Informatica - Area scientifica
Dipartimento di Scienze matematiche, informatiche e multimediali
Università di Udine

Progetto di Architetture Parallele

Rasera Giovanni (143395)

Anno accademico 2024/2025

Indice

1 Descrizione del problema affrontato

Scrivere un programma che, dato un nodo x in V (diverso da s), determini un sottoinsieme D di nodi tale che:

- 1) la sorgente s non appartiene a D e il nodo x non appartiene a D
- 2) ogni cammino diretto da s a x passa per almeno un nodo in D
- 3) D è minimale rispetto alla cardinalità

Un algoritmo che risolve un problema analogo è l'algoritmo MaxFlowMinCut con la differenza che permette di trovare gli archi che rappresentano il massimo flusso che può attraversare un grafo.

1.1 Una visualizzazione del problema in un contesto reale

Scenario bellico: Gestione strategica delle vie di comunicazione
In un contesto bellico, un comandante deve impedire al nemico di trasportare rifornimenti dalla loro base principale (sorgente s) a un obiettivo strategico (t) attraverso una rete stradale.
La rete è rappresentata come un grafo, dove:

- I nodi rappresentano le intersezioni stradali.
- Gli archi rappresentano le strade che collegano queste intersezioni, con capacità che indicano la quantità massima di rifornimenti che possono attraversare ciascuna strada.

Obiettivo:
Individuare il set minimo di strade (arco taglio) che, se bloccate o distrutte, interromperebbero efficacemente tutti i rifornimenti dal punto s al punto t.

2 Metodoligia adottata per la soluzione

2.1 Sfruttare l'algoritmo MinCut per risolvere il problema richiesto

Come suggerito da: Professor Andrea Formisano
Un nodo v in G diventa v', composto da (v'even -> v'odd).

- Tutti gli archi originali in(v) che arrivavano a v sono ora collegati a v'even con costo infinito.
- Tutti gli archi originali out(v) che partivano da v partono ora da v'odd con costo infinito.
- Il costo da v'odd a v'even è pari a 1.

Esempio:

- G : 0 -> 1 -> 2 -> 3
  - / 0 -> 2
- G' : (0 -> 1) -> (2 -> 3) -> (4 -> 5) -> (6 -> 7)
  - / 1 -> 4

Il risultato del MinCut nel grafo G' darà come risultato il taglio di archi di costo 1 e indicheranno i nodi da eliminare con un semplice calcolo nodo_da_eliminare = (v'even / 2).

Un esempio pratico:

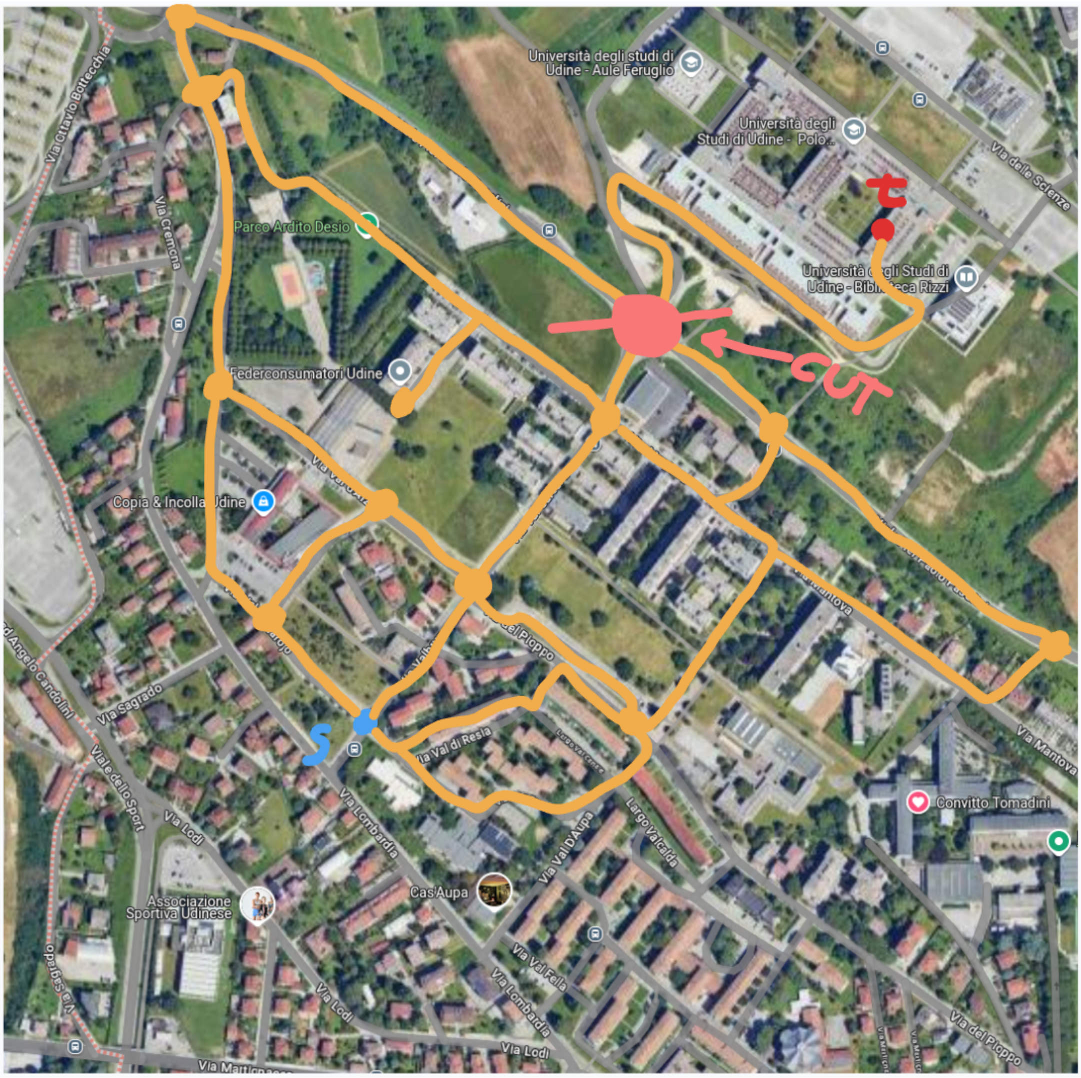

Figura 2.1: Esempio di difesa dell'aula A036

In questo caso il minimo taglio di un nodo che si può fare per separare (s) e (t) è il taglio del nodo rosso in figura.

2.2 MinCutMaxFlow

Esistono diversi algoritmi di MinCutMaxFlow, i due che vengono presi in considerazione sono:

- Ford-Fulkerson
- Goldberg-Tarjan

2.2.1 Ford-Fulkerson

Di seguito viene presentato lo peseudo codice.

Ford-Fulkerson

```
auto minCutMaxFlow(graph,rGraph, source, to){
    // init structs
    //...

    while(bfs(rGraph, parent, source, to)){
        path_flow = INFINITY;
        // a path from to -> source
        for (v = to; v != source; v = parent[v]){
            u = parent[v];
            path_flow = min(path_flow, rGraph[u][v]);
        }

        // update the flow in the residual graph
        for (v = to; v != source; v = parent[v]){
            u = parent[v];
            rGraph[u][v] -= path_flow;
            rGraph[v][u] += path_flow;
        }
    }

    // run dfs on the residual graph
    dfs(rGraph, visited, source);

    // here you can calculate nodes to cut
}
```

Utilizza BFS e DFS per determinare se l'ultimo nodo può essere raggiunto. Tale algoritmo è difficile da parallelizzare perché BFS e DFS sono due algoritmi notoriamente complicati da parallelizzare.

2.2.2 Goldberg-Tarjan

Di seguito viene presentato lo peseudo codice.

Golberg-Tarjan

```
auto minCutMaxFlow(source, sink){
    preflow(source);

    while(any_active()) {
        push(active_node);
        relabel(active_node);
    }
}
```

L'idea che sta alla base è quella di cercare un nodo con delle caratteristiche particolari da cui è possibile far passare del flusso, e procedere in tal modo fino a

quando non è più possibile inserire del flusso.
Per capire l'algoritmo ovviamente c'è bisogno della funzione di push e relabel.

Push e Relabel

```
auto push(x){
    if(active(x)){
        for (y=neighbor(x)) {
            if (height(y) == height(x)-1) {
                flow = min( capacity(x,y), excess_flow(x));

                // update the flow
                excess_flow(x) -= flow;
                excess_flow(y) += flow;
                capacity(x,y) -= flow;
                capacity(y,x) += flow;
            }
        }
    }
}

auto relabel(x){
    if (active(x)) {
        my_height = V;

        // init to max height
        for (y=neighbor(x)){
            if capacity(x,y) > 0 {
                my_height = min(my_height, height(y)+1);
            }
        }
        height(x) = my_height;
    }
}
```

Il nodo x è attivo: se capacity(x) $>$ 0 e height(x) $<$ HEIGHT_MAX:
Nodo attivo x:

- può spingere verso il vicino y: se capacity(x,y) $>$ 0, height(y) $=$ height(x) $-$ 1
- viene relabel: se per tutti capacity(x, *) $>$ 0, height(*) $<=$ height(x)

Tale algoritmo è un'ottimo candidato per la realizzazione di un algoritmo parallelo.

2.3 Goldberg-Tarjan Parallelo

In questi due algoritmi vengono presentati delle strutture dati particolari.
Da notare che vengono usate offsets, roffsets, destinations, rdestinations.
Queste strutture dati servono servono a salvare il grafo in una rappresentazione chiamata CSR().

3 Risultati sperimentali

4 Valutazioni ed osservazioni

Bibliografia

[] URL: https://web.stanford.edu/class/archive/cs/cs161/cs161.1172/CS161Lecture16.pdf.

[] URL: https://www.tutorialspoint.com/data_structures_algorithms/dsa_kargers_minimum_cut_algorithm.htm.

[] URL: https://www.baeldung.com/cs/minimum-cut-graphs.

[] URL: https://it.wikipedia.org/wiki/Algoritmo_di_Ford-Fulkerson.

[] URL: https://www.nvidia.com/content/GTC/documents/1060_GTC09.pdf.

[] URL: https://en.wikipedia.org/wiki/Push%E2%80%93relabel_maximum_flow_algorithm.

[] URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4563095.

[] URL: https://arxiv.org/pdf/2404.00270.

[] URL: https://github.com/NTUDDSNLab/WBPR/tree/master/maxflow-cuda.

[] URL: https://www.adrian-haarbach.de/idp-graph-algorithms/implementation/maxflow-push-relabel/index_en.html.

[] URL: https://www.geeksforgeeks.org/push-relabel-algorithm-set-2-implementation/.