볼록 이분할 그래프에서 최대 매칭을 찾기 위한 불리안 회로[*]

이성희[0] 정유진
한국외국어대학교 컴퓨터공학과
{alexwannarun[0], chungyj}@hufs.ac.kr

A Boolean Circuit For Finding Maximum Matching In A Convex Bipartite Graph.

Sunghee Lee[0] Yoojin Chung
Dept. of Computer Eng, Hankuk Univ. of Foreign Studies

Summary

We' ve developed a Boolean circuit that finds a maximum matching in a convex bipartite graph. This circuit is designed in BC language that was created by K. Park and H. Park[1]. The depth of the circuit is $O(\log^2 n \cdot \log b)$ and the size is $O(bn^3)$. Our circuit gets a triple representation of a convex bipartite graph as its input and produces the maximum matching for its output. We developed some Boolean circuit design techniques that can be used for building other Boolean circuits.

1. Introduction

A convex bipartite graph G is a graph with three sets (A, B, E) where A and B are sets of vertices and E is the set of edges. This graph satisfies a following feature. If $(a_i, b_i) \in E$ and $(a_i, b_{i+k}) \in E$, then $(a_i, a_{i+q}) \in E$, $1 \leq q < k$. $F \subseteq E$ is a matching in the convex bipartite graph G if there are no edges that share an endpoint. F is a maximum matching in G if (a) F is a matching and (b) G contains no matching H such that $|H| > |F|$ ($|H|$ = number of edges in H).

A convex bipartite graph G = (A, B, C), A = $\{a_1, \ldots, a_n\}$, B = $\{b_1, \ldots, b_m\}$ can be represented by the set of triples:

$$T = \{(i, s_i, h_i) \mid 1 \leq i \leq n\}$$

In the triple representation, i stands for index in A and s_i and h_i represents smallest and highest index of vertices in B to which a_i is connected, respectively.

A job scheduling problem with n jobs, each requiring one unit of processing, can be transformed into finding a maximum matching in a convex bipartite graph. Dekel and Sahni[2] developed a parallel algorithm that finds the maximum matching that has complexity $O(\log^2 n)$. It is this algorithm that we implemented in our maximum matching Boolean circuit.

In chapter 2 of this paper, we will introduce the BC language, in which our circuit is designed.

Chapter 3 will discuss the internal structure of our maximum matching circuit. Finally, we will finish this paper by conclusion.

2. Boolean Circuit Language

A Boolean circuit is composed of inputs, outputs and logic gates. Wires connect the inputs, outputs, and logic gates. We can show a Boolean circuit as a graph if we take inputs, outputs, and logic gates as nodes and the wires as edges. Here are the definitions of the depth and the size of a circuit. The depth of a circuit is the longest path from an input to an output. The size of a circuit is the number of inputs, outputs, and logic gates in it.

P. Park and H. Park[1] developed a programming language called the BC Language that is similar to VHDL but more convenient to design parallel algorithms due to its general iterative and recursive structures and the ease of modular design..

P. Park and H. Park[1] also introduced a sorting circuit that has its depth $O(\log n + \log b)$ and its size $O(bn^2)$. The basic idea in the sorting circuit is to rank numbers and send them to the output through a Boolean circuit component ' DEMUX' . We made full use of the sorting circuit for our circuit that finds the maximum matching.

* "본 연구는 한국학술진흥재단 선도연구자지원사업(2003-041-D00483) 지원으로 수행되었음."

3. The Maximum Matching Circuit

We' ve divided Dekel and Sahni' s parallel maximum matching algorithm[2] into three parts. Similarly, our maximum matching circuit has three main parts. The first part divides the input, the set of triples T, and places them into leaves of a computation tree. The computation is a complete binary tree that is used for finding the maximum matching. Let P be any node of the computation tree. Each node P in the computation tree has a subset M(P) of A vertices that are available for matching[2]. The subset M is partitioned into three different subsets MAXM(P), I(P), and T(P). MAXM(P) is called the matched set and is a maximum cardinality subset of M(P) that can be matched with vertices in B. I(P) denotes the infeasible set and T(P) is called transferred set. Dekel and Sahni' s algorithm makes two passes over the computation tree. In the first pass MAXM(P) and T(P) for each vertex in the computation tree are found. This is the second part of Dekel and Sahni' s algorithm. The last part of this algorithm is the implementation of the second pass in Dekel and Sahni' s parallel maximum matching algorithm. In the second pass, MAXM' (P) is found for each vertex in the computation tree.

Appropriately, we divide this chapter into three sections such that, each section explains each part of our circuit.

3.1 First Part

In the first part, we sort our input by s_i and h_i $(1 \le i \le n)$ in the set of triples. Sorting is done using the sorting circuit[1] with some changes. Instead of the original Boolean expression used in the sorting circuit, we use following Boolean expression :

$$(s_i>s_j) \vee (s_i==s_j \wedge h_i>h_j) \vee (s_i==s_j \wedge h_i==h_j \wedge i>j)$$

Using the Boolean expression above will sort the input by s_i and h_i. Next, we find a set R [2], which has distinct s_i values. Finding distinct values are simple. Let $M(i) = s_i$ $(1 \le i \le n)$ and $M(0) = 0$. For every M(i), we compare it with M(i- 1) to see if they are different. If they are different, M(i) is included in the set R and not included, otherwise. Comparison is done using the 2^k-bit Comparator[1], which has the depth of O(k) and the size of $O(2^k)$. Comparing M(i) and M(i-1) is done in parallel so the depth of this function is O(log b) and the size is O(b). Using the set R, we find u and v for each node of the computation tree. Every node in the computation tree has its own u and v values and each value is used to separate the input. The circuit that performs the whole procedure of sorting and dividing has the depth O(log n + log b) and the size $O(bn^3)$.

3.2 Second Part - First Pass

Dekel and Sahni introduced a procedure FEAS[3] that finds the matched set for a node in the computation tree. FEAS uses an array DONE(i) $(1 \le i \le n)$ to find a matching. DONE(i) has the vertex number of B, to which the vertex a_i of A is matched. Dekel and Sahni obtained an equation for DONE(i), which is defined as follows[3]:

$$DONE(i) = \min_{0 \le j \le i}\{h'_j + i - j\}, \quad 1 \le i \le n$$

To implement the equation above, we used an ADDER[4] that produces the sum of two integers. The depth and the size of the ADDER is O(log b) and O(b), respectively. Since i - j can be computed at compile time of a synthesis tool of the BC language, [1] we only need an addition operation. $h'_j = \min\{h_j, v\}$[2]. Let DONE(0) = u - 1. For all DONE(i) $(1 \le i \le n)$, if DONE(i-1) <> DONE(i), then a_i is included in the matched set MAXM. If DONE(i-1) == DONE(i) and $h_i > v$[2], then a_i is included in the transferred set T. Comparing DONE(i-1) and DONE(i) takes the depth of O(log b) and the size of O(b).

For the leaves of the computation tree, we run FEAS directly. The Boolean circuit for FEAS has the depth of O(log n · log b) and the size of $O(bn^2)$. For a non-leaf node, MAXM and T is obtained from its children. Let L and R be the left and the right child of a node P. Let S be the matched set of M(P) after the execution of FEAS with T(L) ∪ MAXM(R). Then MAXM(P) = MAXM(L) ∪ S [2]. To merge MAXM(L) and S, we designed a component MERGE. The basic idea of merging two sets in a Boolean circuit is to put the two sets together without any loss and sort them. After the sorting, we cut half of the merged set because they are unnecessary. For example, assume that we merge MAXM(L) and S. |MAXM(L)| = n and |S| = n. Since, the maximum number of MAXM(L) and S is n, they almost always contain so called ' garbage number' , which are unnecessary numbers but have to be included because of the features of the Boolean circuit. (Usually, the garbage numbers are zeroes.) Let H be the merged set of MAXM(L) and S. Then |H| = 2n. It is guaranteed that at least n number of elements in H are garbage numbers. The depth of MERGE is O(log n + log b) and the size is $O(bn^2)$.

The first pass of Dekel and Sahni' s parallel maximum matching algorithm begins at leaves and moves upward to the leaves. Since the depth of the computation tree is O(log n) and at each depth, we run FEAS and MERGE in parallel, the depth of the first pass is $O(\log^2 n \cdot \log b)$ and the size is $O(bn^3)$.

3.3 Third Part – Second Pass

In the second pass, we find MAXM' (P) for each node of the computation tree. MAXM' (P) is the final matched set that will appear as an output of the circuit. This pass starts at the root and moves downward to the leaves.

If P is the root node, then MAXM' (P) = MAXM(P). Let P be any non-leaf node for which MAXM' (P) has been computed. Let L and R be the left and right children, respectively. Let V = {j | j ∈ MAXM' (P) and $s_j < v_L$}. Let W be the ordered set obtained by merging together V and MAXM(L). MAXM' (L) consists of the first min {|W|, $v_L - u_L + 1$} vertices in W[2]. MAXM' (R) = MAXM' (P) – MAXM' (L).

Hence, we made a component that finds the set V and also a component for performing the subtraction of two sets. When merging V and MAXM(L), some elements may coincide. Therefore, we need to sort them by i (Remember that the input is the set of triples T = {i, s_i, h_i}) and we need to cast off elements with same i. Finding the set V, merging MAXM(L) and V, and subtracting MAXM' (L) from MAXM' (P) has the depth of O(log n + log b) and the size of $O(bn^2)$. Therefore, the depth of the second pass is $O(\log^2 n)$ and the size is $O(bn^3)$.

After the second pass, we obtain the matched set of the convex bipartite graph. As we can see, the total depth of the maximum matching circuit is O(log2n · log b) and the size is $O(bn^3)$.

4. Conclusion

Although the Boolean circuit is simple and realistic, not many algorithms have been developed because of its awkwardness of programming. The BC language made it easy to design the Boolean circuit. Still, it is true that there are many restrictions and limitations for developing algorithms for the Boolean circuit. We hope that our circuit will contribute to the developing of other Boolean circuits. We are now searching for a way to make our circuit more simple.

References

[1] K. Park and H. Park, Boolean Circuit Programming : A New Paradigm to Design Parallel Algorithms,

[2] Dekel, E. and Sahni, S. A Parallel Matching Algorithm for Convex Bipartite Graphs and Applications to Scheduling, Journal Of Parallel And Distributed Computing 1, 185-205 (1984).

[3] Dekel, E., and Sahni, S. Parallel scheduling algorithms. Oper. Res. 31, 1 (1984), 24-49.

[4] T.H. Cormen, C.E. Leiserson, R.L. Rivest, Introduction to Algorithms, MIT Press 1990.