{n,n}) = n^2(n-1)$. Consider the set of $\binom{n}{2}^2$ cycles of length four. We split all prohibited turns in the minimum cycle-breaking set of turns into $n(n-1)$ groups, putting any two turns (a,b,c) and (x,y,z) in one group if and only if $a = x$, $c = z$. Denote the number of turns in these groups as s_j, $j = [1, 2, \ldots, n(n-1)]$. Then, the number of prohibited turns is $Z(K_n, n) = \sum_{j=1}^{n(n-1)} s_j$. Now, we consider the number c_j of cycles of length four, covered by turns from group j, $c_j = (n-1) + (n-2) + \ldots, (n - s_j)$. For the total number of cycles of length four we obtain,

$$\binom{n}{2}^2 \leq \sum_{j=1}^{n(n-1)} c_j = \sum_{j=1}^{n(n-1)} \frac{s_j(2n - s_j - 1)}{2} = \frac{2n-1}{2} Z(K_{n,n}) - \sum_{j=1}^{n(n-1)} \frac{s_j^2}{2}\,.$$

Since

$$\sum_{j=1}^{n(n-1)} \frac{s_j^2}{2} \geq \frac{n(n-1)}{2} \left(\frac{Z(K_{n,n})}{n(n-1)}\right)^2,$$

we have

$$(Z(K_{n,n}))^2 - n(n-1)(2n-1)Z(K_{n,n}) + \frac{1}{2}n^3(n-1)^3 \leq 0\,.$$

Solving this inequality for $Z(K_{n,n})$, we obtain the fraction of prohibited turns as

$$z(K_{n,n}) \geq 1 - \frac{1}{2n} - \frac{1}{2}\sqrt{2 - \frac{2}{n} + \frac{1}{n^2}}\,.$$

□

Hence, the $\lim_{n\to\infty} z(K_{n,n}) \geq 1 - \frac{\sqrt{2}}{2} \approx 0.2929$. Therefore, asymptotically, bound (12) gives better results than bound (11). The bounds for $Z(K_{n,n})$ are given in Table 2 for $n = 2, \ldots, 8$.

Table 2: Bounds for the number of prohibited turns $Z(K_{n,n})$

n	2	3	4	5	6	7	8
$Z(K_{n,n}) \geq$	1	5	14	28	50	81	123
$Z(K_{n,n}) \leq$	1	5	14	30	55	91	140

We conjecture that the CB algorithm generates a minimum cycle-breaking set for any complete bipartite graph $K_{n,m}$.

Next we consider complete t-partite graphs $K_{n_1,n_2,\ldots,n_t} = K_n^t$, where $n_i = n$, $i = 1, 2, \ldots, t$ [8], with $N = nt$ nodes and $M = n^2t(t-1)/2$ edges.

Theorem 6 *For complete t-partite graphs K_n^t,*

$$\frac{4n^3(t-2) + 3(n-1)^2(n+2)}{12n^2(nt-n-1)} \leq z(K_n^t) \leq \frac{2n^2t - 2n^2 - 3n + 1}{6(n^2t - n^2 - n)} . \tag{13}$$

Proof: To prove the upper bound, we estimate the number of prohibited turns $Z(K_n^t)$, generated by the CB algorithm. Number of prohibited turns can be calculated as

$$\begin{aligned} Z(K_n^t) &= Z(K_{n-1}^t + \binom{n(t-1)}{2} + \binom{n(t-1)-1}{2} + \ldots + \binom{(n-1)(t-1)}{2} \\ &= \sum_{j=2}^{n(t-1)} \binom{j}{2} + \sum_{j=1}^{n-1} \binom{j(t-1)}{2} . \end{aligned} \tag{14}$$

With

$$\sum_{j=2}^{n(t-1)} \binom{j}{2} = \frac{1}{6} n(t-1)(nt-n+1)(nt-n-1)$$

and

$$\sum_{j=1}^{n-1} \binom{j(t-1)}{2} = \frac{1}{12} n(t-1)(n-1)(2nt-2n-t-2) ,$$

the total number of turns $T(K_n^t) = nt \binom{n(t-1)}{2}$ equation (10) follows from (14).

To prove the lower bound, consider all cycles of length three, containing nodes from three different parts, and all cycles of length four, containing nodes from two different parts. There are $C_1(K_n^t) = n^3 \binom{t}{3}$ cycles of the first type and $C_2(K_n^t) = \binom{n}{2}^2 \binom{t}{2}$ cycles of the second type. To cover all cycles of the second type, by Corollary 1, $Z_2(K_n^t) \geq \frac{(n-1)^2(n+2)}{4} \binom{t}{2}$ turns are needed. Since no cycles of the first type have common turns with each other and with cycles of the second type, using Theorem 1, we obtain

$$Z(K_n^t) \geq C_1(K_n^t) + Z_2(K_n^t) \geq n^3 \binom{t}{3} + \frac{(n-1)^2(n+2)}{4} \binom{t}{2},$$

$$Z(K_n^t) \geq \frac{t(t-1)}{2}(\frac{n^3(t-2)}{3} + \frac{(n-1)^2(n+2)}{4}).$$

Note that $T(K_n^t) = nt \binom{n(t-1)}{2}$, thus

$$z(K_n^t) \geq \frac{1}{n(t-1)-1}(\frac{n(t-2)}{3} + \frac{(n-1)^2(n+2)}{4n^2}$$

$$z(K_n^t) \geq \frac{4n^3(t-2) + 3(n-1)^2(n+2)}{12n^2(nt-n-1)}.$$

□

For example, by Theorem 6, for $n = 2, 3$ and any $t \geq 3$, the upper and lower bounds coincide to yield $z(K_n^t) = 11/36$, and for these cases the CB algorithm is optimal.

Corollary 4 *If* $n \to \infty$, *then* $\frac{1}{3} - \frac{1}{12(t-1)} \leq \lim_{n\to\infty} z(K_n^t) \leq \frac{1}{3}$. *If also* $t \to \infty$, *then* $z(G) \to \frac{1}{3}$.

Thus lower bound (13) is asymptotically tight and the maximum difference between the upper and the lower bound is achieved when $n \to \infty$.

7 Application of the CB Algorithm to Graphs with Small Degree Nodes

Now we consider an arbitrary graph G, which is constrained to have only nodes of small degrees. This corresponds to the practical case where the number of possible point-to-point connections at each node is restricted by the number of output buffers in the router [10]. Assume that degrees of all nodes do not exceed 3. As shown below, in this case the upper bound on the fraction of prohibited turns given in Section 5 can be substantially improved.

Consider first the case when initially graph G includes a node of degree smaller than 3. Then, at each recursive call of the CB algorithm, each connected component will have a node of degree smaller than 3. If the component is not a simple cycle or a path without repeating nodes it includes also nodes of degree 3. Since recursive calls that select nodes of degree 1 do not prohibit any turns, we assume that there are no nodes of degree 1 in the initial graph. Note that if there are at least two nodes of degree 2 in a component, then at least one of them is not a delayed node and is a neighbor of a node of degree 3. Also, there is only one possibility that a node a of degree 3 will be selected, namely, if the component includes exactly one delayed node b of degree 2. It is easy to show by contradiction that in this case there exists a non-cut-node of degree 3 that is not a neighbor of the delayed node. Therefore, only a non-cut-node of degree 3 can be selected. Let G be such a connected graph with M edges and $N = N_2 + N_3$ nodes, where N_2 and N_3 are the number of nodes of degree 2 and 3, respectively. Furthermore, assume that CB algorithm is applied to this graph until the remaining subgraph becomes a collection of k simple, disjoint cycles of length $C_j, j = 1, \ldots, k$. Denote by A_i $(i = 1, 2, 3)$ the number of nodes of degree i selected during the execution of the algorithm up to the point when a collection of cycles is obtained. Then $A_2 = B_2 + P_2$, where B_2 and P_2 denote the number of non-cut-nodes and number of cut-nodes of degree 2, respectively. Obviously, exactly one turn in each of k simple cycles need to be prohibited. (Nodes selected in these cycles are not counted in A_1 or A_2.) Hence, the total number of prohibited turns is

$$Z(G) \leq B_2 + 3A_3 + k\,. \tag{15}$$

It is readily seen that C_j, A_1, A_2, and A_3 satisfy the following equations

$$\sum_{j=1}^{k} C_j + A_1 + A_2 + A_3 = N, \tag{16}$$

and

$$\sum_{j=1}^{k} C_j + A_1 + 2A_2 + 3A_3 = M. \tag{17}$$

Since each cut-node increases the number of components by one we have

$$P_2 = k - 1, \tag{18}$$

and since the minimum length of a cycle is 3

$$\sum_{j=1}^{k} C_j \geq 3k. \tag{19}$$

Inequality (19) can be strengthened. First, note that at every step of the algorithm there exists a component without a delayed or a forcing node. If the initial graph is not a cycle of length $N = 3$, it is easy to see that at least one non-cut-node of degree 2 will be selected in this component, in the course of the algorithm, before this component turns into a cycle. Thus,

$$B_2 \geq 1 \tag{20}$$

provided that $N > 3$ and there exists anode of degree 2 in the initial graph. Note that the same is true in the case when the initial graph has nodes of degree 3, provided that $N > 4$.

Consider now components with delayed nodes.

Lemma 3 *If a component has a delayed node and all other nodes are of degree 3, then either a cycle of length 4 appears or a non-cut-node of degree 2 is selected in the course of the algorithm.*

Proof: After a node of degree 3 is selected in this component, three more nodes of degree 2 will appear. Together with the delayed node, they may form a cycle of length four as shown in Figure 6a. Suppose now that this is not the case and consider the component to which the delayed node belongs after selection of node of degree 3. Note that henceforth only nodes of degree 2 will be selected in this component. Two cases can take place in the course of the algorithm as follows:

(i) The delayed node becomes a node in a simple cycle.

(ii) The delayed node becomes a forcing node.

Consider the case (i). Since in addition to the delayed node, at least two nodes must become simultaneously of degree 2 to form a cycle, it follows that when it occurs, a non-cut-node of degree 2 was selected as in Figure 6(a).

Case (ii) includes two sub cases. First sub case takes place if the delayed node becomes a cut-node of degree 2. Then the node a whose selection caused this result must be a non-cut-node, as shown in Figure 7(a). The second sub case occurs if the delayed node becomes a node of degree 1. Then the selected node a is a neighbor of the delayed node, Figure 7(b). Node a must be a non-cut-one, since otherwise the delayed node must have been turned into a forcing node (a cut-node) already, Figure 7(c). □

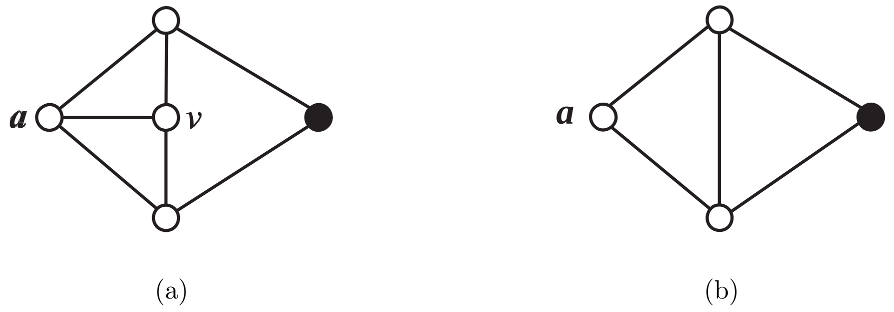

Figure 6: Graphs illustrating the cases where the only node of degree two is a delayed node, showing delayed nodes as solid

It follows from Lemma 3 that every selection of node of degree 3 either creates a cycle of length $C_j = 4$, or leads to a selection of non-cut-node of degree 2, increasing B_2 by 1. These considerations, together with (19) and (20) imply the following inequality

$$\sum_{j=1}^{k} C_j + B_2 - 1 \geq 3k + A_3. \tag{21}$$

Finally, since at most one node of degree 3 can be selected in each component, except the component without a delayed node, it follows that

$$A_3 \leq k - 1. \tag{22}$$

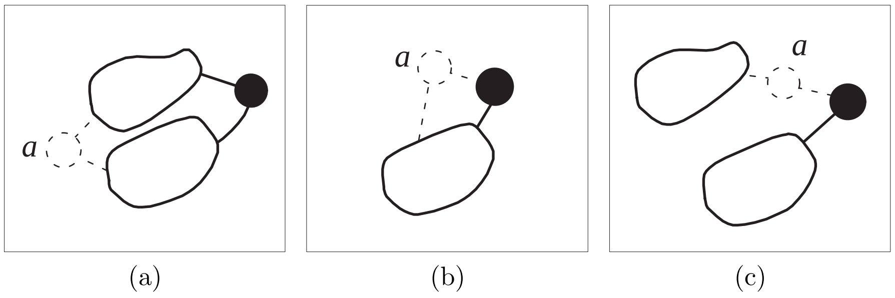

Figure 7: Graphs illustrating the runs that would make a node of degree 2 a forcing node in (a), a node of degree 1 forcing node in (b), and node that should have been a forcing node rather than a delayed node in (c)

Theorem 7 *Let G be a connected graph with M edges and $N > 3$ nodes, where all nodes have degrees not exceeding 3 and at least one node is of degree smaller than 3. Then*

$$Z(G) \leq \left\lfloor \frac{1}{6}(6M - 5N + 2) \right\rfloor, \tag{23}$$

and

$$\begin{aligned} z(G) &\leq \frac{\left\lfloor \frac{1}{6}(6M - 5N + 2) \right\rfloor}{(4M - 3N)} \\ &\leq \frac{1}{4} - \frac{N-4}{12(4M-3N)} \\ &\leq \frac{1}{4} - \frac{N-4}{12(4M-3N)}. \end{aligned} \tag{24}$$

Proof: The proof follows from the system of equations (16)– (18) and inequalities (15),(21), and (22). Subtracting (16) from (21) and substituting P_2 from (18), we get

$$2A_3 \leq N - 4k - A_1. \tag{25}$$

Multiplying (22) by 4 and adding (25) yields

$$6A_3 \leq N - A_1 - 4. \tag{26}$$

Subtracting (16) from (17), we obtain

$$A_2 + 2A_3 = M - N. \tag{27}$$

From (15) and (18):

$$Z(G) \leq A_2 + 3A_3 + 1. \tag{28}$$

Taking into account that $Z(G)$ is an integer and $A_1 \geq 0$, from (26), (27), and (28), we obtain:

$$Z(G) \quad \leq \lfloor \tfrac{1}{6}(6M - 5N + 2) \rfloor .$$

Since the total number of turns is $T(G) = 4M - 3N$, the fraction of prohibited turns is upper bounded by

$$z(G) \quad \leq \quad \frac{\lfloor \frac{1}{6}(6M - 5N + 2) \rfloor}{4M - 3N},$$

or

$$z(G) \leq \frac{1}{4} - \frac{N - 4}{12(4M - 3N)} .$$

□

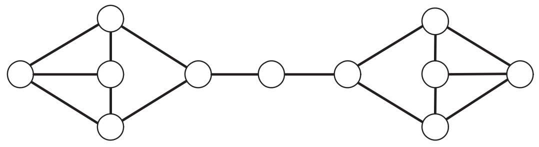

Figure 8: An example graph with $Z(G) \leq \lfloor \frac{43}{6} \rfloor = 7$ and $z(G) \leq \frac{7}{31}$

Corollary 5 *For any degree 3 regular graph G with $N > 4$,*

$$Z(G) \leq \left\lfloor \frac{4N + 7}{6} \right\rfloor, \tag{29}$$

$$\begin{aligned} z(G) \quad &\leq \quad \frac{\lfloor \frac{4N+7}{6} \rfloor}{3N}, \\ &\leq \quad \frac{2}{9} + \frac{7}{18N}. \end{aligned} \tag{30}$$

Proof: After selecting the first node (which is a non-cut-node of degree 3), prohibiting 3 turns, permitting 6 turns, deleting the node and its edges, we obtain a graph considered in Theorem 7, with $N' = N - 1$ nodes and $M' = M - 3 = \frac{3N}{2} - 3$ edges. Hence the number of prohibited turns $Z(G)$ becomes

$$\begin{aligned} Z(G) &\leq 3 + \left\lfloor \frac{1}{6}(6M' - 5N' + 2) \right\rfloor, \\ &\leq 3 + \left\lfloor \frac{1}{6}(6(M-3) - 5(N-1) + 2) \right\rfloor, \\ &\leq \left\lfloor \frac{4N+7}{6} \right\rfloor, \end{aligned}$$

where we used $M = 3N/2$. With the total number of turns given by $T = 3N$, (30) follows immediately. □

Bounds (24) and (30) are tight. The first is achieved, e.g., for the graph shown in Figure 8 with $z(G) \leq \frac{7}{31}$, and the second is achieved, for example for the Petersen graph, $K_{3,3}$, among many others. Bound (30) is also achieved when the number of repeated groups of six nodes tends to infinity as in the graph shown in Figure 9. In spite of the fact that both the upper and lower bounds are tight, there is a gap between the upper bounds (24), (30) and the lower bound $Z(G) \geq M - N + 1$. This variation is due to the effect of nodes of degree 3 that are selected in the course of the algorithm, as the following theorem shows.

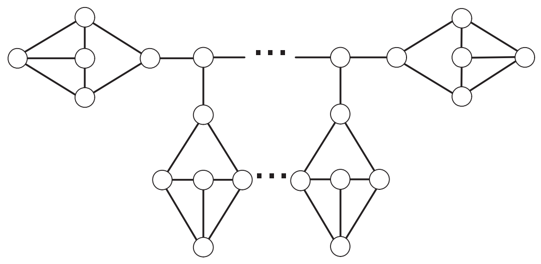

Figure 9: A topology in which the upper bound (30) is attained asymptotically when number of five-node clusters tends to infinity

Theorem 8 *For graphs described in Theorem 7, if no node of degree 3 is ever selected in the course of the algorithm, then*

$$z(G) = \frac{M - N + 1}{4M - 3N}. \tag{31}$$

Proof: If $A_3 = 0$, then from (16), (17), (28), and (2) we obtain

$$Z(G) = M - N + 1 \tag{32}$$

which gives (31). □

Thus in this case upper and the lower bounds coincide and the CB algorithm is optimal.

A similar result holds for a degree 3 regular graph if the number of connected components remains equal to one throughout the execution of the algorithm.

Corollary 6 *If graph G is a degree 3 regular graph and $k = 1$, then*

$$z(G) = \frac{1}{6} + \frac{2}{3N}. \tag{33}$$

Proof: Since $k = 1$, then $A_2 = B_2 + P_2 = B_2$, and $\sum_{j=1}^{k} C_j = C_1$. Total number of prohibited turns $Z(G)$ then becomes

$$\begin{aligned} Z(G) &\leq B_2 + 3A_3 + k \\ &\leq 4 + B_2 \\ &\leq 4 + A_2\,. \end{aligned} \tag{34}$$

Number of nodes and number of edges are equal to $N = C_1 + A_1 + A_2 + 1$ and $M = C_1 + A_1 + 2A_2 + 3$ we obtain

$$M - N = A_2 + 2. \tag{35}$$

It follows from (34),(35) and Theorem 2 that

$$Z(G) = M - N + 2. \tag{36}$$

Since total number of turns in a degree 3 regular graph G is $T(G) = 3N$, we obtain

$$z(G) = \frac{1}{6} + \frac{2}{3N}\,.$$

□

8 Experimental Results

To illustrate the effectiveness of the CB algorithm we compared it with the popular Up/Down approach [16] by means of simulation experiments. To compare the two algorithms, we generated a family of graphs with a given bisection width B. The bisection width is the minimum number of edges that when deleted partition the graph into two separate connected components with equal number of nodes in each component. In Figure 10 (a), (b), (c), and (d) we show the results of our simulation experiments for topologies of bisection widths two, four, six, and eight respectively. These histograms of the fraction of prohibited turns were obtained using the Up/Down and the CB algorithms. For each distribution in Figure 10, we evaluated 100 randomly generated graphs with the given bisection width. All graphs had 64 nodes with each graph having an average node degree of four. We then applied both algorithms to the same set of topologies,

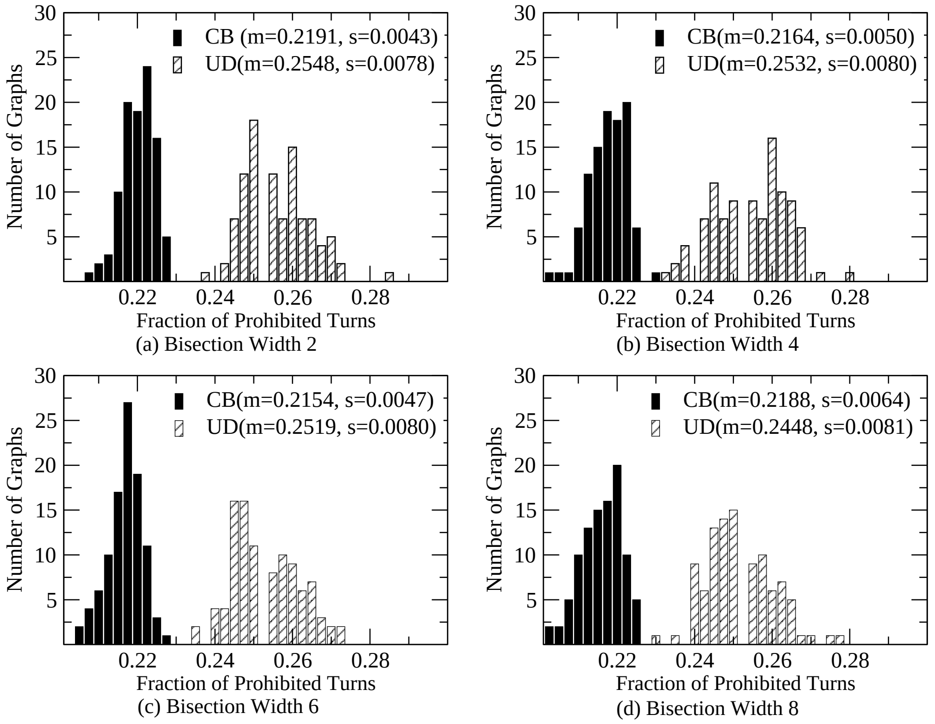

Figure 10: Experimental comparison of fractions of prohibited turns of Up/Down and CB-algorithms. In the figures, m, and s stand for the mean and the standard deviation of the distributions respectively.

determining the mean and the variance for the fraction of prohibited turns. In Figure 11, we plotted the mean of the fraction of prohibited turns versus the bisection width B. It can be seen that the Up/Down approach has a larger variance than the CB algorithm. The mean fraction of prohibited turns in the Up/Down approach are consistently larger by about 15% than those generated by the CB algorithm.

In next set of experiments we analyzed the average distance as a function of bisection width in a large number of topologies. Given the randomly generated family of connected topologies as discussed above, we first computed the average distance in each topology without any turn prohibition. We then applied turn prohibitions using the Up/Down algorithm and computed the average distance to determine the effect of prohibitions on the average distance. Subsequently we applied the CB algorithm to prohibit turns to the same original topology and computed the average distance after the CB algorithm is applied. We repeated these set of experiments for the same 100 randomly generated topologies as used in previous sets of experiments, and computed the mean of average distances. We repeated the set of experiments, varying the bisection width between 2

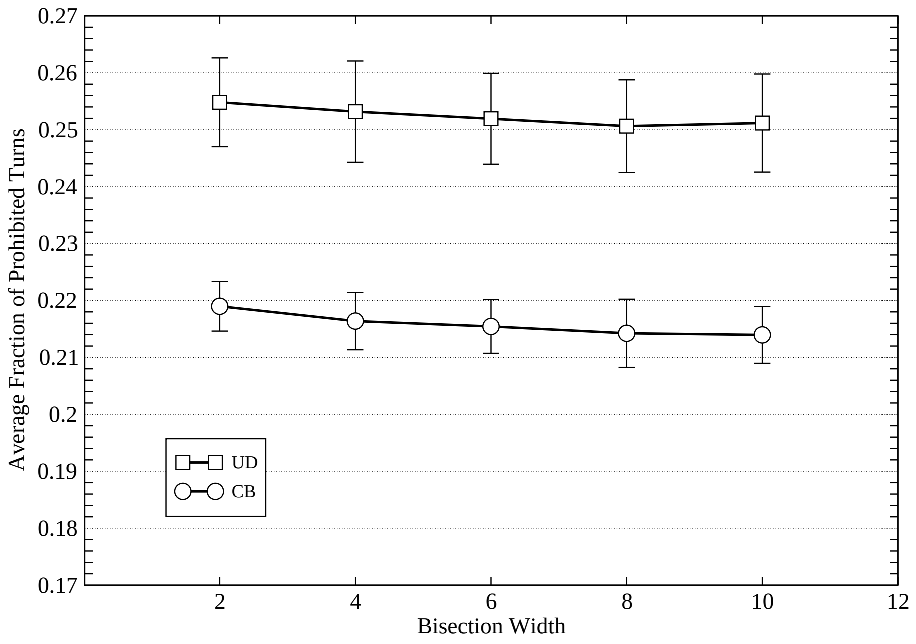

Figure 11: Average fraction of prohibited turns vs bisection width B. All topologies had 64 nodes and an average node degree of four.

and 30. Experimental results are shown in Figure 12. Surprisingly, as can be seen in Figure 12(a), for the 64 node topologies that we investigated, the Up/Down algorithm has smaller average distance values than the CB algorithm when bisection width values are between 2 and 10, and the CB algorithm has smaller average distance values than CB algorithm when bisection width values between 16 and 30. When the bisection width is in the range of 12 to 14 both algorithms perform approximately the same. As expected, in all cases, both CB and the Up/Down algorithms increase the average distance of the original topology. Defining the average distance dilation as the ratio of the average distance with a given prohibition scheme to the average distance with no prohibition, we obtain the plot in Figure 12(b). For the topologies that we investigated,the CB algorithm introduces approximately the same dilation in topologies with a bisection width of 2 as does the Up/Down algorithm at a bisection width of 30. For bisection widths 12 and 14 both algorithms cause approximately same dilation. For topologies with bisection widths greater than 14, the dilation introduced by the CB algorithm is always less than the dilation introduced by the Up/Down algorithm.

Finally, in Figure 13 we show our simulation results comparing saturation points of the CB and the Up/Down algorithms. In these experiments, we used uniform traffic model where every node is equally likely to be selected as a source and as a destination. Messages generated by the source node are of fixed

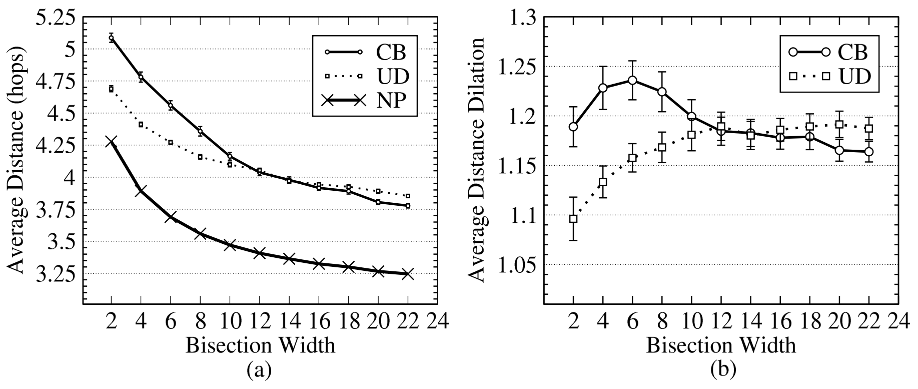

Figure 12: Comparing average distance in hops (a) and average dilation (b) in Up/Down (UD) and CB algorithms against the reference No Prohibition (NP).

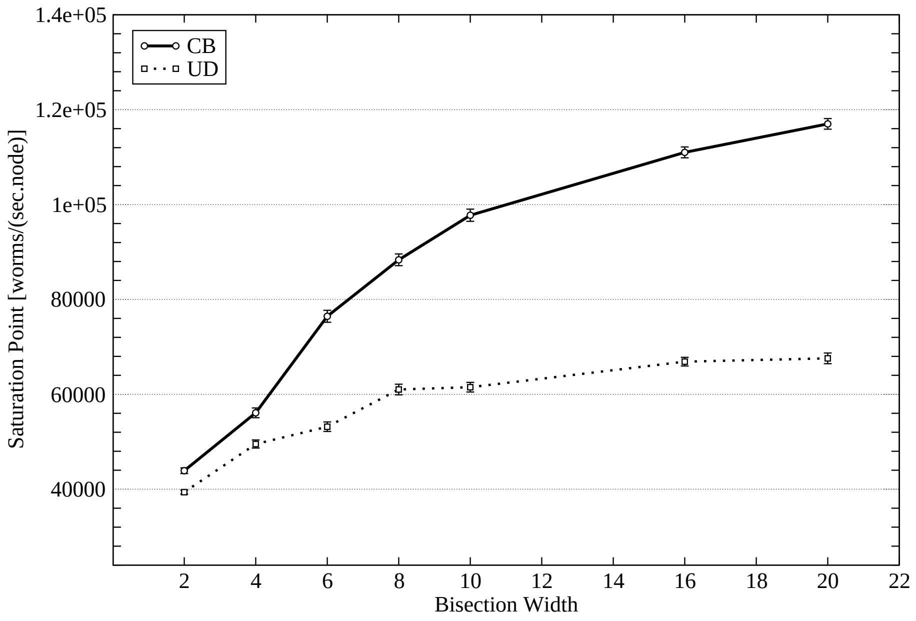

Figure 13: Experimental results comparing saturation points of the Up/Down and the CB algorithms. Saturation point is in terms of number of messages (also referred to as worms) per second per node.

length of 200 flits. In our experiments each flit contains eight bits of payload data and three bits of control information. Control bits are used to indicate if a flit is a header flit, a payload flit, or a tail flit. During the simulations, we observed message latencies as the message generation rate is increased. As the observed latencies tend to infinity, the corresponding message generation rate is noted as the saturation point (maximum sustainable message generation rate) for that topology. In the figure, each experimental point represents the average saturation point for 100 different topologies for both the CB and the Up/Down approaches. Even with the minimum bisection width constraints in the topologies as shown, we see that for all topologies with the minimum bisection widths (between two and twenty) that we investigated, the CB algorithm outperformed the Up/Down algorithm by as much as 73%. We note that for 15% reduction in the fraction of prohibited turns, Figure 2 predicts (with no bisection width constraints in the topologies) an increase in the saturation point of greater than 116%.

9 Conclusions

In this paper we analyzed the problem of constructing minimal cycle-breaking sets of turns in a given graph. This problem is important in networks with irregular topologies in which wormhole routing is used. We developed an algorithm, which we called the CB algorithm, with $O(N^2\Delta)$ complexity. This algorithm generates irreducible sets of prohibited turns, the size of which is no more than one third of the total number of turns for any graph. Furthermore, this set breaks all cycles and maintains connectivity in the original graph. The results of computer simulations illustrate that the proposed approach performs consistently and considerably better than the existing Up/Down approach. With randomly generated 64-node topologies with nodes of average degree four, the CB algorithm based approach outperformed the Up/Down approach by approximately 15% for the fraction of prohibited turns. For message latencies, we observed the improvement in saturation points by the CB algorithm over the Up/Down algorithm to be as large as 73%.

Acknowledgments

Authors wish to thank the reviewers for their many useful remarks and suggested improvements in the paper.

References

[1] V. S. Adve and M. K. Vernon. Performance analysis of mesh interconnection networks with deterministic routing. *IEEE Trans. Parallel Distrib. Syst.*, 5(3):225–246, 1994.

[2] B. Bollobas. *Modern Graph Theory.* Springer, 2002.

[3] N. Christofides. *Graph Theory An Algorithmic Approach.* Academic Press, Inc., 1975.

[4] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction To Algorithms.* The MIT Press, 1989.

[5] W. Dally and H. Aoki. Deadlock-free adaptive routing in multiprocessor networks using virtual channels. *IEEE Trans. on Parallel and Distributed Systems*, pages 466–475, 1997.

[6] W. Dally and C. Seitz. Deadlock-free message routing in multiprocessor interconnection networks. *IEEE Trans. on Comput.*, 36:547–553, 1987.

[7] F. De Pellegrini, D. Starobinski, M. G. Karpovsky, and L. B. Levitin. Scalable, distributed cycle-breaking algorithms for Gigabit Ethernet backbones. *Journal of Optical Networks*, 5(1):1–23, 2006.

[8] R. Diestel. *Graph Theory.* Springer, 2000.

[9] J. Duato. A new theory of deadlock-free adaptive routing in wormhole networks. *IEEE Trans. on Parallel and Distributed Systems*, 4:1320–1331, 1993.

[10] J. Duato, S. Yalamancili, and L. Ni. *Interconnection Newtworks and Engineering Approach.* IEEE Computer Society Press, 1997.

[11] E. Fleury and P. Fraigniaud. A general theory for deadlock avoidance in wormhole-routed networks. *IEEE Trans. on Parallel and Distributed Systems*, 9:626–638, 1998.

[12] C. Glass and L. Ni. The turn model for adaptive routing. *Proc. of the 19th Annual Int.. Symp. on Computer Architecture*, pages 278–286, 1992.

[13] C. Glass and L. Ni. The turn model for adaptive routing. *Journal of ACM*, 5:874–902, 1994.

[14] F. Harary. *Graph Theory.* The Advanced Book Program. Perseus Books, 1998.

[15] P. Kermani and L. Kleinrock. Virtual cut-through: A new computer communication switching technique. *Computer Network*, 3:267–286, 1979.

[16] R. Libeskind-Hadas, D. Mazzoni, and R. Rajagopalan. Tree-based multicasting in wormhole-routed irregular topologies. In *Proceedings of the International Parallel Processing Symposium*, pages 244–249, Mar. 30 - Apr. 3, 1998.

[17] O. Lysne, T. Skeie, S.-A. Reinemo, and I. Theiss. Layered routing in irregular networks. *IEEE Transactions on Parallel and Distributed Systems*, 2(1):51–65, January 2006.

[18] M. Mustafa, M. Karpovsky, and L. Levitin. Cycle breaking in wormhole routed computer communication networks. In *Opnetwork 2005*, http://www.opnet.com/opnetwork2005/proceedings, 304.pdf, August 2005. Opnet Technologies.

[19] L. M. Ni and P. K. McKinley. A survey of wormhole routing techniques in directed networks. *Computer*, 26:62–76, 1993.

[20] M. D. Schroeder, A. D. Birrell, M. Burrows, H. Murray, R. M. Needham, T. L. Rodeheffer, E. H. Satterthwaite, and C. P. Thacker. Autonet: A high-speed self configuring local area network using point-to-point links. Technical Report SRC Research Report 59, Digital Equipment Corporation, SRC, April 1990.

[21] L. Schwiebert. Deadlock-free oblivious wormhole routing with cyclic dependencies. *IEEE Trans. on Computers*, 50(9):865–876, 2001.

[22] T. Skeie, O. Lysne, and I. Theiss. Layered shortest path (LASH) Routing in irregular system area networks. In *Proceedings International Parallel and Distributed Processing Symposium (IPDPS)*, April 2002.

[23] D. Starobinski, M. Karpovsky, and L. Zakrevski. Application of network calculus to general topologies using turn prohibition. *IEEE/ACM Transactions on Networking*, 11(3):411–421, 2003.

[24] L. Zakrevski, S. Jaiswal, L. Levitin, and M. Karpovsky. A new method for deadlock elimination in computer networks with irregular topologies. *Pro. of the IASTED Conf. PDCS-99*, 1:396–402, 1999.

[25] L. Zakrevski and M. Karpovsky. Fault-tolerant message routing for multiprocessors. In J. Rolim, editor, *Lecture Notes in Computer Science - Parallel and Distributed Processing*, pages 714–731. Springer, 1998.

[26] L. Zakrevski and M. G. Karpovsky. Fault-tolerant message routing in computer networks. *Proc. of Int. Conf. on PDPA-99*, pages 2279–2287, 1999.

[27] L. Zakrevski, M. Mustafa, and M. Karpovsky. Turn prohibition based routing in irregular computer networks. *Proc. of the IASTED International Conference on Parallel and Distributed Computing and Systems*, pages 175–179, 2000.