Parallel algorithm for Maximum Matching

VladislavBrion

November 21 2015

1.Classic maximum matching algorithm

1.1. Definitions

Let $G(U,V,E)$be a bipartite graph consisting of two disjoint groups $U = \{u_1, \dots, u_n\}$ and $V = \{v_1, \dots, v_n\}$ connected by$m$non-weightededges $(u_i, v_j) \in E$. A **matching** in a bipartite graph is a set of edges $M \subseteq E$if no two edges have a common vertex.

A **maximal matching**$M_1 \subseteq E$ is a matching which can't be extended by the addition of any edge u_i, v_j. However, another matching,$M_2 \subset E$, where $|M_2| > |M_1|$ can exist [4].

A **maximum matching**$M_1 \subseteq E$ is a matching of maximum cardinality. No other matching $M_2 \subset E$ where $|M_2| > |M_1|$ exists. A graph can have multiple maximum matchings.

A **perfect matching**$M \subseteq E$ is a matching where$|M| = n$. Any perfect matching is a maximum matching. A graph can have multiple perfect matchings. If edges in a graph have weights, a minimum weightperfect matching can be defined as a perfect matching with minimal sum of edge weights for all perfect matching in G.

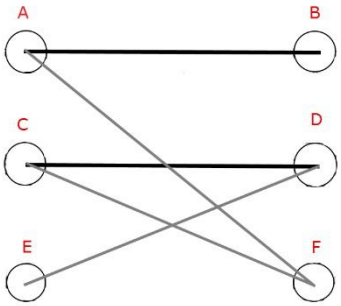

Fig.1. Maximal matching

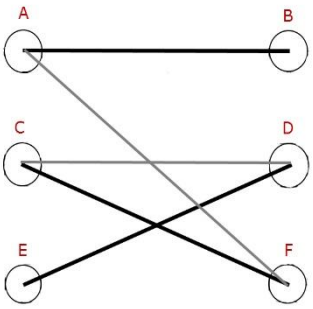

Fig. 2. Maximum and perfect matching

A vertex v is **matched** if it is the endpoint of an edge in a matching M.

Augmenting path with respect to M is a path whose endpoints are non-matched and edges are alternately not in M and in M. Augmenting paths always have odd lengths. Odd edges including beginning and ending edges are not in M, whereas even edges are in M.

1.2. Finding a maximum matching

For a bipartite graph $G(U,V,E)$ a maximum matching can be found using the following algorithm [1]:

1. Start at $M = \emptyset$.

2. While an augmenting path exists:

    - Find an augmenting path P with respect to M
    - $M = M \oplus P$

For example, a graph on Fig.1 has augmenting path ED – DC – CF, and the new maximal matching (that is also maximum matching) is shown on Fig. 2.

At each iteration, $|M|$ is increased by 1. If an augmenting path doesn't exist a matching M is a maximum matching (Berge's theorem).

The complexity of the algorithm is $O(|V||E|)$. The more efficient algorithm (Hopcroft-Carp algorithm) has complexity $O\left(\sqrt{|V|}|E|\right) = O\left(m\sqrt{n}\right)$ [3].

For further reducing complexity of the algorithm we consider randomized parallel algorithms.

2. The parallel matching algorithm

2.1 Randomized algorithms: definitions

There are two approaches for algorithms:

- **Deterministic** algorithms producing exact solutions. Reapplying a deterministic algorithm for the same input will not change the result obtained or the time it takes. Efficiency of these types of algorithms is described by average-case time analysis. It is a probabilistic value depending on distribution of inputs.
- **Randomized** algorithms which may produce incorrect answers with a low probability. Due to the randomness of these algorithms, reapplying them for the same input may take a different amount of time. In many cases such algorithms are simpler or faster [2]. Many parallel algorithms are randomized. There are two classes of randomized algorithms:
    1. A randomized algorithm is called a **Las Vegas** algorithm if it always gives the correct solution but its running time can vary. A randomized quick sort algorithm belongs to this class.
    2. A randomized algorithm is called a **Monte Carlo** algorithm if its execution can fail with low probability. Its execution can be repeated, which significantly decreases the failure probability. Algorithms of this class are much faster.

An algorithm is called $\boldsymbol{RNC^c}$ **algorithm**, if constants c and $k$exist such that it can be executed in $O(log^c(n))$ time using $O(n^k)$ parallel processors.

2.2 The Isolating Lemma

Definition: A **set system** $(\boldsymbol{S}, \boldsymbol{F})$ consists of a finite set $S = \{x_1, x_2, \dots, x_n\}$ and a family of subsets $F = \{S_1, S_2, \dots, S_k\}, S_j \subseteq S$, for $1 \leq j \leq k$.

A weight w_i is assigned to each element $x_i \in S$, and the weight of a set $S_j \subseteq S$ is defined as $\sum_{x_i \in S_j} w_i$. A set with minimal weight may not be unique.

Lemma [5]: Let (S, F) be a set of dimension n whose elements are assigned by random integer weights chosen uniformly and independently from $[1, N]$. Then,

$$\Pr[\textit{There is a unique minimum weight set in } F] \geq \frac{n}{N}$$

Note:the size of F is arbitrary:$k \in [1 : 2^n - 1]$. The weights of all sets lie in the range $[1, \dots, 2n^2]$. The weights of the sets are not independent.

Proof:[6] Without loss of generality, assume that each element of S occurs in at least one setin F. Select random weights of all elements except x_i. Suppose that we have two minimal sets: set $S_m$contains x_i, and set $S_l$doesn't. Let W_i be the weight of S_m, excluding the weight of x_i, and $\overline{W_i}$ be the weight of S_l. Define $\alpha_i = \overline{W_i} - W_i$, the value of which can be positive or negative.Note that α_i, which is the threshold for x_i, does not depend on $w(x_i)$.

If $w(x_i) < \alpha_i$ thenweight$W_i + w(x_i)$ = weight of minimal set $S_m$containing $x_i < \overline{W_i}$. Therefore, every minimum set must containx_i.

Similarly, if $w(x_i) > \alpha_i$ then weight$W_i + w(x_i)$ = weight of minimal set$S_m$containing $x_i > \overline{W_i}$.Therefore, x_i is in no minimum weight subset.

If $w(x_i) = \alpha_i$, no conclusion can be made about whether an arbitrary minimal weight set contains x_i or not. In this case we say that this element is singular. The presence of a singular element means that for two distinct minimum weight sets S_m and $S_l$only one of them contains x_i, i.e. a minimum weight set is not unique. Therefore, the presence of a singular element is equivalent to non-uniqueness of the minimum weight set.

Since $w(x_i)$ is a randomly distributed integer in $[1, N]$, the probability that x_i is singular is

$$\Pr(w_i = \alpha_i) \leq \frac{1}{N}$$

The probability that there exists a singular element among n elements is at most

$$n \times \frac{1}{N} = \frac{n}{N}$$

For further analysis we will use $N = 2n$, and the isolating lemma is redefined as follows:

The probability that there is unique minimum setwhen integer weights of all sets are selected randomly and uniformly from $[1, 2n]$ is at least half.

■

By the same reasoning we can consider that the maximum weight set will also be unique with probability at least half.

3. The perfect matching algorithm

Suppose we have a bipartite graph $G(U, V, E)$ having a perfect matching. Let $m = |E|$. Random integer weights chosen uniformly and independently from $[1, 2m]$ are assigned to the edges of the graph. By isolating lemma, the minimum weight perfect matching will be unique with probability at least $1/2$. Let Bbe a$n \times n$ adjacency matrix such as:

$$d_{ij} = \begin{cases} 2^{w_{ij}}, (u_i, v_j) \in E \\ 0, \quad otherwise \end{cases}$$

Lemma 1: LetMbe the unique minimum weight perfect matching for $G(U, V, E)$ with weight w (if other perfect matching exist, its weight is higher). Then:

1. $|B| \neq 0$

2. $2^w$is the highest power of 2 which divides $|B|$.

Proof: For each permutation σon $\{1, \dots, n\}$, define

$$value(\sigma) = \prod_{i=1}^{n} b_{i\sigma(i)}$$

For M, value$(\sigma) = 2^{\sum_{i=1}^{n} w_{ij}} = 2^w$. If another permutation doesn't correspond to a perfect matching, its value is 0. For other perfect matching with higher weight, the corresponding permutation will have higher weight $2^\lambda, \lambda > w$. Using definition of determinant:

$$|B| = \sum_\sigma sign(\sigma) \times value(\sigma) = \pm 2^w + 0 \dots \pm 2^\lambda \pm \cdots$$ ■

Lemma 2: Let M be the unique minimum weight perfect matching in G with weight w.

The edge $(u_i, v_j) \in M$ iff $\frac{|B_{ij}|2^{w_{ij}}}{2^w}$ is odd.

Proof: If B is an$n \times n$ matrix, then a minorB_{ij} represents the $(n-1) \times (n-1)$ matrix obtained by deleting the i^{th} row and j^{th} column. Note that:

$$\left|B_{ij}\right|2^{w_{ij}} = \sum_{\sigma:\,\sigma(i)=j} sign(\sigma)value(\sigma)$$

If $(u_i, v_j) \in M$, then the contribution of the permutation corresponding to this sum will be 2^w, whereas the contributions of other permutations will be zero (no perfect matching corresponding to this permutation) or a higher power of 2 (other perfect matching with higher weight). Therefore,

$\frac{\left|B_{ij}\right|2^{w}ij}{2^w} = \frac{\pm 2^w + 0 \ldots \pm 2^\lambda \pm \cdots}{2^w} = 1 + 2^{\lambda - w}$ is odd.

If $(u_i, v_j) \notin M$, the numerator will not contain the term 2^w, and this expression will be even.

■

Based on the lemmas, the algorithm of finding minimum weight perfect matching is shown below [6]:

Parallel algorithm for perfect matching

Input: Bipartite graph $G(U, V, E)$ with at least one perfect matching

Output: A perfect matching $M \subseteq E$

1. For eachedge$(u_i, v_j) \in E$ do in parallel:
   - select random integer weights $w_{ij} = uniform[1, 2m]$.
2. Compute the matrix B , as well as $|B|$ and obtain w.
3. Compute adjoint matrix $adj(b)$ whose $(j, i)^{th}$ entry is minor $|B_{ij}|$.
4. For each edge $(u_i, v_j) \in E$ do in parallel:
   - compute $\frac{\left|B_{ij}\right|2^{w}ij}{2^w}$;
   - if this value is odd, then add (i, j) into M.
5. Check if the selected edges form a perfect matching. If yes: exit. Otherwise, (the probability of this case is$\leq 1/2$) repeat Step 1 onwards.

Complexity of the algorithm:RNC^2using $O(n^{3.5}m)$ parallel processors. The most expensive computation steps are evaluation of determinant and adjoint of the matrix B. They can be computed by Pan's randomized parallel algorithm [5] that requires $O(log^2 n)$ time and $O(n^{3.5}m)$ processors for inverting $n \times n$ matrix with $O(m)$-bit integers.

This is a Monte Carlo algorithm that with probability at least $1/2$ produces a correct answer (by isolating lemma). In the case of incorrect result (step 5) the algorithm should be rerun. The probability of incorrect outputdecreases exponentially.

4. Algorithm for maximum matching

To compute maximum matching in a bipartite graph $G(U, V, E)$when the size of the maximum matching is unknown we can use the algorithm of the perfect matching [2]:

1. Extend G to complete bipartite graph. The total number of edges will be $m = n^2$.
2. Select random integer weights $w_{ij} = uniform[1, 2m]$.
3. Add $2mn$ to the weight of each new edge.
4. Repeat the steps 2-5 from the algorithm of perfect matching.
5. Remove edges added at the first step.

This algorithm will find a minimum-weight perfect matching for complete graph, and it should include a minimum matching of the original graph. Suppose that the original graph has a maximum matching with the size $k \leq n$. Its maximal weight will be at most $2mn$ (perfect matching with maximal weight for each edge). However, the weight of any single edge added to the graph for completeness will be at least $2mn + 1$ considering minimal random weight. Therefore, any perfect matching containing one or more artificial edge will have higher weight that the entire maximum matching, i.e. will not have minimal weight.

The complexity of the algorithm does not change because the most expensive computation is in the part for the perfect matching.

5. Parallel algorithms for related problems

- Finding perfect matchingfor general graphs. In this case the matrix B is obtained from the Tutte matrix.
- Finding a minimum weight perfect matching in a graph $G(V, E)$ with integer edge weights $w(e)$. Each weight should be assigned by $mnw(e) + r_e$ where r_e is integer selected independently and uniformly from $[1, 2m]$. Adding random weight will isolate one of the minimum weight matchings. This problem is also RNS^2 with $O(n^{3.5}mW)$ processors where $W$is the weight of the heaviest edge.
- For a graph with positive weight for each vertex, finding a matching with maximal vertex-weight.
- Combinatorial search problem.

References

[1] A.V. Aho, J.E. Hopcroft and J.D. Ulman. *Data Structures and Algorithms*. Addison-Wesley, Reading, MA, 1983.

[2] J. Jaja. An Introduction to Parallel Algorithms. Addison-Wesley, Reading, MA, 1992

[3]J.E. Hopcroft and R.M. Karp. An $n^{5/2}$ algorithm for maximum matching in bipartite graph. *SIAM Journal on Computing*, 2:225-231, 1973.

[4] W. Kocay and D.L. Kreher. *Graphs, Algorithms and Optimization*. Discrete Mathematics and its Application Series, Chartman & Hall, 2004.

[5] K. Mulmuley, U.V. Vazirani and V.V. Vazirani. Matching is as easy as matrix inversion. *Combinatorica*, 7:105-113, 1987.

[6] R. Motwani and P. Raghavan. *Randomized Algortihms*. Cambridge University Press, 1995.