1 Closure Problem

- *Algorithm:* Max-weight closure finding (algo. 1)
- *Input:* A directed graph
- *Complexity:* $\mathcal{O}(|V| \cdot |E|^2)$
- *Data structure compatibility:* directed graph
- *Common applications:* open pit mining, military application, job scheduler, network flow design

Closure Problem

Given a directed graph, with each vertex has some weight. Find the maximum-weight directed graph that has no edges start within the graph and end outside the graph.

Description

1. Closure: A closure is defined as a set of vertices, for a directed graph, there is no outgoing edges. And this means that there shall only be edges pointing from any node outside the closure to the nodes inside the closure.

2. Problem clarification: In this project I will focus on the maximum weight closure problem, that is to find the closure with the greatest sum of vertices' weight.

3. Problem solution: To solve the closure problem, it could be **reduced** into a maximum flow problem in the following way, which will be discussed in algo 1.

   (a) Add two nodes, s and t to the original graph.

   (b) For all nodes with positive weight, add a directed edge from s to the node and set the capacity of the edge to be the absolute value of node's weight.

   (c) For all nodes with negative weight, add a directed edge from the node to t and set the capacity of the edge to be the absolute value as node's weight.

   (d) For all other existed edges, set their capacity to be ∞.

Algorithm 1: Reduction to the max flow

Input : A directed graph $G = (V, E)$
Output: A new directed graph after redution

Function `reduce(`G`)`**:**
    $H \leftarrow \{\}$;
    $H.V = G.V + \{s\} + \{t\}$;
    for *All nodes n_1 in $G.V$ with $G.V.weight > 0$* **do**
        Add edge $< s, n_1 >$ to $H.E$;
        $< s, n_1 > .capacity = n_1.weight$;
    end for
    for *All nodes n_2 in $G.V$ with $G.V.weight < 0$* **do**
        Add edge $< n_2, t >$ to $H.E$;
        $< s, n_1 > .capacity = n_1.weight$;
    end for
    for *All other edges in $G.E$* **do**
        $H.E.capacity = \infty$;
    end for
    /* Apply Edmond Karp algorithm on H, to get the max cut. Edmond Karp algorithm already discussed in class, no need to write it out here. */
    `edmond_karp(`H`)`;
    return $H.left - \{s\}$
end

After performing this algorithm, the returned part are the nodes within the closure.

Time complexity

For this problem, since the original problem can be reduced in a maximum flow problem within polynomial time $\mathcal{O}(|V|+|E|)$, so there are in the same complexity space, and the total time complexity is decided by the time complexity of the Edmond Karp algorithm, which is given by

$$\mathcal{O}(|V| \cdot |E|^2)$$

References

- Schrijver, A. (2002). "On the history of the transportation and maximum flow problems". Mathematical Programming. 91 (3): 437445.
- Ahuja, Ravindra K.; Magnanti, Thomas L.; Orlin, James B. (1993), "19.2 Maximum weight closure of a graph", Network flows, Englewood Cliffs, NJ: Prentice Hall Inc., pp. 719724, ISBN 0-13-617549-X, MR 1205775.
- Picard, Jean-Claude (1976), "Maximal closure of a graph and applications to combinatorial problems", Management Science, 22 (11): 12681272.