# A Fast Algorithm for Enumerating Bipartite Perfect Matchings

Takeaki UNO
Foundations of Informatics Research Division, National Institute of Informatics, 2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo 101-8430, Japan. uno@nii.ac.jp

**Abstract:** In this paper, we propose an algorithm for enumerating all the perfect matchings included in a given bipartite graph $G = (V, E)$. The algorithm is improved by the approach which we proposed at ISAAC98. Our algorithm takes $O(\log |V|)$ time per perfect matching while the current fastest algorithm takes $O(|V|)$ time per perfect matching.

**Keyword:** enumeration, enumerating algorithm, perfect matching.

## 1 Introduction

Enumeration is a fundamental problem for optimization, data bases, decision making, and many other scientific problems. Numerous problems are solved, or investigated by enumerating related objects. Therefore, enumeration algorithms need to be intensively analyzed in order to find ways to solve these problems.

At ISAAC'98, we proposed a new approach for speeding up enumeration algorithms. Currently, there had been only few studies on speeding up enumeration algorithms. Almost all their techniques are depend on the structures of their problems, hence their techniques can not be applied to other algorithms immediately. Those algorithms often use data structures, which is also make the improvement difficult to be generalized. Our approach, which we named "trimming and balancing," is a general method for speeding up enumeration algorithms. It is not depend on structures of problems, and does not rely on data structures. Therefore, by using the approach, we can speedup several algorithms which we can not with the existing methods. In this paper, we speed up an algorithm for enumerating bipartite perfect matching by using the approach.

Let $G = (V = V_1 \cup V_2, E)$ be an undirected bipartite graph with vertex sets $V_1$ and $V_2$ and an edge set composed of edges in $V_1 \times V_2$. A matching $M$ of the graph $G$ is an edge set such that no two edges of $M$ share their endpoints. If all vertices of $G$ are incident to some edges of a matching $M$, then we say that $M$ is a *perfect matching*. Let $N$ be the number of perfect matchings in $G$. We consider the problem of enumerating all the perfect matchings in a given bipartite graph.

For this problem, some algorithms have been proposed. In 1993, K. Fukuda and T. Matsui proposed an enumeration algorithm [1]. The running time of the algorithm is $O(|V|^{1/2}|E|+N(|E|+|V|))$ time. In 1997, we proposed an algorithm [3] running in $O(|V|^{1/2}|E| + N|V|)$ time. Our algorithm in this paper reduces the time complexity to $O(|V|^{1/2}|E| + N \log |V|)$ time.

In the next section, we explain the framework of "trimming and balancing." In Section 3, we explain the basic algorithm arising from Fukuda and Matsui's algorithm, and we describe our improvement in section 4.

## 2 Approach for Speeding Up Enumeration Algorithms

This section explains our approach, which we proposed at ISAAC 98. Here, we omit the details and proofs. Readers should refer [4, 5]. The approach uses an amortized analysis. The analysis bounds time complexities of enumeration algorithms with two parameters. Since decrease of these two parameters result smaller time complexities, the goal of the approach is to improve algorithms to get small parameters. The way of improvement is to add two phases to each iteration of the algorithms, which decreases each parameter, respectively.

Firstly, we explain the amortized analysis. Consider enumeration algorithms based on recursive. For a given enumeration algorithm and its input, we define the *enumeration tree* by $\mathcal{T} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of all iterations occurring in the algorithm, and an edge of $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ connects two vertices iff one of them occurs in the other. In this paper, we define an iteration by computation in a recursive call excluding the computation in recursive calls occurring in the recursive call. For a vertex $v$ of a tree, let $D(v)$ be the set of descendants of $v$, $Ch(v)$ be the set of children of $v$. For a vertex $x \in \mathcal{V}$, we denote the computation time in $x$ by $t(x)$, and define $\hat{t}(\mathcal{T}) = \max_{x \in \mathcal{T}}\{t(x)/|D(x)|\}$.

The idea of the amortized analysis is to distribute the computation time of an iteration $x$ to all the children of $x$ such that each children $y$ receives computation time proportional to $t(y)$ or $|D(y)|$. This is for the balance of amount of computation time which the descendants of children receive. This distribution almost amortizes computation time of iterations. By adding several modifications to this idea, we can avoid the bad cases, and can state that the sum of computation time in an enumeration tree $\mathcal{T}$ is $\mathrm{O}(\hat{t}(\mathcal{T})x^*(\mathcal{T}))$ per iteration. Here $x^*(\mathcal{T})$ is a parameter of $\mathcal{T}$ which is bounded by the following ways.

Let $\mathcal{P}$ be the set of paths of $\mathcal{T}$ from the root to a leaf, and $\alpha > 1$ be a constant number. $x^*(\mathcal{T})$ is less than or equal to the maximum number of vertices in a path $P \in \mathcal{P}$ satisfying $T(x) > \dfrac{\alpha - 1}{\alpha} \displaystyle\sum_{u \in Ch(x)} T(u)$.

This is a result of [4, 5]. From this, we can get the following lemma.

**Lemma 1.** *If the enumeration tree satisfies the following conditions for a constant $c$, then $x^*(\mathcal{T}) = O(\log_{c/(c-1)} t(x_0))$.*

*(1) $t(x) \geq t(y)$ for any child $y$ of a vertex $x$*
*(2) If a vertex $w$ satisfies $t(w) < 4c^2$, then $|D(w)|$ is constant.*
*(3) If a vertex $w$ satisfies $t(w) \geq 4c^2$, then $Ch(w)$ can be split into two subset $Ch_1(w)$ and $Ch_2(w)$ such that $\sum_{u \in Ch_1(w)} t(u)$, $\sum_{u \in Ch_2(w)} t(u) \geq (1/c)t(w) - c$ satisfies.*

*Proof.* We set $\alpha = 2c + 1$. On vertex $w$ satisfying $t(w) > \frac{\alpha-1}{\alpha}\sum_{u \in Ch(w)}$, $\frac{2c+1}{2c}t(w) > \sum_{u \in Ch_1(w)} t(u) + \sum_{u \in Ch_2(w)} t(u)$ holds. Hence, from the assumption (3), we have

$$\begin{aligned}\sum_{u\in Ch_2(w)} t(u) &\le \frac{2c+1}{2c}t(w) - \sum_{u\in Ch_1(w)} t(u)\\ &\le \frac{2c+1}{2c}t(w) - \frac{2}{2c}t(w) + c\\ &\le \frac{2c-1}{2c}t(w) + \frac{t(w)}{4c}\\ &= \frac{4c-1}{4c}t(w).\end{aligned}$$

Similarly, we have $\sum_{u\in Ch_1(w)} t(u) \le \frac{4c-1}{4c}t(w)$. Hence, we get $t(u) \le \frac{4c-1}{4c}t(w)$, for any child $u$ of $w$. From the assumption (2), there are at most constant number of vertices satisfying $t(w) < 4c^2$ on any path $P \in \mathcal{P}$. Hence, $P$ has at most $\log_{4c/(4c-1)} t(x_0) + \mathrm{O}(1)$ vertices $x$ satisfying $T(x) > \dfrac{\alpha-1}{\alpha} \displaystyle\sum_{u\in Ch(x)} T(u)$. Therefore, $x^*(\mathcal{T}) = \mathrm{O}(\log_{c/(c-1)} t(x_0))$. □

From this, we can improve the algorithm by decreasing $\hat{t}(\mathcal{T})$ and bounding $x^*(\mathcal{T})$ with the three conditions of the lemma. For this purpose, our approach "trimming and balancing" does these by adding two phases. The first phase "trimming phase" reduces the input, i.e., removes unnecessary parts from the inputs, to decrease $t(x)$ so that the order of $\hat{t}(\mathcal{T})$ is reduced. The second phase "balancing phase" balance the size of subproblems so that each subproblem $y$ has not so small size after the trimming phase, to satisfy the conditions of the lemma. We describe the framework of trimming and balancing approach.

**Algorithm** ENUMERATION_INIT $(X)$
**Step 1:** $X :=$ trimming phase $(X)$
**Step 2:** **Call** ENUMERATION $(X)$

**Algorithm** ENUMERATION $(X)$
**Step 1:** **For** $i := 1$ **to** (the number of subproblems)
**Step 2:** Generate the input $X_i$ of subproblem $i$ by balancing phase
**Step 3:** $X_i :=$ trimming phase to $(X_i)$
**Step 4:** **Call** ENUMERATION $(X_i)$
**Step 5:** **End for**

## 3 An Algorithm for Perfect Matchings

In this section, we explain the basic algorithm arising from Fukuda and Matsui's algorithm[1]. In the next section, we improve this algorithm by "trimming and balancing" approach. For a given bipartite graph $G = (V_1 \cup V_2, E)$, we denote the set of all the perfect matchings in $G$ by $\mathcal{M}(G)$. For an edge subset $E'$, let $G \setminus E'$ be the graph obtained by deleting all the edges of $E'$ from $G$. The algorithm utilizes the following properties to enumerate perfect matchings.

*Property 1.* Let $E_1$ and $E_2$ be edge sets such that $E_1 \cup E_2$ is the set of edges incident to a vertex $v$, and $E_1 \cap E_2 = \emptyset$. Then, $\mathcal{M}(G \setminus E_1) \cap \mathcal{M}(G \setminus E_2) = \emptyset$. and $\mathcal{M}(G \setminus E_1) \cup \mathcal{M}(G \setminus E_2) = \mathcal{M}(G)$.

*Proof.* A perfect matching $M$ of $G$ including an edge of $E_1$ is a perfect matching of $G \setminus E_2$ and vice versa. A perfect matching $M$ of $G$ including an edge of $E_2$ is a perfect matching of $G \setminus E_1$ and vice versa. $M$ includes exactly one edge of $E_1 \cup E_2$, hence the statement holds. □

By using this property, the enumeration problem can be partitioned into two subproblems of $G \setminus E_1$ and $G \setminus E_2$, if both $G \setminus E_1$ and $G \setminus E_2$ include a perfect matching, respectively. $G \setminus E_i$ has a perfect matching iff a perfect matchings $M$ satisfy $M \cap E_i = \emptyset$. Hence, we find two distinct perfect matchings $M$ and $M'$, and set $E_1$ and $E_2$ so that $E_1$ includes an edge $e \in M \setminus M'$ and $E_2$ includes an edge $e \in M' \setminus M$.

A perfect matching $M$ can be found in $O(|V|^{1/2}|E|)$ time [2]. To find another perfect matching $M'$, we use *alternating cycles*. For a perfect matching $M$ and a cycle $C$, if any two edges in $C \setminus M$ are not adjacent, then we call $C$ an *alternating cycle*. In an alternating cycle, edges of $M$ and edges not in $M$ appear alternatively. By exchanging edges along an alternating cycle, we can obtain a perfect matching different from $M$. Alternating cycles satisfy the following condition [1].

*Property 2.* For a perfect matching $M$, there exists another perfect matching iff there exists an alternating cycle. □

To find alternating cycles, we utilize a directed graph $DG(G, M)$ defined for a graph $G$ and a matching $M$. The vertex set of $DG(G, M)$ is given by $V$. The arc set of $DG(G, M)$ is given by orienting edges of $M$ from $V_1$ to $V_2$, and edges of $E \setminus M$ in the opposite direction. For any directed cycle $C$ in the graph $DG(G, M)$, arcs of $M$ and the other arcs appear alternatively in the cycle of $G$ corresponding to $C$. Hence, we can find an alternating cycle by finding a directed cycle of $DG(G, M)$. For conciseness, we treat an edge $(u, v)$ ( or $(v, u)$ ) of $G$ and an arc $(u, v)$ of $DG(G, M)$ as the same object, for example, arcs of $DG(G, M)$ which are included in $M$ means arcs of $DG(G, M)$ corresponding to the edges of $M$.

By using these properties, we can construct the following enumeration algorithm. We note that we do not need to find a perfect matching in each iteration since we give $M$ or $M'$ to subproblems when we generate recursive calls.

**ALGORITHM** Basic_Algorithm $(G)$
**Step 1: If** ( $G$ includes no perfect matching ) **then** stop.
**Step 2:** $M :=$ ( a perfect matching of $G$ )
**Step 3:** Call Basic_Algorithm_Iter $(G, M)$

**ALGORITHM** Basic_Algorithm_Iter $(G, M)$
**Step 1:** Construct $DG(G, M)$.
**Step 2:** Find an alternating cycle $C$ by finding a directed cycle of $DG(G, M)$.
**Step 3: If** ( no directed cycle exists ) **then output** $M$ ; **stop**

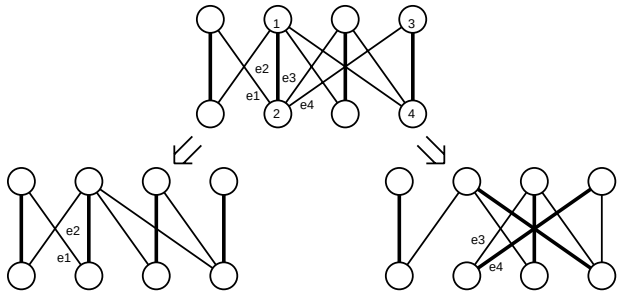


**Fig. 1.** An example of partitioning a problem: $E_1$ is composed of $e_1$ and $e_2$, and $E_2$ is composed of $e_3$ and $e_4$. $M'$ is obtained from $M$ with an alternating cycle (1,2,3,4).

**Step 4:** $M' :=$ the perfect matching obtained from $M$ and $C$
**Step 5:** $e :=$ an edge in $M \setminus M'$ ; $v :=$ an endpoint of $e$
**Step 6:** $E_1 := \{e\}$ ; $E_2 :=$ { all the edges incident to $v$ except for $e$}
**Step 7: Call** BASIC_ALGORITHM_ITER $(G \setminus E_2, M)$
**Step 8: Call** BASIC_ALGORITHM_ITER $(G \setminus E_1, M')$

Let $x$ be a vertex of an enumeration tree of the basic algorithm, and $G_x = (V_x, E_x)$ and $M_x$ be the input graph and input matching of $x$, The time complexity of $x$ is $O(|E_x| + |V_x|)$, which is the computation time in Steps 1 through 8 except for the computation done in generated recursive calls in Steps 7 and 8. Since each leaf of an enumeration tree corresponds to an output, and each internal vertex of the tree has two children, the number of iterations is less than twice the number of outputs, which is $2N$. Hence, the time complexity of this basic algorithm is $O(|E||V|^{1/2} + (|E| + |V|)N)$.

## 4 Improving the Basic Algorithm

In this section, we improve the basic algorithm by adding a trimming phase and a balancing phase. The trimming phase is composed of two parts, removing edges included in no perfect matching or all perfect matchings, and replacing consecutive degree 2 vertices by an edge.

To explain the first part, we prove a lemma. Let $Trim'(DG(G, M))$ be the graph obtained by removing the arcs included in no directed cycle, and $Trim'(G)$ be the undirected version of $Trim'(DG(G, M))$. We denote the edges of $M$ included in $Trim'(G)$ by $Trim'(M)$. Let $IS(G)$ be the graph obtained by removing all the isolated vertices of $G$.

**Lemma 2.** $\mathcal{M}(G) = \{M' \cup (M \setminus Trim'(M)) | M' \in \mathcal{M}(IS(Trim'(G)))\}$

*Proof.* An edge $e$ is included in no directed cycle of $DG(G, M)$ if and only if $e$ is included in all the perfect matchings, or no perfect matching. Hence, all the edges in $M \setminus Trim'(M)$ are included in any perfect matching of $G$. Since

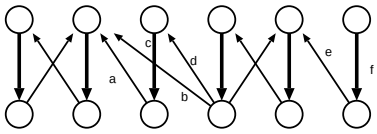


**Fig. 2.** An instance of $DG(G, M)$. Bold lines are edges of $M$. Arcs $a, b, c, d, e$ and $f$ are included in no directed cycle. $a, b, d$ and $e$ are included in no perfect matching, and $c$ and $f$ are included in all the perfect matchings.

any edge of $Trim'(G)$ is incident to no edge of $M \setminus Trim'(M)$, $M' \cup (M \setminus Trim'(M))$ is included in $\mathcal{M}(G)$ for any $M' \in \mathcal{M}(IS(Trim'(G)))$. Moreover, for any $M \in \mathcal{M}(G)$, if a vertex $v$ is incident to no edge of $Trim'(M)$, then no edge of $Trim'(G)$ is incident to $v$. Hence, $Trim'(M)$ is a perfect matching of $IS(Trim'(G))$. Therefore, the lemma holds. □

Arcs included in no directed cycle can be detected by strongly connected component decomposition. Hence, we obtain $Trim'(G)$ in $O(|E| + |V|)$ time. Next we state the following lemma to explain the second part of the trimming algorithm.

**Lemma 3.** *Suppose that two vertices $u$ and $v$ are incident to only edges $(w_1, u), (u, v)$ and $(v, w_2)$, and $w_1 \neq w_2$. Let $G'$ be the graph obtained by removing $(w_1, u), (u, v)$ and $(v, w_2)$ from $G$, and adding $(w_1, w_2)$ to it. Then, $\mathcal{M}(G) = \{M \cup \{(u, v)\} | M \in \mathcal{M}(IS(G')), (w_1, w_2) \notin M\} \cup \{M \setminus \{(w_1, w_2)\} \cup \{(w_1, u), (v, w_2)\} | M \in \mathcal{M}(G'), (w_1, w_2) \in M\}$ holds.*

*Proof.* For any $M \in \mathcal{M}(G)$, exactly one of $(w_1, u), (v, w_2) \in M$ and $(u, v) \in M$ hold. $(w_1, u), (v, w_2) \in M$ if and only if $M \setminus \{(w_1, u), (v, w_2)\} \cup \{(w_1, w_2)\} \in \mathcal{M}(IS(G'))$. $(u, v) \in M$ if and only if $M \setminus \{(w_1, u), (v, w_2)\} \cup \{(w_1, w_2)\} \in \mathcal{M}(IS(G'))$. Hence, the lemma holds. □

Let $Trim(DG(G, M))$ be the graph obtained by applying this operation to $Trim'(DG(G, M))$ while $G$ includes a pair of vertices with degree 2 adjacent to each other, and removing isolated vertices. Let $Trim(G)$ be the undirected version of $Trim(DG(G, M))$. $Trim(G)$ is obtained in $O(|E| + |V|)$ time. We note that $Trim'(DG(G, M)) = DG(Trim'(G), M')$ and $Trim(DG(G, M)) = DG(Trim(G), M'')$ hold for some perfect matchings $M'$ of $Trim'(G)$ and $M''$ of $Trim(G)$.

In the trimming phase operated before beginning of an iteration $x$, we construct $Trim(G_x)$ and set $G_x$ to $Trim(G_x)$. After the trimming phase, we output all edges of $M_x \setminus Trim'(M_x)$, and the changes by the operation of Lemma 3. By this, when an iteration inputs an empty graph and output a perfect matching $M$, the all edges of $M$ are already outputted, hence we can construct $M$ by previous outputs. Thus, we output only a word "matching" when we have to output a perfect matching. since they are included in any perfect matching of the original $G$. At the end of the iteration $x$, we cancel the outputs generated in the above. By using this outputting method, we can reduce the computation time for the output as much as the other part of the iteration.

Here we describe the trimming algorithm, inputting $G, M$ and outputting $Trim(G)$.

**ALGORITHM** TRIMMING_PERFECT_MATCHING $(G, M)$
**Step 1:** $G := G \setminus$ ( edges corresponding to arcs included in no directed cycle of $DG(G, M)$ )
**Step 2:** **If** ( $u$ and $v$ are incident to only edges $(w_1, u), (u, v)$ and $(v, w_2)$, and $w_1 \neq w_2$ )
**then** $E := E \setminus \{(w_1, u), (u, v)(v, w_2)\} \cup (w_1, w_2)$ ; **Go to Step 2**
**Step 3** **Output** $G$

In a trimming and balancing algorithm, we operate the trimming phase for the generated subproblem before generating a recursive call, hence we assume that the input graph $G$ in each iteration satisfies $G = Trim(G)$. This assumption gives a lemma. Let $cc(G)$ be the number of connected components of $G$, and $f(G)$ be $|E| - |V| + cc(G)$.

**Lemma 4.** $|\mathcal{M}(G)| \geq f(G) \geq |E|/5$.

*Proof.* To prove the lemma, we estimate the lower bound of the number of directed cycles in $DG(G, M)$. For a strongly connected component $D_i = (V_i, E_i)$ of $DG(G, M)$, we set a graph $C = (V_C, E_C)$ to a directed cycle of $D_i$. The number of directed cycles in $C$ is $|E_C| - |V_C| + 1$. If $E_i \setminus E_C \neq \emptyset$, then the graph $(V_i, E_i \setminus E_C)$ contains a directed path $P = (V_P, E_P)$ whose endpoints are both included in $C$ and whose internal vertices and edges are not in $C$ since $D_i$ is strongly connected. $P$ satisfies $|E_P \setminus E_C| - |V_P \setminus V_C| = 1$. By adding $P$ to $C$, at least one directed cycle including $P$ is generated since $C$ is strongly connected. This addition does not make $C$ non-strongly connected. $|E_C| - |V_C| + 1$ increases only one by this addition. Hence, when $E_C = E_i$ holds, we have that the number of directed cycles in $D_i$ is at least $|E_i| - |V_i| + 1$. Therefore, $DG(G, M)$ includes at least $\sum_{i=1}^{cc(Trim(G))}(|E_i| - |V_i| + 1) = |E| - |V| + cc(G) = f(G)$ directed cycles.

If $D_i$ is a directed cycle with length 2, then $|E_i| - |V_i| + 1 = 1 > 0.2|E_i|$. If $D_i$ is not a directed cycle with length 2, $D_i$ does not include consecutive vertices with degree 2. Hence, $|E_i| \geq 1.25|V_i|$ holds, and we have $|E_i| - |V_i| + 1 = 0.2|E_i|$. Therefore, $f(G) \geq 0.2|E_t|$. □

From this lemma, we can see that $G_x$ has at least $f(G_x)$ perfect matchings, thus $D(x) \geq f(G_x)$. Since the trimming phase and the balancing phase explained in below takes only $O(|E_x|)$ time, we have $\hat{t}(\mathcal{T}) = O(1)$. Next we explain the balancing phase. In the balancing phase, we select edge sets $E_1$ and $E_2$ such that $f(Trim(G \setminus E_i)) \geq f(G)/4 - 2$.

If connected components $D_1, ..., D_k$ of $G$ are at least two, there exists $D_i$ satisfying $f(D_j) \leq f(G)/2$. Since $f(G) = \sum_{i=1}^{k} f(D_i)$, any subsets $E_1$ and $E_2$ of edges incident to a vertex of $D_i$ satisfies $f(Trim(G \setminus E_i)) \geq f(G)/4$.

In the case that $G$ is connected, we get $E_1$ and $E_2$ by partitioning edges incident to a vertex $r \in V_2$. If $f(Trim(G \setminus E_i)) \geq f(G)/4$ does not hold, then we re-select $E_1$ and $E_2$. Suppose that $f(Trim(G \setminus E_2)) < f(G)/4$. Let $M$ be a perfect matching of $G$ including an edge $e^* \in E_1$. In $DG(G, M)$, $r$ is the head of $e^*$ since $e^* \in M$. We denote the tail of $e^*$ by $r'$. To re-select, we construct a directed graph $DG'$ satisfying the following conditions.

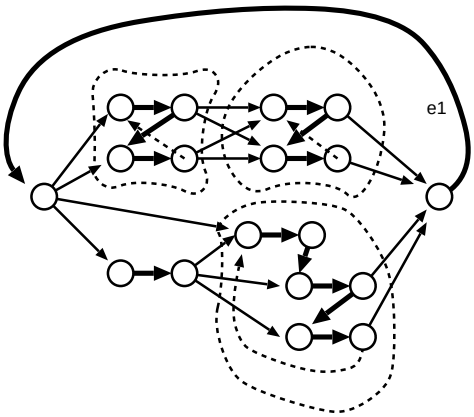


**Fig. 3.** An instance of $DG'$: dotted lines are arcs of $DG(G, M)$ not in $DG'$, and each dotted circle is $DG'_i$.

*Property 3.* There exists a directed subgraph $DG'$ of $DG(G, M)$ satisfying:
$(a)$ any directed cycle in $DG'$ includes $e^*$,
$(b)$ any arc $e$ of $DG'$ is included in a directed cycle, and
$(c)$ $f(DG') \geq 3f(G)/4$.

*Proof.* Let $D_i = (V_i, E_i)$ be each strongly connected component of $DG(G \setminus E_2, M)$, and $E'$ be the set of the edges not included in any $D_i$. We denote the set of vertices in $V_i$ which are heads of edges in $E'$ by $VH_i$, and those which are tails of edges $E'$ by $VT_i$. Since $DG(G, M)$ is strongly connected, $VH_i, VT_i \neq \emptyset$. Here we obtain $DG'_i$ by the following operations for each $i$.

(1) Choose a vertex $v \in VH_i$. Set $DG'_i = (V'_i, E'_i)$ to $(\{v\}, \emptyset)$
(2) If there exists a vertex $u \in VT_i \setminus V'_i$, then find a directed path $P$ from a vertex of $V'_i$ to $u$ such that all internal vertices of $P$ are not included in $V'_i$, add $P$ to $DG'_i$, and go to (2).
(3) If there exists a vertex $u \in VH_i \setminus V'_i$, then find a directed path $P$ from $u$ to a vertex of $V'_i$ such that all internal vertices of $P$ are not included in $V'_i$, add $P$ to $DG'_i$, and go to (3).

Here we set $DG'$ to $(\bigcup V'_i, E' \cup \bigcup E'_i)$. Since any arc of $E'$ is included in only directed cycles of $DG(G, M)$ including $e^*$, and $(V_i, E_i)$ includes no directed cycle, we can see that any directed cycle of $DG'$ includes $e^*$, thus $DG'$ satisfies (a). Since any vertex $v$ of $DG'$ is the tail of an arc of $DG'$, and is also the head of an arc of $DG'$, we can see that $DG'$ includes directed paths from $v$ to $r$ and $r$ to $v$. Hence, $DG'$ satisfies (b).

Since removals of isolated vertices does not change the value of $f$, we have $f(H) = f(IS(H))$ for any graph $H$. Since $f(G) = f(G')$ holds in Lemma 3, we have $f(Trim'(H)) = f(Trim(H))$ for any graph $H$. Thus, from $|E'_i| - |V_i| +$

$cc((V_i, E'_i)) \geq 0$ and $cc((V, E' \cup \bigcup E'_i)) \geq \sum cc((V'_i, E'_i)) - cc(Trim(G \setminus E_2)) + 1$, $DG'$ satisfies $(c)$ from the following inequation.

$$\begin{aligned}
f(DG') &= f((V, E' \cup \bigcup E'_i)) \\
&= |E'| + (\sum |E'_i|) - |V| + cc((V, E' \cup \bigcup E'_i)) \\
&\geq |E'| + (\sum |E'_i| - |V_i| + cc((V_i, E'_i))) - cc(Trim(G \setminus E_2)) + 1 \\
&\geq |E'| - cc(Trim'(G \setminus E_2)) + 1 \\
&= |E'| - (f(Trim'(G \setminus E_2)) - (|E| - |E'|) + |V|) + 1 \\
&= |E| - |V| + 1 - f(Trim'(G \setminus E_2)) \\
&= f(G) - f(Trim'(G \setminus E_2)) \\
&\geq 3f(G)/4. \square
\end{aligned}$$

Let $d'(v)$ be the out-going degree of $v$ in $DG'$, which is the number of arcs of $DG'$ whose tails are $v$. We note that $f(DG') = cc(DG') + \sum_{v \in V'}(d(v) - 1)$ where $V'$ is the vertex set of $DG'$. This holds for any directed graph. Let $Q$ be a directed path from $r$ to $r'$ including a maximum out-going degree vertex $w$ of $DG'$. Note that $w \neq r'$ since $d'(r') = 1$. Let $T$ be a directed spanning tree of $DG'$ including $Q$ whose root is $r$. For a vertex $v \in T$, we recall that $D(v)$ is the set of all the descendants of $v$. We note that $v$ is a descendant of $v$. We also denote the set of all the arcs whose tails are $v$ by $L(v)$, and the set of all the arcs whose tails are in $D(v)$ by $L(D(v))$. For an arc set $F \subseteq L(v)$, we define $D(F) = \{v\} \cup \bigcup_{(v,v') \in F} D(v')$, and $L(D(F)) = F \cup \bigcup_{(v,v') \in F} L(D(v'))$. $|L(D(r))| - D(r) \geq 3f(G)/4$ holds. Since $w$ is not a leaf of $T$, any leaf $v$ of $T$ satisfies $d'(v) \leq \sum_{v \in V'} d(v)/2$, hence $|L(D(v))| - D(v) \geq 3f(G)/4$ holds. By using this, we re-construct $E_1$ and $E_2$ as follows.

(1) Find a vertex $v^*$ such that $|L(D(v^*))| - |D(v^*)| \geq 2f(G)/4$, and $|L(D(u))| - |D(u)| < 2f(G)/4$ for any child $u$ of $v^*$.
(2) If an edge $e \in L(v^*)$ satisfies $L(D(\{e\})) \geq f(G)/4$, then we set $E_2$ to $\{e\}$. If not, we add an arc of $L(v)$ to $E_2$ iteratively until $|L(D(E_2))| - |D(E_2)| \geq f(G)/4$.

The obtained $E_2$ satisfies $|L(D(E_2))| - |D(E_2)| < 2f(G)/4$. Since $|L(D(L(v^*)))| - |D(L(v^*))| \geq |L(D(v^*))| - |D(v^*)|$, $E_2 \neq L(v^*)$. Let $E_1$ be the set of edges incident to $v^*$ and not included in $E_2$. Then, the following lemma holds.

**Lemma 5.** *$E_1$ and $E_2$ satisfy $f(Trim(G \setminus E_1)) \geq f(G)/4 - 2$, $f(Trim(G \setminus E_2)) \geq f(G)/4 - 2$.*

*Proof.* First, we show $f(Trim(G \setminus E_2)) \geq f(G)/4 - 2$. Let $e$ be an arc whose tail $v$ is in $V \setminus D(E_2)$, and $C$ be a directed cycle of $DG'$ including $e$. Suppose that $C$ includes an arc of $E_2$. Since $DG'$ includes only directed cycles including $e^*$, at most one arc of $E_2$ is included in $C$. We obtain a directed cycle including no arc of $E_2$ as follows.

(1) If a directed $r$-$v$ path in $C$ includes an arc of $E_2$, we replace the path of $C$ by the directed $r$-$v$ path of $T$.

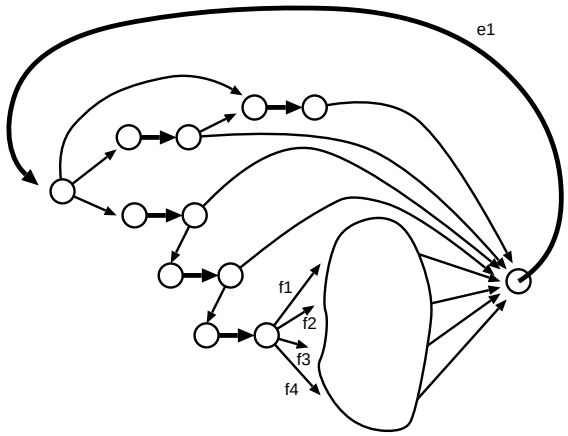


**Fig. 4.** An instance of re-selected $E_1 = \{f_1, f_2\}$ and $E_2 = \{f_3, f_4\}$. The circle is a subgraph with a large number of arcs.

(2) If a directed $v$-$r$ path of $C$ includes an arc of $C_1$, we replace the directed $v^*$-$r$ path of $C$ by a directed $v^*$-$r$ path including an arc of $E_1$. We note that the directed path exists since $L(v^*)$ includes at least one arc of $E_1$.

Therefore, $e$ is included in $Trim'(DG(G \setminus C_1, M))$. From this, the out-going degree of $v$ in $Trim'(DG(G \setminus E_2, M))$ is $d'(v)$. Similarly, the out-going degree of $v^*$ in $Trim'(DG(G \setminus E_2, M))$ is $|E_1| - 1$. Thus,

$$\begin{aligned} f(Trim(G \setminus E_2)) &= f(Trim'(DG(G \setminus E_2, M))) \\ &\geq 1 + ((|E_1| - 1) - 1) + \sum_{v \in V \setminus D(E_2)} (d'(v) - 1) \\ &= f(DG') - 2 - (|E_2| + \sum_{v \in D(E_2) \setminus \{v^*\}} (d'(v) - 1)) \\ &\geq 3f(G)/4 - 2 - (|L(D(E_2))| - |D(E_2)|) \\ &\geq f(G)/4 - 2. \end{aligned}$$

We next show that $f(Trim'(DG(G \setminus E_1, M'))) \geq f(G)/4 - 2$. Suppose that $C$ is an alternating cycle respect to $M$ including an edge of $E_2$ and $M'$ is the perfect matching obtained by $C$ from $M$.

If an arc $e$ of $DG(G, M)$ satisfies the following two conditions, then $G \setminus E_1$ contains both perfect matchings including $e$, and those not including $e$, hence $e$ is included in $Trim'(DG(G \setminus E_1, M'))$.

(1) There exists a directed cycle in $DG(G, M)$ including $e$ and an arc of $E_2$.
(2) There exists a directed cycle in $DG(G, M)$ including an arc of $E_2$ and not including $e$.

Any arc $e$ of $L(D(E_2))$ satisfies (1). If $e$ is not included in $C$, then $e$ satisfies (2) from the existence of $C$. Let $B$ be the set of arcs of $C \cap L(D(v^*))$ not included in $Trim'(DG(G \setminus E_1, M'))$, and $d''(v)$ be the out-going degree of $v$ in

$Trim'(DG(G \setminus E_1, M'))$. For $v \in D(E_2) \setminus \{v^*\}$, $d''(v) = d'(v) - 1$ if $v$ is the tail of an arc of $B$, and $d''(v) = d'(v)$ otherwise. Similarly, we can see $d''(v^*) \geq |E_2| - 1$.

Since any arc of $B$ is included in no directed cycle of $Trim'(DG(G \setminus E_1, M'))$, each strongly connected component on which an arc of $B$ has its tail is distinct. Hence, we have $cc(Trim'(DG(G \setminus E_1, M'))) \geq |B|$. From these, we obtain

$$
\begin{aligned}
& f(Trim(G \setminus E_1, M')) \\
&= f(Trim'(DG(G \setminus E_1, M'))) \\
&\geq cc(Trim'(DG(G \setminus E_1, M'))) + \sum_{v \in D(E_2)} (d''(v) - 1) \\
&= cc(Trim'(DG(G \setminus E_1, M'))) + (|E_2| - 1) - 1 + \sum_{v \in D(E_2) \setminus \{v^*\}} (d'(v) - 1) - |B| \\
&= cc(Trim'(DG(G \setminus E_1, M'))) + (|L(D(E_2))| - |B| - 1) - |D(E_2)| \\
&\geq |L(D(E_2))| - |D(E_2)| - 1 \\
&\geq f(G)/4 - 1. \square
\end{aligned}
$$

We describe the framework of our balancing phase as follows.

**ALGORITHM** Balancing_Perfect_Matching $(G, M)$
**Step 1:** $r$ :=(a vertex of strongly connected component with the minimum value of $f$)
**Step 2:** $E_1$ :=(the set composed of an edge $e^*$ incident to $r$)
**Step 3:** $E_2$ :=(the set of edges incident to $r$ except for $e^*$)
**Step 4: If** $(f(Trim(G \setminus E_i)) \leq f(G)/4)$ **then**
**Step 5:** Construct $DG'$
**Step 6:** $E_2$ :=( a subset of $L(r)$ with $f(G)/4 \leq |L(D(E_2))| - |D(E_2)| < 2f(G)/4$)
**Step 7:** $E_1$ :=(the set of edges incident to $r$ and not included in $E_2$ )
**Step 8: End if**
**Step 9 Output** $E_1, E_2$

Adding this balancing algorithm, we describe our trimming and balancing algorithm.

**ALGORITHM** Enum_Perfect_Matchings_Iter $(G, M)$
**Step 1: If** $G$ includes no edge, then **output** "matching" ; **return**
**Step 2:** $E_1, E_2$ :=Balancing_Perfect_Matching $(G, M)$
**Step 3:** $C$ := a directed cycle of $DG(G, M)$
**Step 4:** $M'$ := the perfect matching obtained by $C$ from $M$
**Step 5: For** $i := 1$ **to** 2 **do**
**Step 6:** $G$ := Trimming_Perfect_Matching $(G \setminus E_i, M)$
**Output** all edges of $M$ not included in $G$
**Step 8:** **Call** Enum_Perfect_Matchings_Iter $(G, Trim(M))$
**Step 9:** **Output** "delete" and all edges of $M$ not included in $G$
**Step 10:** Recover the original $G$ by doing the reverse operation of **Step 6**
**Step 11: End for**

For this algorithm, $t(x) = \mathrm{O}(|E_x|)$ and $|D(x)| \geq |E_x| - |V_x| + cc(G_x)$ for any iteration $x$. Thus we have $\hat{t}(\mathcal{T}) = \mathrm{O}(1)$. Moreover, we obtain the following properties.

(1) For any child $y$ of $x$, $t(x) \geq t(y)$.
(2) If $t(x)$ is constant, then $|D(x)|$ is constant since the size of $G_x$ is constant.
(3) For any child $y$ of $x$, $t(y) \geq t(x)/4 - 1$ from the balancing phase.

Hence, from lemma 1, any enumeration tree $\mathcal{T}$ generated by this algorithm satisfies $x^*(\mathcal{T}) = \mathrm{O}(\log |E|) = \mathrm{O}(\log |V|)$. Therefore, we obtain the following theorem.

**Theorem 1.** *Perfect matchings in a bipartite graph $G = (V, E)$ can be enumerated in $O(|E||V|^{1/2})$ preprocessing time and $O(\log |V|)$ time per perfect matching.* □

We note that the memory complexity of the algorithm is $\mathrm{O}(|E| + |V|)$. The analysis of the memory complexity is same as [3].

# Acknowledgment

We would like to thank Professor Akihisa Tamura of the University of Electro-Communications, and all the members of "myogadani-club" for their valuable advice.

## References

1. K. Fukuda and T. Matsui, "Finding All the Perfect Matchings in Bipartite Graphs," Appl. Math. Lett. **7**, No. 1, 15-18 (1994).
2. J. E. Hopcroft and R. M. Karp, "An $n^{5/2}$ Algorithm for Maximum Matching in Bipartite Graphs," SIAM J. Comp. **2**, 225-231 (1973).
3. T. Uno, "Algorithms for Enumerating All Perfect, Maximum and Maximal Matchings in Bipartite Graphs," Lecture Note in Computer Science **1350**, Springer-Verlag, Algorithms and Computation, 92-101 (1997).
4. T. Uno, "A New Approach for Speeding Up Enumeration Algorithms," Lecture Note in Computer Science **1533**, Springer-Verlag, Algorithms and Computation, 287-296 (1998).
5. T. Uno, "A New Approach for Speeding Up Enumeration Algorithms and Its Application for Matroid Bases," Lecture Note in Computer Science **1627**, Springer-Verlag, Computing and Combinatorics (Proceeding of COCOON99), 349-359, (1999)