Matching Randomly in Parallel

Shang-Hua Teng

May 1989
CMU-CS-89-149

Abstract

In this paper, the parallel complexity of the *Random Matching Problem*–a problem of generating a perfect matching in a bipartite graph uniformly in random–is considered. We show that the only known polynomial time random matching algorithm, due to Broder, Jerrum and Sinclair, can not be parallelized in NC, unless $NC = P$. The reduction is from the *Lexical First Maximal Independent Set Problem*. This result shows many interesting structural properties between matching and lexical first maximal independent sets. It also leaves many interesting and important open questions. We also show that any polynomial time scheme (NC scheme) for the *Random Maximal Independent Set Problem* implies $NP = RP$ ($NP = RNC$). This provides another example that the problem of uniform random generation is harder than the corresponding construction problem.

This work was supported in part by National Science Foundation Grant CCR-87-13489.

The views and conclusions contained in this paper are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the National Science Foundation or the US Government.

Matching Randomly in Parallel

Shang-Hua Teng*

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

May 1989

Abstract

In this paper, the parallel complexity of the *Random Matching Problem*–a problem of generating a perfect matching in a bipartite graph uniformly in random–is considered. We show that the only known polynomial time random matching algorithm, due to Broder, Jerrum and Sinclair, can not be parallelized in NC, unless $NC = P$. The reduction is from the *Lexical First Maximal Independent Set Problem*. This result shows many interesting structural properties between matching and lexical first maximal independent sets. It also leaves many interesting and important open questions. We also show that any polynomial time scheme (NC scheme) for the *Random Maximal Independent Set Problem* implies $NP = RP$ ($NP = RNC$). This provides another example that the problem of uniform random generation is harder than the corresponding construction problem.

1 Introduction

The *Matching Problem* is a natural and important problem in computing theory. Like the *Maximal Independent Set* problem, it has been used as an important subroutine in several computational problems.

A set of edges, M, of a graph $G(V, E)$ with no self-loops, is a *perfect matching* if every vertex is incident to exactly one edge of M. The problem of computing a perfect matching in a *bipartite* graph is also known as the *marriage problem*.

Generating a perfect matching in a general graph is first solved in polynomial time by Edmonds [6] and improved by Even and Kariv [7]. By reducing the bipartite matching problem to the maximum flow problem, Hopcroft and Karp first present a simple $O(|V|^{1/2} \cdot E)$ time algorithm. On the other hand, Karp, Upfal and Wigderson [12], Karloff [11],

*This work was supported in part by National Science Foundation under Grant CCR-87-13489.

and Mulmuley, Vazirani and Vazirani [17] showed that parallel RNC algorithms exist for computing a perfect (maximum) matching in general graphs.

However, the *Random Matching Problem*– a problem of generating a perfect matching in a bipartite graph uniformly in random–had been left open for a many years. Until very recently, Border and Jerrum and Sinclair [3,9] showed how to computing a random matching in polynomial time.

In this paper, the parallel complexity of the Random Matching Problem is considered. The efficient generation of a random perfect matching plays a critical role in efficiently approximating the number of perfect matching in a bipartite graph, or equivalent the permanent of a 0–1 matrix [3,9,18].

We show that the only known polynomial time random matching algorithm, due to Broder, Jerrum, and Sinclair, can not be parallelized in NC, unless $NC = P$. The reduction is from the *Lexical First Maximal Independent Set Problem.* Some interesting structural properties between matching and the lexical first maximal independent sets are presented. Our results provides some evidence to the following conjectures.

Conjecture 1.1

1. *There is no RNC algorithm to generate a perfect matching in a bipartite graph uniformly in random unless $P = RNC$.*

2. *There is no NC algorithm to approximate permanent of a 0–1 matrix unless $P = RNC$.*

We also show that any polynomial time scheme (NC scheme) for the Random Maximal Independent Set Problem implies $NP = RP$ ($NP = RNC$). This provides another example that the problem of uniform random generation is harder than the corresponding construction problem [10].

2 Definitions and Notations

2.1 Perfect Matchings

Let $G(V, E)$ be a undirected graph where $V = \{v_1, \ldots, v_{|V|}\}$ is the *vertex set* and E is the *edge set.* G is ***bipartite*** if V can be partitioned into X and Y such that each edge has one end vertex in X and on in Y.

A set of edges, M, of a graph $G(V, E)$ with no self-loops, is called a *matching* if every vertex incident to at most one edge of M. A matching M is a *perfect matching* if every vertex incident to exactly one edge of M.

2.2 The Lexical First Maximal Independent Set Problem

A set of vertices $I \in V$ of a graph $G(V, E)$ is an *independent set* if there is no edge between two vertices of I. An independent set I is *maximal* if for each vertex $v \in V - I$, there is $u \in I$ such that $(v, u) \in E$.

For each permutation $\pi \in S_{|V|}$, define $order(v_i) = \pi(i)$. The *Lexical First Maximal Independent Set* of a graph $G(V, E)$ and a permutation π, denoted by $LFMIS_{G,\pi}$, is an independent set such that

1. $v_{\pi^{-1}(1)} \in LFMIS_{\pi,G}$;
2. for each $v_i \notin LFMIS_{G,\pi}$, there is $j : \pi^{-1}(j) < \pi^{-1}(i)$, such that $(v_i, v_j) \in E$.

Problem 2.1 (Lexical First Maximal Independent Set Problem) *Given a graph $G(V, E)$ and a permutation π, compute the lexical first maximal independent set of G.*

Without loss of the generality, it is assumed that $\pi(i) = i$. The Lexical First Maximal Independent Set Problem can be solved by the following simple greedy algorithm in linear time.

Algorithm GREEDY LFMIS
$I = \emptyset$, $V' = V$;;
while $V' \neq \emptyset$
    $I = I \cup \min(V')$;;
    $V' = V - N_G^+(\min(V'))$;;
output I.

where $\min(V') = v_{\min\{i|v_i \in V'\}}$ and $N_G^+(v)$ is the set of neighbors of v in G of order larger than that of v.

Lemma 2.1 (Cook [4]) *The Lexical First Maximal Independent Set Problem is log-space complete for P.*

2.3 Notations

For each graph $G(E, V)$,

- $Ed(G) = E$, the set of edges of a graph G;
- $Vr(G) = V$, the set vertices of a graph G;
- $\Delta_G(v)$, the degree of a vertex v in a graph G;
- $\Delta_G^-(v)$, the number of lower order neighbors of v in G;
- $N_G(v)$, $N_G^+(v)$, and $N_G^-(v)$, the sets of all neighbors, all higher order neighbors, and all lower order neighbors, respectively, of v in G;
- $$Rank_{G,v}^-(u) = \begin{cases} -1 & \text{if } u \notin N_G^-(v) \\ |\{w|w \in N_G^-(v) \ \&\ order(w) \leq order(u)\}| & \text{otherwise} \end{cases}$$

3 Random Generation Schemes

3.1 Random Generation Problems

Definition 3.1 (Random Generation Problem over Sets) *Let S be a finite set. The* **Random Generation Problem** *over S is to generate uniformly, at random, an element of S.*

The Random Generation Problem over many interesting sets has been studied [2,19,16]. The following are some examples.

- **Random Permutation Problem:** $S = S_n$, the set of all n-element permutations [16,15,13];
- **Random Factored Composite Problem:** $S = \{$factored composite numbers of length $N\}$, [2];
- **Random Labeled Tree Problem:** $S = \{$labeled trees of n vertices$\}$ [16];
- **Random Unlabeled Graph Problem:** $S = \{$unlabeled graphs of n vertices$\}$ [5];

The Random Generation Problem can also be defined over binary relations.

Definition 3.2 (Random Generation Problem over Relations [10]) *Let Σ be an alphabet and $R \subseteq \Sigma^* \times \Sigma^*$ be a binary relation. The* **Random Generation Problem** *over R is to generate uniformly, at random, a $y \in \Sigma^*$ which satisfies $R(x, y)$ for a given $x \in \Sigma^*$.*

Definition 3.3 (Random Matching Problem) *Given a bipartite graph $G(V_1, V_2, E)$, generate a perfect matching of G uniformly in random. In other words, the Random Matching Problem is a Random Generation Problem over the relation R, where $R(G, M)$ if M is a perfect matching of G.*

The following are some other interesting Random Generation Problems.

- **Random Maximal Independent Set Problem:** $R(G, I)$ if I is a maximal independent set of G;
- **Random Cycle Problem:** $R(G, C)$ if C is a cycle of G;
- **Random Topological Labeling Problem:** $R(G, \pi)$ if π is topological labeling of a DAG G;

Follows from Jerrum, Valiant, Vazirani [10], the relation in the Random Generation Problem is restricted to the following set of *p–relations* where the relation can be checked efficiently.

Definition 3.4 (P-relations [10]) *$R \subseteq \Sigma^* \times \Sigma^*$ is a* **p–relation** *if*

1. *there is a polynomial $p(n)$ such that $R(x, y) \Rightarrow |y| \leq p(|x|)$;*
2. *the predicate $R(x, y)$ can be tested in deterministic polynomial time.*

3.2 Probabilistic Turing Machines

The model of computation used in this paper is the *general probabilistic Turing machines (PTM)* [8,10]. A probabilistic Turing machine is a Turing machine [1] equipped with an output tape and having a set of distinguished coin–tossing states. In each con–tossing state, two possible transitions of machine are specified. The computation of a PTM is deterministic, except when the PTM is in a coin–tossing state, in which case the next transition is decided by the toss of a *biased coin* which is a ratio of two integers computed previously by the machine.

The PTM can be viewed as a random generator [10], with the distribution depending on the input x, and on some underlined relation R. A PTM is a *uniform generator* for a relation $R \subseteq \Sigma^* \times \Sigma^*$ iff

1. for each $x, y \in \Sigma^*$,

$$Pr(\text{given input } x,\ M \text{ outputs } y) = \begin{cases} 0 & \text{if}(x,y) \notin R \\ \frac{1}{|\{y|R(x,y)\}|} & \text{if } (x,y) \in R \end{cases}$$

2. for all inputs $x \in \Sigma^*$ such that $\{y \in \Sigma^* : R(x,y)\}$ is not empty,

$$Pr(\text{M accpets } x) \geq \frac{1}{2}.$$

A PTM M is $f(n)$ time-bounded iff, for each natural number n and for each input $x \in \Sigma^n$, every accepting computation of M halt within $f(n)$ steps. A PTM M is *polynomially time–bounded* if there exists a polynomial $p(n)$ such that M is $p(n)$ time–bounded.

In the context of parallel computation, the computation model used in this paper is *probabilistic PRAM* which is a PRAM where each processor is a PTM.

3.3 Random Generation Schemes

There are four types of *(uniform) random generation schemes* (for examples, see Miller and Teng [16]).

- **Type(0)–Scheme**: A random generation scheme is a Type(0)–Scheme for a relation R, iff for each input x, the scheme always halts successfully and each $y : R(x,y)$ is *completely unbiasedly* generated;

- **Type(1)–Scheme**: A random generation scheme is a Type(1)–Scheme for a relation R, iff for each input x, the scheme may not halt successfully but each $y : R(x,y)$ is *completely unbiasedly* generated;

- **Type(2)–Scheme**: A random generation scheme is a Type(2)–Scheme for a relation R, iff for each input x, the scheme always halts successfully and the difference of the probabilities between each pair of each $y_1, y_2 : R(x,y_1), R(x,y_2)$ is bounded by a small epsilon;

- **Type(3)–Scheme:** A random generation scheme is a Type(3)–Scheme for a relation R, iff for each input x, the scheme may not halt successfully and the difference of the probabilities between each pair of each $y_1, y_2 : R(x, y_1), R(x, y_2)$ is bounded by a small epsilon.

Definition 3.5 (**(ϵ, δ)–Schemes**) *A random generation scheme S is an* **(ϵ, δ)–schemes** *for a relation R if for each input $x \in \Sigma^n$, S halts successfully with probability $1 - \delta$ and for each pair $y_1, y_2 : R(x, y_1), R(x, y_2)$,*

$$|Pr(M \text{ outputs } y_1) - Pr(M \text{ outputs } y_2)| \leq \frac{\epsilon}{|\{y : R(x, y)\}|}$$

An (ϵ, δ)–Schemes is a *fully polynomial-time* scheme if there is a polynomial $p(n, 1/\epsilon, 1/\delta)$ such that the scheme always halts in $p(n, 1/\epsilon, 1/\delta)$ steps.

An (ϵ, δ)–Schemes is an NC scheme if there is a polynomial $p(n, 1/\epsilon, 1/\delta)$ and a $k \in N$ such that the scheme uses $p(n, 1/\epsilon, 1/\delta)$ processors and always halts in $\log^k(n, 1/\epsilon, 1/\delta)$ steps.

4 A Polynomial Time Random Matching Algorithm

The first polynomial time algorithm for the Random Matching Problem was given by Broder [3] and was rigorously verified by Jerrum and Sinclair [9]. The algorithm is very simple. The algorithm uses a Markov chine that converges to a uniform distribution on the space of perfect matchings for any given bipartite graph.

Given a bipartite graph $G(V_1, V_2, E)$ ($|V_1| = |V_2|$), define $M_n(G)$ be the set of all perfect matchings of G and $M_{n-1}(G)$ be the set of all matchings of size $|V_1| - 1$.

The following notations will be used throughout the paper.

- For each $M \in M_n(G) \cup M_{n-1}(G)$, for each $v \in V_1$, $M(v)$ denotes the vertex in V_2 that $(v, M(v)) \in M$;
- For each $M \in M_{n-1}(G)$, $B_1(M)$ stands for the unmatched vertex in V_1 and $B_2(M)$ the unmatched vertex in V_2.

The random matching generating scheme of Broder, Jerrum, and Sinclair uses a Markov chain $\mathcal{MC}(G)$ with state space $\mathcal{N} = M_n(G) \cup M_{n-1}(G)$ in which transitions are made by adding and/or deleting edges locally.

The simpler transition was given by Broder, which are specified as follows.

MC1–Broder's Transition

1. Choose a vertex v uniformly at random in V_1;
2. If $M \in M_n(G)$, then move to $M - (v, M(v))$;

3. If $M \in M_{n-1}(G)$,

   (a) if $(v, B_2(M)) \in E$, then move to $M + (v, B_2(M)) - (v, M(v))$;

   (b) otherwise do nothing.

A slight modified transition was given by Jerrum and Sinclair. The transitions in $\mathcal{MC}(G)$ are specified as follows, in any state $M \in N$, choose an edge $e = (u, v) \in E$ uniformly at random and

Jerrum and Sinclair's Transition

1. if $M \in M_n(G)$ and $e \in M$, move to state $M' = M - e$;
2. if $M \in M_{n-1}(G)$ and u and v are unmatched in M, move to $M' = M + e$;
3. if $M \in M_{n-1}(G)$, $v = B_2(M)$ and $u \neq B_1(M)$, move to $M + e - (u, M(u))$, and symmetrically with u and v interchanged;
4. in all other case, do nothing.

For each M_0 and $M \in M_n(G) \cup M_{n-1}(G)$, for each integer t, let $Pr(M_0 \stackrel{t}{\Rightarrow} M)$ denote the t–step transition probability from matching M_0 to matching M.

Jerrum and Sinclair proved that the above Markov chain is *rapidly mixing.* i.e., that after a short period of evolution in the distribution of the final state is essentially independent of the initial state. Here *short* means bounded by a polynomial in the input size. Formally,

Theorem 4.1 (Jerrum and Sinclair) *There is a $k \in \mathcal{N}$ independent of $G(V_1, V_2, E)$, such that for each $\epsilon > 0$, for each $M_0, M \in M_G$,*

$$\frac{1-\epsilon}{|M_n(G) \cup M_{n-1}(G)|} \leq Pr(M_0 \stackrel{|V_1|^k}{\Rightarrow} M) \leq \frac{1+\epsilon}{|M_n(G) \cup M_{n-1}(G)|}$$

The following is a polynomial time algorithm for the Random Matching Problem.

Random Matching Scheme

1. Generate a perfect matching M using any known polynomial time algorithm;
2. **apply** one of the above transitions $|V_1|^k$ times, letting M' be the resulting matching;
3. **if** $M' \in M_n(G)$, **output** M' and **halt, otherwise goto** Step 1.

It follows easily from Theorem 4.1, that the above scheme randomly generates a perfect matching uniformly. The following theorem due to Broder implies that the above scheme runs in polynomial time.

Theorem 4.2 (Broder) *For each bipartite graph $G(V_1, V_2, E)$,*

$$\frac{1}{|V_1|^2} \leq \frac{|M_n(G)|}{|M_{n-1}(G)|} \leq |V_1|^2$$

5 The Walking Problem for Matching

In this section, we show that the polynomial time Random Matching Scheme presented in the last section can not be parallelized in NC, unless $NC = P$. We shall prove the case when the Broder's transition is used. The proof also applied to the case when Jerrum and Sinclair's transition is used.

For each bipartite graph $G(V_1, V_2, E)$, define a binary operator $\oslash : M_n(G) \cup M_{n-1}(G) \times V_1 \rightarrow M_n(G) \cup M_{n-1}(G)$ as, for each $M \in M_n(G) \cup M_{n-1}(G)$ and $v \in V_1$,

$$M \oslash v = \begin{cases} M - (v, M(v)) & \text{if } M \in M_n(G) \\ M & \text{if } (v, B_2(M)) \notin E \\ M + (v, B_2(M)) & \text{if } v = B_1(M) \\ M + (v, B_2(M)) - (v, M(v)) & \text{otherwise} \end{cases}$$

Assume $\oslash$ is *left associative*, that is $M \oslash v_1 \oslash v_2 = ((M \oslash v_1) \oslash v_2)$.

Definition 5.1 (The Walking Problem for Matching) *Given a bipartite graph $G(V_1, V_2, E)$, $M \in M_n(G) \cup M_{n-1}(G)$, and $v_1, \ldots, v_N \in V_1$, compute $WPM(M, v_1, \ldots, v_N)$, where*

$$WPM(M, v_1, \ldots, v_N) = M \oslash v_1 \oslash \cdots \oslash v_N.$$

The Walking Problem for Matching can be viewed as an execution of a random walk of the algorithms of Broder, Jerrum, and Sinclair. Clearly, if the Walking Problem for Matching is in NC or RNC, then the Random Matching Problem is also in NC or RNC, respectively.

However, we shall show

Theorem 5.1 (Walking Problem for Matching) *The Walking Problem for Matching is log-space complete for P.*

The reduction is from the Lexical First Maximal Independent Set Problem.

5.1 Some Structure Properties

We will present some interesting and important properties of the Walking Problem for Matching. These properties will be used in the NC reduction from the Lexical First Maximal Independent Set Problem to the Walking Problem for Matching.

Lemma 5.1 (Idempotent Properties) *For all $M \in M_n(G) \cup M_{n-1}(G)$, for all $v \in V_1$,*

$$WPM(M, v, v) = M \oslash v \oslash v = M$$

[**PROOF**]: There are four cases,

1. if $M \in M_n(G)$, then clearly, by the definition of $\oslash$, $WPM(M, v, v) = M$;

2. if $v = B_1(M)$ and $(v, B_2(M)) \in E$, then $M \oslash v = M + (v, B_2(M))$, hence, $WPM(M, v, v) = M + (v, B_2(M)) - (v, B_2(M)) = M$;

3. if $(v, B_2(M)) \notin E$, then $M \oslash v = M$, hence, $WPM(M, v, v) = M$;

4. if $(v, B_2(M)) \in E$ and $v \neq B_1(M)$ then $M' = M \oslash v = M - (v, M(v)) + (v, B_2(M))$. Hence, $B_2(M') = M(v)$ and $M'(v) = B_2(M)$, which implies that $WPM(M, v, v) = M' \oslash v = M' + (V, M(v)) - (v, B_2(M)) = M$;

A *Bi-valued Graph* of name j and k, denoted as $BG_{j,k}$, is a bipartite graph of four vertices as shown in the following figure (Figure 1).

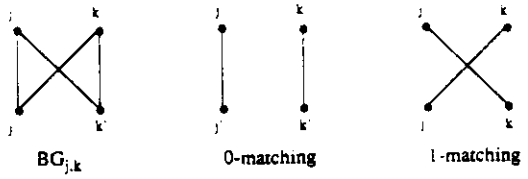

Figure 1: A Bi-Valued Graph of Name j and k and two Matchings

Each Bi-valued Graph $BG_{j,k}$ has two matchings. They are $M_0 = \{(j, j'), (k, k')\}$ and $M_1 = \{(j, k'), (k, j')\}$. M_0 and M_1 are called the 0-*matching* and the 1-*matching* of $BG_{j,k}$, respectively.

We can attach a Bi-valued Graph $BG_{j,k}$ to an edge (i, i') of a bipartite graph $G'(V_1', V_2', E')$, by connecting vertex j with vertex i'. In other words, we define another bipartite graph, denoted by $Attatching_{G'}((i, i'), BG_{j,k}) = (V_1, V_2, E)$, where $V_1 = V_1' \cup \{j, k\}$, $V_2 = V_2' \cup \{j', k'\}$, and $E = E' \cup Ed(BG_{j,k}) \cup \{(j, i')\}$ (see Figure 2).

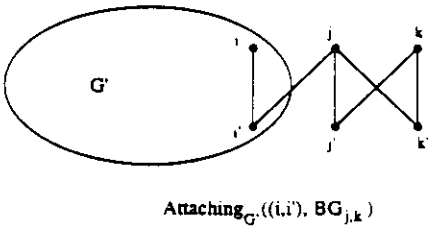

Figure 2: An Attached Graph

Lemma 5.2 (0–1 Devices) *For each* $M = M' \cup \{(j, j'), (k, k')\} \in M_{n-1}(G)$ *where* $G(V_1, V_2, E) = Attatching_{G'}((i, i'), BG_{j,k})$ *and* $M' \in M_{n-1}(G')$ *such that* $M'(i) = i'$,

$$WPM(M, i, j, k, j, i) = \begin{cases} M & \text{if } (i, B_2(M)) \notin E \\ M' \cup \{(j, k'), (k, j')\} & \text{if } (i, B_2(M)) \in E. \end{cases}$$

See figure 3.

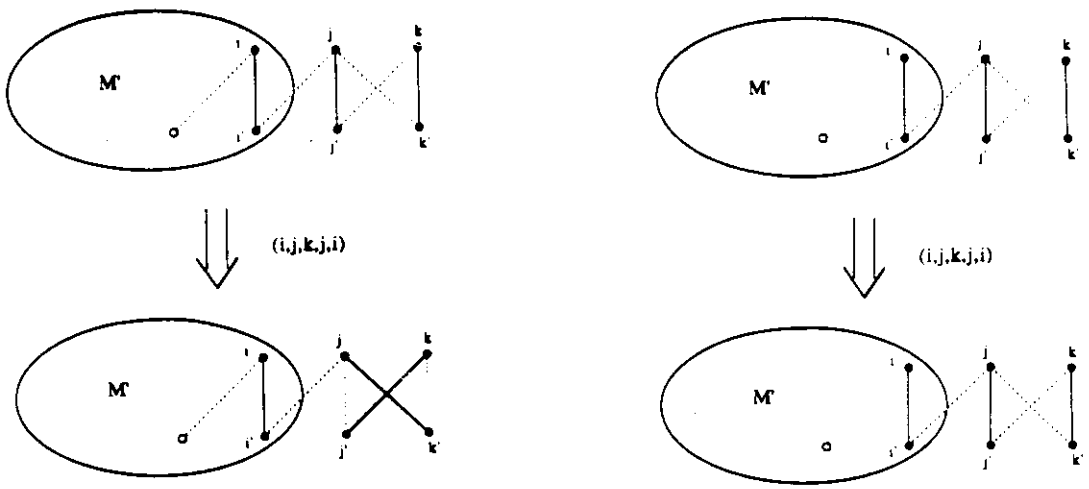

Figure 3: Two Cases of the 0–1 Devices Lemma

[**PROOF**]: if $(i, B_2(M)) \notin E$, then

$$M \oslash i = M \oslash j = M \oslash k = M.$$

Hence, $WPM(M, i, j, k, j, i) = M$.

if $(i, B_2(M)) \in E$, then the proof is in the following figure (Figure 4).

□

Lemma 5.2 shows that there is a very simple walking sequence that the matching obtained after the walking sequence sets the Bi-valued graph $BG_{j,k}$ to the 0–matching if i is not connected with the unmatched vertex in V_2 and to the 1–matching otherwise. Moreover, the changing is localized inside $BG_{j,k}$. Such a walking sequence is denoted as $C_{i,j,k}$, i.e.,

- $C_{i,j,k} = (i, j, k, j, i)$.

A *L-place Bi-valued graph* of name j and k, denoted as $BG^L_{j,k}$, is a bipartite graph of $2(L+1)$ vertices as shown in the following figure (Figure 5).

Each $BG^L_{j,k}$ contains L Bi-valued graphs $BG_{j_1,k_1}, ..., BG_{j_L,k_L}$ as subgraphs. Note that $BG^L_{j,k}$ has $2^L + 1$ perfect matchings: one contains the edge (k, j'_1), called the *infeasible matching* and all others are called feasible matchings, which are formed by a matching from each of its Bi-valued subgraph plus the edges (j, j') and (k, k'). In other words, each feasible matching M of $BG^L_{j,k}$ can be written as

$$M = \left(\bigcup_{l=1}^{L} M_l \right) \cup \{(k, k'), (j, j')\},$$

where M_l is a matching of BG_{j_l,k_l}. M is the *all 1's*–matching if for all $1 \le l \le L$, M_l is the 1–matching of BG_{j_l,k_l}. M is the *all 0's*–matching if for all $1 \le l \le L$, M_l is the 0–matching of BG_{j_l,k_l}.

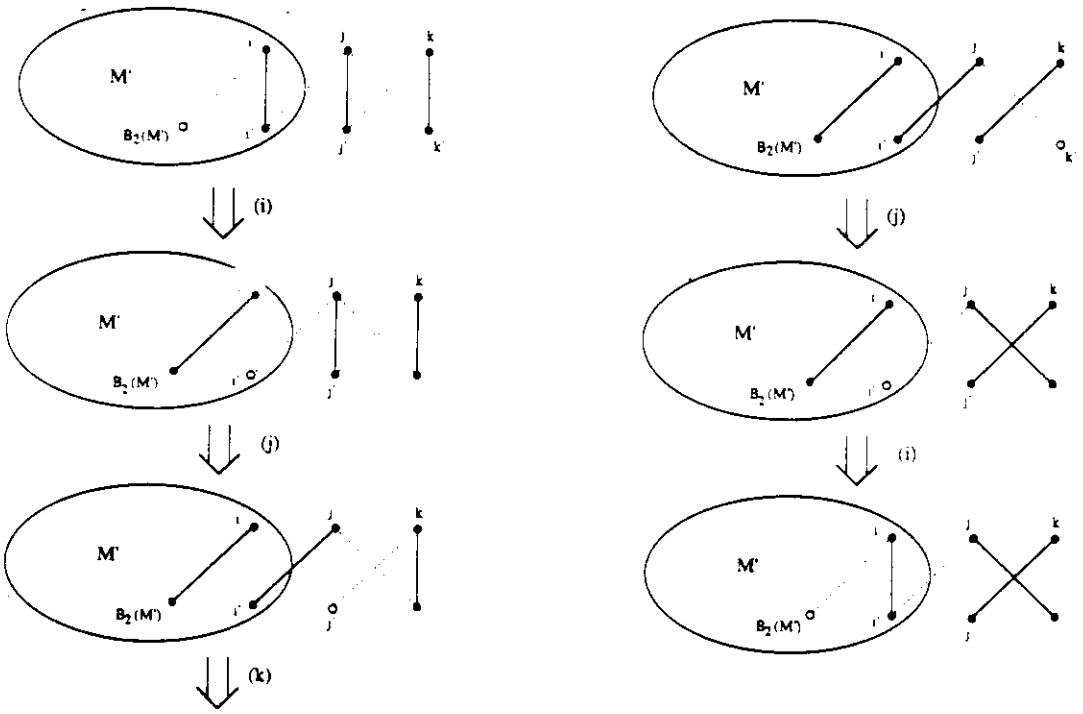

Figure 4: A Step-wise Proof of the Second Case

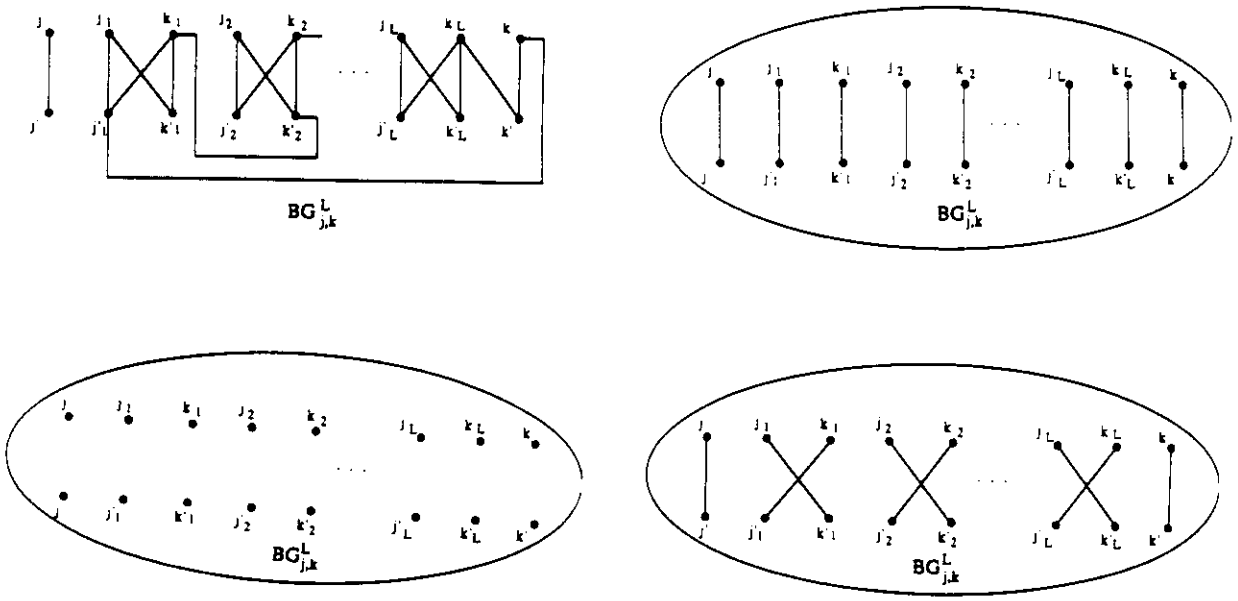

Figure 5: $BG^L_{j,k}$, the All 0's-Matching, and the All 1's-Matching

We can attach a Bi-valued Graph $BG^L_{j,k}$ to an edge (i,i') of a bipartite graph $G'(V'_1, V'_2, E')$, by connecting vertex j with vertex i', k with all neighbors of i but i', and attaching each Bi-valued subgraph to (i,i'). In other words, we define another bipartite graph (see Figure 6), denoted by $Attatching_{G'}((i,i'), BG^L_{j,k}) = (V_1, V_2, E)$, where

$$\begin{aligned} V_1 &= V'_1 \cup \{j, k, j_1, \ldots, j_L, k_1, \ldots, k_L\} \\ V_2 &= V'_2 \cup \{j', k', j'_1, \ldots, j'_L, k'_1, \ldots, k'_L\} \\ E &= E' \cup Ed(BG^L_{j,k}) \cup \{(j,i')\} \cup \{(j_l, i') | 1 \le l \le L\} \cup \{(k,v') | (i,v') \in E', v' \neq i'\} \end{aligned}$$

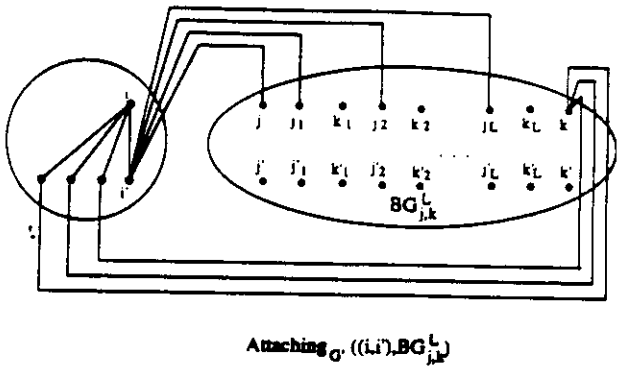

Figure 6: An Attached Graph

Lemma 5.3 (Checking Devices) *For each* $M = M' \cup M'_{BG^L_{j,k}} \in M_{n-1}(G)$ *where* $G(V_1, V_2, E) = Attatching_{G'}((i,i'), BG^L_{j,k})$, $M' \in M_{n-1}(G')$ *such that* $M'(i) = i'$, *and* $M'_{BG^L_{j,k}}$ *is a feasible matching of* $BG^L_{j,k}$, *such that if*

$$WPM(M, k, k_L, j_L, k_{L-1}, j_{L-1}, ..., j_2, k_1, k, i, j, k_1, j_2, k_2, ..., j_L, k_L, k, j) = M'' \cup M''_{BG_{j,k}}$$

then

$$M'' = \begin{cases} M' - (i,i') + (i, B_2(M')) & \text{if } M'_{BG^L_{j,k}} \text{ is a all 1's-matching and } (i, B_2(M')) \in E \\ M' & \text{otherwise} \end{cases}$$

where (i) $M'' \in M_{n-1}(G')$ *and* (ii) $M''_{BG^L_{j,k}}$ *is a feasible matching of* $BG^L_{j,k}$.

[**PROOF**]: We prove the case when $L = 1$. For $L < 1$, the similar argument applied. The proof is in Figure 7.

□

Lemma 5.3 shows that there is a very simple walking sequence that the matching obtained after the walking sequence moves the unmatched vertex to a particular node if the matching in an attached L–place Bi-valued graph is a all 1's matching. Such a walking sequence is denoted as $ONE_{i,j,k}$, i.e.,

- $ONE_{i,j,k,L} = (k, k_L, j_L, k_{L-1}, j_{L-1}, ..., j_2, k_1, k, i, j, k_1, j_2, k_2, ..., j_L, k_L, k, j)$.

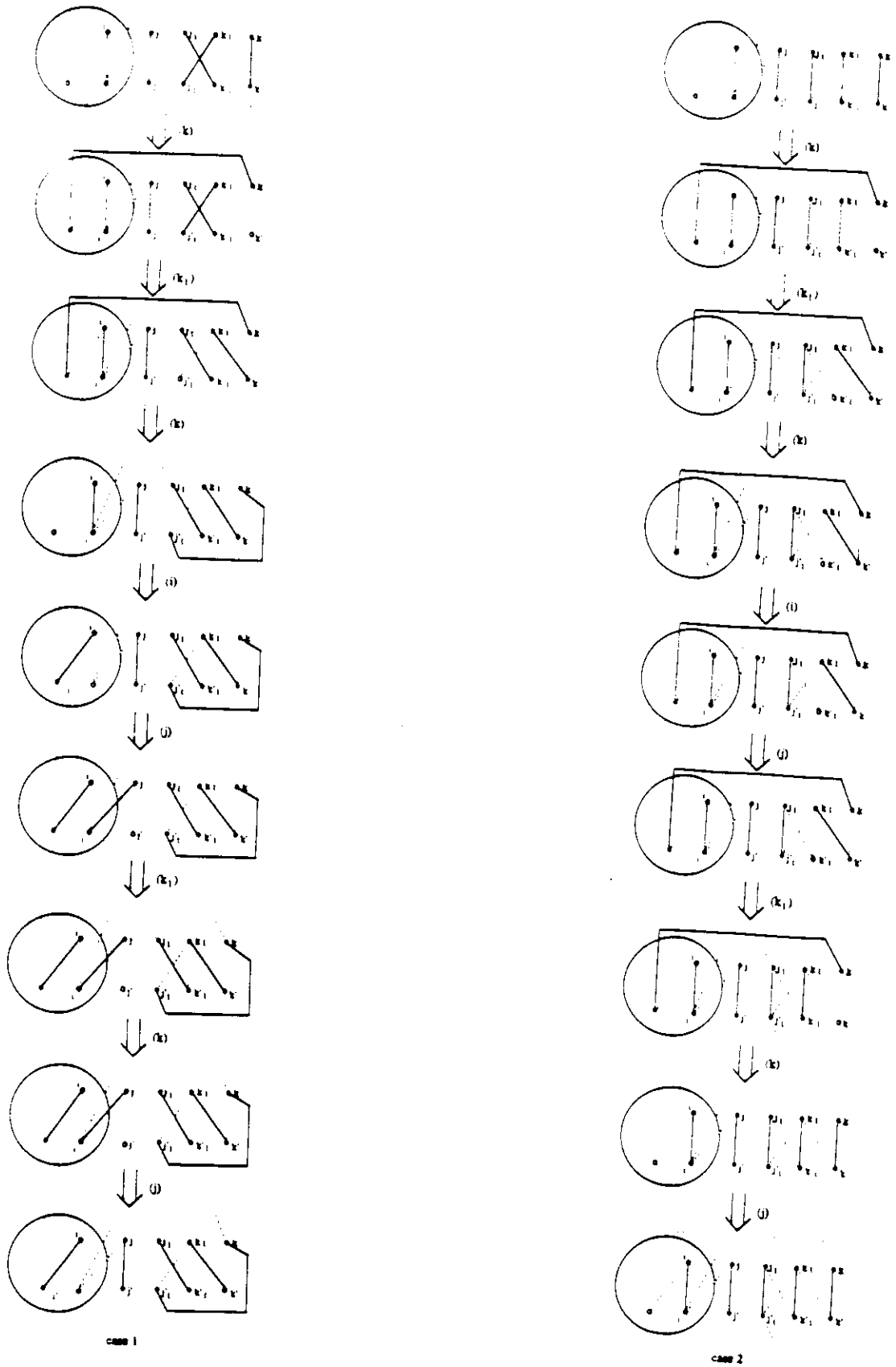

Figure 7: A Proof to the Case When $L = 1$

6 The Reduction

In this section, we will show that the Lexical First Maximal Independent Set Problem is NC-reducible to the Walking Problem for Matching. The reduction hinges on the structure properties of the matching problem and the $\oslash$ operator.

Theorem 6.1 *The Lexical First Maximal Independent Set Problem is NC-reducible to the Walking Problem for Matching.*

6.1 Informal Discussions

The reduction is carried out by using Lemma 5.2 and 5.3. For each vertex v in the given graph $G(V, E)$, we introduce another vertex v' and make (v, v') an edge in the bipartite graph and also an edge in the initial matching, where $V = \{v_1, \ldots, v_n\}$, and $order(v_i) < order(v_{i+1})$. We introduce another pair of vertices o and o' such that o and o' are the unmatched pair in the initial matching. The vertex o' is connected with all vertices in V. We will show that a walking sequence can be constructed such that the matching obtained after the walking sequence has the properties that for each $v \in V$, v is in the lexical first maximal independent set iff (v, v') is not in the matching.

Informally, the walking tests the vertices in V sequentially according to their order. The vertex v_1 is tested first. Note that initially, the unmatched vertex is o' which is connected with all vertices in V. The test is performed by a walking sequence of Lemma 5.2 followed by a walking sequence of Lemma 5.3. The Lemma 5.2 sequence tests whether a vertex, say v_i, is connected the current unmatched vertex. If the test succeeds, then the Lemma 5.3 sequence tests whether v_i had been disconnected from the unmatched vertex. If v_i has never been disconnected from the unmatched vertex, then as a side-effect, the test makes (v_i, *the current unmatched vertex*) an edge of the matching and make v_i' the current unmatched vertex.

In the construction of the bipartite graph, if (v_i, v_j) is an edge of E, then neither (v_i, v_j') nor (v_j, v_i') is an edge of the bipartite graph. Hence, if v_i' becomes the unmatched vertex (v_i is placed in the independent set), then all its higher order neighbors are disconnected from the unmatched vertex. This connectivity can be recorded by a walking sequence of Lemma 5.2 on the higher order neighbors.

Hence, the walking sequence performing the following tests.

1. test the whether a vertex v_i had been disconnected from the unmatched vertex. If v_i was disconnected from the unmatched vertex before, then some of v_i's lower order neighbor had been placed in the independent set. hence, v_i is not in the independent set. If v_i has not been disconnected from the

unmatched vertex, then no of v_i's lower order neighbor had been placed in the independent set. hence, v_i is in the independent set.

2. after testing v_i, all of v_i's higher order neighbors are tested. If v_i is placed in the independent set, then its higher order neighbors are disconnected from the unmatched vertex. Hence, by the time when they are tested, they can not be placed in the independent set.

6.2 The Proof

For each graph $G(V, E)$, a *higher neighborhood graph* $H_G(H_1, H_2, E_H)$ is a bipartite graph defined as following,

$$\begin{aligned} H_1 &= V \cup \{o\} \\ H_2 &= \{v' | v \in V\} \cup \{o'\} \\ E_H &= \{(v, v'), (v, o') | v \in V\} \cup \{o, o'\} \cup \{(u, v') | v \in V, u \notin N_G(v)\} \end{aligned}$$

Clearly, given a graph $G(V, E)$, $H_G(H_1, H_2, E_H)$ can be computed in $O(\log |V|)$ time, using $|V| + |E|$ processors.

[PROOF]: (Theorem 6.1) The reduction procedure on input a graph $G(V, E)$, generates a bipartite graph $B_G(V_1, V_2, E_G)$, a perfect matching M of B_G, and a walking sequence WS_G, where $B_G(V_1, V_2, E_G)$ is called the *matching bipartite graph* of G and M is called the *natural matching* of B_G.

- $B_G(V_1, V_2, E_G)$ has $|V| + 1$ subgraphs, S_G and S_i, for all $i : 1 \leq i \leq |V|$. S_i is a subgraph for the i^{th} vertex in G.

  The subgraphs are defined as,

  - $S_G = H_G$, the higher neighborhood graph of G;
  - $S_i = B^{\Delta^-(v_i)+1}_{j(i),k(i)}$, a $\Delta^-(v_i) + 1$ place Bi-valued graph.

  B_G is defined in such way that for each v_i,

  $$B_G = Attaching_{B_G - S_i}((v_i, v_i'), S_i) \tag{1}$$

  It follows the Equation (1) that B_G can be computed in $O(\log |V|)$ time, using $O(|V| + |E|)$ processors.

- The natural matching, M, of B_G is defined as,

  $$M = \left(\bigcup_{i=1}^{n} M_i \right) \cup \{(v, v') | v \in V\} \cup \{(o, o')\}$$

  where M_i is the all 0's–matching of S_i.

  Clearly, M can be computed in $O(1)$ time, using $O(|V| + |E|)$ processors.

- The walking sequence consists of $|V|+1$ subsequences, that is

$$WS_G = (o) \circ WS_1 \circ WS_2 \circ \cdots \circ WS_n$$

where WS_i is a walking sequence defined on the vertex v_i.

WS_i consists of three subsequences, i.e., $WS_i = C_i \circ IN_i \circ NE_i$, where letting $N^+(v_i) = (u_1, \ldots, u_s)$ and $r_j = Rank^{-}_{G,u_j}(v_i)$, $1 \leq j \leq s$,

$$\begin{aligned} C_i &= C_{v_i,j(i)_{\Delta^-+1},k(i)_{\Delta^-+1}} \\ IN_i &= ONE_{v_i,j(i),k(i),\Delta^-+1} \\ NE_i &= C_{u_1,j(r_1),k(r_1)} \circ \cdots \circ C_{u_s,j(r_s),k(r_s)} \end{aligned}$$

Clearly, WS_G can be computed in $O(\log |V|)$ time, using $|V| + |E|$ processors.

The correctness of the reduction follows from the following lemma. □

Lemma 6.1 *For each $G(V,E)$, $M_G = WPM(M, WS_G)$ has the property that for each $v \in V$, $(v,v') \notin M_G$ iff v is in the lexical first maximal independent set of G.*

Lemma 6.1 follows the following lemma.
For each $1 \leq i \leq n$, let $LMIS_i = \max\{j | j \leq i, v_j \in LFMIS(G)\}$.

Lemma 6.2

1. *For each $1 \leq i \leq n$, $B_2(WPM(M, WS_{G,i})) = v'_{LMIS_i}$;*
2. *For each $v_i \in V$, if v is in the lexical maximal independent set, then for all $1 \leq j < i$, $(v_i, B_2(WPM(M, WS_{G,j}))) \in E(B_G)$, where $WS_{G,i} = (o) \circ WS_1 \circ \cdots \circ WS_i$.*

[**PROOF**]: The lemma is proven by induction on i.
The lemma is clearly true for $i = 1$.
Assume that the lemma is true for all $j < i$, we shall show that it is true for i.
Let us first consider the case when $v_i \in LFMIS_G$. Since for all $j < i$, $B_2(WPM(M, WS_{G,j})) = v_{LMIS_j}$, clearly, $(v_i, v_{LMIS_j}) \notin E(G)$. Hence $(v_i, B_2(WPM(M, WS_{G,j})) \in E(B_G)$. Therefore, it follows from Lemma 5.2 and 5.3 and the definition of B_G, and WS_G that in $WPM(M, WS_{G,i-1}))$ the matching edges from S_i forms an all 1's matching of S_i. Therefore, $B_2(WPM(M, WS_{G,i})) = v'_i$.
We now consider the case when $v_i \notin LFMIS_G$. Since there is a $j < i$ such that $v_j \in LFMIS_G$ and $(v_j, v_i) \in E(G)$. By the induction hypothesis, $B_2(WPM(M, WS_{G,j}))) = LMIS_j$. Hence, $(v_1, B_2(WPM(M, WS_{G,j}))) \notin E(B_G)$. Thus, in $WPM(M, WS_{G,i-1}))$ the matching edges from S_i does not form an all 1's matching of S_i. Therefore, it follows from Lemma 5.2 and 5.3 and the definition of B_G, and WS_G that $B_2(WPM(M, WS_{G,i})) = B_2(WPM(M, WS_{G,i-1})) = v'_{LMIS_i}$. □

7 The Random Maximal Independent Set Problem

In this section, we prove the following theorem.

Theorem 7.1

1. *If there is a fully polynomial time bounded PTM for the Random Maximal Independent Set Problem, then* $NP = RP$;

2. *If there is an* NC *Random Maximal Independent Set Scheme, then* $NP = RNC$.

[**PROOF**]: A similar technique of Jerrum, Valiant, and Vazirani [10] is used. The reduction is from the *Maximum Independent Set Problem*–an NP–complete problem.

The basic idea, as that of the Jerrum, Valiant, and Vazirani, is to transform a given graph G to a one G' in which more that half of its maximal independent sets are maximum ones. Moreover, a maximum independent set of G can be computed in polynomial time when a maximum independent set of G' is given.

For each undirected graph $G(V, E)$ of n vertices, let $G(V', E')$, the *clique graph of* G, be an undirected graph derived from G by replacing each vertex v in G by a graph G_v of n n–cliques, complete graphs of n vertices, such that if (v, u) is an edge of G, then each vertex of G_v is connected with each vertex of G_u (See Figure 8).

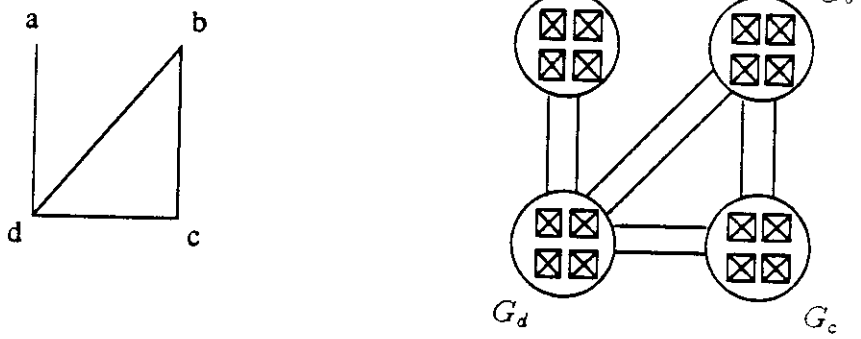

Figure 8: A Graph and its Clique Graph

Formally, the vertex and edge sets of G' are given by

$$\begin{aligned} V' &= V \times \{1, ..., n\} \times \{1, ..., n\} \\ E' &= \{(< v, i, j >, < u, k, l >) | (v, u) \in E\} \bigcup \{(< v, i, j >, < v, k, j >)\} \end{aligned}$$

Clearly, G' contains a maximal independent set of size kn iff G contains one of size k, and if G' has an independent set of size kn, then it contains at least n^{kn} independent set of

such size. Moreover, each maximal independent set $I = \{u_1, \ldots, u_k\}$ of size k in G defines exactly n^{kn} maximal independent sets of size kn in G'. They are of forms

$$\{< u_1, 1, i_{1,1} >, ..., < u_1, n, i_{1,n} >, ..., < u_k, 1, i_{k,1} >, ..., < u_k, 1, i_{k,n} >\}, \quad (2)$$

where $1 \leq i_1, \ldots, i_k \leq n$. And each maximal independent set of G' is of the form as (2).

It is easy to check that each maximal independent set of G' is of size kn for some k and the number of maximal independent sets of size less than kn is bounded above by

$$\sum_{j=1}^{k-1} \binom{n}{j} n^{jn} \leq kn^{k-1} n^{(k-1)n} < n^{nk}$$

Thus, it G has a maximum independent set of size k, the probability that a randomly generated maximal independent set of G' is of size kn is at least $\frac{1}{2}$.

It is easy to show that a maximal independent set of size k can be computed in polynomial time given a maximal independent set of size kn in G'. Moreover, this transformation is NC computable.

Since each step of the reduction is NC computable, the second part of the theorem follows. □

Since the Maximal Independent Set Problem is polynomial time and NC solvable [14], our reduction provides another example that the problem of uniform random generation is harder that the corresponding construction problem. For other examples, see [10].

8 Final Remarks and Open Questions

It is known that the Random Permutation Problem can be solved polylogarithmic time, using polynomial number of processors. In fact, Miller and Teng gave an $O(\log n)$ time, n processor Type(0)-Scheme for the Random Permutation Problem [16]. Therefore the Random Matching Problem over the set of complete bipartite graphs can be solved efficiently in parallel. It is important and interesting to show whether this is true for the class of general bipartite graphs. It is also important to show whether the Random Matching Problem over some of other naturally restricted classes of bipartite graphs can be solved in NC in parallel. One natural class is the planar bipartite graphs, for which Miller and Naor showed that the perfect matching problem is in deterministic NC and Vazirani showed that the number of perfect matching of a planar bipartite graph can be computed in NC.

8.1 Open Questions

1. Is the Random Matching Problem over planar bipartite graphs in NC?
2. Is the Random Maximal Independent Set Problem over planar graphs in P (NC)?
3. Is the Random Maximal Independent Set Problem over chordal graphs, circular arc graphs, and circle graphs in P (NC)?

8.2 Conjectures

Conjecture 8.1

1. *There is no RNC algorithm to generate a perfect matching in a bipartite graph uniformly in random unless $P = RNC$.*

2. *There is no NC algorithm to approximate permanent of a 0–1 matrix unless $P = RNC$.*

Conjecture 8.2 *The Approximation Counting Problem is not NC equivalent to the Random Generation problem.*

Acknowledge We would like to thank Eric Bach, Alan Frieze, Ming-Deh Huang, Ravi Kannan, Gary Miller, and Daniel Sleator for valuable discussion. We would like to thank Mei-Ting Lu for helping the figures.

References

[1] A. Aho, J. Hopcroft, and J. Ullman. *The Design and Analysis of Computer Algorithms.* Addison-Wesley, 1974.

[2] E. Bach. How to generate random integers with known factorization. In *Proceedings of the 15th Annual ACM Symposium on Theory of Computing*, pages 184–188, ACM, 1983.

[3] A. Z. Broder. How hard is to marry at random? (on the approximation of the permanent. In *Proceedings of the 18th Annual ACM Symposium on Theory of Computing*, pages 50–58, ACM, 1986.

[4] S. A. Cook. A taxonomy of problems with fast parallel algorithms. *Information and Control*, 64:2–22, 1985.

[5] J. D. Dixon and H. S. Wilf. The random selection of unlabelled graphs. *J. of Algorithms*, 4:205–213, 1983.

[6] J. Edmonds. Paths, trees, and flowers. *Canadian J. of Math.* 17:449–467, 1965.

[7] S. Even and O. Kariv. An $O(n^{2.5})$ algorithm for maximum matching in general graphs. In *16th Annual Symposium on Foundations of Computer Science*, pages 100–112, IEEE, 1975.

[8] J. Gill. Computational complexity of probabilistic turing machines. *SIAM J. Comput.*, 6:675–695, 1977.

[9] M. R. Jerrum and A. J. Sinclair. Conductance and the rapid mixing property for markov chains: the approximation of the permanent resolved. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing*, pages 235–244, ACM, 1988.

[10] M. R. Jerrum, L. G. Valiant, and V. V. Vazirani. Random generation of combinatorial structures from a uniform distribution. *Theoretical Computer Science*, 43:169–188, 1986.

[11] H. Karloff. A randomized parallel algorithm for the odd set cover problem. *Combinatorica*, 387–391, 1986.

[12] R. Karp, E. Upfal, and A. Wigderson. Finding a maximum matching is in random *NC*. In *Proceedings of the 17th Annual ACM Symposium on Theory of Computing*, pages 22–32, ACM, 1985.

[13] Donald E. Knuth. *The Art of Computer Programming.* Volume 2 /Seminumerical Algorithms of *Computer Science and Information Processing*, Addison-Wesley, second edition, 1981.

[14] M. Luby. A simple parallel algorithm for the maximal independent set problem. *SIAM J. Comput.*, 15(4):1036–1053, November 1986.

[15] Gary L. Miller and Shang-Hua Teng. On uniform permutation networks. Manuscript, School of Computer Science, CMU, 1988.

[16] Gary L. Miller and Shang-Hua Teng. Perfectly random permuting in parallel. Manuscript, School of Computer Science, CMU, 1988.

[17] K. Mulmuley, U. V.Vazirani, and V. V. Vazirani. Matchig is as easy as matrix inversion. *Combinatorica*, 7(1):105–113, 1987.

[18] L. G. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8:189–201, 1979.

[19] H. S. Wilf. The uniform selection of free trees. *J. of Algorithms*, 2:204–207, 1984.