An adjustable linear time parallel algorithm for maximum weight bipartite matching

Morteza Fayyazi, David Kaeli, Waleed Meleis

Department of Electrical and Computer Engineering, Northeastern University, Boston, MA 02115, USA

Abstract

We present a parallel algorithm for finding a maximum weight matching in general bipartite graphs with an adjustable time complexity of $O(\frac{n}{\omega})$ using $O(n^{max(2\omega,4+\omega)})$ processing elements for $\omega \geq 1$. Parameter ω is not bounded. This is the fastest known strongly polynomial parallel algorithm to solve this problem. This is also the first adjustable parallel algorithm for the maximum weight bipartite matching problem in which the execution time can be reduced by an unbounded factor. We also present a general approach for finding efficient parallel algorithms for the maximum matching problem.

Key words: Parallel algorithms, Graph algorithms, Bipartite matching

1 Introduction

A *bipartite graph* is a graph that can be partitioned into two groups of nodes such that every edge of the graph is incident on at most one node from each group. Given a graph, a *matching* is a subset of edges none of which are incident on any common node. A *perfect matching* is a matching in which every node of the graph is an end point to an edge in the matching. For a given graph, a set X is a maximum matching if there is no matching Y that is larger than X. The size of a matching is the number of edges in the matching. A weighted bipartite graph is a bipartite graph in which each edge is assigned a weight. The weight of a matching in a weighted bipartite graph is the sum of the weights of the edges in the matching. The problem of finding maximum (size or weight) matching is a fundamental graph problem. A *maximum size (or cardinality) bipartite matching (MSBM) algorithm* is an algorithm that finds a matching with the maximum size. A *maximum weight bipartite matching (MWBM) algorithm* is an algorithm that finds a matching with the maximum weight. The MSBM problem is a special case of the MWBM problem in which every edge has a weight of one.

An efficient parallel algorithm for the MWBM problem is an algorithm that runs in polylogarithmic time using a polynomial number of processing elements. It is still an open question if there exists an efficient parallel algorithm for the MWBM problem. Goldberg et al. [1] introduced a sublinear parallel algorithm to solve the MWBM problem which runs in $O(n^{\frac{2}{3}} \times \log^3 n \times \log(nC))$ time using $O(n^3/\log n)$ processing elements, where n is the number of nodes in the bipartite graph and C is the maximum weight of the graph edges. This is the fastest known weakly polynomial [1] parallel algorithm for the MWBM problem. Driscoll et al. [2] introduced an adjustable [2] parallel algorithm for the MWBM problem running in $O(nm/p)$ time, where m is the number of edges, p is the number of processing elements, and $p \leq m/(n \times \log n)$. Using the largest number of processing elements allowed, this algorithm runs in $O(n^2 \log n)$ time. Kim and Chwa [3] described a parallel algorithm to solve the MSBM problem in $O(n \times \log n \times \log\log n)$ time using $O(n^2 \times \lceil n/\log n \rceil)$ processing elements.

In this paper, we describe an adjustable parallel algorithm to solve the MWBM problem in $O(\frac{n}{\omega})$ time using $O(n^{max(2\omega,4+\omega)})$ processing elements, where $\omega \geq 1$ is an integer. Our algorithm is the fastest strongly polynomial parallel algorithm for the MWBM problem. This is also the first parallel adjustable algorithm for maximum weight bipartite matching problem in which the execution time can be reduced by an unbounded factor.

In the next section, we describe our general approach to finding parallel algorithms for the MWBM problem. In Section 3, we provide an example of the general approach. In Section 4, we give our adjustable parallel MWBM algorithm.

2 Definitions and preliminary results

Assume $G(U, V, E)$ is a weighted bipartite graph with $2n$ nodes, where $U = \{u_1, ..., u_n\}$ are the nodes on the left, $V = \{v_1, ..., v_n\}$ are the nodes on the right and $E = \{(u, v) | u \in U, v \in V\}$ are the edges of G. Edge (u_i, v_j) has weight of $w_{ij} \geq 0$. [3] We assume G is a fully-connected bipartite graph and that edges can have zero weight. Therefore, every matching can be extended to be a perfect matching. For a given graph, we only consider perfect matchings unless we indicate otherwise. A matching $M = \{(u_1, v_{x1}), (u_2, v_{x2}), ..., (u_n, v_{xn})\}$ can therefore be referred to more simply as a sequence of nodes in V: $(v_{x1} v_{x2} ... v_{xn})$.

[1] An algorithm is strongly(weakly) polynomial if its time complexity is bounded polynomially by the number of nodes and edges and it is (not) independent of the weights of edges.

[2] Adjustable algorithms' runtime can be reduced by a varying factor.

[3] We can assume $w_{ij} \geq 0$ without loss of generality; any negative weight can be made nonnegative by increasing all weights by the magnitude of the most negative weight.

Given a matching for a graph, a matched edge is an edge contained in the matching, and a free edge is an edge that is not in the matching. An alternating path (cycle) is a simple path (cycle) whose edges are alternately matched and free. The length of an alternating path is the number of edges it contains and its weight is the total weight of its free edges minus the weight of its matched edges. An augmenting path (cycle) is a positive weight alternating path (cycle).

Given two sets of edges E_1 and E_2, we define $E_1 \oplus E_2$ to be the symmetric difference of E_1 and E_2. Given a graph $G(U, V, E)$, a matching M, and a set of edges P that represent a set of disjoint alternating cycles, $M \oplus P$ is also a matching for graph G. Given a graph $G(U, V, E)$ and two matchings M_1 and M_2, $\Pi = M_1 \oplus M_2$ represents a set of disjoint cycles in G because every node in G is incident on exactly one edge in the perfect matching M_1 and exactly one edge in the perfect matching M_2. Therefore, every node in G is incident on exactly zero or two edges in Π. Note that for matchings M_1 and M_2, $M_1 \oplus M_2 = \Pi \Leftrightarrow M_1 \oplus \Pi = M_2$, which implies that for a matching M, there is a one-to-one correspondence between sets of disjoint alternating cycles and matchings. Note also that $M \oplus M = \emptyset$.

An MWBM algorithm for a graph with $2n$ nodes finds a matching of maximum weight in the space of $n!$ matchings. Given a graph, if we let S be the set of all matchings, we can find an efficient parallel algorithm for the MWBM problem as follows. We define a *basis* $B = \{b_1, ..., b_m\} \subseteq S$ for some m where b_i is a matching, and a *feasible set* of matchings for each b_i, with the following three basis properties:

- **Property 1.** $|B| = O(n^k)$, where k is a constant integer (i.e., the size of the basis is polynomial in n).
- **Property 2.** For each matching b_i of the basis, the feasible set for b_i includes b_i, and every matching in S is in at least one feasible set. Note that the feasible sets are not necessarily disjoint.
- **Property 3.** For each matching b_i of the basis, there is an algorithm *NC-search* whose time complexity is polylogarithmic in n, and which finds a maximum weight matching among the feasible matchings of b_i using a polynomial number of processing elements.

Lemma 1: There is an NC algorithm [4] for the MWBM problem if and only if it is possible to satisfy the basis properties.

Proof: If there exists an NC algorithm X that solves the MWBM problem, then the basis properties can be satisfied by setting the basis B equal to any matching b, setting the feasible set for b equal to S, and using X as the NC-search algorithm. To prove the second implication, we assume there exist a basis $B = \{b_1, ..., b_m\}$, feasible sets, and a search algorithm NC-search that satisfy the basis properties.

[4] An NC algorithm is a polylogarithmic time complexity parallel algorithm that uses polynomial number of processing elements.

We apply the search algorithm to every element of the basis in parallel. Each application of the algorithm selects the best matching from the matching's feasible set in polylogarithmic time. We can select the best matching from m matchings in polylogarithmic time using a polynomial number of processing elements. Therefore, there is an NC algorithm for the MWBM problem if and only if it is possible to satisfy the basis properties. □

Satisfying the basis properties is equivalent to proving that $NC = P$, i.e., that there is an efficient parallel algorithm (NC algorithm) for every problem that has a polynomial time complexity sequential solution. This is one of the most important open questions in the theory of algorithms. Conversely, proving that the basis properties cannot be satisfied is equivalent to proving that $NC \neq P$.

In Section 3, we define a basis, a search algorithm, and feasible sets that satisfy the second and third basis properties. However, since the number of elements in this basis is not polynomial in n, which violates the first basis property, this basis cannot be used to construct an NC algorithm for MWBM. We will use this basis to create an adjustable parallel MWBM algorithm which runs in $O(\frac{n}{\omega})$ time using $O(n^{max(2\omega,4+\omega)})$ processing elements (i.e., the runtime can be decreased by a constant factor $\omega \geq 1$).

3 Providing a basis, feasible sets and NC-search algorithm

Assume $G(U, V, E)$ is a bipartite graph with $2n$ nodes and $M = (v_1v_2...v_n)$ is a matching for G. We let $\Phi(M)$ be a feasible set of matchings for M such that $M' \in \Phi(M)$ iff $M \oplus M'$ includes at most one cycle.

Next, we create a basis β. The first element of β is $b_{11} = (v_1v_2...v_n)$. The second group of elements of β is created from b_{11} by exchanging all combinations of n nodes taken two at a time. For example, exchanging nodes v_1 and v_2 creates $b_{21} = (v_2v_1v_3...v_n)$ and exchanging v_1 and v_3 creates $b_{22} = (v_3v_2v_1...v_n)$, etc. The third group of elements of β is created from b_{11} by rotating all combinations of n nodes taken three at a time one node to the right. For example, rotating nodes v_1, v_2 and v_3 one node creates $b_{31} = (v_3v_1v_2v_4...v_n)$ and rotating nodes v_1, v_2 and v_4 one node creates $b_{32} = (v_4v_1v_3v_2...v_n)$, and so on. In general, the *i*th group of elements of β is created from b_{11} by rotating all combinations of n nodes taken *i* nodes at a time one node to the right, where $i \in \{2, 3, ..., \lfloor \frac{n}{2} \rfloor\}$. The size of β is $1 + \binom{n}{2} + \binom{n}{3} + ... + \binom{n}{\lfloor \frac{n}{2} \rfloor} = O(2^n)$. Note that when n is odd, the size of β is $2^{n-1} - 2n$.

Lemma 2: Given a graph G, a basis β, and feasible sets Φ, every matching is in the feasible set of some matching in β.

Proof: Let $M = b_{11}$ and M' be any matching. We showed that $M \oplus M'$ consists of a set of k disjoint cycles for some k. Based on this, we can show that M' is in the feasible set of one of the elements in the kth group of elements of the basis. We find this matching by first selecting one node from V for each of the k cycles in $M \oplus M'$. We start with matching M and rotate the selected nodes forward one position, leaving non-selected nodes unchanged. This gives a matching b_{ki} in β for some i. The edges $b_{ki} \oplus M'$ give a graph similar to $M \oplus M'$, except the k cycles have been joined to create one large cycle. Therefore, M' is in the feasible set of b_{ki}. Since M' was selected arbitrarily, the proof is complete. □

For a given graph G, we have described a basis β and feasible sets Φ that satisfy the second basis property. It remains to show that given a matching M, an NC algorithm exists that finds a maximum weight matching in the feasible set of M. In general, given a graph G and a matching M, M is a maximum weight bipartite matching if and only if no augmenting path exists [4]. We can find an MWBM by iteratively selecting a maximum weight augmenting path P and applying it to the current matching M.

Since $\Phi(M)$ includes all matchings M' such that $M \oplus M'$ is one cycle, every matching in $\Phi(M)$ can be constructed from M by xor-ing it with a single alternating cycle. A maximum weight bipartite matching within $\Phi(M)$ is therefore $M \oplus P$, where P is a maximum weight augmenting path. A maximum weight augmenting path can be found using the parallel shortest path algorithm described in [5]. The time complexity of this algorithm can be reduced to $O(\log n)$ using the $O(1)$ parallel selection algorithm described in [6] with $O(n^4)$ processing elements using a CRCW PRAM model. This $O(\log n)$ parallel algorithm is also described in [7].

4 Adjustable parallel MWBM algorithm

In this section, we describe our parallel algorithm for the MWBM problem. Given a graph G and a matching M, we have shown that we can construct a basis β and feasible sets Φ, and use an NC-search algorithm of the type described in Section 3 to find an MWBM in polylogarithmic time using $O(2^n)$ processing elements. This result assumes that the matching is chosen arbitrarily. In this section, we show that if M is selected more carefully, our algorithm only requires a polynomial number of processing elements.

Assume $G_t(U_t, V_t, E_t)$ is a weighted bipartite graph with $2t$ nodes and M_t is an MWBM for G_t. We construct a new graph G_{t+k} by adding k nodes to each side of G_t and adding edges to make G_{t+k} a fully-connected bipartite graph. Therefore, $G_{t+k}(U_{t+k}, V_{t+k}, E_{t+k})$ is a weighted bipartite graph with $2t + 2k$ nodes such that $U_{t+k} = U_t \bigcup \{u_{t+i} | i \in [1, k]\}$, $V_{t+k} = V_t \bigcup \{v_{t+i} | i \in [1, k]\}$, $E_{t+k} = E_t \bigcup E1 \bigcup E2$, where $E1 = \{(u_{t+i}, v) | \forall v \in V_{t+k}, i \in [1, k]\}$ and $E2 = \{(u, v_{t+i}) | \forall u \in U_{t+k}, i \in$

$[1,k]\}$. Edge (u_i, v_j) has weight of $w_{ij} \geq 0$.

Lemma 3: If M_t is an MWBM for graph G_t and $M = M_t \bigcup \{(u_{t+i}, v_{t+i}) | i \in [1,k]\}$ for $t \in [1, n-1]$, a maximum weight bipartite matching M_{t+k} exists for graph G_{t+k} such that $M \oplus M_{t+k}$ includes at most k disjoint cycles.

Proof: Since M_t is an MWBM for graph G_t, any set of augmenting cycles P for graph G_{t+k} with respect to matching M cannot contain a loop in G_t. Any cycle in P must contain at least one of the edges in $\{(u_{t+i}, v_{t+i}) | i \in [1,k]\}$, so P cannot contain more than k disjoint augmenting cycles. □

As described in Lemma 3, since $M \oplus M_{t+k}$ consists of k or fewer disjoint cycles, M_{t+k} is in the feasible set of one of the elements in groups 1 through k of basis β. The number of elements in this basis is $2^k - k$ because we only need to consider combinations of k nodes taken 2, 3,..., k nodes at a time, plus the original matching. Edges of M_t are included in all elements of the basis.

Assume $G(U, V, E)$ is a weighted bipartite graph with $m = 2n$ nodes. We reconstruct the graph by starting with a graph G_k which has $2k$ nodes from G. Starting with an arbitrary matching M_k for G_k, we construct the basis β_k as described in Section 3. We use an NC-search algorithm to find an MWBM within each feasible set of the basis in parallel. An MWBM for G_k is a matching with the maximum weight among the selected matchings. In the second step, graph G_{2k} is constructed from G_k by adding $2k$ additional nodes from G and their incident edges. We can find an MWBM for G_{2k} by constructing a basis for G_{2k} and finding a maximum weight augmenting path for each element of the basis. By continuing this way, an MWBM for G can be found in $\frac{n}{k}$ steps.

Theorem 4: Given a graph G with $2n$ nodes, the MWBM algorithm runs in $O(\frac{n}{\omega})$ time using $O(n^{max(2\omega, 4+\omega)})$ processing elements for a fixed $\omega \geq 1$.

Proof: The NC-search algorithm runs in $O(\log n)$ time using $O(n^4)$ processing elements. Therefore, in the ith step of the MWBM algorithm, an MWBM within each feasible set of the basis for graph G_{ik} can be found in $O(\log(i \times k))$ time using $O((i \times k)^4)$ processing elements. Since we have $2^k - k$ elements in the basis, we need $O((2^k - k) \times (i \times k)^4)$ processing elements to find an MWBM within each feasible set of the basis in parallel. Finding a maximum weight matching among $2^k - k$ selected matchings in constant time requires $O((2^k - k)^2)$ processing elements. Therefore, an MWBM for G_{ik} can be found in $O(\log(i \times k))$ time using $O(max((2^k - k)^2, (2^k - k) \times (i \times k)^4))$ processing elements. Since an MWBM for graph G can be found in $\frac{n}{k}$ steps, the overall complexity of the MWBM algorithm is $T = O(\sum_{i=1}^{\frac{n}{k}} \log(i \times k))$ using $O(max((2^k - k)^2, (2^k - k) \times n^4))$ processing elements. We have $T = O(\log((\frac{n}{k})! \times k^{\frac{n}{k}}))$. Since $x^{\frac{x}{2}} \leq x! \leq x^x$ for any $x \geq 1$, we have $O(\log x!) = O(x \log x)$. Therefore, $T = O(\frac{n}{k} \times \log n)$. For $k = 1$, the MWBM algorithm runs in $O(n \log n)$ time using $O(n^4)$ processing elements. We

set k such that the number of processing elements remains polynomial in n. Selecting $k = \lceil \omega \times \log n \rceil$ for $\omega \geq 1$ gives a time complexity of $O(\frac{n}{\omega})$ for the algorithm using $O(n^{max(2\omega, 4+\omega)})$ processing elements. □

Acknowledgements

This work was supported by CenSSIS the Center for Subsurface Sensing and Imaging Systems, under the Engineering Research Centers Program of the NSF (Award Number EEC-9986821).

References

[1] A. V. Goldberg, S. A. Plotkin, P. M. Vaidya, Sublinear-time parallel algorithms for matching and related problems, in: IEEE symposium on foundations of computer science, 1988, pp. 174–185.

[2] J. R. Driscoll, H. N. Gabow, R. Shrairman, R. E. Tarjan, An alternative to fibonacci heaps with applications to parallel computation, Communications of the ACM 31 (11) (1988) 1343–1354.

[3] T. Kim, K. Y. Chwa, An O(nlognloglogn) parallel maximum matching algorithm for bipartite graphs, Information Processing Letters 24 (1) (1987) 15–17.

[4] C. Berge, Graphs and hypergraphs, North-Holland, New York, 1973.

[5] E. Dekel, D. Nassimi, S. Sahni, Parallel matrix and graph algorithms, SIAM Journal on Computing 10 (4) (1981) 657–675.

[6] J. Jaja, An introduction to parallel algorithms, Addison-Wesley, 1992.

[7] M. Fayyazi, D. Kaeli, W. Meleis, Parallel maximum weight bipartite matching algorithms for scheduling in input-queued switches, in: IEEE International Parallel and Distributed Processing Symposium (IPDPS), 2004, pp. 26–30.

[8] T. H. Cormen, C. E. Leiserson, R. L. Rivest, Introduction to algorithms, The MIT press, 1999.