Algorithm Homework 4

Jingwei Zhang 201528013229095

2015-11-26

1 Problem 2

1.1 LP Formulation

Let $x_i(i = 1, \ldots, n)$ be the arrival time of the i-th airplane. Since the i-th airplane should land between s_i and t_i for all $i = 1, \ldots, n$. We have two constraints for each i: $s_i \leq x_i$ (this guarantees that $x_i \geq 0$) and $x_i \leq t_i$. Furthermore, we would like to maximize the minimum of $x_{i+1} - x_i(i = 1, \ldots, n-1)$. Let m be the minimum of $x_{i+1} - x_i$. Then, we have constraints: $x_{i+1} - x_i \geq m$ for all $i = 1, \ldots, n-1$. Finally, the objective function is maximize m.

LP Formulation:

$$\begin{array}{ll} \text{Max} & m \\ \text{s.t.} & x_i \geq s_i, \qquad\qquad i = 1, \ldots, n \\ & x_i \leq t_i, \qquad\qquad i = 1, \ldots, n \\ & x_{i+1} - x_i \geq m, \; i = 1, \ldots, n-1 \end{array}$$

1.2 An Instance

Instance: We use the instance in this problem:

$s_i(min)$	600	680	720
$t_i(min)$	660	700	740

Result: GLPK gets the following answer:

m	x_i	x_2	x_3
60	600	6800	740

2 Problem 4

2.1 LP Formulation

Let $x_i(i = 1, \ldots, n)$ be the distance of the i-th gas station from the end point. Since the i-th gas station should be placed at most r far away from the i-th twon, which has distance d_i. We have two constraints for each i: $d_i - r \leq x_i$ and $x_i \leq d_i + r$. Furthermore, we would like to minimize the maximum of $x_{i+1} - x_i(i = 1, \ldots, n-1)$. Let m be the maximum of $x_{i+1} - x_i$. Then, we have constraints: $x_{i+1} - x_i \leq m$ for all $i = 1, \ldots, n-1$. Finally, the objective function is minimize m.

LP Formulation:

$$\begin{array}{ll} \text{Min} & m \\ \text{s.t.} & x_i \geq d_i - r, \qquad i = 1, \ldots, n \\ & x_i \leq d_i + r, \qquad i = 1, \ldots, n \\ & x_{i+1} - x_i \leq m, \; i = 1, \ldots, n-1 \end{array}$$

3 Problem 8

3.1 LP Formulation

LP Formulation:

$$\begin{array}{lll} \text{Min} & \sum_{i,j} \varepsilon_{ij} & \\ \text{s.t.} & \varepsilon_{ij} \geq 0, & \forall i,j \\ & x_i - x_j - d_{ij} \leq \varepsilon_{ij}, & \forall i,j \\ & x_i - x_j - d_{ij} \geq -\varepsilon_{ij}, & \forall i,j \end{array}$$

3.2 Result

Solving this problem by GLPK gets the optimum 1627494, with 6 seconds(including file input) and 310.3 Mb of memory. The algorithm mentioned in part 2 gets 1627617 with 0.015 second and very few memory usage. From the two results, we can conclude that this algorithm does not necessarily converge to the optimum, but runs faster and uses much less memory.

3.3 C++ Code

```cpp
#include <iostream>
#include <cstdio>
#include <vector>
#include <cstdlib>
#include <algorithm>

using namespace std;

const int EPS = 1E-5;

struct Edge{
    int t;
    int w;
    bool reversed;
    Edge(){}
    Edge(int tt,int ww,bool rr):t(tt),w(ww),reversed(rr){}
};

vector< vector<Edge> > g; // graph, adjcency list
vector<int> x;// vertics
int sum; // the objective function value

void get_input(){
    int s,t,w;
    g.clear();

    while(cin>>s>>t>>w){
        while(s >= g.size() || t >= g.size()){
            g.push_back(vector<Edge>());
        }
        g[s].push_back(Edge(t, w, false));
        g[t].push_back(Edge(s, -w, true));
    }

}

void BFS(){
    x.resize(g.size());
```

```
    x[1] = 0;

    vector<int> q;// queue
    int tail = 0;// points to the tail of queue
    q.push_back(1);

    vector<bool> vis(x.size(), false);
    while(tail < q.size()){
        int u = q[tail++];
        vis[u] = true;
        for(Edge &e: g[u]){
            int v = e.t;
            if(!vis[v]){
                x[v] = x[u] - e.w;
                q.push_back(v);
            }
        }
    }

}

void solve(){
    BFS();
    // optimization
    vector<int> weights;
    bool is_conv = false;
    //int cnt = 0;
    while(!is_conv){
        //cout<<"\t"<<++cnt<<endl;
        is_conv = true;
        for(int u = 1;u < g.size();u++){
            weights.clear();
            for(Edge &e: g[u]){
                weights.push_back(x[e.t] + e.w);
            }
            //cout<<"\t"<<weights.size()<<endl;
            nth_element(weights.begin(),
                        weights.begin() + weights.size() / 2,
                        weights.end());
            int new_x = weights[weights.size() / 2];
            if(abs(new_x - x[u]) > EPS){
                is_conv = false;
                x[u] = new_x;
            }
        }
    }
    // make x[0] be 0
    for(int i = x.size()-1;i >= 1;i--){
        x[i] -= x[1];
    }
    //for(int w: x){cout<<w<<endl;}
    // compute the objective function value
    sum = 0;
    for(int u = 1;u < g.size(); u++){
        for(Edge &e:g[u]){
            if(!e.reversed){
```

```
                int v = e.t;
                sum += abs(x[u] - x[v] - e.w);
            }
        }
    }
}

int main()
{
    freopen("8_CInput.in","r",stdin);
    get_input();
    solve();
    cout<<sum<<endl;
    return 0;
}
```