Refutation of Aslam's Proof that $NP = P$

Frank Ferraro Garrett Hall Andrew Wood

{fferraro, ghall3, awood8}@u.rochester.edu

Department of Computer Science
University of Rochester
Rochester, NY 14627

May 14, 2009

Abstract

Aslam presents an algorithm he claims will count the number of perfect matchings in any incomplete bipartite graph with an algorithm in the function-computing version of NC, which is itself a subset of FP. Counting perfect matchings is known to be $\#P$-complete; therefore if Aslam's algorithm is correct, then $NP = P$. However, we show that Aslam's algorithm does not correctly count the number of perfect matchings and offer an incomplete bipartite graph as a concrete counter-example.

1 Introduction

We provide preliminary definitions from Aslam's paper, which are necessary to understanding Aslam's algorithm and what it purportedly proves. After presenting these definitions, claims and examples, we look at an overview of the algorithm and why it purports $NP = P$. In Section 2, we refute some major arguments of his paper which, under his current construction, invalidate his current claim that $NP = P$. Then in Section 3, we provide a sound counter-example which demonstrates his algorithm does not correctly enumerate all perfect matchings in any bipartite graph. In order to avoid confusion, our definitions, theorems, and lemmas are labeled contiguously, without regard to the section in which they appear. Unless noted otherwise, it can be assumed that all other theorems and lemmas are from Aslam's paper [1].

We would also like to note that we are critiquing version 9 of Aslam's paper. Although at time of writing this critique, Aslam has released two additional versions, 10 and 11, both of those versions are simply summaries of version 9. As a result, they rely heavily on the claims made in version 9 and thus it is sufficient to analyze version 9. Whenever we cite Aslam as [1], we refer to version 9 of his paper.

Aslam represents perfect matchings in bipartite graphs as permutations. These permutations are elements of the symmetric group S_n, the group of all permutations of n elements. A review of general group theory can be found in any introductory abstract algebra book such as [2].

Perfect Matchings as Permutations

A perfect matching in a bipartite graph BG_n with vertices $V \cup W$ is a set of edges represented as

$$\bigcup_{i=1}^{n} ij,$$

where each ij represents the edge $v_i w_j$ in BG_n with each $w_j \in W$ and $v_i \in V$ occuring exactly once and $|V| = |W| = n$. We can also represent a perfect matching as a permutation $\pi = (a_1, a_2, \ldots, a_n)$, $1 \leq a_r \leq n$ for all r; in other words, every element in the permutation cycle is can be represented as number between 1 and n, inclusive. Letting $a_r^\pi = a_{r+1}$ and $a_n^\pi = a_1$, the perfect matching corresponding to π is

$$E(\pi) = \bigcup_{i=1}^{n} ii^\pi.$$

For instance, the permutation $(2, 3, 1)$ corresponds to the perfect matching $\{12, 23, 31\}$, and $(2, 3, 1, 5, 4)$ corresponds to $\{15, 23, 31, 42, 54\}$. Futhermore we can decompose each permutation π into a product of transpositions $\pi = \psi_n\psi_{n-1}\dots\psi_1$ where ψ_i is the transposition (i,k) and $i \leq k \leq n$. From the previous example, $(2,3,1,5,4) = (5,5)(4,5)(3,5)(2,3)(1,5)$. Note that every unique $\psi_n\psi_{n-1}\dots\psi_1$ represents a unique permutation, i.e., a unique perfect matching for all $n!$ perfect matchings in a complete bipartite graph.

Perfect Matchings from the Generating Graph $\Gamma(n)$

A generating graph $\Gamma(n)$ is a DAG (directed acyclic graph) defined by Aslam to have $O(n)$ vertices called *nodes*. Each node contains a unique pair of edges (ik, ji) such that $1 \leq i < k, j \leq n$ or $1 \leq i = j = k \leq n$. The graph $\Gamma(n)$ is designed so that traversal of special paths called complete valid multiplication paths, *CVMP*s [1, Definition 4.33], will enumerate every $n!$ perfect matching in a bipartite graph with $2n$ vertices. There are two ways to find the corresponding perfect matching given a *CVMP*. If $p = a_1a_2...a_n$ is a *CVMP* in $\Gamma(n)$, then one way to find the perfect matching is from the permutation π given by

$$\pi = \psi(a_n)\psi(a_{n-1})\dots\psi(a_1).$$

where $\psi(a_i) = \psi(ik, ji) = (i,k)$. The second way to find a perfect matching is through the union of all edge pairs in each a_i without the surplus edges (SE), which will be jk from each edge a_ia_j in p, which written formally is

$$\begin{aligned} E(p) &= EP(p) - (SE(p) \cap EP(p)) \\ &= \left(\bigcup_{a_i \in p} EP(a_i)\right) - \left(\left(\bigcup_{a_i, a_j \in p} SE(a_ia_j)\right) \bigcap \left(\bigcup_{a_i \in p} EP(a_i)\right)\right) \end{aligned}$$

where $EP(ik, ji) = \{ik, ji\}$ and $SE((ik, ji)(jk, qj)) = \{jk\}$, $i < k, j < q$ or $i < k = j = q$. As an example, consider the following *CVMP* from the graph $\Gamma(9)$ in Figure 1:

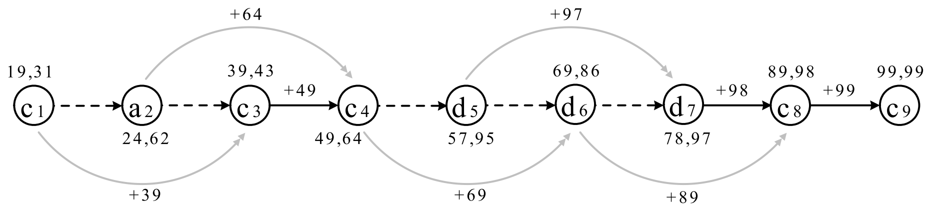

Figure 1: A subgraph of $\Gamma(9)$ containing a *CVMP* along the dark single-arrowed edges. Dotted edges are refered to as *S-edges*, solid edges are *R-edges*, and the edges between nonadjacent nodes with double-arrows are *jump edges*. Each node is labeled with a pair of edges (ik, ji). Surplus edges along the *CVMP* are indicated with a $+jk$ above them.

A *CVMP* can only be along S-edges or R-edges which are not jump edges, so the *CVMP* pictured above is $p = c_1a_2c_3c_4d_5d_6d_7c_8c_9$. For each node we can easily find ψ_i, e.g. $\psi(c_1) = \psi(19, 31) = (19)$, giving the permutation

$$\begin{aligned}\pi &= (99)(89)(78)(69)(57)(49)(39)(24)(19)\\ &= (1, 9, 5, 7, 8, 6, 2, 4, 3).\end{aligned}$$

We can also easily compute $E(p)$ from the set of all edges in all the nodes minus all of the surplus edges as

$$\begin{aligned}E(p) &= EP(c_1) \cup EP(a_2) \cup ... \cup EP(c_9) - SE(c_1) \cup SE(a_2) \cup ... \cup SE(c_9)\\ &= \{19, 31, 24, 62, ..., 98, 99\} - \{39, 64, ..., 99\}\\ &= \{19, 24, 31, 43, 57, 62, 78, 86, 95\}.\end{aligned}$$

Clearly, $E(\pi) = E(p)$ so both methods have arrived at the same perfect matching.

*CVMP*s in Incomplete Graphs

Each of the $n!$ *CVMP*s in $\Gamma(n)$ corresponds to a unique perfect matching in a complete bipartite graph BG_n. Of course, given an incomplete bipartite graph BG'_n only a subset of all the *CVMP*s will contain edges not in BG'_n which cannot be counted as a perfect matchings. These edges not in BG'_n

are refered to as the edge requirements (*ER*) of a *CVMP* and are formally defined for a *CVMP* p as

$$\begin{aligned} ER(p) &= \left(\bigcup_{a_i \in p} ER(a_i)\right) - \\ &\qquad \left(\left(\bigcup_{a_i, a_j \in p} SE(a_i a_j)\right) \bigcap \left(\bigcup_{a_i \in p} ER(a_i)\right)\right), \end{aligned}$$

where $ER(a_i) = \{e | e \in EP(a_i), e \notin BG'_n\}$. Clearly, the *ER* of a *CVMP* will be the empty set if and only if the *CVMP* corresponds to a valid perfect matching in BG'_n. In other words, $ER(p) = \emptyset \iff E(p) \subseteq BG'_n$. Performing a *valid enumeration* of *CVMP*s in $\Gamma(n)$ given a corresponding BG'_n, i.e. enumerating every *CVMP* with $ER(p) = \emptyset$, is equivalent to counting every valid perfect matching.

Obviously, naive valid enumeration of the $n!$ possible *CVMP*s cannot be done in polynomial time. However, if Aslam's algorithm is correct, the special properties of $\Gamma(n)$ allow it to be reduced to sets of subpaths which can be joined while preserving the *ER* of all *CVMP*s.

Algorithm Overview

A generating graph $\Gamma(n)$ contains $O(n^3)$ nodes [1, Property 4.21] and can be created in polynomial time. $VMPSet(a_i, a_j)$ is a data structure representing a subset of *VMP*s between nodes a_i and a_j that have the same *ER*. Within the algorithm, all of the $VMPSet(a_i, a_{i+1})$ are initializated first and all possible $VMPSet(a_i, a_j)$ are stored in a matrix.

During each iteration, the algorithm performs two reduction operations: adding and multiplying *VMPSet*s. If one multiplies two *VMPSet*s, $VMPSet(a, b)$ and $VMPSet(b, c)$, effectively doubles the length of the paths they represent to get $VMPSet(a, c) = VMPSet(a, b) \times VMPSet(b, c)$, and increases the number of *VMP*s in the new $VMPSet(a, c)$ to

$$|VMPSet(a, c)| = |VMPSet(a, b)| \times |VMPSet(b, c)|.$$

When two sets $VMPSet(a, b)$ and $VMPSet'(a, b)$ can be combined so as to satisfy conditions for multiplication they are added together, and the number of *VMP* in the new $VMPSet''(a, b) = VMPSet(a, b) \cup VMPSet'(a, b)$ is

$$|\,\mathit{VMPSet}''(a,b)| = |\,\mathit{VMPSet}(a,b)| + |\,\mathit{VMPSet}'(a,b)|.$$

Since every iteration doubles the length of *VMP*s represented in the *VMPSet*s, after $O(\log(n))$ iterations the entire generating graph $\Gamma(n)$ is reduced to several disjoint sets of $\mathit{VMPSet}(1,n)$ representing all *CVMP*s. Summation over all $|\,\mathit{VMPSet}(1,n)|$ that have $ER = \emptyset$ then gives the total number *CVMP*s and likewise perfect matchings. The summation step is what necessitates keeping *ER* the same for all *VMPSet*s.

Since the problem of counting perfect matchings in any bipartite graph is $\#P$-complete and the class of parallel algorithms running in $(\log n)^{O(1)}$ on a polynomial number of processors, which is analogous to and commonly referred to as NC, is a subset of *FP*, the algorithm Aslam purports to have will be able to solve any $\#P$ problem in polynomial time, meaning $\#P = FP$ and $NP = P$.

2 Refutation

The main flaw in Aslam's reasoning is that the *ER* of every $CVMP$ can be preserved during decomposition and subsequent reduction operations over *VMPSet*s. However, the *ER* of a *VMPSet* does not capture the *SE* of the *VMP*s it contains, and ultimately *SE* will determine *ER*. We will show that multiplication and addition of *VMPSet*s does not always give a *VMPSet* with the same *ER* for all *VMP*s. As proof this problem is inherent in all sufficiently large generating graphs we give a counter-example in Section 3.

Lemma 1. The product *VMPSet* AC found from multiplying two *VMPSet*s $A = \mathit{VMPSet}(a,b)$ and $C = \mathit{VMPSet}(b,c)$ must be a single set and contain only *VMP*s with the same *ER*.

Proof. The condition on *ER* follows from the definition of *VMPSet* and the inductive result of the algorithm itself. Since after the final iteration all *VMPSet*s with $ER = \emptyset$ will be counted, if multiplication resulted in a *VMPSet* containing some *CVMP*s with $ER = \emptyset$ (valid perfect matchings) and some with $ER \neq \emptyset$ (invalid), then summation would result in an incorrect number of perfect matchings. □

By Aslam's definition of multiplication, AC must be a single *VMPSet*. We note, however, even if we allowed multiplication to produce more than one *VMPSet*, the number of sets produced would have to be constant with respect to $|A| \times |C|$, since multiplication is what allows the fast enumeration in $O(\log(n))$ iterations.

Theorem 2. *The conditions for multiplying VMPSets $A \times C = AC$ given in Lemma 5.8 and Lemma 5.9 of Aslam's proof are insufficient conditions for multiplication. Lemma 5.9 is not a necessary condition for multiplication.*

Proof. Lemma 5.8 states that $\forall p \in A$ and $\forall q \in C$, p must multiply (form a *VMP*) with q. Obviously, this is a necessary condition since we are considering all pq to be *VMP*s in AC. Lemma 5.9 states that every node covered by a *VMP* in C from the same partition must have the same ER. Note that partition number is equal to the depth in $\Gamma(n)$, so for every $x_i, x'_i \in q$ we must have $ER(x_i) = ER(x'_i)$. The proof of this lemma is conspicuously omitted and we found it to be incorrect. Assume the only nodes with unequal ER in C are $x_i \in q$ and $x'_i \in q'$, all the *VMP*s in A have the same SE, and let $ER(x_i) = e$ and $ER(x'_i) = \emptyset$. If $e \notin SE(A)$ the resultant AC will not have every *VMP* with the same ER violating the condition laid out in Lemma 1. However, if $e \in SE(A)$ then $ER(pq') = ER(pq) = \emptyset$ and AC is a *VMPSet* with valid *VMP*s all with $ER = \emptyset$, so Lemma 5.9 is not a necessary condition for multiplication. Note we use the following definitions:

$$\begin{aligned} e \in SE(A) &\iff (\forall p \in A)\ [e \in SE(p)]\,. \\ e \in ER(A) &\iff (\forall p \in A)\ [e \in ER(p)]\,. \end{aligned}$$

Now we will show these lemmas are insufficient conditions for multiplication. Let p and p' be *VMP*s in A such that $e \in SE(p)$ and $e \notin SE(p')$. Let the node x_i covered by all *VMP*s in C have $ER = e$ and all other nodes have $ER = \emptyset$. Note that C satisfies Lemma 5.9 since every *VMP* in C covers x_i there is no node x'_i in C and every other node has the same $ER = \emptyset$. Proof that A and C can satisfy Lemma 5.8 (every path through a and c is a *VMP*) relies on properties of the generating graph $\Gamma(n)$ itself and so we will demonstrate that in our counter-example in Section 3. For now we assume Lemma 5.8 is satisfied.

After multiplication of A and C, we have $ER(pq) = \emptyset$ and $ER(p'q) = e$ for every $q \in C$. The resultant *VMPSet* AC does not contain *VMP*s with the same ER, violating the conditions for multiplication set forth in Lemma 1.

We see this is because the *VMPSet* representation does not capture cases where $SE(p) \neq SE(p')$. □

Further Discussion

Although Aslam does not give the sufficient conditions for performing multiplication to reduce all *VMPSet*s in $O(\log(n))$ iterations, if his proof is valid until that point, then proving whether such conditions exist or do not exist may be equivalent to proving whether $P = NP$. In the rest of the section we explore why performing multiplication while preserving *ER* over *VMPSet*s is difficult. In the following lemma we show why the various *SE*s of all the paths in a *VMPSet* are significant.

Lemma 3. There are at least $(n-1)!$ *CVMP*s where $SE(p) \subsetneq EP(p)$ for every *CVMP* p. In these *CVMP*s, if $a_i \in p$ and all edges $SE(a_i)$ are not present in the bipartite graph BG'_n, then changing any one node in p results in $ER(p) \neq \emptyset$.

Proof. Each *CVMP* p in $\Gamma(n)$ is n nodes long. For simplicity, we only consider the $(n-1)!$ cases where p is composed of non-identity nodes (except the last node). Recall

$$\begin{aligned} E(p) &= EP(p) - (SE(p) \cap EP(p)) \\ &= \left(\bigcup_{a_i \in p} EP(a_i) \right) - \left(\left(\bigcup_{a_i, a_j \in p} SE(a_i a_j) \right) \bigcap \left(\bigcup_{a_i \in p} EP(a_i) \right) \right). \end{aligned}$$

Every node $a_i = (ik, ji)$, $1 \leq i < j, k < n$ contributes two unique edges ik, ji to $EP(p)$ and the last node contributes one so that $|EP(p)| = 2n-1$. Every node except the last contributes an edge to $SE(p)$ so $|SE(p)| = n-1$. Since $|E(p)| = n$, every edge in $SE(p)$ must be unique in p, eliminate one edge from $EP(p)$, and $SE(p) \subsetneq EP(p)$. Note that each $a_i = (ik, ji) \in \Gamma(n)$ is unique, so for all nodes $SE(a_i) \neq SE(b_i)$. Therefore if the edge $SE(a_i)$ is not present in the bipartite graph BG'_n, then changing the node $a_i \in p$ to any $b_i \in \Gamma(n)$ results in an $ER(p) \neq \emptyset$, where

$$\begin{aligned} ER(p) &= \left(\bigcup_{a_i \in p} ER(a_i)\right) - \\ &\left(\left(\bigcup_{a_i, a_j \in p} SE(a_i a_j)\right) \bigcap \left(\bigcup_{a_i \in p} ER(a_i)\right)\right). \end{aligned}$$

It follows there are at least $(n-1)!$ *CVMP*s with corresponding BG'_ns in which satisfying $ER = \emptyset$ may be dependent on the *SE* of every node in p. If we are concerned with satisfying the conditions on multiplication outlined in Lemma 1, we can keep all *VMP*s with differing *SE* in separate *VMPSet*s, but this leads to a large number of *VMPSet*s. □

Lemma 4. The number of *VMPSet*s with the same *SE* from partition 1 to i is at least $\binom{n}{i}$.

Proof. By definition, the generating graph $\Gamma(n)$ contains $n!/(n-i)!$ *VMP*s from partitions 1 to i. Note this is necessary for $\Gamma(n)$ to be able to enumerate $n!$ perfect matchings. Since all these *VMP*s are unique, for any two *VMP*s p_1 and p_2, each will contain at least one different node $x_i \in p_1$ and $y_i \in p_2$, with $SE(a_i) \neq SE(b_i)$. Consequently, no two *VMP*s have surplus edges occuring in exactly the same order, so the maximum size of any *VMPSet* with the same set of *SE* edges from 1 to i will be $i!$ which is the number of permutations of *SE* edges from i nodes. To get a lower bound on the number of *VMPSet*s with the same *SE* we divide the number of *VMP*s by the upper bound on set size:

$$\frac{n!}{(n-i)!i!} = \binom{n}{i}.$$ □

3 Counter-example

We present γ, a subgraph of every $\Gamma(n)$ with $n \geq 9$, which enumerates five perfect matchings. We then offer an incomplete bipartite graph BG'_n and show that Aslam's algorithm will incorrectly count some number of perfect matchings in BG'_n using γ. This serves as an example that the graph

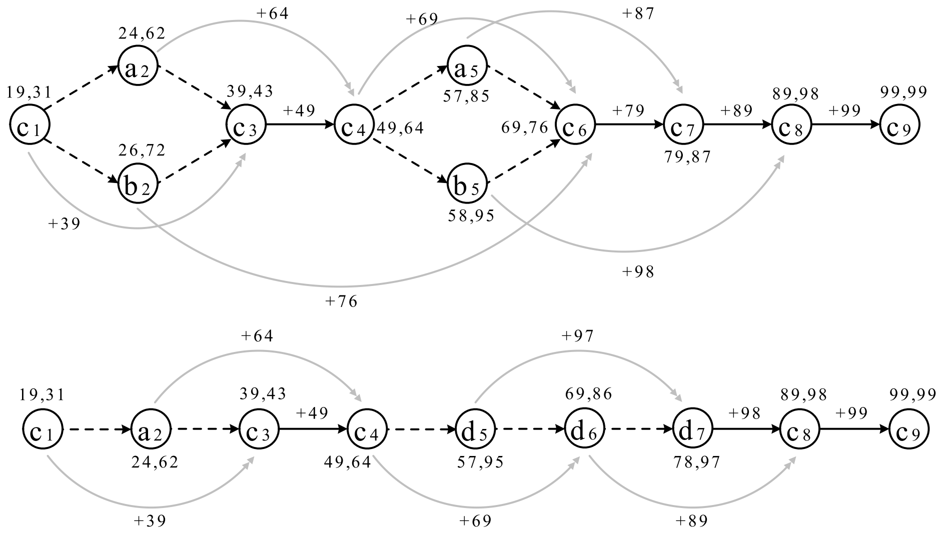

Figure 2: The above figure represents graph γ. Double-arrowed solid-lines are jump edges, single-arrowed solid-lines are R-edges, and dotted-lines are S-edges.

described in Theorem 2 can be realized in $\Gamma(n)$, satisfying Lemma 5.8 of Aslam's paper.

Graphical Representation of the Example In Figure 2 we provide a graphical representation of the counter-example.

Lemma 5. γ, in Figure 2, represents a subgraph of every $\Gamma(n)$, where $n \geq 9$.

Proof. The generating graph $\Gamma(n)$ is defined over all $1 \leq i < k \leq n$, where vertices are $a_i \in g(i)$, R-edges are $a_i a_j$ such that $|a_i R a_j| = 1$, jump edges are R-edges such that $j \neq i+1$, and S-edges are $a_i a_{i+1}$ such that $a_i S a_{i+1}$ [1, Definition 4.12]. From this definition we should note that the generating graph $\Gamma(n)$ is a subgraph of $\Gamma(n+1)$.

Now we will show that γ from is a valid subgraph of $\Gamma(9)$. It can be easily verified every node $a_i = (ik_1, k_2 i)$ in γ has either $k_1, k_2 > i$ or $i = k_1 = k_2$, so $a_i \in g(i)$ is true for all nodes. Any relation R in γ between nodes a_i and $a_j = (jt_1, t_2j)$ appears iff $t_1 = k_2$, $t_2 = k_1$, and $i < j$, which satisfies $|a_iRa_j| = 1$. The relation appears as an R-edge if $j = i + 1$, otherwise it appears as a jump-edge. Any S-edge appears if and only if $j = i + 1$, and either $k_1, t_2 < k_2, t_1$, $k_2, t_1 < k_1, t_2$, or $k_1 = k_2 < t_1, t_2$. The disjointness of a_iSa_j given these conditions is made clear from Remark 4.5 [1] since no R-cycle $C_{a_ia_j}$can be constructed which has a strictly increasing or decreasing traversal.

Since for all nodes in γ we have $a_i \in g(i)$, and all edges appear if and only if they satisfy their respective definitions (at least for the nodes appearing in γ), γ is a subgraph of $\Gamma(9)$ and every $\Gamma(n)$, where $n \geq 9$. □

CVMPS in $\boldsymbol{\gamma}$. Let $P(a, b)$ denote the path in γ starting at c_1 going through a, b and ending at c_9. Then γ only contains five *CVMP*s: $p_{aa} = P(a_2, a_5)$, $p_{ab} = P(a_2, b_5)$, $p_{ba} = P(b_2, a_5)$, $p_{bb} = P(b_2, b_5)$, and $p_{ad} = P(a_2, d_5)$ which correspond to five unique perfect matchings.

Proof. By definition the permutation $\pi(p)$ that realizes the perfect matching corresponding to *CVMP* p is given by $\psi_n \dots \psi_1$, where for every node $x_i = (ik, ji) \in p$, $\psi_i = (ik)$. The set of edges in the perfect matching corresponding to *CVMP* p is given by $E(p) = (\bigcup\{e|e \in x_i\}) - (\bigcup SE(x_i))$, where $SE(x_i)$ gives the surplus edges in p.

It is trival to verify $\pi(p)$ and $E(p)$ are consistent:

$$\pi(p_{aa}) = (99)(89)(79)(69)(57)(49)(39)(24)(19)$$
$$E(p_{aa}) = \{19, 24, 31, 43, 57, 62, 76, 85, 98\}$$

$$\pi(p_{ab}) = (99)(89)(79)(69)(58)(49)(39)(24)(19)$$
$$E(p_{ab}) = \{19, 24, 31, 43, 58, 62, 76, 87, 95\}$$

$$\pi(p_{ba}) = (99)(89)(79)(69)(57)(49)(39)(26)(19)$$
$$E(p_{ba}) = \{19, 26, 31, 43, 57, 64, 72, 85, 98\}$$

$$\pi(p_{bb}) = (99)(89)(79)(69)(58)(49)(39)(26)(19)$$
$$E(p_{bb}) = \{19, 26, 31, 43, 58, 64, 72, 87, 95\}$$

$$\pi(p_{ad}) = (99)(89)(78)(69)(57)(49)(39)(24)(19)$$
$$E(p_{ad}) = \{19, 24, 31, 43, 57, 62, 78, 86, 95\}.$$

Note that $P(b_2, d_5)$ is not a *CVMP* because it does not contain the mdag $MDG(b_2, c_3, c_6)$.

Choosing BG' The initialization of PTM, which Aslam defines as the matrix containing all *VMPSet*s, is vague regarding how it incorporates information from the adjacency matrix BGX of the bipartite graph, considering there is no clear relation between removal of S-edges or R-edges, and edges from the bipartite graph. To allow this ambiguity, we have chosen a BG' so that no edge might be removed from γ without losing a perfect matching. Formally,

$$BG' = \left(\bigcup E(p)\right) - \{76\}$$

In other words, all *CVMP*s in γ are valid matchings with $ER(p) = \emptyset$ except for those containing the edge $\{76\}$ which will have $ER(p) = \{76\}$. Clearly only p_{ba}, p_{bb}, and p_{ad} are *CVMP*s which represent valid perfect matchings so the result of enumerating with γ should be 3. In addition, these *CVMP*s cover every edge of γ, so no edge may be removed without losing at least one *CVMP*.

Iterations of Add and Join on γ

During the first iteration of the algorithm, the *VMPSet*s $VMPSetA(c_1, c_3)$, $VMPSetB(c_1, c_3)$, $VMPSetA(c_4, c_6)$, $VMPSetB(c_4, c_6)$, and *VMPSet*s over all other pairs of nodes are created. In the next iteration two additions between the corresponding A and B sets occur: $VMPSetA(c_1, c_3)$ and $VMPSetB(c_1, c_3)$ are added together, and $VMPSetA(c_4, c_6)$ and $VMPSetB(c_4, c_6)$ are added. So we have

$$\begin{aligned} VMPSet(c_4, c_6) &= VMPSetA(c_4, c_6) + VMPSetB(c_4, c_6) \\ VMPSet(c_1, c_3) &= VMPSetA(c_1, c_3) + VMPSetB(c_1, c_3). \end{aligned}$$

Crucially, $|VMPSet(c_4, c_6)| = |VMPSet(c_1, c_3)| = 2$ and the set of surplus edges SE of $VMPSet(c_4, c_6)$ equals the ER of $VMPSet(c_1, c_3)$ which is $\{76\}$.

Once these two sets are multiplied to get 4 and the additional *VMPSet* representing p_{ad} is counted, the number of perfect matchings the algorithm will return will be 2 more than the actual number of perfect matchings because the *SE* of the *VMP*s in $VMPSet(c_4, c_6)$ was combined. □

Thus, as the above example demonstrates, Aslam's algorithm does not correctly enumerate all perfect matchings for all cases. Therefore, his current proof that $\#P \subseteq FP$ (and hence that $P = NP$ and the polynomial-time hierarchy collapses) does not validly establish that claim.

4 Acknowledgements

This work was completed in partial fulfillment of the requirements for an Honors Bachelor of Science Degree in Computer Science from the Department of Computer Science at the University of Rochester, in Rochester, NY, USA. This paper was also written as the Honors Project for the course CSC200H during the Spring 2009 semester. We would like to thank Professor Lane A. Hemaspaandra, the course T.A. Adam Sadilek and others in the community for their feedback, support and suggestions.

References

[1] Aslam, Javaid. "The Collapse of the Polynomial Hierarchy: $NP = P$." arXiv.org. 9 Mar. 2009. <http://arxiv.org/pdf/0812.1385v9>.

[2] Fraleigh, John B. *A First Course in Abstract Algebra*. 7th ed. New York: Addison Wesley. 2003.