OS202 – TD2: Ford Fulkerson

Santiago Florido Gomez

***Abstract*—This report presents Ford–Fulkerson applications on flow networks, including maximum-flow computations, minimum cuts, and unit-capacity models for matching and disjoint-path problems.**

I. Introduction

In the present document, we will present applications of flow network modeling aimed at optimizing the circulation of resources in systems with limited capacities, mainly by applying the Ford–Fulkerson algorithm. With this objective, we seek to find maximum flows (i.e., the maximum throughput), detect bottlenecks, and enable assignment and matching in resource-allocation problems, such as the lock-and-key problem for opening a safe. Finally, we consider the use of Ford–Fulkerson on graphs with unit capacities in order to find disjoint routes in systems that require redundancy and robustness.

II. Ford–Fulkerson

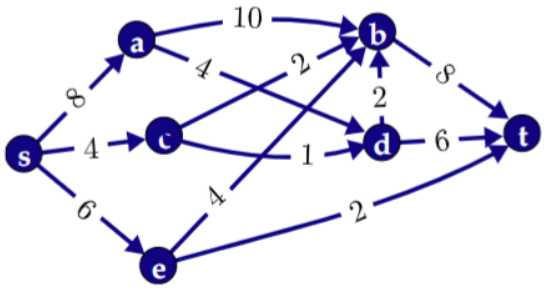

Fig. 1. Flow network for Ford–Fulkerson (arc labels are capacities).

Capacities

$$s \to a : 8, \quad s \to c : 4, \quad s \to e : 6,$$
$$a \to b : 10, \quad a \to d : 4,$$
$$c \to d : 1,$$
$$e \to t : 2,$$
$$d \to t : 6,$$
$$b \to t : 8.$$

Augmenting paths using ontly paths in alphabetical order

1) $P_1 : s \to a \to b \to t, \quad \Delta_1 = \min(8, 10, 8) = 8$
2) $P_2 : s \to c \to d \to t, \quad \Delta_2 = \min(4, 1, 6) = 1$
3) $P_3 : s \to e \to t, \quad \Delta_3 = \min(6, 2) = 2$

Final flow (nonzero arcs)

$$f(s,a) = 8, \quad f(a,b) = 8, \quad f(b,t) = 8,$$
$$f(s,c) = 1, \quad f(c,d) = 1, \quad f(d,t) = 1,$$
$$f(s,e) = 2, \quad f(e,t) = 2.$$

$$|f| = 11.$$

By applying the Ford–Fulkerson algorithm, the maximum flows that can be obtained while taking the capacities into account are as follows:

- `'s' - 'a' (8.0)`
- `'s' - 'c' (3.0)`
- `'s' - 'e' (4.0)`
- `'a' - 'b' (4.0)`
- `'a' - 'd' (4.0)`
- `'b' - 't' (8.0)`
- `'c' - 'b' (2.0)`
- `'c' - 'd' (1.0)`
- `'d' - 't' (5.0)`
- `'e' - 'b' (2.0)`
- `'e' - 't' (2.0)`

To identify an appropriate way to improve the system's minimum flow, it is useful to analyze the minimum cuts resulting from applying Ford–Fulkerson, which are:

$$S^* = \{s, a, b, c, e\}, \qquad T^* = \{d, t\}.$$

Given these minimum cuts, it becomes clear that it is advisable to increase the capacity of the arcs that go from S^* to T^*, since those arcs act as bottlenecks that restrict the flow.

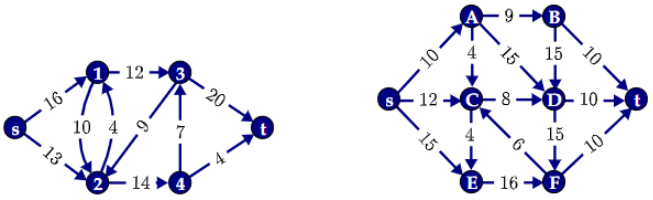

Fig. 2. Additional Ford–Fulkerson examples (arc labels are capacities).

The Ford–Fulkerson algorithm was applied to the two networks shown in Figure 2. The resulting maximum flows and minimum cuts are:

Graph 1 (left network). Maximum flow value: $|f| = 23$. Nonzero flow arcs:

- `'s' - '1' (12.0)`

- 's' - '2' (11.0)
- '1' - '3' (12.0)
- '2' - '4' (11.0)
- '3' - 't' (19.0)
- '4' - '3' (7.0)
- '4' - 't' (4.0)

Minimum cut: $S^* = \{s, 1, 2, 4\}$, $T^* = \{3, t\}$.

Graph 2 (right network). Maximum flow value: $|f| = 30$. Nonzero flow arcs:

- 's' - 'A' (10.0)
- 's' - 'C' (10.0)
- 's' - 'E' (10.0)
- 'A' - 'B' (8.0)
- 'A' - 'D' (2.0)
- 'C' - 'B' (2.0)
- 'C' - 'D' (8.0)
- 'E' - 'F' (10.0)
- 'B' - 't' (10.0)
- 'D' - 't' (10.0)
- 'F' - 't' (10.0)

Minimum cut: $S^* = \{s, C, E, F\}$, $T^* = \{A, B, D, t\}$.

I would propose solving the problem as follows: define a source node s from which arcs of capacity 1 go to each key that can be used, since each key can only be used once. Then, each key is connected by arcs of capacity 1 to the drawers it can open. Finally, from those drawers, define edges of capacity 1 to the sink t. I would implement a Ford–Fulkerson algorithm in order to identify the maximum flow, i.e., the maximum number of drawers that can be opened simultaneously given the available keys.

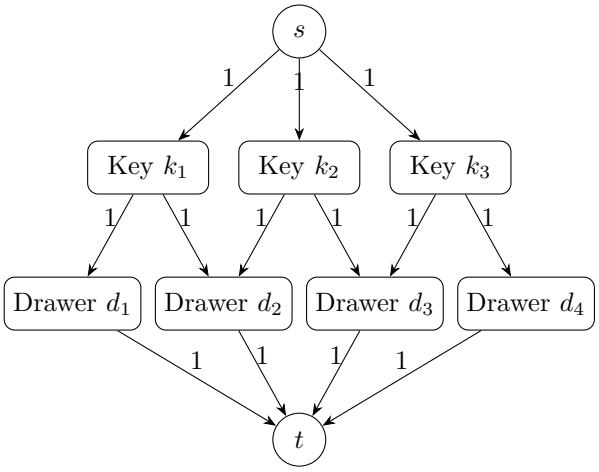

Fig. 3. Flow network model: each key can be used at most once, and each opened drawer contributes one unit of flow.

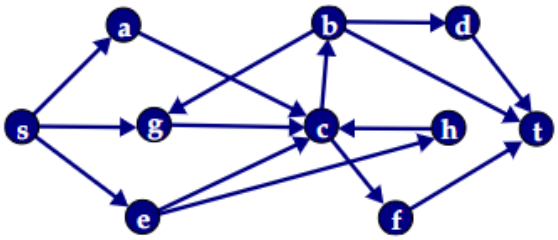

Fig. 4. Graph used for the disjoint paths problem.

The *disjoint paths* problem is addressed analogously to the keys-and-drawers formulation: it is transformed into a **maximum flow** problem and solved with **Ford–Fulkerson** using **unit capacities**. These capacities ensure that the resulting paths are **unique** and **disjoint** according to the chosen criterion.

There are two main levels of separation:

- **Arc-disjoint:** assign capacity 1 to every arc. This prevents any additional unit of flow from reusing a saturated arc, so the set of paths induced by the maximum flow is edge-disjoint.
- **Vertex-disjoint:** in addition to unit capacities, forbid two paths from sharing an intermediate vertex. This is done via *node splitting*: each vertex v is replaced by two subnodes v_{in} and v_{out}, connected by an arc $(v_{\text{in}}, v_{\text{out}})$ of capacity 1. Then, all incoming edges to v are redirected to v_{in} and all outgoing edges leave from v_{out}. This enforces a unit capacity per vertex.

Therefore, to solve the instance in Figure 4, both strategies must be implemented: the first to obtain arc-disjoint paths, and the second (with node splitting) to obtain vertex-disjoint paths.

The Ford–Fulkerson results for the graph in Figure 4 are:

Arc-disjoint (unit capacities on arcs). Maximum flow value: $|f| = 3$. Nonzero flow arcs:

- 's' - 'a' (1.0)
- 's' - 'g' (1.0)
- 's' - 'e' (1.0)
- 'a' - 'b' (1.0)
- 'g' - 'c' (1.0)
- 'e' - 'h' (1.0)
- 'b' - 'd' (1.0)
- 'c' - 'f' (1.0)
- 'h' - 't' (1.0)
- 'f' - 't' (1.0)
- 'd' - 't' (1.0)

Vertex-disjoint (node splitting with unit capacities). Maximum flow value: $|f| = 3$. Nonzero flow arcs:

- 's' - 'a_in' (1.0)
- 's' - 'g_in' (1.0)
- 's' - 'e_in' (1.0)
- 'a_in' - 'a_out' (1.0)

- 'a_out' - 'b_in' (1.0)
- 'g_in' - 'g_out' (1.0)
- 'g_out' - 'c_in' (1.0)
- 'e_in' - 'e_out' (1.0)
- 'e_out' - 'h_in' (1.0)
- 'b_in' - 'b_out' (1.0)
- 'b_out' - 'd_in' (1.0)
- 'c_in' - 'c_out' (1.0)
- 'c_out' - 'f_in' (1.0)
- 'h_in' - 'h_out' (1.0)
- 'h_out' - 't' (1.0)
- 'f_in' - 'f_out' (1.0)
- 'f_out' - 't' (1.0)
- 'd_in' - 'd_out' (1.0)
- 'd_out' - 't' (1.0)