I/O Efficient Algorithms for the Minimum Cut Problem on Unweighted Undirected Graphs

Alka Bhushan and G. Sajith
Department of Computer Science and Engineering,
Indian Institute of Technology Guwahati, India-781039
{alka,sajith}@iitg.ac.in

Abstract. The problem of finding the minimum cut of an undirected unweighted graph is studied on the external memory model. First, a lower bound of $\Omega((E/V)\text{Sort}(V))$ on the number of I/Os is shown for the problem, where $V$ is the number of vertices and $E$ is the number of edges. Then the following are presented, for $M = \Omega(B^2)$, (1) a minimum cut algorithm that uses $O(c\log E(\text{MSF}(V,E)+\text{Sort}(V)))$ I/Os; here $\text{MSF}(V,E)$ is the number of I/Os needed to compute a minimum spanning tree of the graph, and $c$ is the value of the minimum cut. The algorithm performs better on dense graphs than the algorithm of [7], which requires $O(E+c^2V \log(V/c))$ I/Os, when executed on the external memory model. For a $\delta$-fat graph (for $\delta > 0$, the maximum tree packing of the graph is at least $(1+\delta)c/2$), our algorithm computes a minimum cut in $O(c\log E(\text{MSF}(V,E)+\text{Sort}(E)))$ I/Os. (2) a randomized algorithm that computes minimum cut with high probability in $O(c\log E \cdot \text{MSF}(V,E)+\text{Sort}(E)\log^2 V + \frac{1}{2}\text{Sort}(V)\log V)$ I/Os. (3) a $(2+\epsilon)$-minimum cut algorithm that requires $O((E/V)\text{MSF}(V,E))$ I/Os and performs better on sparse graphs than our exact minimum cut algorithm.

1 Introduction

The minimum cut problem on an undirected unweighted graph seeks to partition the vertices into two sets while minimizing the number of edges from one side of the partition to the other. While efficient in-core and parallel algorithms for the problem are known [4,10,11], this problem has not been explored much from the perspective of massive data sets. However, it is shown in [2] that the minimum cut can be computed in a polylogarithmic number of passes using only a polylogarithmic sized main memory on the stream sort model.

In this paper we consider the minimum cut problem on the external memory model proposed in [1]. To the best of our knowledge, this problem has so far not been investigated on the external memory model. This model has been used to design algorithms intended to work on large data sets that do not fit in the main memory. The external memory model defines the following parameters: $N$ (= $V+E$) is the input size, $M$ is the size of the main memory and $B$ is the size of a disk block. It is assumed that $2B < M < N$. In an I/O operation one block
of data is transferred between the disk and the internal memory. The measure of performance of an algorithm on this model is the number of I/Os it performs. The number of I/Os needed to read (write) \( N \) contiguous items from (to) the disk is \( \text{Scan}(N) = \Theta(N/B) \). The number of I/Os required to sort \( N \) items is \( \text{Sort}(N) = \Theta((N/B) \cdot \log_{M/B}(N/B)) \). For all realistic values of \( N, B, \) and \( M \), \( \text{Scan}(N) < \text{Sort}(N) \ll N \).

For an undirected unweighted graph \( G = (V, E) \), a cut \( X = (S, V - S) \) is defined as a partition of the vertices of the graph into two nonempty sets \( S \) and \( V - S \). An edge with one endpoint in \( S \) and the other endpoint in \( V - S \) is called a crossing edge of \( X \). The value \( c \) of the cut \( X \) is the total number of crossing edges of \( X \). The minimum cut problem is to find a cut of minimum value. We assume that the input graph is connected, since otherwise the problem is trivial.

A cut in \( G \) is \( \alpha \)-minimum, for \( \alpha > 0 \), if its value is at most \( \alpha \) times the minimum cut value of \( G \).

A tree packing is a set of spanning trees, each with a weight assigned to it, such that the total weight of the trees containing a given edge is at most one. The value of a tree packing is the total weight of the trees in it. A maximum tree packing is a tree packing of largest value. (When there is no ambiguity, we will use “maximum tree packing” to refer also to the value of a maximum tree packing, and a “minimum cut” to the value of a minimum cut.) A graph \( G \) is called a \( \delta \)-fat graph for \( \delta > 0 \), if the maximum tree packing of \( G \) is at least \( \left( \frac{1}{1 + \delta} \right) c^2 \), where \( c \) is the minimum cut [10].

As in [10], we say that a cut \( X \) \( k \)-respects a tree \( T \) (equivalently, \( T \) \( k \)-constrains \( X \)), if \( X \) cuts at most \( k \) edges of \( T \).

Several approaches have been tried in designing in-core algorithms for the minimum cut problem [6,7,9,10,14,17]. Significant progress has been made in designing parallel algorithms as well [3,10,11]. The current best in-core algorithm computes the minimum cut in \( O(E + c^2V \log(V/c)) \) time [7] and when executed on the external memory model, performs \( O(E + c^2V \log(V/c)) \) I/Os.

Karger [10] presents a near linear time randomised algorithm that computes a minimum cut with high probability in \( O(\min\{E \log^3 V, V^2 \log V\}) \) time. This algorithm also computes all \( \alpha \)-minimum cuts, for \( \alpha < 3/2 \). These cuts can be stored in a data structure that uses \( O(k + V \log V) \) space, where \( k \) is the total number of cuts found, in \( O(V^2 \log V) \) time. With this data structure, we can verify whether a given cut is \( \alpha \)-minimum in \( O(V) \) time. If we execute this algorithm on the external memory model then the I/O complexity of the cuts computation is \( O(\min\{E \log^3 V, V^2 \log V\}) \), the construction of the data structure is \( O(V^2 \log V) \), and the answering of a query is \( O(V) \).

A linear time algorithm for computing a \((2 + \epsilon)\)-minimum cut [12] is also known. This executes on the external memory model in \( O(V + E) \) I/Os.

The best known RNC algorithm presented in [10] can be simulated in \( O(\log^2 V \cdot \text{Sort}(V^2/\log^2 V)) \) I/Os in the external memory model using the PRAM simulation presented in [5]. The poly-logarithmic passes stream-sort algorithm presented in [2] implies an external memory algorithm that uses \( O(\text{Sort}(E) \cdot \text{polylog}(V)) \) I/Os under the assumption of \( M = \Omega(\text{polylog}(V)) \).