Parallel Algorithm for Bipartite Graph Matching

Vladislav Brion

COMP 5703

November 23 2015

Outline

- Definitions
- Classic maximum matching algorithm
- Randomized algorithms - definitions
- The Isolating Lemma
- The parallel perfect matching algorithm
- The parallel maximum matching and related problems

Definitions

- **Bipartite graph** $G(U, V, E)$: consists of two disjoint groups U= $\{u_1, \dots, u_n\}$, $V = \{v_1, \dots, v_n\}$ connected by m non-weighted edges $(u_i, v_j) \in E$.

- **Matching** $M \subseteq E$: no two edges have a common vertex.

- **Maximal matching** $M_1 \subseteq E$: can't be extended.
  $M_2 \subset E$: $|M_2| > |M_1|$can exist

- **Maximum matching** $M_1 \subseteq E$: maximum cardinality.
- No $M_2 \subset E$: $|M_2| > |M_1|$ exists.
  A graph can have multiple maximum matchings.

Definitions (cont.)

- **Perfect matching** $M \subseteq E: |M| = n$.

  Any perfect matching is a maximum matching. A graph can have multiple perfect matchings. For a graph with weighted edges, a minimum weight perfect matching is a perfect matching with minimal sum of edge weights for all perfect matchings in G.

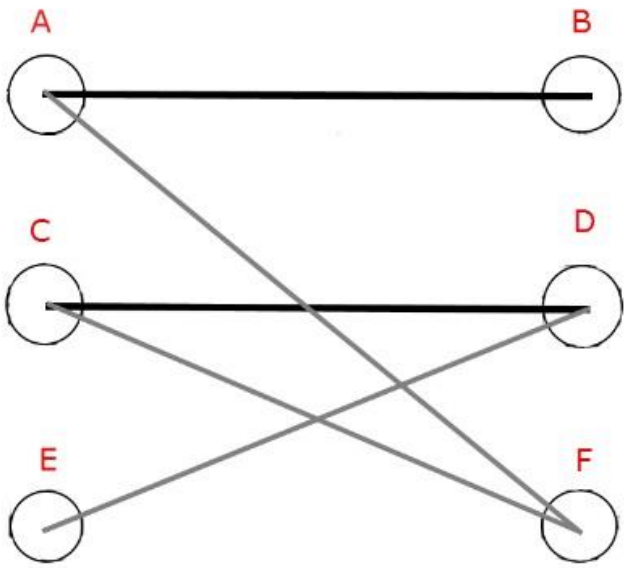
- A vertex v is **matched** if it is the endpoint of an edge in a matching M.

- **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 have odd lengths. Odd edges including beginning and ending edges $\notin M$; even edges $\in M$.

Classic maximum matching algorithm

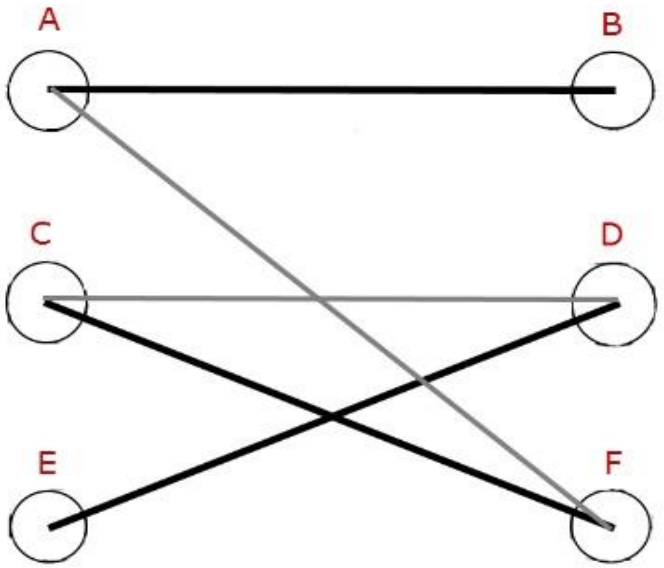
Maximum matching algorithm for a bipartite graph $G(U, V, E)$:

- 1. Start at $M = \emptyset$.
- 2. While an augmenting path exists:
  - Find an augmenting path P with respect to M
  - $M = M \oplus P$



Augmenting path:

ED – DC – CF



Classic maximum matching algorithm (cont.)

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: $O(|V|\,|E|)$.

The complexity of more efficient algorithm (Hopcroft-Carp algorithm): $O\left(\sqrt{|V|}\,|E|\right) = O(m\sqrt{n})$.

For further reducing complexity of maximum matching:

randomized parallel algorithms.

Randomized algorithms - definitions

Two approaches for algorithms:

- **Deterministic** algorithms: exact solutions.

  The same input: the same output for the same time.

  Efficiency: average-case/worst case time analysis.
- **Randomized** algorithms: possibly incorrect answer with a low probability.

  The same input: different time.

  Simpler or faster.

  Many parallel algorithms are randomized.

Randomized algorithms – definitions (cont.)

Two classes of randomized algorithms:

- **Las Vegas** algorithm: always the correct solution.
  Various running time.
  Example: a randomized quick sort algorithm.

- **Monte Carlo** algorithm: can fail with low probability.
  Reapplying decreases the failure probability.
  Much faster.

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

The Isolating Lemma

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 $x_i \in S$, 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: 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 $\in [1, \dots, 2n^2]$. The weights of the sets are not independent.

The Isolating Lemma (cont.)

Proof: Assume that each element of S occurs in at least one set in 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 $w(x_i) < \alpha_i$ then weight $W_i + w(x_i)$ = weight of minimal set S_m containing $x_i < \overline{W_i}$. Therefore, every minimum set must contain x_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.

The Isolating Lemma (cont.)

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. 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 Isolating Lemma (cont.)

The probability that there exists a singular element among n elements:

$$\text{At most } n \times \frac{1}{N} = \frac{n}{N}$$

The probability that there is unique minimum set when 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.

The parallel perfect matching algorithm

A bipartite graph $G(U, V, E)$ has a perfect matching. For $e(u_i, v_j)$:

w_{ij} is an integer weight chosen uniformly and independently from $[1, 2m]$.

By isolating lemma, the minimum weight perfect matching will be unique with probability at least $1/2$. Let B be a $n \times n$ adjacency matrix:

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

Lemma 1: Let M be the unique minimum weight perfect matching for $G(U, V, E)$ with weight w. Then:

1. $|B| \neq 0$
2. 2^w is the highest power of 2 which divides $|B|$.

The parallel perfect matching algorithm(cont.)

- **Proof**: For each permutation σ on $\{1, \dots, n\}$, define:

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

For M, value(σ) $= 2^{\sum_{i=1}^{n} w_{ij}} = 2^{w}$. 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:

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

The parallel perfect matching algorithm(cont.)

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: For $n \times n$ matrix B, minor B_{ij} is the $(n-1) \times (n-1)$ matrix obtained by deleting the i^{th} row and j^{th} column.

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

If $(u_i, v_j) \in M$: $\frac{|B_{ij}|\, 2^{w_{ij}}}{2^w} = \frac{\pm 2^w + 0 \ldots \pm 2^\lambda \pm \cdots}{2^w} = 1 + 2^{\lambda - w}$: odd

The parallel perfect matching algorithm(cont.)

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

Output: A perfect matching $M \subseteq E$

- For each edge $(u_i, v_j) \in E$ do in parallel:

  select random integer weights $w_{ij} = uniform[1, 2m]$.
- Compute the matrix B , as well as $|B|$ and obtain w.
- Compute adjoint matrix $adj(b)$ whose $(j, i)^{th}$ entry is minor $|B_{ij}|$.
- For each edge $(u_i, v_j) \in E$ do in parallel:

  compute $\frac{|B_{ij}|\, 2^{w_{ij}}}{2^w}$;

  if this value is odd, then add (i, j) into M.
- If the selected edges don't form a perfect matching: repeat step 1.

The parallel perfect matching algorithm(cont.)

Complexity of the algorithm: 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 the algorithm should be rerun. The probability of incorrect output decreases exponentially.

The algorithm of the perfect matching is used to compute maximum matching in a bipartite graph $G(U, V, E)$ when the size of the maximum matching is unknown.

Algorithm for maximum matching

1. Extend G to complete bipartite graph ($m = n^2$).
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.

maximal weight of maximum matching $\leq 2mn$

weight of artificial edge > $2mn$

Complexity: RNC^2 using $O(n^{3.5}m)$ parallel processors

Parallel algorithms for related problems

- Finding perfect matching for 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.

Thank you !

Questions?