Technical Memorandum Number 748

An efficient Algorithm for the Complementary Hamiltonian Cycle Problem in Bipartite Graphs

by

George Vairaktarakis
Daniel Solow

August 2001

Department of Operations
Weatherhead School of Management
Case Western Reserve University
10900 Euclid Avenue
Cleveland, Ohio 44106-7235

An Efficient Algorithm for the Complementary Hamiltonian Cycle Problem in Bipartite Graphs

George Vairaktarakis Daniel Solow *

Abstract

In this paper, the problem of finding a complementary Hamiltonian cycle (CHC) in a bipartite graph B is considered. Given a perfect matching M of B, the CHC problem is to find another perfect matching I of B so that $I \cup M$ forms a Hamiltonian cycle for B. The matching I *complements* M in forming a Hamiltonian cycle. This graph theoretic problem has numerous workforce planning applications. It is shown that the CHC problem on general bipartite graphs is equivalent to the problem of finding a Hamiltonian cycle on a general graph and is therefore NP-complete. Even the CHC problem on the special bipartite graphs that arise in our motivating applications is shown to be strongly NP-complete. Basic properties of these special bipartite graphs are presented and used to obtain structural properties of complementary Hamiltonian cycles. These properties employ two different types of cuts—connectivity and cycle cuts—which decompose the bipartite graph into smaller subgraphs. For each such subgraph, special-purpose algorithms are developed for solving the CHC problem by exploiting the structure of the subgraph. The properties presented in this article form the basis of a very efficient algorithm that solves CHC efficiently. Detailed computational experiments are presented.

*Weatherhead School of Management, Dept. of Operations, Case Western Reserve University, 10900 Euclid Avenue, Cleveland, OH 44106-7235

1 Introduction

Applications of the complementary Hamiltonian cycle problem are found in labor-intense assembly lines where workforce planning is essential to reduce labor-related costs. Such costs are captured by the size of the workforce or by hiring/firing costs. The workforce size is a measure of the total labor cost when the manager does not use firing or hiring to dynamically adjust the available labor force. In case hiring and/or firing is allowed, the relevant costs are captured by a minisum or minimax cost function of the difference between the workforce sizes of any two consecutive production periods. Before providing details of the motivating applications, a brief description of the CHC problem is given. To this end, the following definitions are introduced.

Definition 1 *A* **matching** M *in a graph* G *is a set of edges, no two of which share a common endpoint. The matching* M *is* **perfect** *if and only if each node of* G *belongs to an edge in* M*.*

Definition 2 *Let* $B(U, V)$ *be an arbitrary bipartite graph on the node sets* U *and* V*. A cycle* C *that includes all nodes in* $U \cup V$ *is referred to as a* **Hamiltonian cycle***.*

Let $B(U, V)$ be a bipartite graph with edge set $E(B)$ and M be a set of edges, not necessarily in $E(B)$, that constitutes a perfect matching of $B(U, V)$. The objective of the problem—hereafter called the *Complementary Hamiltonian Cycle* (CHC) problem—is to find a matching I of B so that $C = I \cup M$ is a Hamiltonian cycle of $B(U, V)$. Any such matching I of $B(U, V)$ must be such that $I \cap M = \emptyset$. A necessary condition for a Hamiltonian cycle C to exist in the bipartite graph $B(U, V)$ is that $|U| = |V| = n$. Then C is a cycle whose vertices alternate between nodes of $U = \{u_1, \ldots, u_n\}$ and $V = \{v_1, \ldots, v_n\}$ and the number of edges in C is $2n$.

Applications of complementary Hamiltonian cycles from synchronous assembly lines are now described. A *synchronous* or *paced* assembly line consists of m stations arranged in series. A set J of jobs is to be processed so that each job visits every station in the same order, say, 1 to m. Every station has the same production cycle of c units of time, referred to as a *period*. Hence, in a paced assembly line, jobs move one station downstream at the end of every period. At that time, a new job enters the first station and a completed job exits the last station. Given that the labor requirements of each job in each station are different, the production manager must assign the necessary number of workers to each workstation in each period so that each operation is completed in that period. To that end, for each $1 \leq i \leq n$, let $(W_{i1}, W_{i2}, \ldots, W_{im})$ be the work requirements associated with job J_i, where W_{ij} is the number of workers necessary to complete job J_i at station j in exactly one period, for $1 \leq j \leq m$. The problem is to determine a sequence

for processing the jobs so that the resulting workforce schedule satisfies some overall objective.

The problem of minimizing the maximum workforce size over all periods in the paced assembly line just described has been considered by Lee and Vairaktarakis, 1997. Objectives other than minimizing the maximum workforce size over all production periods have been considered by Vairaktarakis and Cai, 2001 who define a number of objective functions of the differences between the workforce requirements of different production periods. These objectives attempt to find level worker schedules that smooth the workforce fluctuations from one period to the next. Such schedules are particularly useful in automobile assembly because they help preserve overall smoothing of operations. One such objective, called *range*, is to minimize the difference between the maximum and minimum number of workers required by any worker schedule. In the next section we show how one can use the complementary Hamiltonian cycle problem to solve for the range objective.

The outline of the rest of the paper is as follows. In Section 2, the problem is formally defined and applications of the problem are described. In Section 3 the problem complexity is discussed. Structural properties of complementary Hamiltonian cycles are presented in Section 4. Using these properties a solution algorithm is presented in Section 5. This algorithm is tested in Section 6. Concluding remarks are made in Section 7.

2 Problem Definition and Motivation

In this section, the problem considered in this paper is formulated together with the notation, definitions, and terminology used throughout the article.

First, an integer programming formulation of the CHC problem is given. Observe that finding a complementary Hamiltonian cycle C is equivalent to finding an ordering of the edges in the given matching M so that if, say, $M = \{e_1, e_2, \ldots, e_n\}$ and the nodes of U and V are renumbered so that $e_i = (u_i, v_i)$, for $1 \leq i \leq n$, then the order $e_1, e_2, \ldots, e_n$ induces the matching $I = \{(v_1, u_2), (v_2, u_3), \ldots, (v_{n-1}, u_n), (v_n, u_1)\}$ of $B(U, V)$, for which $I \cup M = C$. In other words, CHC can be cast as the problem of finding a sequence σ of the edges of M so that the nodes of any two consecutive edges, including edges n and 1, in σ are linked in $B(U, V)$ by an edge. Equivalently, let

$$c_{ik} := \begin{cases} 1 & \text{if } (v_i, u_k) \in E(B) \\ 0 & \text{otherwise} \end{cases} \quad \text{and} \quad x_{ij} := \begin{cases} 1 & \text{if edge } i \text{ of } M \text{ is in position } j \text{ of } \sigma \\ 0 & \text{otherwise} \end{cases}$$

$$(\mathbf{CHC}) \qquad Max \quad \sum_{j=1}^{n} \sum_{i=1}^{n} \sum_{k=1}^{n} c_{ik} x_{ij} x_{k,(j+1) \ mod \ n} \tag{1}$$

$$s.t. \qquad \sum_{j=1}^{n} x_{ij} = 1 \qquad 1 \le i \le n \tag{2}$$

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad 1 \le j \le n \tag{3}$$

$$x_{ij} \in \{0, 1\} \tag{4}$$

where *mod* is the modulus operation for integers and, by definition, $n \ mod \ n = n$. By construction, $c_{ik} = 1$ if and only if the edges (u_i, v_i) and (u_k, v_k) are linked by the edge $(v_i, u_k) \in E(B)$. Equations (2) and (3) correspond to the assignment constraints that provide a sequencing of the edges in M. Consider an optimal ordering σ of the edges in M. Then, for every $1 \le j \le n$, the term $\sum_{i=1}^{n} \sum_{k=1}^{n} c_{ik} x_{ij} x_{k,(j+1) \ mod \ n}$ equals 1 if and only if the edges e_i and e_k assigned to positions j and $j+1$, respectively, are linked with the correct edge in $B(U, V)$. The cyclic nature of σ is captured by the modulus operation when $j = n$ because $x_{ij} x_{k,(j+1) \ mod \ n} = x_{in} x_{k1}$. As a result, a complementary Hamiltonian cycle exists if and only if the optimal objective function value of CHC is n.

In addition to the general CHC problem, special cases are considered in which the incidence matrix $[c_{ik}]$, and hence the bipartite graph B itself, has a special form arising from the job-sequencing applications described in Section 1. From here on, it is assumed that the nodes of $B(U, V)$ are numbered and fixed with $U = \{u_1, \ldots, u_n\}$ and $V = \{v_1, \ldots, v_n\}$. When i and j are integers with $1 \le i < j \le n$, the node u_i (v_i) is said to be to the *left* of u_j (v_j). Equivalently, u_j (v_j) is said to be to the *right* of u_i (v_i).

Definition 3 *For every node $u \in U$, let $N(u)$ denote the subset of V defined by $N(u) = \{v \in V : (u, v) \in E(B)\}$. Similarly, for every node $v \in V$, let $N(v) = \{u \in U : (u, v) \in E(B)\}$. For any $w \in U \cup V$, $N(w)$ is referred to as the* **neighborhood** *of w.*

Definition 4 *Given integers l and r with $1 \le l \le r \le n$, define the* **node intervals** *$[u_l, u_r] = \{u_l, u_{l+1}, \ldots, u_r\}$ and $[v_l, v_r] = \{v_l, v_{l+1}, \ldots, v_r\}$.*

Definition 5 *Graph $B(U, V)$ is called a* **beam bipartite graph** *if and only if there exists a numbering of the vertices such that for every $1 \le i \le n$, the neighborhood of v_i is a node interval, that is, there exist integers l_i and r_i with $1 \le l_i \le r_i \le n$ such that $N(v_i) = [u_{l_i}, u_{r_i}]$.*

A beam bipartite graph is shown in Figure 1(a). A special case of a beam bipartite graph is one in which the node intervals comprising the neighborhoods of the vertices in V overlap in U in such a way that if $1 \le i < j \le n$, then the leftmost neighbor of v_i is to the left of the leftmost neighbor of v_j and the rightmost neighbor of v_i is to the left of the rightmost neighbor of v_j, as

shown for example in Figure 1(b). A formal definition follows.

INSERT FIGURE 1 HERE

Definition 6 *Let $B(U,V)$ be a beam bipartite graph. Graph B is called a* **neighboring-beam bipartite graph** *if and only if there exists an ordering of the vertices such that, whenever i and j are integers with $1 \leq i < j \leq n$, the neighborhoods $N(v_i) = [u_{l_i}, u_{r_i}]$ and $N(v_j) = [u_{l_j}, u_{r_j}]$ are such that $l_i \leq l_j$ and $r_i \leq r_j$.*

The following theorem shows that Definition 6 holds when the neighborhoods $N(u_i)$ and $N(u_j)$ are used rather than $N(v_i)$ and $N(v_j)$, that is, that neighboring-beam bipartite graphs are well defined irrespective of which set of vertices is labeled U and which is labeled V.

Proposition 1 *Let $B(U,V)$ be a neighboring-beam bipartite graph on the ordered node sets $U = \{u_1, u_2, \ldots, u_n\}$ and $V = \{v_1, v_2, \ldots, v_n\}$. Then for any integers i and j with $1 \leq i < j \leq n$, there are integers l_i, r_i, l_j, and r_j with $1 \leq l_i \leq r_i \leq n$ and $1 \leq l_j \leq r_j \leq n$ such that*

(i) $N(u_i) = [v_{l_i}, v_{r_i}]$ and $N(u_j) = [v_{l_j}, v_{r_j}]$ and

(ii) $l_i \leq l_j$ and $r_i \leq r_j$.

Proof: To prove (i), let l_i and r_i be such that v_{l_i} and v_{r_i} are the leftmost and rightmost nodes of $N(u_i) \cap V$, respectively. It is shown that the nodes $v_{l_i+1}, v_{l_i+2}, \ldots, v_{r_i-1}$ are also in $N(u_i)$, or equivalently, that the edges $(v_k, u_i) \in E(B)$ for every $k \in \{l_i + 1, \ldots, r_i - 1\}$ [see Figure 2(a)]. Indeed, $u_i \in N(v_{l_i})$ and $u_i \in N(v_{r_i})$ and, by choice of v_{l_i} and v_{r_i}, $l_i \leq r_i$. Consider an arbitrary integer k with $l_i < k < r_i$. Because B is a neighboring-beam bipartite graph, $N(v_k) = [u_{l_k}, u_{r_k}]$ for some integers $l_k \leq r_k$. It is enough to show that $l_k \leq i \leq r_k$. Indeed, since u_{r_k} is the rightmost u-node adjacent to v_k and $l_i < k$, the beam property for $N(v_{l_i})$ and $N(v_k)$ implies that $i \leq r_k$. Similarly, $k < r_i$ and hence $l_k \leq i$. Combining the two inequalities yields that $u_i \in N(v_k)$. Similarly, $N(u_j) = [v_{l_j}, v_{r_j}]$. This completes the proof of (i).

INSERT FIGURE 2 HERE

To prove (ii), it is first shown that $l_i \leq l_j$. By contradiction, suppose that there is an integer l' with $v_{l'} \in N(u_j)$ such that $l' < l_i$ [see Figure 2(b)]. Then, because $u_i \in N(v_{l_i})$ and $i < j$, the neighboring-beam property for $v_{l'}$ and v_{l_i} yields that $u_i \in N(v_{l'})$. This contradicts the assumption that v_{l_i} is the leftmost node of V adjacent to u_i in B and so it must be that $l_i \leq l_j$. A symmetric argument shows that $r_i \leq r_j$, thus completing the proof. □

In view of the foregoing definitions, the following three versions of the CHC problem are considered in this paper:

CHC_A: the CHC problem where B is an arbitrary bipartite graph,

CHC_B: the CHC problem where B is a beam bipartite graph, and

CHC_N: : the CHC problem where B is a neighboring-beam bipartite graph.

These versions of the CHC problem can be cast in terms of properties of the incidence matrix $[c_{ik}]$, as follows. When $B(U,V)$ is an arbitrary bipartite graph, $\mathcal{C} = [c_{ik}]$ is an arbitrary incidence matrix. The case where $B(U,V)$ is a beam bipartite graph corresponds to a CHC in which $\mathcal{C} = [c_{ik}]$ has the *consecutive ones property* for the rows, that is, the 1's in every row i of $\mathcal{C}$, corresponding to those vertices in U that are adjacent to v_i, are consecutive. In other words, the consecutive 1's property is equivalent to the requirement that $N(v_i) = [u_{l_i}, u_{r_i}]$. Also note that, in general, the incidence matrix of a beam bipartite graph does not possess the consecutive 1's property for the columns. That is, the 1's in column i of $\mathcal{C}$, corresponding to those vertices in V that are adjacent to u_i, are not necessarily consecutive. This reflects the fact that in Definition 5, the neighborhood $N(u_i)$ is not necessarily of the form $[v_{l_i}, v_{r_i}]$. Finally, the case where $B(U,V)$ is a neighboring-beam bipartite graph corresponds to a CHC in which the incidence matrix $\mathcal{C}$ possesses the consecutive 1's property for both the rows and columns. This is a result of Definition 6 and Proposition 1.

Among the above 3 versions of the CHC problem, the main focus of this article is on CHC_N. For the other 2 versions we only provide results that follow easily from our main problem, or results that help our presentation. The motivation to solve CHC_N comes from the problem of minimizing the workforce size in a paced assembly line. The special case of $m = 2$ stations can be formulated as a complementary Hamiltonian cycle problem on a neighboring-beam bipartite graph, as follows. Let $B(U,V)$ be the bipartite graph where $U = \{W_{i1}\}_{i=1}^{n}$ and $V = \{W_{i2}\}_{i=1}^{n}$. The given matching M is defined by $M = \{(W_{i1}, W_{i2})\}_{i=1}^{n}$ and corresponds to the n jobs. Every permutation σ of the edges of M gives rise to a matching $I = \{(W_{[1],2}, W_{[2],1}), (W_{[2],2}, W_{[3],1}), \ldots, (W_{[n],2}, W_{[1],1})\}$ such that $C = I \cup M$ is a Hamiltonian cycle that complements M, where $(W_{[k],1}, W_{[k],2})$ denotes the k-th edge of σ. Let $e = (W_l, W_{l'})$ be an edge in I. Then, $W_l + W_{l'}$ denotes the number of workers required in the two assembly stations during a particular period. The edge set $E(B)$ depends on the workforce objective of interest. If the objective is to minimize the maximum workforce size over the production

horizon, then B is a complete bipartite graph, that is, $E(B)$ consists of the edges from every $v \in V$ to every $u \in U$. On the other hand, consider the range objective described earlier. One way to solve this problem is by finding a cycle $C = I_R \cup M$ such that the difference between the maximum and minimum workforce size associated with edges of I_R lies within prespecified upper and lower bounds, say, $R_L \leq W_l + W_{l'} \leq R_U$, for every $(W_l, W_{l'}) \in I_R$. Then, bisection search on the possible values for R_U and R_L yield the optimal value of the range. Therefore, for given values R_U and R_L, $E(B)$ consists of all the edges (W_{i1}, W_{j2}) such that $R_L \leq W_{i1} + W_{j2} \leq R_U$. For this reason, in the rest of this article, it is assumed that $E(B)$ is given. Moreover, as we will show, in the foregoing applications B possesses the neighboring-beam property.

Indeed, let $w_{i_1} \leq w_{i_2} \leq \ldots \leq w_{i_n}$ be an ordering of the elements in $\{W_{i1}\}_{i=1}^{n}$, and $w'_{j_1} \geq w'_{j_2} \geq \ldots \geq w'_{j_n}$ be an ordering of the elements in $\{W_{i2}\}_{i=1}^{n}$. Then, order the nodes in U according to their w_{i_k} values and the nodes in V according to their w'_{i_k} values. Then, for any bounds R_L and R_U, it is easy to see that the edge set of $E(B)$ consists of edges (w_{i_k}, w'_{j_k}) for which $R_L \leq w_{i_k} + w'_{j_k} \leq R_U$ satisfies the neighboring-beam property of Definition 6.

Example 1: As an example, consider the 13-job instance with $J_1 = (2, 23)$, $J_2 = (6, 26)$, $J_3 = (6, 26)$, $J_4 = (6, 16)$, $J_5 = (8, 16)$, $J_6 = (11, 7)$, $J_7 = (13, 20)$, $J_8 = (15, 16)$, $J_9 = (18, 20)$, $J_{10} = (20, 13)$, $J_{11} = (20, 2)$, $J_{12} = (26, 6)$, and $J_{13} = (27, 8)$. Figure 3 depicts the bipartite graph B in which the nodes in U correspond to the nondecreasing order of W_{i1} values, the nodes in V correspond to the nonincreasing order of W_{i2} values, the job matching M consists of the solid edges, and $E(B)$ includes all edges (W_{i1}, W_{j2}) such that $R_L = 25 \leq W_{i1} + W_{j2} \leq 33 = R_U$.

INSERT FIGURE 3 HERE

By ordering the nodes in U (V) according to the nondecreasing (nonincreasing) order of the W_{i1} (W_{i2}) values, the resulting graph B possesses the neighboring-beam property (see Figure 3). Hence, there exists a worker schedule for the above instance requiring from 25 to 33 workers per period if and only if there exists a complementary Hamiltonian cycle for the graph B and the given matching M. The complexity of the problem of finding a complementary Hamiltonian cycle in bipartite graphs is now addressed.

3 Complexity Results

In Theorem 1, a simple proof for the NP-completeness of CHC_A is given that provides insight as to why the complementary Hamiltonian cycle problem in bipartite graphs is equivalent to the Hamiltonian cycle problem in general graphs.

Theorem 1 *The problem CHC_A is strongly NP-complete.*

Proof: The reduction is from the Hamiltonian-cycle problem on an arbitrary graph G with node set $V(G) = \{v_1, v_2, \ldots, v_n\}$ and edge set $E(G)$. A bipartite graph $B(X, Y)$ is constructed from G, as follows. Let $X = Y = V(G)$, that is, each bipartition is a copy of the set $V(G)$. Let $X = \{x_1, x_2, \ldots, x_n\}$, and $Y = \{y_1, y_2, \ldots, y_n\}$. Hence, x_i (y_i) is the node of X (Y) corresponding to $v_i \in V$. Let the edge set $E(B)$ contain all edges (x_i, y_j) such that $(v_i, v_j) \in E(G)$. Clearly, $B(X, Y)$ is constructed from G in polynomial time. Consider the instance of CHC_A defined by this bipartite graph $B(X, Y)$ and the matching $M = \{(x_i, y_i)\}_{i=1}^{n}$. It is shown that there exists a Hamiltonian cycle C for G if and only if there exists a Hamiltonian cycle for $B(X, Y)$ complementing the matching M.

Indeed, let $C = v_{[1]}, v_{[2]}, \ldots, v_{[n]}, v_{[1]}$ denote a permutation of the vertices in V, that is, a Hamiltonian cycle for G. Then, it is obvious that the cycle

$$C' = y_{[1]}, x_{[2]}, y_{[2]}, x_{[3]}, y_{[3]}, \ldots, y_{[n-1]}, x_{[n]}, y_{[n]}, x_{[1]}, y_{[1]}$$

traverses all nodes in $X \cup Y$ as well as all edges in $M = \{(x_{[i]}, y_{[i]})\}_{i=1}^{n}$. Hence, C' is a complementary Hamiltonian cycle. On the other hand, given a complementary Hamiltonian cycle C' for $B(X, Y)$, the edges in $C' - M$ induce a Hamiltonian cycle for G. Indeed, $C' - M = \{(y_{[1]}, x_{[2]}), (y_{[2]}, x_{[3]}), \ldots, (y_{[n-1]}, x_{[n]}), (y_{[n]}, x_{[1]})\}$, which means that $v_{[1]}, v_{[2]}, \ldots, v_{[n]}, v_{[1]}$ is a Hamiltonian cycle for G. This completes the proof of the theorem. □

In a related technical report, Vairaktarakis and Solow, 2001 have shown that the CHC_N problem is also strongly NP-complete using a reduction to the problem of finding a Hamiltonian cycle in a cubic graph, that is, a graph where every node has degree 3 (see Garey and Johnson, 1979). Note that the complexity status of CHC_N does not follow from CHC_A (i.e., from Theorem 1) because it lacks the special neighboring-beam structure. For this reason the complexity of CHC_N is significantly more involved and lengthy. All the details of this result are relegated to the above mentioned technical report. Since CHC_N is a special case of CHC_B, the latter problem is at least as difficult.

Because of the foregoing NP-completeness results, the only means currently available for identifying a solution for CHC_N are the use of enumeration algorithms. To improve the efficiency of such algorithms, structural properties of an optimal solution for CHC_N are identified in Section 4. The approach is to use cuts to decompose B into bipartite subgraphs B_k, for $k = 1, 2, \ldots, r$. Doing so means that the desired matching I of B for which $I \cup M$ forms a com-

plementary Hamiltonian cycle is the collection of smaller matchings I_k, one for each bipartite subgraph B_k.

4 Blocks, Cuts, and Patching

In this section, properties of a complementary Hamiltonian cycle (if one exists) for a neighboring-beam bipartite graph $B(U,V)$ and given matching M are presented. As before, assume that $U = \{u_1, u_2, \ldots, u_n\}$ and $V = \{v_1, v_2, \ldots, v_n\}$. To simplify the exposition, the edges $e = (u_j, v_i)$ of M are added to $E(B)$ whenever $l_i \leq j \leq r_i$ and $N(v_i) = [u_{l_i}, u_{r_i}]$. Clearly, no complementary Hamiltonian cycle C can include e because C includes the edge $(u_j, v_i) \in M$. Let $M' = \{(u_j, v_i) \in M : l_i \leq j \leq r_i\}$. Then, the revised edge set of B is $E(B) \cup M'$. Note that $B(U,V)$ is a bipartite multigraph with respect to the edge set $E(B) \cup M' \cup M$, and $B(U,V)$ satisfies the neighboring-beam property with respect to the edge set $E(B) \cup M'$.

Example 2: To clarify this point and to illustrate the results presented in this section, consider the example in Figure 3, in which $N(v_{11})$ is the node interval $[u_9, u_{12}]$. On the other hand, $N(v_1) = \{u_1, u_2, u_4\}$ is not a node interval but, after including u_3 (for which $(u_3, v_1) \in M$) in $N(v_1)$, $N(v_1) = [u_1, u_4]$ which is a node interval. For notational simplicity, in the rest of this paper it is assumed that $E(B)$ already includes all edges in M' and hence $B(U,V)$ satisfies the neighboring-beam property with respect to $E(B)$.

Consider the matching

$$I_0 = \{(u_i, v_i)\}_{i=1}^{n}$$

of B. This matching will be used extensively in the rest of this article. It is possible that $I_0 \cup M$ forms a single cycle. In this case, $C = I_0 \cup M$ is a complementary Hamiltonian cycle. In general, however, $I_0 \cup M$ is a union of subcycles as shown in the following example.

Example 3: The graph $I_0 \cup M$ in Figure 4 gives rise to the 5 subcycles $C_1, C_2, \ldots, C_5$. Moreover, each cycle contains an even number of nodes because when a node v_i is in a subcycle, the corresponding node u_i is also included in the same subcycle. Moreover, each subcycle $C_i \subseteq I_0 \cup M$ can be partitioned into k *blocks* of consecutive nodes, say, $v_{i_1}, u_{i_1}, v_{i_1+1}, u_{i_1+1}, \ldots$ followed by another block of vertices $v_{i_2}, u_{i_2}, v_{i_2+1}, u_{i_2+1}, \ldots$, and so on, with $1 \leq i_1 < i_2 < i_3 < \ldots < i_k \leq n$. The subcycle C_4, for instance, consists of three different blocks of nodes, namely, the block $S_1 = [u_5, u_6] \cup [v_5, v_6]$; the block $S_2 = [u_9, u_{11}] \cup [v_9, v_{11}]$; and the block $S_3 = \{u_{13}\} \cup \{v_{13}\}$. Also, observe that $\{u_2\} \cup \{v_2\}$ and $\{u_{12}\} \cup \{v_{12}\}$ in $I_0 \cap M$ are the blocks of the trivial 2-edge

subcycles C_2 and C_5, respectively.

INSERT FIGURE 4 HERE

The formal definition of a block follows.

Definition 7 *Given the subcycles $C_1, \ldots, C_r$ of $I_0 \cup M$, a set $S = [u_i, u_j] \cup [v_i, v_j]$ of nodes is called a* **block** *if and only if all nodes in S belong to the same subcycle C_k of $I_0 \cup M$, while $u_{i-1}, v_{i-1} \notin C_k$ (if $i > 1$) and $u_{j+1}, v_{j+1} \notin C_k$ (if $j < n$).*

From here on, it is assumed that $I_0 \cup M$ is a collection of subcycles, $C_1, C_2, \ldots, C_r$. When $r \geq 2$, an algorithm is needed to merge or *patch* all subcycles of $I_0 \cup M$ into a single cycle; this is referred to as *patching*. To merge the subcycles of $I_0 \cup M$, the matching I_0 is modified by adding and deleting edges so that the revised matching, say I, results in a single Hamiltonian cycle $I \cup M$. To do this, $B(U, V)$ is first partitioned using *cuts* of the form defined below.

Definition 8 *For $1 \leq i < n$, the* **cut** *c_i is a partitioning of $B(U, V)$ into the two bipartite subgraphs $B_1([u_1, u_i], [v_1, v_i])$ and $B_2([u_{i+1}, u_n], [v_{i+1}, v_n])$.*

In the above definition, let c_j be a second cut such that $i < j$, and c_j partitions $B(U, V)$ into $B_3([u_1, u_j], [v_1, v_j])$ and $B_4([u_{j+1}, u_n], [v_{j+1}, v_n])$. Then the cuts c_i, c_j partition $B(U, V)$ into the subgraphs $B_3, B([u_{j+1}, u_i], [v_{j+1}, v_i])$, and B_2. Similarly, a collection of $r - 1$ cuts partitions $B(U, V)$ into r bipartite subgraphs $B_1, \ldots, B_r$. After these subgraphs are identified, they are used to revise the initial matching I_0 by replacing edges in $E(B_k) \cap I_0$ with edges in $E(B_k) - I_0$. This approach reduces the process of revising I_0 using edges in $E(B)$ to revising part of I_0 using edges from $E(B_k)$ only, for $k = 1, 2, \ldots, r$. Moreover, the subgraphs B_k are shown to possess properties that are useful in finding a complementary Hamiltonian cycle, if one exists. Two types of cuts—*cycle cuts* and *connectivity cuts*— are used. The definition of cycle cuts follows.

Definition 9 *Let $S = [u_i, u_j] \cup [v_i, v_j]$ be a block for a cycle C_k of $I_0 \cup M$. The cuts c_i and c_{j-1} of $B(U, V)$ are called* **cycle cuts** *associated with S.*

Note that when $j = i + 1$, the block S in Definition 9 consists of only two pairs of nodes and the two cuts c_i and c_{j-1} coincide. The two cuts are distinct only if $j \geq i + 2$. If $i = j$, then no cycle cut is associated with S.

Example 4: In Figure 4, c_5 is the only cut associated with the block $[u_5, u_6] \cup [v_5, v_6]$; c_7 is the only cut associated with the block $[u_7, u_8] \cup [v_7, v_8]$; and c_9, c_{10} are the cuts associated with the block $[u_9, u_{11}] \cup [v_9, v_{11}]$. No cuts are associated with the other blocks in Figure 4 because they

consist of a single pair of nodes. The significance of cycle cuts is demonstrated later. First, we introduce another type of cuts; the connectivity cuts.

Definition 10 *For $1 \leq i \leq n-1$, the cut c_i is a* **connectivity cut** *if and only if the edges (u_i, v_i) and (u_{i+1}, v_{i+1}) belong to different subcycles of $I_0 \cup M$ and at least one of the edges $(u_{i+1}, v_i), (u_i, v_{i+1})$ is not in $E(B)$.*

Example 5: In the example of Figure 4, c_8 is a connectivity cut because v_8 and u_8 belong to C_3, v_9 and u_9 belong to C_4, and $(v_8, u_9) \notin E(B)$ (see Figure 3).

The cycle and connectivity cuts just defined are motivated by Theorems 4 and 5 provided in the appendix. These theorems show that, if a CHC exists, say $I \cup M$, then we can construct one such that no edge in I has its endpoints on opposite sides of a cut. Therefore, matching I can be a collection $I = \{I_k\}_{k=1}^{r}$ where I_k is a perfect matching of B_k.

Having found the cycle and connectivity cuts, in the next subsection we provide conditions for the existence of a CHC.

4.1 Necessary Conditions for the Existence of a CHC

In this subsection necessary conditions are presented for the existence of a complementary Hamiltonian cycle. The following lemma states that I_0 must be a subset of $E(B)$ for a CHC to exist. Therefore, the subcycles of $I_0 \cup M$ described earlier are well defined.

Proposition 2 *A necessary condition for the existence of a complementary Hamiltonian cycle for $B(U, V)$ is that $(u_i, v_i) \in E(B)$, for every $1 \leq i \leq n$.*

Proof: By contradiction, suppose that $(u_i, v_i) \notin E(B)$ for some $v_i \in V$. Then, either (i) $N(v_i) \subseteq [u_{i+1}, u_n]$, or (ii) $N(v_i) \subseteq [u_1, u_{i-1}]$. Assume, without loss of generality, that case (i) holds. According to the neighboring-beam property for bipartite graphs (Definition 6), $u_i \notin N(v)$ for any $v \in [v_i, v_n]$, as shown in Figure 5.

INSERT FIGURE 5 HERE

By symmetry, $N(u_i) \subseteq [v_1, v_{i-1}]$, and hence Proposition 1 suggests that $v_i \notin N(u)$, for any $u \in [u_1, u_i]$. Consider the subgraphs $B_1([u_1, u_i], [v_1, v_{i-1}])$ and $B_2([u_{i+1}, u_n], [v_i, v_n])$ of $B(U, V)$. Recall from Section 2 that, in a complementary Hamiltonian cycle $I \cup M$, set I is a perfect matching of B. However, no perfect matching exists for B because i) $E(B)$ contains only one edge with one endpoint in B_1 and the other in B_2, and ii) the disconnected parts B_1 and B_2 do not have the same number of nodes in each of their two bipartition sets. This completes the proof. □

We refer to the condition in Proposition 2 as Condition 1. This condition will later be used by our algorithm to check for existence of a CHC. For a second necessary condition, observe that the cycle and connectivity cuts introduced so far decompose B into smaller subgraphs, say $B_1(U_1, V_1), B_2(U_2, V_2), \ldots, B_r(U_r, V_r)$. Also, the matching I_0 results to subcycles, say $C_1, C_2, \ldots, C_{n_0}$, of $I_0 \cup M$. Given these subgraphs and subcycles we construct a graph G as follows: Let

$$V(G) = \{C_1, C_2, \ldots, C_{n_0}\}, \quad \textit{and}$$

$$E(G) = \{(C_i, C_j) \ : \ \textit{both } C_i \textit{ and } C_j \textit{ traverse at least a pair of nodes in } B_k \textit{ for some } 1 \leq k \leq r\}.$$

We refer to G as the *cycle graph*.

Example 6: The cycle graph associated with the example in Figure 4 is depicted in Figure 6. Nodes C_1, C_2, C_3, C_4 are pairwise connected because they are all represented in subgraph B_1. Node C_4 is linked only to C_5 because of subgraph B_6. Now we can state Condition 2.

INSERT FIGURE 6 HERE

Theorem 2 *If a CHC exists, then the cycle graph G is connected.*

Proof: Given the cycle and connectivity cuts identified in Theorems 5 and 4, the best one can do with subgraphs $B_1(U_1, V_1), B_2(U_2, V_2), \ldots, B_r(U_r, V_r)$ is to patch all subcycles of $I_0 \cup M$ that traverse nodes in B_k, for every $k = 1, 2, \ldots, r$. This may or may not be possible for some of the subgraphs; this depends on $E(B_k)$ and M. Suppose that a CHC exists for B, say $I^* \cup M$. In light of Theorems 4 and 5 we assume that $I^* \subseteq \cup_{k=1}^{r} E(B_k)$.

Using I^* we can construct a spanning subgraph of G as follows. Suppose we traverse $I^* \cup M$ starting from an arbitrary edge $e_1^* \in I^*$. Let $e_1^*, e_2^*, \ldots, e_n^*$ be the list of edges of I^* ordered in the order they are traversed. Let $e_1', e_2', \ldots, e_m'$ be the edges in I^* whose endpoints lie in different subcycles of $I_0 \cup M$. Let C_i, C_i' be the two subcycles associated with edge e_i' for $1 \leq i \leq m$. Consider the ordered list $(C_1, C_1'), (C_2, C_2'), \ldots, (C_m, C_m')$. Since $I^* \cup M$ is a CHC we must have that $C_1' = C_2$, $C_2' = C_3$, etc., and $C_m' = C_1$. Hence, the pairs $(C_1, C_1'), (C_2, C_2'), \ldots, (C_m, C_m')$ can be rewritten as $(C_1, C_2), (C_2, C_3), \ldots, (C_m, C_1)$.

By construction, for every $i = 1, 2, \ldots, m$ the edge (C_i, C_i') is in $E(G)$. Indeed, we have assumed that $e_i' \in I^* \subseteq \cup_{k=1}^{r} E(B_k)$. Since $I^* \cup M$ is a CHC, we can trace all subcycles of $I_0 \cup M$ by traversing the subcycles associated with $e_1', e_2', \ldots, e_m'$. This means that the edges $(C_1, C_2), (C_2, C_3), \ldots, (C_m, C_1)$ span G. Equivalently, G is connected. This completes the proof of the Theorem. $\square$

The next section is dedicated to patching the subcycles of $I_0 \cup M$. This can be done simply by producing all perfect matchings of $\cup_{i=1}^{r} B_r$. Such approach is computationally inefficient. For this reason, more efficient algorithms are developed.

5 Patching the Subcycles of $I_0 \cup M$

In this section, an algorithm for solving CHC_N is presented that takes advantage of all the results developed so far. A related patching algorithm to the one described here can be found in Karp, 1979 for the nonsymmetric TSP. Our algorithm starts by decomposing B into smaller bipartite subgraphs by identifying the cycle and connectivity cuts. For subgraphs B_k defined by cycle cuts we use the partial matching defined in Theorem 5. For the remaining subgraphs B_k, a heuristic procedure called Criss-Cross is applied in hopes of merging all the subcycles that traverse nodes in B_k. Procedure Criss-Cross is described and analyzed in detail later in this section. For all subgraphs B_k for which Criss-Cross cannot merge all the subcycles associated with the nodes of B_k, the algorithm of Fukuda and Matsui, 1994 is used to generate all possible perfect matchings. This algorithm is referred to as *Algorithm FM* and uses a binary search tree to produce all the perfect matchings of an arbitrary bipartite graph B. It only requires an initial perfect matching; this matching is I_0 in our implementation. Hence, we refer to algorithm FM as FM$(B(U,V), I_0)$. Fukuda and Matsui, 1994 show that the effort required by algorithm FM to find all the perfect matchings of B is $\mathcal{O}(c(|V| + |E(B)|))$, where c is the number of perfect matchings. Counting the number of perfect matchings is #P-complete. Finally, the union of all partial matchings (over the subgraphs B_k) yields matching I which results to a CHC $I \cup M$ if one exists. The steps of the algorithm are as follows.

Algorithm CHC_N

Input : A Bipartite subgraph $B(U,V)$ and perfect matching M of B

Output : A CHC if one exists

Begin

$I := \emptyset$, $R := \emptyset$

Let $C_1, C_2, \ldots, C_m$ be the subcycles of $I_0 \cup M$

If $m = 1$ **then stop** (a CHC is found)

If Condition 1 is violated **then stop** (problem is infeasible)

Identify all cycle cuts c_i according to Definition 9

Identify all connectivity cuts c_i according to Definition 10

Let $B_1(U_1, V_1), B_2(U_2, V_2), \ldots, B_r(U_r, V_r)$ be the resulting bipartite subgraphs

If Condition 2 is violated **then stop** (problem is infeasible)

For $k = 1$ **to** r **do begin**

If B_k is defined by two cycle cuts **then** let $I_k = I_0 \cap E(B_k)$ and $I := I + I_k$ **else**

If Criss-Cross$(B_k(U_k, V_k), M)$ merges all subcycles with nodes in $B_k(U_k, V_k)$ **then** $I := I + I_k$

else $R := R + B_k(U_k, V_k)$

[1] Apply algorithm $FM(R, I_0 \cap R)$

For every perfect matching I' of R **do**

If $I \cup I' \cup M$ is a single cycle **then stop** (a CHC is found)

end

End

Identifying cuts takes $\mathcal{O}(n)$ time. Verifying conditions 1 and 2 takes $\mathcal{O}(n)$ time. As we will see shortly, running Criss-Cross on all subgraphs $B_1,\ B_2, \ldots, B_r$ takes $\mathcal{O}(n)$ time as well. Hence, it is expected that the time consuming part of algorithm CHC_N is line [1] (whenever this step is required).

5.1 The Criss-Cross Algorithm

In the rest of this section we introduce and subsequently analyze procedure Criss-Cross. First, we describe its main steps.

Algorithm Criss-Cross(B_k)

Input : Bipartite subgraph $B_k(U_k, V_k)$ of B with node sets $U_k = \{u_j^k\}_{j=1}^{n_k}$ and $V_k = \{v_j^k\}_{j=1}^{n_k}$, and perfect matching M for B

Output : Bipartite matching I_k of B_k

Begin

Set $I_k := \emptyset$, $I_0^k := \{(v_j^k, u_j^k)\}_{j=1}^{n_k}$ and $flag := 0$

Repeat

Let $C_1^k, C_2^k, \ldots, C_{n_k}^k$ be the subcycles of $(I_0^k + I_k) \cup M$ traversing $v_1^k, v_2^k, \ldots, v_{n_k}^k$ respectively, and $C_{i_k+1}^k, C_{i_k+2}^k, \ldots, C_{i_k+s}^k$ be the longest leading substring of consecutive distinct subcycles

[1] **If** $n_k = 1$ **then** $I_k := (v_{i_k+1}^k, u_{i_k+1}^k)$

else begin

[2] **For** $j = 1$ **to** $\lfloor \frac{s}{2} \rfloor - 1$ **do** $I_k := I_k + (v_{i_k+2j-1}^k, u_{i_k+2j+1}^k) + (v_{i_k+2j+2}^k, u_{i_k+2j}^k)$

[3] **If** $s =$*even* **then** $I_k := I_k + (v_{i_k+2}^k, u_{i_k+1}^k) + (v_{i_k+s-1}^k, u_{i_k+s}^k)$

else $I_k := I_k + (v^k_{i_k+2}, u^k_{i_k+1}) + (v^k_{i_k+s}, u^k_{i_k+s-1})$

[4] **If** $s = odd$ **then**

If $(v^k_{i_k+s-2}, u^k_{i_k+s}) \notin M$) **then** $I_k := I_k + (v^k_{i_k+s-2}, u^k_{i_k+s})$

else $I_k := I_k - (v^k_{i_k+s-1}, u^k_{i_k+s}) + (v^k_{i_k+s-1}, u^k_{i_k+s-2}) + (v^k_{i_k+s}, u^k_{i_k+s})$

end

[5] **If** $I_k \not\subseteq E(B_k)$ **then** $flag := 1$ **else** $I^k_0 := I^k_0 - \{(v^k_{i_k+j}, u^k_{i_k+j})\}^s_{j=1}$

Until $(flag = 1)$ **or** ($\{v^k_j\}^{n_k}_{j=1}$ are traversed by a single subcycle of $(I^k_0 + I_k) \cup M$)

If $flag = 0$ then $I_k := I_k + I^k_0$ **else** $I_k := I^k_0$

End

Evidently, Criss-Cross(B_k) produces a perfect matching I_k of subgraph B_k when B_k is defined by one or more connectivity cuts. Any two consecutive subcycles among $C^k_1, C^k_2, \ldots, C^k_{n_k}$ are distinct otherwise a cycle cut would exist between these two subcycles. Hence, a maximum string of distinct subcycles $C^k_1, C^k_2, \ldots, C^k_s$ must exist. Initially, I_k is null. The Repeat-Until loop merges these subcycles by amending I_k iteratively. In each iteration, the subcycles $C^k_1, C^k_2, \ldots, C^k_s$ are merged via I_k into a single subcycle, say C^k_1. Hence, in the next iteration of the Repeat-Until loop the ordered list $C^k_1, C^k_2, \ldots, C^k_{n_k}$ becomes $C^k_1, C^k_1, \ldots, C^k_1, C^k_{s+1}, \ldots, C^k_{n_k}$. This process repeats until all of $C^k_1, C^k_2, \ldots, C^k_{n_k}$ are merged into a single subcycle. Line [1] considers the trivial case where B_k consists of a single pair of nodes. In all other cases lines [2]-[4] are repeated in every iteration of the Repeat-Until loop. Line [2] lends Criss-Cross its name because the matching I_k produced looks as in Figure 7(a) where $n_k = even$. Lines [3] and [4] add the last few edges at the head and the tail of the current string of subcycles depending on whether s is odd or even. Throughout lines [2]-[4] we do not check whether the edges added to I_k are in fact present in $E(B_k)$. For simplicity in our presentation, this test is done in line [5]. If any of the edges of I_k is missing from $E(B_k)$, a flag is raised indicating that Criss-Cross fails to merge all of $C^k_1, C^k_2, \ldots, C^k_{n_k}$ into a single subcycle. In this case Criss-Cross returns the matching $I_k = I^k_0$ which is the restriction of I_0 on the set $E(B_k)$. Otherwise the algorithm proceeds until all of $C^k_1, C^k_2, \ldots, C^k_{n_k}$ are merged.

INSERT FIGURE 7 HERE

Example 7: The application of Criss-Cross on subgraph B_1 of Figure 4 yields the matching I_1 depicted in Figure 7(b). In this case $n_1 = 5$. The first leading substring of distinct subcycles in $C_1C_2C_1C_3C_4$ is C_1C_2. Upon merging C_1C_2 into a single subcycle, say C'_1, the resulting substring is $C'_1C'_1C'_1C_3C_4$. Then the next substring is $C'_1C_3C_4$. Algorithm Criss-Cross applied on these 3 subcycles yields the partial matching $(v_3, u_5), (v_4, u_3)$, and (v_5, u_4). Note that, if we

let $I = I_0 - \{(v_i, u_i)\}_{i=1}^{5} + I_1$, then in $I \cup M$ all of C_1, C_2, C_3, C_4 are merged into a single subcycle.

Since CHC_N is NP-complete, Criss-Cross is not expected to solve all instances of the problem. As we will show however, it solves most instances and dramatically reduces the search space for finding a CHC. We demonstrate this in the following theorem. We need the following definition.

Definition 11 *For $v_i \in V$ and its neighborhood $N(v_i) = [u_{l_i}, u_{r_i}]$, let $d^l(v_i) = i - l_i$ denote the left degree of v_i and $d^r(v_i) = r_i - i$ be the right degree of v_i.*

Intuitively, $d^l(v_i)$ ($d^r(v_i)$) indicates the number of nodes to the left (right) of u_i that are adjacent to v_i. Similarly for the left and right degrees of u_i. With these definitions, note that c_i is a connectivity cut if at least one of $d^r(v_i) = 0$ or $d^l(v_{i+1}) = 0$.

Consider the bipartite subgraphs $B_k(U_k, V_k)$, for $1 \leq k \leq r$ determined by one or more connectivity cuts (those determined by 2 cycle cuts are considered in Theorem 5) and hence Criss-Cross is not appropriate. We show that, when $d^l(v), d^r(v) \geq 2$ for every $v \in V_k - \{v_1^k, v_2^k, v_{n-1}^k, v_n^k\}$ and $d^l(v_1^k) = d^r(v_{n_k}^k) = 0$, $d^l(v_2^k) = d^r(v_{n_k-1}^k) = 1$, algorithm Criss-Cross yields a matching $I_k \in E(B_k)$ that merges the subcycles of $I_0 \cup M$ that contain one or more pairs $u_i, v_i \in U_k \cup V_k$.

Theorem 3 *Let $B_k(U_k, V_k)$ be a bipartite subgraph of the neighboring-beam bipartite graph $B(U, V)$. If $d^l(v), d^r(v) \geq 2$ for every $v \in V_k - \{v_1^k, v_2^k, v_{n_k-1}^k, v_{n_k}^k\}$ the algorithm Criss-Cross merges all subcycles with nodes in V_k into a single subcycle in $\mathcal{O}(|V_k|)$ time.*

The node degree requirements in Theorem 3 are very mild. As a result Criss-Cross usually succeeds in providing a CHC (if one exists) eliminating the need for additional search. Criss-Cross is likely to fail only when the subgraph B_k is sparse. Then, additional search is required but in this case the search space is small. This is very clearly demonstrated in our computational experiments.

6 Computational Experiments

We coded algorithm CHC_N in C++ and ran experiments on a PC running at 266 MHz. We tested the algorithm on randomly generated problems. It has been very difficult to devise a problem instance that requires CPU time noticeably greater than 0 seconds. In particular, we tried neighboring beam bipartite graphs of various degrees of density from around 10% of the edges of a complete bipartite graph up to 90%. We tried several hundreds of problems of sizes varying from 50 to 250 pairs of nodes. We found less than 5 that employed the search

algorithm FM. Even then, the search required less than 10 nodes. The remaining problems where solved in virtually 0 seconds using Conditions 1 and 2, our structural properties, and algorithm Criss-Cross.

To develop insight on how to obtain hard instances, observe the following. The NP-complete instance of CHC_N provided in Vairaktarakis and Solow, 2001 has $d^l(v), d^r(v) \leq 2$ and $d^l(v) + d^r(v) \leq 3$, for every $v \in V$. On the other hand, problem CHC is easily solvable if $d^l(v), d^r(v) \geq 2$ for every $v \in V$. As a result, problem CHC is expected to be hard only for very sparse bipartite graphs. Such graphs however, are likely to violate Condition 2 in which case no CHC exists. For these reasons, we developed a problem generator for sparse graphs. Given a number n of node-pairs, consider the graph $B_0(V, U)$ every node of which (except the first and last 2) have $d^l(v), d^r(v) = 1$ for every $v \in V$. Graph B_0 has $3n - 2$ edges; 3 per node $v \in V$, minus 1 for the first and last node of V. We enumerated these $3n-2$ edges and randomly drew integers uniformly from the interval $[1, 3n - 2]$ without repetition. The edges corresponding to the integers drawn formed the edge set of B_0. To span a variety of sparse graphs we experimented with graphs B_0 with $\lceil \frac{3n-2}{K} \rceil$ edges for $K = 2, 3, \ldots, 9$. In all cases, the matching M was determined by selecting a random permutation of n elements. In this construction, the effectiveness of Criss-Cross deteriorates as K increases (because $E(B)$ has fewer and fewer edges) while the effectiveness of Condition 2 improves. In Table 1 we report our findings for n=50, 100, 150, 200, and 250 pairs of nodes. For each (n, K) combination we run 50 randomly generated problems and report how many of them made use of the search algorithm FM. Over those problems (out of the 50) that search was required, we report the average number of nodes in the search tree and the average CPU time in seconds. All other problems where solved in virtually 0 time.

Table 1: Computational Experiment

K/N	50			100			150			200			250			Overall Average		
	FM	nodes	CPU	FM	nodes	CPU	FM	nodes	CPU	FM	nodes	CPU	FM	nodes	CPU	FM	nodes	CPU
2	4	5.5	0.01	0	0	0	1	3	0.05	0	0	0	0	0	0	1	1.7	0.01
3	1	3	0.03	0	0	0	0	0	0	1	5	0.06	0	0	0	0.4	1.6	0.02
4	2	3	0.03	1	3	0.07	2	4	0.04	0	0	0	0	0	0	1	2.0	0.03
5	2	3	0.04	1	3	0.04	0	0	0	0	0	0	1	3	0.07	0.8	1.8	0.03
6	3	3.67	0.02	1	3	0.07	0	0	0	0	0	0	0	0	0	0.8	1.3	0.02
7	0	0	0	2	3.5	0.02	0	0	0	0	0	0	1	3	0.1	0.6	1.3	0.02
8	0	0	0	1	3	0.09	0	0	0	0	0	0	0	0	0	0.2	0.6	0.02
9	2	4	0.02	0	0	0	1	3	0.06	0	0	0	2	3	0.06	1	2.0	0.03

Based on these results the hardest problems appear to be for very sparse graphs (i.e., $K \leq 5$) with few node-pairs ($n = 50$). Across all combinations our results dramatically reduce the need for additional search. This is precisely the aim of the cycle and connectivity cuts. Whenever

these cuts result to subgraphs that are relatively dense, algorithm Criss-Cross eliminates the need for algorithm FM to consider such subgraphs. On the other hand, whenever the cuts result to very sparse subgraphs, Condition 2 quickly identifies when no CHC exists.

To compare the performance of our algorithms to the performance of a pure search algorithm, we evaluated algorithm FM on problems with various density levels for each value of $n = 50$, 100, 150, 200, and 250. In Table 2 we solve one problem for every (n,density) combination. For each problem we report the number of perfect matchings tested by FM as well as the CPU time in seconds. We terminated the exhaustive algorithm FM after running for 1 hour. In Table 2 the problems corresponding to $(n, density)$ combinations (50,80%) and (100,30%) terminate without a solution. Others are solved very quickly because one of the early trial matchings happened to yield a CHC. Performance is even more unpredictable for $n = 200$ and 250. The boldfaced numbers in Table 2 correspond to problems where the PC ran out of memory. Evidently, as n increases the storage requirements become increasingly prohibitive. In the columns headed by "nodes" we report the number of trial perfect matchings. In all but 4 cases (where the number of nodes is ≤ 3) the trial matchings are many more than those reported in Table 1.

Table 2: Computational Performance of Search Algorithm

	50		100		150		200		250	
Density/n	nodes	CPU	nodes	CPU	nodes	CPU	nodes	CPU	nodes	CPU
30%	65	0.0	3194010	3600.0	15408	60.7	1441	37.8	**999**	43.6
50%	2	0.0	0	0.0	730	26.6	**1599**	420.9	**999**	406.4
70%	99	0.1	771	1.3	632	2.3	718	269.8	517	295.4
80%	11325735	3600.0	1593	2.8	2793	202.1	148	42.8	**999**	472.1
90%	0	0.0	417	0.7	193	16.1	3	0.0	**999**	455.1

In comparison, our algorithms exhibit robust performance across sizes and consistently solve the problem in negligible amount of time by limiting the exhaustive search to only a handful of perfect matchings.

7 Conclusions

A solution to the complementary Hamiltonian cycle problem on bipartite graphs, motivated by workforce planning problems in assembly lines was proposed. The motivating application led to the special class of bipartite graphs that possess the neighboring-beam property. For this class, a number of structural properties of complementary Hamiltonian cycles were presented, as well as

efficient solutions for special (but quite general) cases. Based on these results, an algorithm that exploits these properties was developed. Computational experiments showed that the proposed algorithm dramaticaly reduces the need for additional search. Future research directions will focus on using algorithm CHC_N to solve the workforce planning applications mentioned earlier. In these applications, the various objective functions determine the optimal complementary Hamiltonian cycle.

Acknowledgement: The authors are thankful to Joseph G. Szmerekofsky who provided his constructive input for some of our results, and valuable help in the implementation of all algorithms and the computational experiment.

References

[1] Fukuda K. and T. Matsui. Finding All the Perfect Matchings in Bipartite Graphs. *Applied Mathematics letters*, 7(1):15-18, 1994.

[2] Garey M.R. and D.S. Johnson. Computers and Intractability. *W.H. Freeman*, San Francisco, CA, 1979.

[3] Karp R.M. A Patching Algorithm for the Nonsymmetric Traveling Salesman Problem, *SIAM Journal on Computing*, 8:561-573. 1979.

[4] Lee C.-Y. and G. Vairaktarakis. Workforce Planning in Mixed Model Transfer Lines, *Operations Research*, 45(4):553-567, 1997.

[5] Vairaktarakis G. and X.Q. Cai. Complexity of Workforce Scheduling in Transfer Lines, *Journal of Global Optimization*, submitted, 1998.

[6] Vairaktarakis G and D. Solow. Properties of Complementary Hamiltonian Cycles on Bipartite Graphs, Weatherhead School of Management, Dept. of Operations, Case Western Reserve University, Research Report, 2001.

Appendix

In this section we verify the validity of connectivity and cycle cuts. The following Theorem provides the motivation for Definition **??** of the connectivity cuts. Based on this result, the existence of a connectivity cut indicates that there can't be a CHC $I \cup M$ having an edge $e \in I$ with endpoints in opposite sides of that cut.

Theorem 4 *Let I be an arbitrary matching of $B(U,V)$ such that $I \cup M$ is a complementary Hamiltonian cycle. Suppose that there exist pairs of nodes u_i, v_i and u_{i+1}, v_{i+1} belonging to different subcycles of $I_0 \cup M$. If at least one of (u_i, v_{i+1}), (u_{i+1}, v_i) is not in $E(B)$, then I cannot contain any edge (u,v) in which u and v are on opposite sides of the cut c_i.*

Proof: Suppose, without loss of generality, that $(u_i, v_{i+1}) \notin E(B)$. Because $I \subseteq E(B)$, it must be that $(u_i, v_{i+1}) \notin I$. Also, since (u_i, v_i) and (u_{i+1}, v_{i+1}) belong to different cycles of $I_0 \cup M$, it must be that $(u_i, v_{i+1}) \notin M$. According to Proposition 2, $u_{i+1} \in N(v_{i+1})$ (see Figure 8). This fact together with the observation that $u_i \notin N(v_{i+1})$ imply that $N(v_{i+1}) \subseteq [u_{i+1}, u_n]$. Because B has the neighboring-beam property, $N(v_j) \subseteq [u_{i+1}, u_n]$ for every v_j with $j > i$.

INSERT FIGURE 8 HERE

Since I is a one-to-one correspondence of nodes in V to nodes in U, the nodes in $[v_{i+1}, v_n]$ must be matched in I with nodes from $[u_{i+1}, u_n]$. Hence, I does not contain edges (u,v) with u and v being on opposite sides of the cut c_i. This completes the proof of the lemma. □

The motivation for Definition **??** for cycle cuts is verified next. When we merge a cycle that contains blocks of length 3 or more, one need only consider removing the edges of I_0 connecting the first and last pair of vertices in the block, as shown in the following lemma.

Theorem 5 *If there exists a complementary Hamiltonian cycle $I_C \cup M$ for $B(U,V)$, then there is a cycle $I \cup M$ such that, for every block $S = [u_i, u_{i+r}] \cup [v_i, v_{i+r}]$ of $I_0 \cup M$, $(u_j, v_j) \in I$, for every $i < j < i + r$.*

Proof: Note that the property of the lemma is meaningful only when $r \geq 2$. In this case, the subgraph $B(S)$ with node set S must consist of 3 or more pairs of nodes. Intuitively, the lemma suggests that there exists a matching I that contains all "middle" edges of $I_0 \cap B(S)$.

If $I_0 \cup M$ is a complementary Hamiltonian cycle, then the lemma holds trivially. Hence, suppose that $I_0 \cup M$ is the union $C_1 \cup C_2 \cdots \cup C_m$ of $m \geq 2$ subcycles. Since $I_C \cup M$ forms a single cycle, it must contain edges whose endpoints lie in different cycles $C_1, C_2, \ldots, C_m$. Equivalently, I_C contains edges that lie in different blocks $S_1, S_2, \ldots$. Let S_s and S_{s_0} $(1 \leq s < s_0 \leq n)$ be two blocks such that $(v_{i_0}, u_{j_0}) \in I_C$ with $v_{i_0} \in S_{s_0}$ and $u_{j_0} \in S_s$ [see Figure 9(a)]—the case where $u_{j_0} \in S_{s_0}$ and $v_{i_0} \in S_s$ is symmetric). Also, let $S_{s+1}, \ldots, S_{s_0-1}$ be the blocks of nodes between S_s and S_{s_0}. Given the cycle $I_C \cup M$, another cycle $I \cup M$ is constructed that satisfies $(u_j, v_j) \in I$ for every $i < j < i + r$. Let C be the cycle traversing the nodes of S. Starting from I_0, a matching I is constructed that merges all subcycles traversing the nodes of the blocks $S_s, S_{s+1}, \ldots, S_{s_0}$, while maintaining the edges $(u_j, v_j) \in I$, for every $i < j < i + r$. This construction can be repeated for any two subgraphs S_s and S_{s_0} that contain an endpoint of an edge $e \in I_C$. Since the number of such edges is finite, a matching I that satisfies the lemma is eventually obtained.

To merge the subcycles $C_s, C_{s+1}, \ldots, C_{s_0}$ associated with the blocks $S_s, S_{s+1}, \ldots, S_{s_0}$, respectively, consecutive subcycles C_k and C_{k+1} with $s \leq k < s_0$ are merged iteratively [see Figure 9(b)]. In merging these two subcycles, two cases can arise: (i) S_k and S_{k+1} each contains two or more pairs of nodes and (ii) either S_k or S_{k+1} contains a single pair of nodes.

Case (i): In this case, C_k and C_{k+1} are merged by deleting the edges $(u_{i_k+r_k}, v_{i_k+r_k})$ and $(u_{i_{k+1}}, v_{i_{k+1}})$ from I_0 and adding the edges $(u_{i_{k+1}}, v_{i_k+r_k})$ and $(u_{i_k+r_k}, v_{i_{k+1}})$, as shown in Figure 7(b). Hence, it suffices to show that $(u_{i_{k+1}}, v_{i_k+r_k}), (u_{i_k+r_k}, v_{i_{k+1}}) \in E(B) - M$. To see this observe that since $(u_{j_0}, v_{i_0}) \in I_C$, there cannot exist a connectivity cut c_i with $j_0 \leq i < i_0$. According to Theorem 4, if no such cut exists, it must be that $(u_{i_{k+1}}, v_{i_k+r_k}), (u_{i_k+r_k}, v_{i_{k+1}}) \in E(B)$. Moreover, $(u_{i_{k+1}}, v_{i_k+r_k}), (u_{i_k+r_k}, v_{i_{k+1}}) \notin M$ because these pairs of nodes belong to different blocks and hence different subcycles of $I_0 \cup M$. Observe that in merging C_k with C_{k+1}, the matching I contains the edges (u_j, v_j) of C_k, for $i_k < j < i_k + r_k$, as well as the edges (u_j, v_j) of C_{k+1}, for $i_{k+1} < j < i_{k+1} + r_{k+1}$. Therefore, iterative use of this merging scheme for the blocks $S_s, \ldots, S_{s_0}$ preserves the property of the lemma in all of the blocks.

INSERT FIGURE 9 HERE

Case (ii): For the blocks that consist of a single pair of vertices u_{i_k}, v_{i_k}, the construction illustrated in Figure 10 is used. Let (u_{i_j}, v_{i_j}) be the single pair of nodes of a block S_j, for some j with $s < j < s_0$. Let S_k be the first block, having 2 or more pairs of nodes, that precedes S_j in Figure 10. Also, assume that S_{k+q} is the first block, having 2 or more pairs of nodes, that follows S_j in Figure 10. If all the blocks preceding (following) S_j have exactly 2 nodes, let $S_k = S_s$ ($S_q = S_{s_0}$). As argued before, since $(u_{j_0}, v_{i_0}) \in I_C$, it must be that $(u_{i+1}, v_i) \in E(B) - M$, for $i = i_k + r_k, \ldots, i_k + r_k + q - 1$, and $(u_{i_k+r_k}, v_{i_k+r_k+q}) \in E(B) - M$. Hence, it is possible to use the construction in Figure 10 to merge the blocks $S_k, S_{k+1}, \ldots, S_{k+q}$ while preserving the property of the lemma for the subcycles C_k and C_{k+q}.

INSERT FIGURE 10 HERE

In both of the foregoing cases, the only edges in $I_0 \cap S_k$ that can be deleted from I_0 are the boundary edges (v_{i_k}, u_{i_k}) and $(v_{i_k+r_k}, u_{i_k+r_k})$, for every S_k among $S_s, S_{s+1}, \ldots, S_{s_0}$. This completes the proof of the lemma. □

Example 6: For the example of Figure 4, Theorem 5 suggests that the edge (u_{10}, v_{10}) is part of a complementary Hamiltonian cycle, if one exists. Evidently, Theorems 4 and 5 decompose the bipartite graph B into smaller subgraphs $B_1, \ldots, B_6$ and CHC_N is equivalent to finding a matching I_k in each B_k, $k = 1, 2, \ldots, 6$ so that $(\cup_k I_k) \cup M$ forms a complementary Hamiltonian cycle. In Figure 4, the subgraphs B_3 and B_5 consist of a single pair of nodes and B_4 is defined by two cycle cuts. Hence, the problem reduces to finding appropriate matchings for B_1, B_2 and B_6. In other words, there exists an optimal solution such that the matchings I_3, I_4 and I_5 are the ones indicated in Figure 3 for the subgraphs B_3, B_4 and B_5 respectively.

Proof of Theorem 3: If B_k consists of a single pair of nodes, then there is a unique choice for I_k as in line [1] of Criss-Cross. Consider the case that B_k consists of precisely 2 pairs of nodes, say (u_i, v_i) and (u_{i+1}, v_{i+1}). Moreover, the edges (u_{i+1}, v_i) and (u_i, v_{i+1}) belong in $E(B_k)$ otherwise a connectivity cut would exist between (u_i, v_i) and (u_{i+1}, v_{i+1}). These pairs must

belong to different subcycles of $I_0 \cup M$. Hence, if we set $I_k = \{(u_{i+1}, v_i), (u_i, v_{i+1})\}$, a subcycle is created that merges the two subcycles of $I_0 \cup M$ that contain the pairs of nodes u_i, v_i and u_{i+1}, v_{i+1}, respectively, as in Figure 9(b). This is precisely the matching produced by procedure Criss-Cross in line [3] (line [2] is inactive in this case).

If the number of pairs of nodes in $U_k \cup V_k$ is greater than 2, consider a construction similar to that in Figure 7(a). The subcycles considered in this construction are $C^k_{i_k+1}, C^k_{i_k+2}, \ldots, C^k_{i_k+s}$ found in each iteration of the Repeat-Until loop of Criss-Cross. The edges depicted in Figure 7(a) are those produced in lines [2]-[4] depending on whether s is odd or even. The edges utilized in this construction are of the form (u_{i+1}, v_i), (u_i, v_{i+1}), (u_{i+2}, v_i) and (u_i, v_{i+2}), all of which are in $E(B_k) - M$. To see that these edges belong in $E(B_k) - M$ note that $d^l(v), d^r(v) \geq 2$, for every $v \in V_k - \{v^k_1, v^k_2, v^k_{n_k-1}, v^k_{n_k}\}$. Also, $C^k_{i_k+j}$ and $C^k_{i_k+j+2}$ are distinct by construction and hence $(u^k_{i_k+j}, v^k_{i_k+j+2})$, $(u^k_{i_k+j+2}, v^k_{i_k+j}) \notin M$.

Now, consider the case where $s > 2$. We will show that lines [2]-[4] of Criss-Cross merge $C^k_{i_k+1}, C^k_{i_k+2}, \ldots, C^k_{i_k+s}$. Rather than dealing with pairs of nodes (v_{i_k+j}, u_{i_k+j}) for $1 \leq j \leq s$, replace subcycle C_{i_k+j} by node w_j. Consider the nodes $w^k_{i_k+1}, w^k_{i_k+2}, \ldots, w^k_{i_k+s}$. Then, traversing C_{i_k+j} is equivalent to entering w_{i_k+j} from - and exiting w_{i_k+j} to - nodes in $\{w^k_{i_k+1}, w^k_{i_k+2}, \ldots, w^k_{i_k+s}\} - \{w_{i_k+j}\}$. Then, merging all of $C^k_{i_k+1}, C^k_{i_k+2}, \ldots, C^k_{i_k+s}$ is equivalent to traversing all of $w^k_{i_k+1}, w^k_{i_k+2}, \ldots, w^k_{i_k+s}$ by a single cycle. We will show that this is precisely the result of lines [2]-[4] of Criss-Cross. We consider the case where $s = \textit{even}$; the case $s = \textit{odd}$ is analogous.

Indeed, the for-loop in [2] iteratively connects the nodes $w_{i_k+1}, w_{i_k+3}, \ldots, w_{i_k+s-1}$ into a path by means of the edges $(v^k_{i_k+2j-1}, u^k_{i_k+2j+1})$ for $j = 1, 2, \ldots, \lfloor \frac{s}{2} \rfloor - 1$. Also, in [2] the nodes $w_{i_k+2}, w_{i_k+4}, \ldots, w_{i_k+s-2}$ are linked into a path by means of the edges $(v^k_{i_k+2j+2}, u^k_{i_k+2j})$ for $j = 1, 2, \ldots, \lfloor \frac{s}{2} \rfloor - 1$. Then, line [3] links w_{i_k+1} with w_{i_k+2}, and node w_{i_k+s-1} with w_{i_k+s} completing the cycle $w^k_{i_k+1}, w^k_{i_k+2}, \ldots, w^k_{i_k+s}, w_{i_k+1}$.

To see that the time complexity of Criss-Cross is $\mathcal{O}(|V_k|)$ observe that no more than $2s$ edges are added in I_k in every iteration of lines [2]-[4]. Also, identifying the string $C^k_{i_k+1}, C^k_{i_k+2}, \ldots, C^k_{i_k+s}$ of subcycles takes $\mathcal{O}(s)$ time. Therefore, merging all of $C^k_{i_k+1}, C^k_{i_k+2}, \ldots, C^k_{i_k+s}$ takes $\mathcal{O}(|V_k|)$ time. This completes the proof of the theorem. $\square$

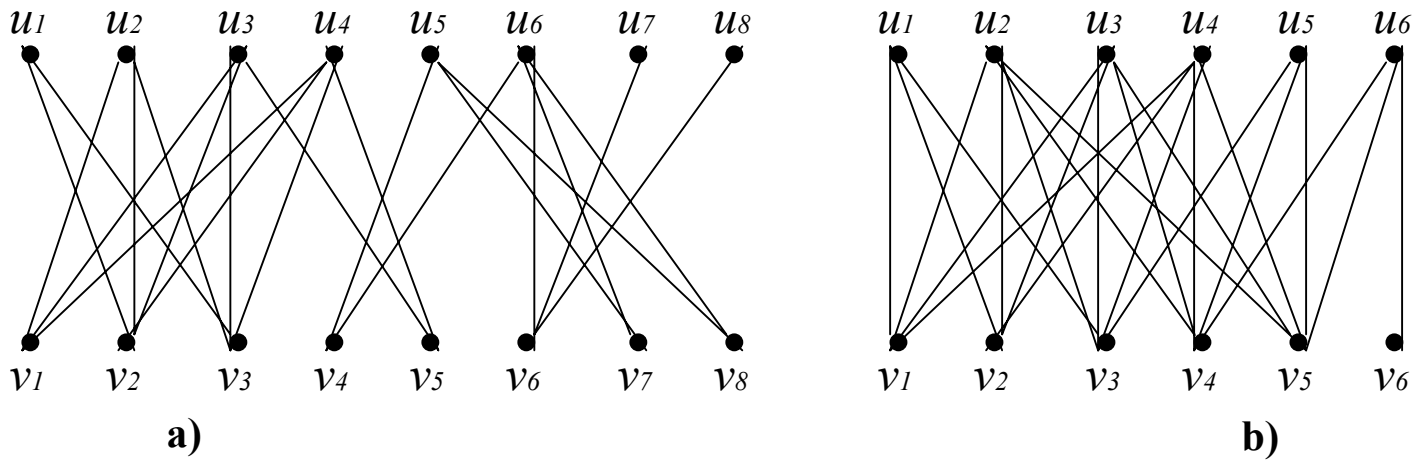

Figure 1: Examples of beam and neighboring-beam bipartite graphs

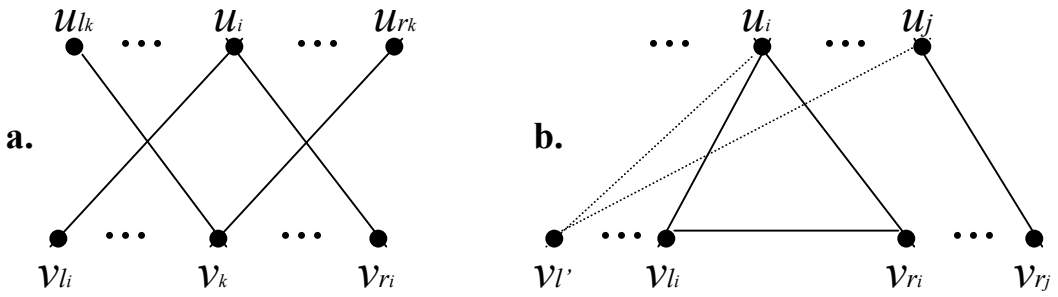

Figure 2

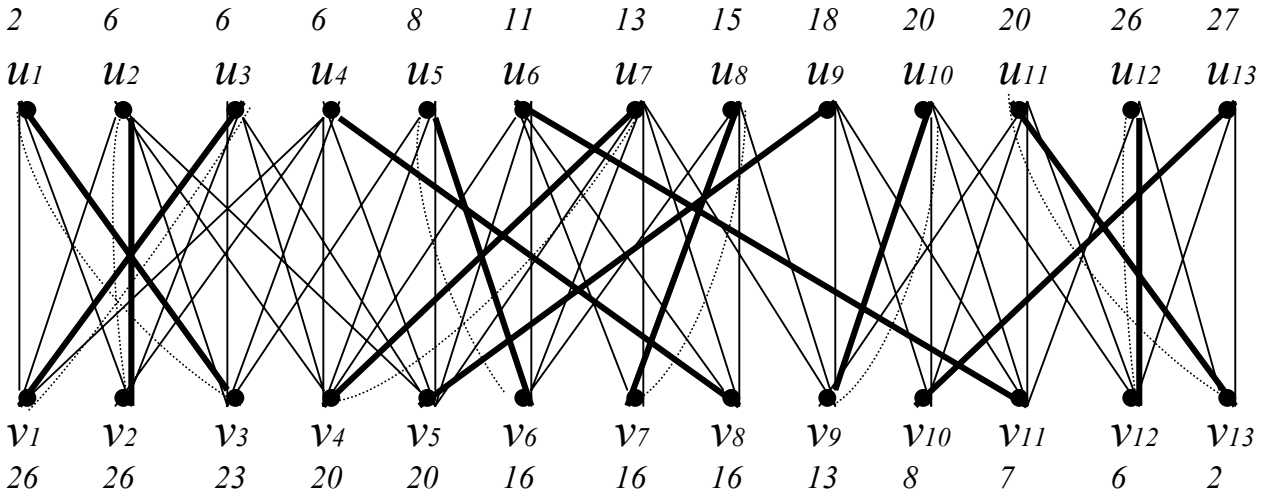

Figure 3: An example problem

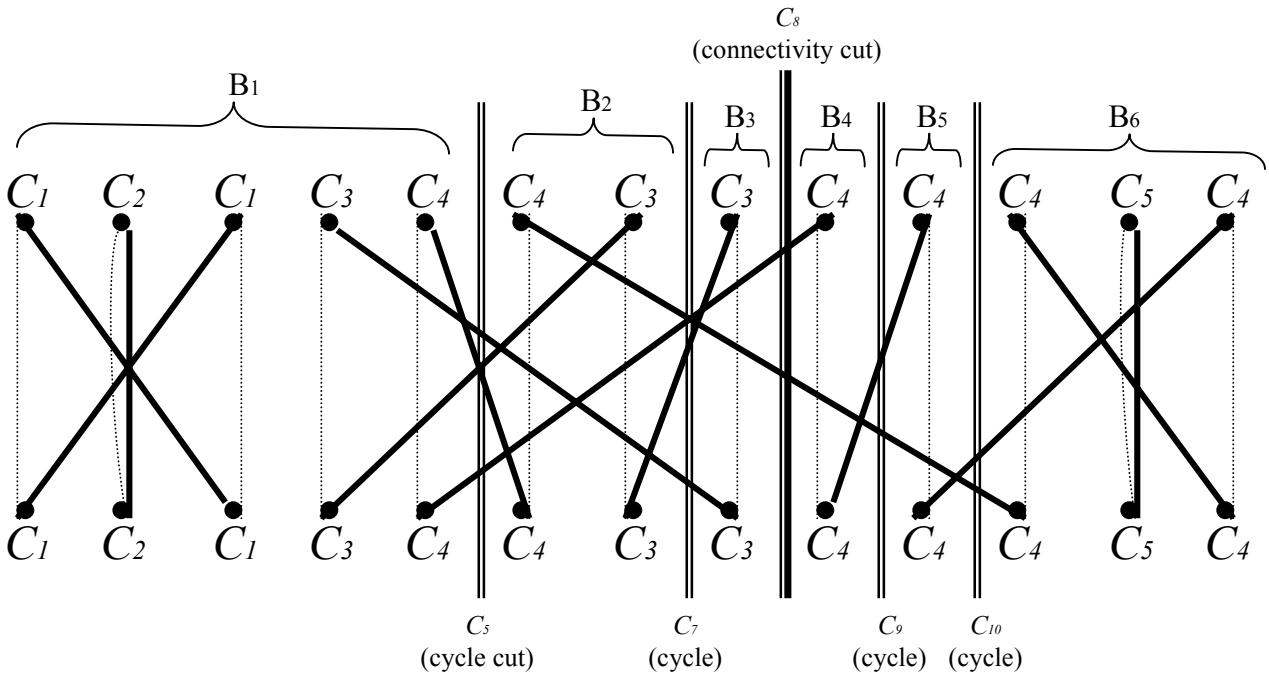

Figure 4: Sub-cycles of I_0UM, cycle cuts, and connectivity cuts

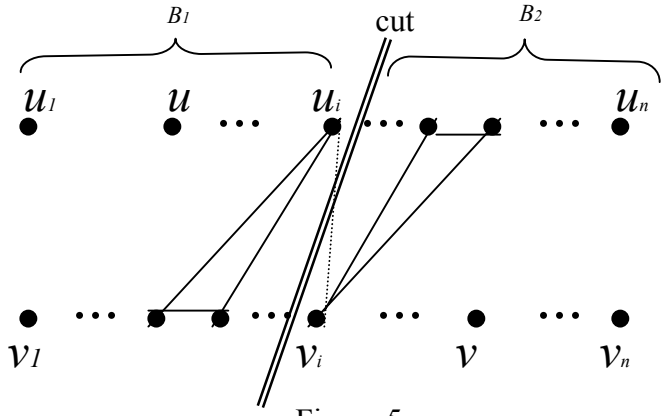

Figure 5

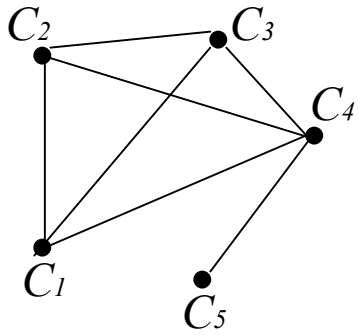

Figure 6: The cycle graph

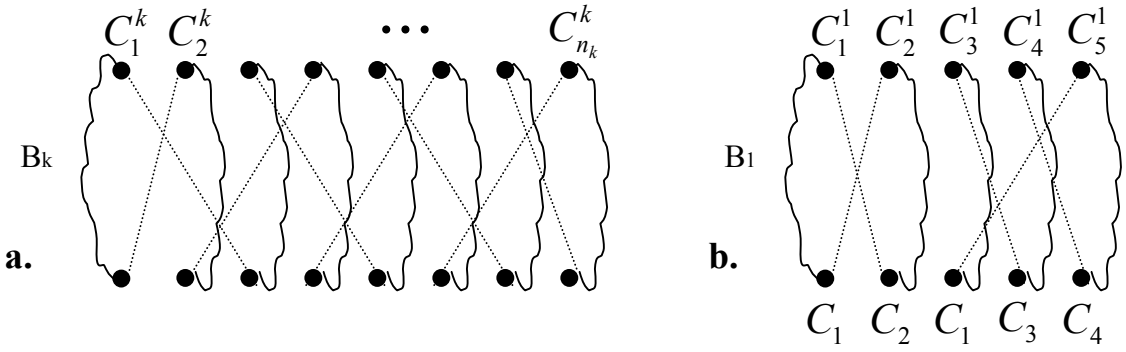

Figure 7: Matchings produced by the Criss-Cross Algorithm

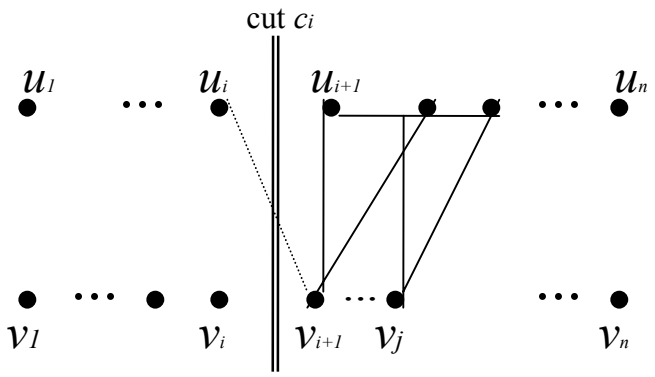

Figure 8

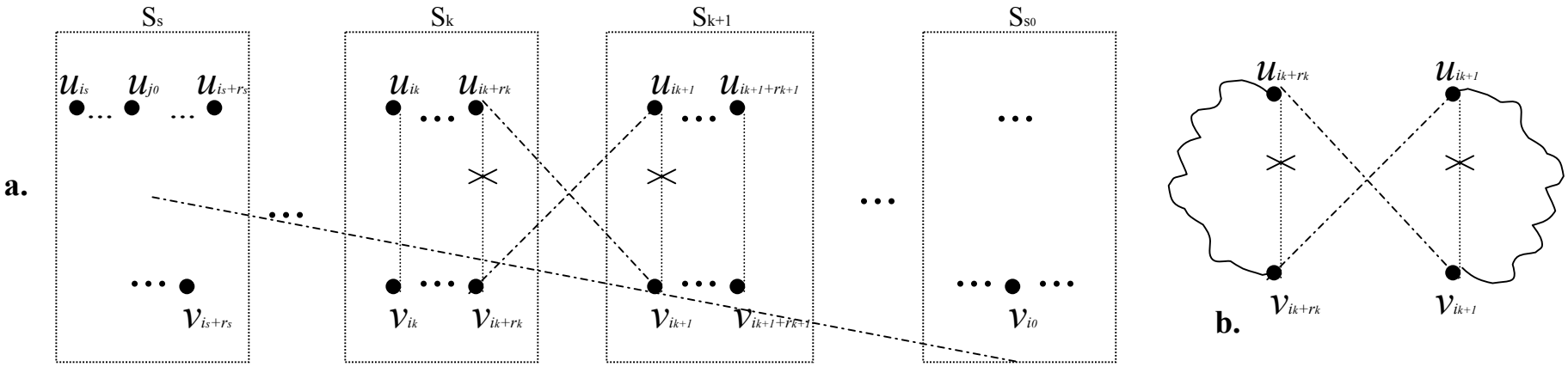

Figure 9

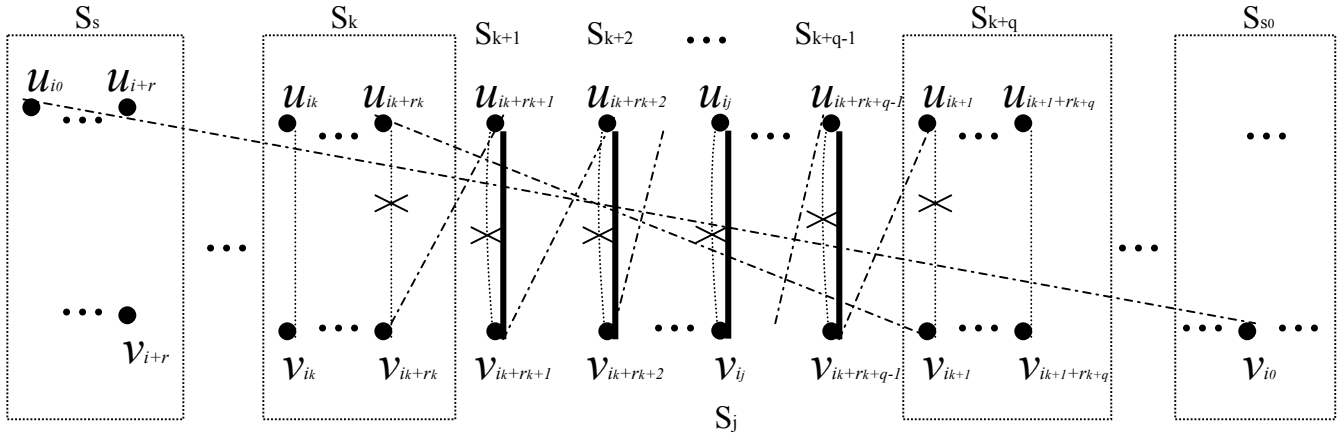

Figure 10