TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA CÔNG NGHỆ THÔNG TIN

BÀI TẬP LỚN :TOÁN TỔ HỢP VÀ ĐỒ THỊ

PROJECT : COMBINATIONAL OPTIMIZATION

Người hướng dẫn : **G.V Lê Đình Thận**
Người thực hiện : **Đỗ Ngọc Khải - 51703107**
Mai Vinh Hiển - 51703078
Lớp : **17050301**
Khóa : **21**

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2018

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG

KHOA CÔNG NGHỆ THÔNG TIN

BÀI TẬP LỚN :TOÁN TỔ HỢP VÀ ĐỒ THỊ

PROJECT : COMBINATIONAL OPTIMIZATION

Người hướng dẫn : **G.V Lê Đình Thận**

Người thực hiện : **Đỗ Ngọc Khải - 51703107**

Mai Vinh Hiển - 51703078

Lớp : **17050301**

Khóa : **21**

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2018

LỜI CẢM ƠN

Chúng em xin gửi lời cảm ơn chân thành nhất đến Thầy Lê Đình Thận - G.V dạy môn thực hành môn Toán Tổ Hợp đã chỉ bảo tụi em tận tình , trang bị cho chúng em những kiến thức thật sự bổ ích để chúng em có thể hoàn thành bài tập lớn này.

Chúng em xin chân thành cảm ơn !

CÔNG TRÌNH ĐƯỢC HOÀN THÀNH TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG

Chúng tôi xin cam đoan đây là công trình nghiên cứu của riêng nhóm tôi và được sự hướng dẫn khoa học của G.V Lê Đình Thận. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong luận văn còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫn và chú thích nguồn gốc.

Nếu phát hiện có bất kỳ sự gian lận nào nhóm tôi xin hoàn toàn chịu trách nhiệm về nội dung luận văn của mình. Trường đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do nhóm tôi gây ra trong quá trình thực hiện (nếu có).

TP. Hồ Chí Minh, ngày 20 tháng 3 năm 2019.

Tác giả
(Ký tên và ghi rõ họ tên)

(Đã ký)
Đỗ Ngọc Khải

(Đã ký)
Mai Vinh Hiển

PHẦN ĐÁNH GIÁ CỦA GIẢNG VIÊN

Phần xác nhận của GV hướng dẫn :

Tp.Hồ Chí Minh,ngày 20 tháng 3 năm 2019

(Kí và ghi rõ họ tên)

Phần đánh giá của GV chấm bài:

Tp.Hồ Chí Minh,ngày 20 tháng 3 năm 2019

(Kí và ghi rõ họ tên)

TÓM TẮT

Combinatorial Optimization là một chủ đề bao gồm việc tìm kiếm các đối tượng tối ưu từ một tập các đối tượng. Đối tượng ở đây có thể là đường đi , có thể là một Node trong Spanning Tree...

Trong bài báo cáo này, chúng em sẽ trình bày rõ về 3 giải thuật tiêu biểu trong chủ đề Combinatorial Optimization, đó là : **Maximum Network Flow, Shortest Path** và **Minimum Spanning Tree.**

MỤC LỤC

PHẦN I : INTRODUCTION

1.2 Shortest Path :

Shortest Path là một chủ đề trong lĩnh vực Combinatorial Optimization. Giải thuật Dijkstra (1959) để tìm đường đi ngắn nhất trong đồ thị với trọng số dương là một trong những giải thuật quan trọng trong ngành Khoa Học Máy Tính về chủ đề này. Bên cạnh đó, giải thuật Linear-Time (Lawler 1976) ra đời cũng nhằm mục đích giải quyết vấn đề này đối với đồ thị có chu kỳ.

PHẦN II : STATE-OF-THE-ART

PHẦN III : APPROACHES

3.1 Maximun Network Flow :

Thuật toán tiêu biểu để giải quyết bài toán Maximum Network Flow trong vấn đề tối ưu hóa kết nối (Combinatorial Optimization) là thuật toán Ford Fulkerson.Ý tưởng thuật toán này khá đơn giản ,với điều kiện tất cả các cung trên đường đi đó vẫn còn khả năng thông qua, thì ta sẽ gửi đi một luồng dọc theo đường đi đó. Sau đó chúng ta tìm một đường đi khác, và tiếp tục như vậy.

Đầu vào : Đồ thị G với khả năng thông qua c, một nút nguồn s , và một nút thoát t.

Kết quả : Luồng f sao cho f là cực đại từ s đến t.

Pseudocode of Ford Fulkerson's Algorithm :

1.f(u,v) <- 0 trên tất cả các cung u,v

2. Trong khi còn có một đường đi từ đến t trong Gf:

1. Tìm một đường đi u1,u2,....,uk với u1 = s và uk = t, sao cho cf(ui,ui+1) > 0

2.Tìm m = min(cf(ui,ui+1))

3.f(ui,ui+1) <-f(ui , ui+1) + m(gửi luồng dọc theo đường đi)

4.f(ui+1,ui) <- f(ui+1,ui) - m(luồng có thể "quay lại" sau)

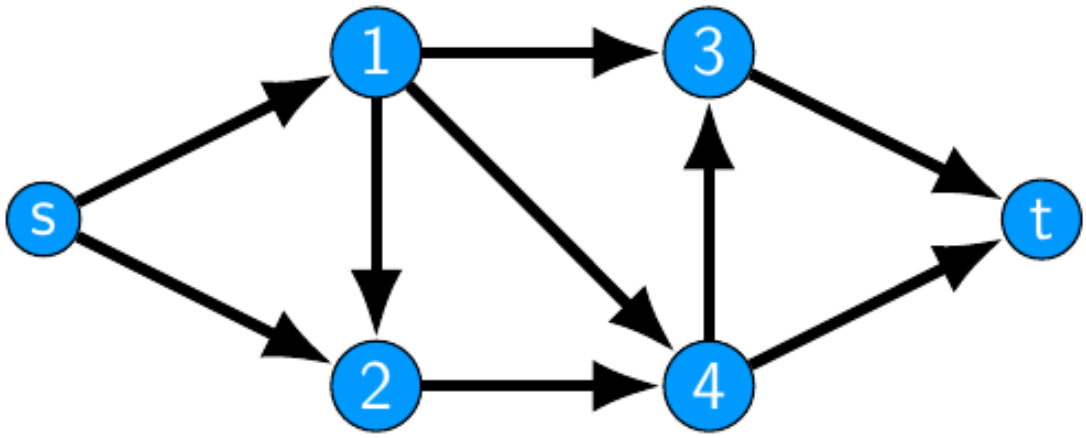
Có thể tìm đường đi trong Gf(V,Ef) bằng các phương pháp chẳng hạn như tìm kiếm theo chiều rộng(BFS) hoặc tìm kiếm theo chiều sâu(DFS). Nếu sử dụng BFS, thuật toán sẽ được gọi là EDmonds-Karp.

3.2 Shortest Path :

Một trong các giải thuật tối ưu để giải quyết Shortest Path Problem có thể nói đến đó chính là giải thuật Dijkstra.

Giải thuật này áp dụng cho Connected(di-)Graph cho phép tìm đường đi ngắn nhất từ một đỉnh s cho đến tất cả các đỉnh còn lại. Thuật toán

được xây dựng trên cơ sở dán Nhãn tạm thời cho mỗi đỉnh. Nhãn dán tạm thời cho biết cận trên của chiều dài đường đi ngắn nhất từ s cho đến đỉnh đó. Nhãn dán của các đỉnh sẽ thay đổi qua từng bước lặp, mà ở mỗi bước lặp đó sẽ có một Nhãn dán tạm thời trở thành Nhãn dán chính thức. Khi một Nhãn dán của một đỉnh trở thành chính thức thì đó cũng chính là chiều dài ngắn nhất từ s cho đến đỉnh đó. Thuật toán được lặp đi lặp lại qua mỗi đỉnh cho tới khi tìm được đường đi ngắn nhất đến đỉnh t.



Example: Finding the Shortest-Path

Pseudocode of Dijkstra's Algorithm :

```
initSSSP(s)
PQ.enqueue((0,s))      //store pair of (dist[u],u)
while PQ is not empty     //order : increasing dist[u]
   (d,u) <- PQ.dequeue()
   if d == dist[u]     //important check , lazy DS
      for each vertex v adjacent to u
         if dist[v] > dist[u] + weight(u,v)    //can relax
            dist[v] = dist[u] + weight[u,v]     //relax
            PQ.enqueue((dist[v],v))    //(re)enqueue this
```

3.3 Minimum Spanning Tree :

Một vấn đề được biết đến khá phổ biến trong việc tối ưu kết nối là Minmum Spanning Tree(MST Problem - Vấn đề cây bao trùm nhỏ nhất). Cho một đồ thị liên thông (connected) , đồ thị vô hướng(undirected) ,cây bao trùm nhỏ nhất của đồ thị là một đồ thị con, mà bản thân nó là một cây và liên thông tất cả các đỉnh. Cho một trọng số mặc định với mỗi đỉnh, Cây bao trùm nhỏ nhất là cây với trọng số nhỏ hơn hoặc bằng trọng số của các cây bao trùm khác.

Vấn đề MST là một ví dụ đơn giản cho vấn đề Combination Optimization(Tối ưu hóa kết nối). Hai thuật toán phổ biến để giải quyết MST Problem là Prim'Algorithm và Kruskal's Algorithm. Trong bài báo cáo này , chúng em xin trình bày một trong hai thuật toán tiêu biểu trên , đó là thuật toán Kruskal.

Pseudocode of Kruskal's Algorithm :

```
sort E edges by increasing weight //O(E log E)
T <- // while there are unprocessed edges left //O(E)
   pick an unprocessed edge e with min cost //O(1)
   if adding e to T does not form a cycle //O(alpha(V))= O(1)
      add e to the T //O(1)
T is an MST
```