Chapter 59

Introduction

Overview. In earlier chapters, we have mostly covered techniques for solving problems on graphs that were developed in the context of sequential algorithms. Some of the algorithms we considered were parallel while others were not. For example, we saw that BFS has some parallelism since each level can be explored in parallel but there was no parallelism in DFS . There was no parallelism in Dijkstra's algorithm , but there was plenty of parallelism in the Bellman-Ford algorithm and Johnson's algorithm .

In this part of the book, we cover the "graph contraction" technique. This technique was specifically designed to be used in parallel algorithms and allows obtaining polylogarithmic span for certain graph problems. This chapter presents an overview of graph contraction. The following chapters present two specializations Edge Contraction and Star Contraction of graph contraction, and apply the technique to graph connectivity .

Video: Motivation. https://www.youtube.com/embed/bNO6LzXzcgQ

1 Preliminaries

Video: Graph Theory for Graph Contraction. https://www.youtube.com/embed/sP2T2IGBT1U

Note. The material here and the followup chapters on graph contraction relies on the graph terminology introduced in the background chapter on graph theory .

Definition 59.1 (Graph Partition)**.** Given a graph G, a ***graph partition*** of G is a collection of graphs

$$H_0 = (V_0, E_0), \ldots, H_{k-1} = (V_{k-1}, E_{k-1}),$$

such that $\{V_0, \ldots, V_{k-1}\}$ is a set partition of V and $H_0, \ldots, H_{k-1}$ are vertex-induced subgraphs of G with respect to $V_0, \ldots, V_{k-1}$.

We refer to each subgraph H_i as a ***block*** or ***part*** of G.

Definition 59.2 (Internal and Cut Edges)**.** Given a partition $H_0 = (V_0, E_0), \ldots, H_{k-1} = (V_{k_1}, E_{k-1})$ of a graph $G = (V, E)$, we define two kinds of edges: internal edges and cut edges.

- We call an edge $\{v_1, v_2\}$ an ***internal edge***, if $v_1 \in V_i$ and $v_2 \in V_i$. Note that $\{v_1, v_2\} \in E_i$.
- We call an edge $\{v_1, v_2\}$ a ***cut edge***, if $v_1 \in V_i$ and $v_2 \in V_j$ and $i \neq j$.

Exercise 59.1. One way to partition a graph is to make each connected component a block. What are the internal and cut edges in such a partition?

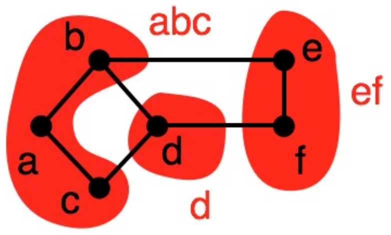
Solution. There are no cut edges between the partitions. All edges of the graph are internal edges.

Definition 59.3 (Partition Map)**.** We sometimes describe a graph partition with a tuple consisting of

1. a set of labels for the blocks, and
2. a ***partition map*** that maps each vertex to the label of its block.

The labels can be chosen arbitrarily but it is usually conceptually and computationally easier to use a vertex inside a block as a representative for that block.

Example 59.1. The partition $\{\{\texttt{a}, \texttt{b}, \texttt{c}\}, \{\texttt{d}\}, \{\texttt{e}, \texttt{f}\}\}$ of the vertices $\{\texttt{a}, \texttt{b}, \texttt{c}, \texttt{d}, \texttt{e}, \texttt{f}\}$, defines three blocks as the corresponding vertex-induced subgraphs .



The edges $\{\texttt{a}, \texttt{b}\}$, $\{\texttt{a}, \texttt{c}\}$, and $\{\texttt{e}, \texttt{f}\}$ are internal edges, and the edges $\{\texttt{c}, \texttt{d}\}$, $\{\texttt{b}, \texttt{d}\}$, $\{\texttt{b}, \texttt{e}\}$ and $\{\texttt{d}, \texttt{f}\}$ are cut edges.

By labeling the blocks 'abc','d' and 'ef', we can specify the graph partition with following partition map:

$$(\{\texttt{abc}, \texttt{d}, \texttt{ef}\}, \tag{59.1}$$
$$\{\texttt{a} \mapsto \texttt{abc}, \texttt{b} \mapsto \texttt{abc}, \texttt{c} \mapsto \texttt{abc}, \texttt{d} \mapsto \texttt{d}, \texttt{e} \mapsto \texttt{ef}, \texttt{f} \mapsto \texttt{ef}\}). \tag{59.2}$$

Instead of assigning a fresh label to each block, we can choose a representative vertex. For example, by picking a, d, and e as representatives, we can represent the partition above using the following partition map

$$(\{\mathtt{a},\mathtt{d},\mathtt{e}\}, \tag{59.3}$$
$$\{\mathtt{a}\mapsto\mathtt{a},\mathtt{b}\mapsto\mathtt{a},\mathtt{c}\mapsto\mathtt{a},\mathtt{d}\mapsto\mathtt{d},\mathtt{e}\mapsto\mathtt{e},\mathtt{f}\mapsto\mathtt{e}\}). \tag{59.4}$$

2 Graph Contraction

Video: Graph Contraction Technique. https://www.youtube.com/embed/jaNhjt8F9ww

Graph contraction is a contraction technique for computing properties of graphs in parallel. As a contraction technique, it is used to solve a problem instance by reducing it to a smaller instance of the same problem.

Graph contraction plays important role in parallel algorithm design, because divide-and-conquer can be difficult to apply to graph problems efficiently. Divide-and-conquer techniques usually require partitioning graphs into smaller graphs in a balanced fashion such that the number of cut edges is minimized. Because graphs can be highly irregular, they can be difficult to partition. In fact, graph partitioning problems are typically NP-hard.

Quotient Graph. The key idea behind graph contraction is to contract the input graph to a smaller ***quotient graph***, solve the problem on the quotient graph, and then use that solution to construct the solution for the input graph. We can specify this technique as an inductive algorithm-design technique as follows.

Definition 59.4 (Graph-Contraction Technique)**.** Graph contraction technique has a base case and an inductive case. Each application of the inductive step is called a ***round*** of graph contraction. In a graph contraction, rounds are repeated until the graph is small, e.g., the graph has no remaining edges.

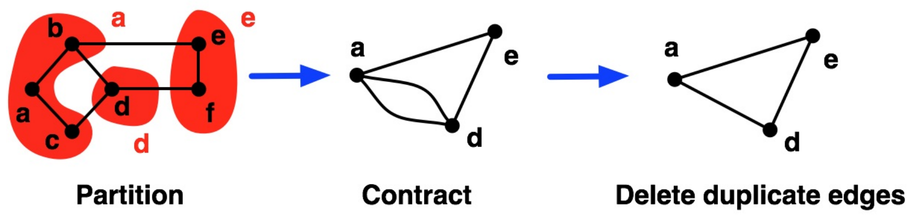
Base case: If the graph is small (e.g., it has no edges), then compute the desired result.

Inductive case:

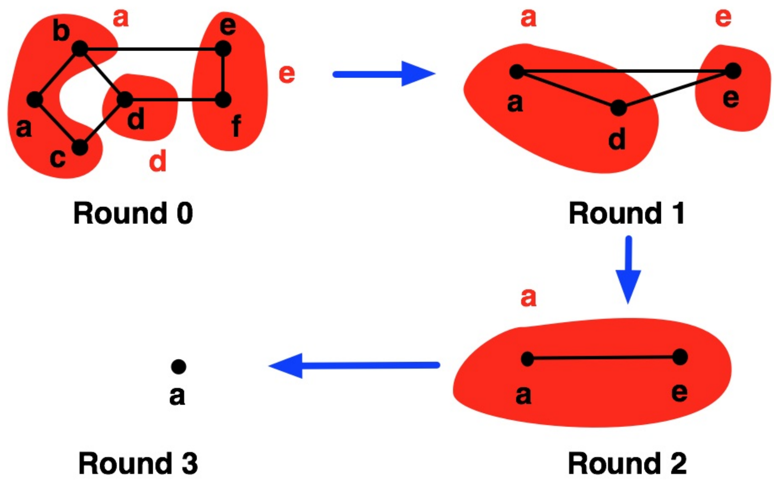
- **Contraction step:** contract the graph into a smaller quotient graph.
  - Partition the graph into blocks.
  - Contract each block to a single super-vertex.
  - Drop internal edges.
  - Reroute cut edges to corresponding super-vertices.
- **Recursive step:** Recursively solve the problem for the quotient graph.

- **Expansion step:** By using the result for the quotient graph, compute the result for the input graph.

Example 59.2. One round of graph contraction:



Contracting a graph down to a single vertex in three rounds:



Construction of the Quotient Graph. To construct a quotient graph, we represent each block in the partition with a vertex, which we call a ***super-vertex***. We then "map" the edges of the graph to the quotient graph. Consider each edge (u, v) in the graph.

- If the edge is an internal edge, then we skip the edge.
- If the edge is a cut edge, then we create a new edge between the super-vertices representing the blocks containing u and v.

Because there can be many cut edges between two blocks, this approach may create multiple edges between two super-vertices. We may remove duplicate edges or leave them in the graph, in which case we would be working with multigraphs. Either approach has its benefits and may, depending on the application, be preferable over the other.

Important. Graph contraction is guided by a graph partition, which leads to blocks whose vertices are disjoint. During the construction of the quotient graph, each vertex in the graph is therefore mapped to a unique vertex in the quotient graph.

Applying Graph Contraction. The ultimate goal of graph contraction technique is to reduce the size of the graph by a constant fraction (possibly in expectation) at each round of contraction. Depending on the graphs of interest many different graph-partition techniques can be used to achieve this goal. As described, the graph-contraction technique is generic in the kind of graph partition used. In the following chapters on Edge Contraction and Star Contraction we consider two techniques, edge partitioning and star partitioning, and the resulting graph-contraction algorithms.