Chapter 60

Edge Contraction

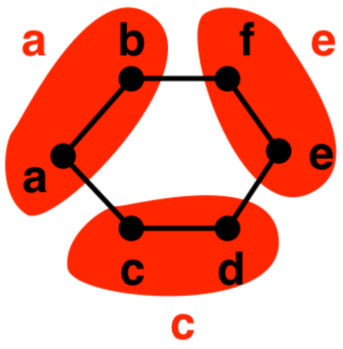
This section describes the edge partition and edge contraction. Edge contraction is an instance of a graph-contraction where blocks being contracted correspond to edges.

Video: Edge Contraction. https://www.youtube.com/embed/Bcbq9-dqWPo

1 Edge Partition

Definition 60.1 (Edge Partition)**.** An ***edge partition*** is a graph partition where each block is either a single vertex or two vertices connected by an edge.

Example 60.1. An example edge partition in which every block consists of two vertices and an edge between them.



Exercise 60.1. Give an example graph whose edge partitions always contain a block that consists of a single vertex.

Solution. Any graph which has an isolated vertex, i.e., a vertex with no incident edges would work.

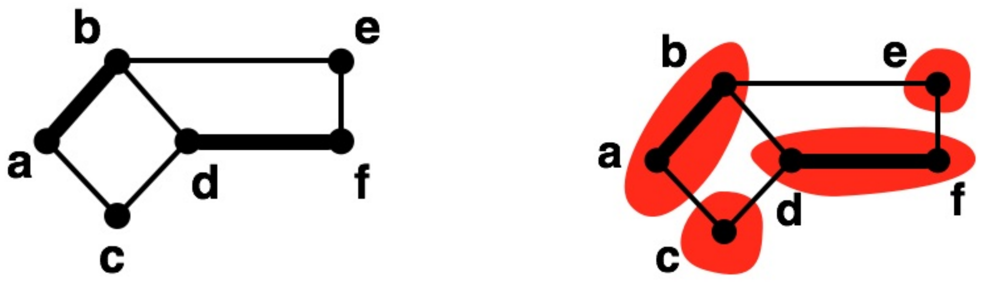
Edge Partitions and Vertex Matching. Finding an edge partition of a graph is closely related to the problem of finding an independent edge set or a vertex matching. A vertex matching in a graph is a subset of the edges that do not share an endpoint, i.e., no two edges are incident on the same vertex. We can construct an edge partition from a vertex matching by constructing a block for each edge in the matching and placing all the remaining vertices into their own singleton blocks.

Definition 60.2 (Vertex Matching)**.** A ***vertex matching*** for an undirected graph $G = (V, E)$ is a subset of edges $M \subseteq E$ such that no two edges in M are incident on the same vertex. In other words, each vertex in M have degree at most 1.

The problem of finding the largest vertex matching for a graph is called the ***maximum vertex matching*** problem.

Algorithms for Maximum Vertex Matching. Maximum Vertex Matching is a well-studied problem and many algorithms have been proposed, including one that can solve the problem in $O(\sqrt{|V|}|E|)$ work.

Example 60.2 (Vertex Matching)**.** A vertex matching for a graph (highlighted edges) and the corresponding blocks.



The vertex matching defines four blocks (circled), two of them defined by the edges in the matching, $\{\texttt{a}, \texttt{b}\}$ and $\{\texttt{d}, \texttt{f}\}$, and two of them are the unmatched vertices c and e.

Note. For edge contraction, we do not need a maximum matching but one that it is sufficiently large.

Algorithm 60.3 (Greedy Vertex Matching)**.** We can use a greedy algorithm to construct a vertex matching by iterating over the edges while maintaining an initially empty matching M. The greedy algorithm considers each edge and proceeds as follows:

- if no edge in M is already incident on its endpoints, then the algorithm adds the edge to M,

- otherwise, the algorithm tosses away the edge.

Exercise 60.2. Does the greedy vertex matching algorithm always returns a maximum vertex matching?

Solution. No.

Exercise 60.3. Prove that the greedy algorithm finds a solution within a factor two of optimal.

Exercise 60.4. Is the greedy algorithm parallel?

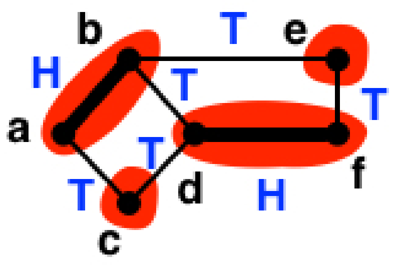
Solution. The greedy algorithm is sequential, because each decision depends on previous decisions.

Randomized Symmetry Breaking. To find a vertex matching in parallel, we want to make local and parallel decisions at each vertex independent of other vertices. One possibility is for each vertex to select one of its neighbors arbitrarily but in some deterministic fashion. Such a selection can be made in parallel but there is one problem: multiple vertices might select the same vertex to match with.

We therefore need a way to *break the symmetry* that arises when two vertices try to match with the same vertex. To this end, we can use randomization. There are several different ways to use randomization but they are all essentially the same and yield the same bounds with a constant factor.

Algorithm 60.4 (Parallel Vertex Matching)**.** To compute a vertex matching the ***parallel vertex matching algorithm*** flips a coin for each edge in parallel. The algorithm then selects an edge (u, v) and matches u and v, if the coin for the edge comes up heads and all the edges incident on u and v flip tails.

Example 60.3 (Parallel Vertex Matching)**.** An example run of the parallel vertex matching algorithm.



Exercise 60.5. Prove that the algorithm produces a vertex matching, i.e., it guarantees that a vertex is matched with at most one other vertex.

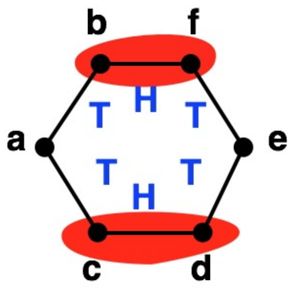
1.1 Analysis of Parallel Edge Partition

We analyze the effectiveness of the parallel edge partition algorithm in selecting a matching that consists of as many edge blocks (equivalently as few singleton blocks) as possible. We first consider cycle graphs and then general graphs.

1.1.1 Cycle Graphs

Probability of Selecting an Edge in a Cycle. We want to determine the probability that an edge is selected in a cycle, where each vertex has exactly two neighbors. Because the coins are flipped independently at random, and each vertex has degree two, the probability that an edge picks heads and its two adjacent edges pick tails is $\frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{8}$.

Example 60.4 (Edge Partition of a Cycle)**.** A graph consisting of a single cycle.



Each edge flips a coin that comes up either heads (H) or tails (T). We select an edge if it turns up heads and all other edges incident on its endpoints are tails. In the example the edges $\{\mathtt{c}, \mathtt{d}\}$ and $\{\mathtt{b}, \mathtt{f}\}$ are selected.

Expected Number of Edges Selected. To analyze the number of edges (blocks) selected in expectation, let R_e be an indicator random variable denoting whether e is selected or not, that is $R_e = 1$ if e is selected and 0 otherwise. Recall that the expectation of indicator random variables is the same as the probability it has value 1 (true). Therefore we have $E[R_e] = 1/8$. Thus summing over all edges, we conclude that expected number of edges selected is $\frac{m}{8}$ (note, $m = n$ in a cycle graph). Thus we conclude that in expectation, a constant fraction $\left(\frac{1}{8}\right)$ of the edges are selected to be their own blocks.

Exercise 60.6. Modify the algorithm to improve the expected number of edges selected.

Improving the Expectation. There are several ways to improve the number of select edges. One way is for each vertex to pick one of its neighbors and to select an edge (u, v) if it was picked by both u and v. In the case of a circle, this increases the expected number of selected edges to $\frac{m}{4}$.

Another way is let each edge pick a random number in some range and then select and edge if it is the local maximum, i.e., it picked the highest number among all the edges incident on its end points. This increases the expected number of selected edges to $\frac{m}{3}$.

1.1.2 Star Graphs

Limitation of Edge Partition. Although our edge partition algorithm works quite well on cycle graphs, it does not work well for arbitrary graphs. The problem is in an edge partition, only one edge incident on a vertex can be its own block. Therefore if there is a vertex with high degree, then only one of its edges can be selected. Star graphs are a canonical example of such graphs, although there are many others.

Definition 60.5 (Star Graph)**.** A ***star graph*** $G = (V, E)$ is an undirected graph with

- a ***center*** vertex $v \in V$, and
- a set of edges E that attach v directly to the rest of the vertices, called ***satellites***, i.e., $E = \{\{v, u\} : u \in V \setminus \{v\}\}$.

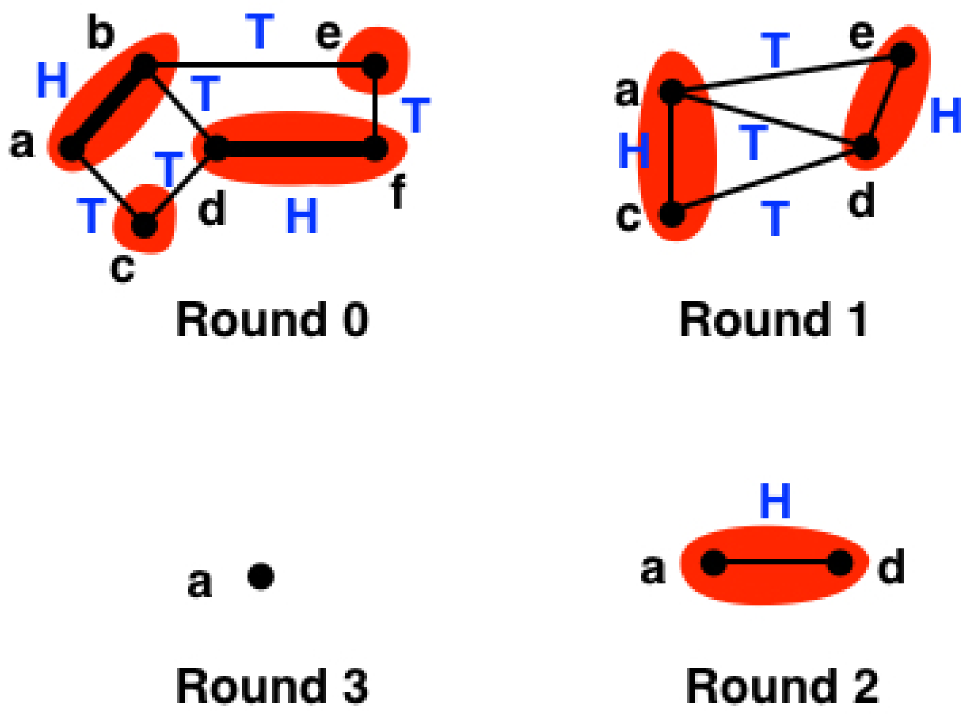
Example 60.5. The following are star graphs:

- a single vertex, and
- a single edge.

2 Edge Contraction

Algorithm 60.6 (Parallel Edge Contraction)**.** Parallel edge contraction algorithm is a specialization of the graph contraction technique that uses the parallel vertex matching algorithm to partition the graph for contraction.

Example 60.6 (Edge contraction)**.** An example parallel edge contraction illustrated.



Analysis of Edge Contraction. The analysis of edge partition established that using edge partition, we are able to select in expectation $\frac{1}{8}$ of the edges as their own blocks if the graph is a cycle. Therefore, after one round of contraction, the number of vertices and edges in a cycle decrease by an expected constant fraction.

In randomized algorithms chapter , we showed that if each round of an algorithm reduces the size by a constant fraction in expectation, and if the random choices in the rounds are independent, then the algorithm will finish in $O(\lg n)$ rounds with high probability. Recall that all we needed to do is multiply the expected fraction that remain across rounds and then use Markov's inequality to show that after some $k \lg n$ rounds the probability that the problem size is a least 1 is very small. For a cycle graph, this technique leads to an algorithm for graph contraction with linear work and $O(\lg^2 n)$ span.

Analysis for Star Graphs. Edge contraction works quite poorly on other graphs such as star graphs, and can result in a partition with many singleton blocks. This is because in an edge partition, only one of the edges incident on a vertex can be its own block (Section 1.1.2), leading to a poor contraction ratio. Edge contraction therefore is not effective for general graphs.