Homework 2

Algorithm Design

Mariani Matteo
Matricola ID: 1815188

Contents

Exercise 1

Given a set A of n 2-dimensional points, we can use a Bounding Box to include all the points s.t. its edges are determined using the maximum and minimum coordinates among the points $(x_{min}, x_{max}, y_{min}, y_{max})$. In this way, we can obtain the length of the two edges in a simple way:

- h, is the heigth of the bounding box (for our example, we assume that is $x_{max} - x_{min}$);
- w, is the width of the bounding box (so, $y_{max} - y_{min}$).

Given this bounding box, its diagonal *diag* represents the maximum possible distance between any two points in it; where $diag = \sqrt{h^2 + w^2}$ (for our example, we assume that $w > h$). Instead, remember that the diameter *diam* of A represents the maximum Euclidean distance between any two points in A, so $diam = max_{a,b \in A} distance(a, b)$.

For this reason, we know for sure that:

$$diam \leq diag = \sqrt{h^2 + w^2} \leq \sqrt{2 \cdot w^2} = \sqrt{2} \cdot w$$

So, $diam \leq \sqrt{2} \cdot w \rightarrow \frac{diam}{w} \leq \sqrt{2}$.

This is exactly the approximation that we are looking for, defined as:

$$\frac{max_{a,b \in A} distance(a,b)}{\Delta} \leq \alpha$$

where $\Delta = w$ (remember that $w = max(w, h)$) and $\alpha = \sqrt{2}$.

Code

The algorithm does the following:

1. compute the $x_{min}, x_{max}, y_{min}, y_{max}$ scanning all the points $p_i \in A$. With those coordinates, the bounding box is defined.
2. compute the height h and the witdh w of the bounding box .
3. compute the approximate diameter as the maximum value between h and w and return it as output.

The complexity is $O(n)$, linear w.r.t. the number of points in the set A.

ex1.py

```
import math

class Point(object):
    def __init__(self,x,y):
        self.x = x
        self.y = y

def getBoundingBoxSize(A):
    x_max = x_min = A[0].x
    y_max = y_min = A[0].y
    for point in A[1:]:
        if point.x > x_max:
            x_max = point.x
        if point.x < x_min:
            x_min = point.x
        if point.y > y_max:
            y_max = point.y
        if point.y < y_min:
            y_min = point.y
    h = y_max - y_min
    w = x_max - x_min
    return h,w

def getApproximateDiameter(h,w):
    approximateDiameter = 0
    if h>w:
        approximateDiameter = h
    else:
        approximateDiameter = w
    return approximateDiameter

#assuming five points in the 2dimensional space
p1 = Point(0,2)
p2 = Point(2,3)
p3 = Point(1,1)
p4 = Point(1,4)
p5 = Point(7,3)

A = [p1,p2,p3,p4,p5]

h,w = getBoundingBoxSize(A)

approximateDiameter = getApproximateDiameter(h,w)
print "Output: " + str(approximateDiameter)
```

Exercise 2

Consider the a Bernoulli variable X_i, we have that:

- $X_i = 1$ if the edge i of the set S belongs to the max-cut;
- $X_i = 0$, otherwise.

Consider a cut (A, B) and en edge e, we have that:

$$P(X_i = 1) = P(e \in (A \times B) \cap E) = \mu = \frac{|(A \times B) \cap E|}{|E|}$$

where, $P(X_i = 1)$ is the probability that $e \in$ *max-cut*.
With respect to the set S, we can write the equation as:

$$\sum_{i=0}^{|S|} X_i = \sum_{i=0}^{|S|} \frac{|(A\times B)\cap S|}{|S|} = |S| \cdot \frac{|(A\times B)\cap S|}{|S|} = |(A \times B) \cap S|$$

Taking into account the *Chernoff bounds* we can formulate the following:

- $P[|(A \times B) \cap S| > (1+\epsilon) \cdot |S| \cdot \frac{|(A\times B)\cap E|}{|E|}] < exp(-\frac{\epsilon^2 \cdot |S| \cdot |(A\times B)\cap E|}{3 \cdot |E|})$;
- $P[|(A \times B) \cap S| < (1-\epsilon) \cdot |S| \cdot \frac{|(A\times B)\cap E|}{|E|}] < exp(-\frac{\epsilon^2 \cdot |S| \cdot |(A\times B)\cap E|}{2 \cdot |E|})$.

The inequality that we want to prove is the following:

$$(1-\epsilon) \cdot |(A \times B) \cap E| \leq |(A \times B) \cap S| \cdot \frac{|E|}{|S|} \leq (1+\epsilon) \cdot |(A \times B) \cap E|$$

that should holds with a probability greater than $\frac{1}{2}$. Instead of directly prove that, we can use the Chernoff bounds and Union bounds showing that:

$$P[|(A \times B) \cap S| > (1+\epsilon) \cdot |S| \cdot \frac{|(A\times B)\cap E|}{|E|} \cup$$

$$\cup\ |(A \times B) \cap S| < (1-\epsilon) \cdot |S| \cdot \frac{|(A\times B)\cap E|}{|E|}] < \frac{1}{2}$$

Knowning that:

1. $|S| = \frac{24 \cdot n}{\epsilon^2}$
2. $\mu = \frac{|(A\times B)\cap E|}{|E|} < \frac{\frac{|E|}{2}}{|E|} = \frac{1}{2}$

we can re-formulate the previous equations:

- $P[|(A \times B) \cap S| > (1+\epsilon) \cdot |S| \cdot \frac{|(A\times B)\cap E|}{|E|}] < exp(-\frac{\epsilon^2 \cdot |S| \cdot \mu}{3}) = exp(-8 \cdot n \cdot \mu)$
- $P[|(A \times B) \cap S| < (1-\epsilon) \cdot |S| \cdot \frac{|(A\times B)\cap E|}{|E|}] < exp(-\frac{\epsilon^2 \cdot |S| \cdot \mu}{2}) = exp(-6 \cdot n \cdot \mu)$

$$\Longrightarrow$$

- $exp(-8 \cdot n \cdot \mu) = e^{-4 \cdot n} \Rightarrow P[|(A \times B) \cap S| > (1+\epsilon) \cdot |S| \cdot \frac{|(A\times B)\cap E|}{|E|}] < e^{-4 \cdot n}$
- $exp(-6 \cdot n \cdot \mu) = e^{-6 \cdot n} \Rightarrow P[|(A \times B) \cap S| < (1-\epsilon) \cdot |S| \cdot \frac{|(A\times B)\cap E|}{|E|}] < e^{-6 \cdot n}$

Given the estimated probabilities, we use the Union bound to say the following:

$$P[|(A \times B) \cap S| > (1+\epsilon) \cdot |S| \cdot \frac{|(A\times B)\cap E|}{|E|} \ \cup\ |(A \times B) \cap S| < (1-\epsilon) \cdot |S| \cdot \frac{|(A\times B)\cap E|}{|E|}] <$$

$$P[|(A \times B) \cap S| < (1-\epsilon) \cdot |S| \cdot \frac{|(A\times B)\cap E|}{|E|}] + P[|(A \times B) \cap S| < (1-\epsilon) \cdot |S| \cdot \frac{|(A\times B)\cap E|}{|E|}] <$$

$$e^{-4 \cdot n} + e^{-6 \cdot n} < \frac{1}{4} + \frac{1}{4} = \frac{1}{2} \quad , \forall n > 0$$

Exercise 3

[1]

Exercise 3.1

Consider a set of facilities F - where f_i is the cost of opening a facility i - and a set of cities C - where $c_{i,j}$ is the cost to connect the city j to the facility i. Moreover, the cost of connecting city j to facility i is also a function of the amount of demand d_j which is expected from the city j ($c_{i,j}d_j$). In the following we are assuming that we are still in a metric space: the triangle inequality still holds.

We have to find a subset of facilities that should be opened to minimize the total cost.

$$\min \sum_{i \in F, j \in C} d_j c_{ij} x_{ij} + \sum_{i \in F} f_i y_i$$

$$\sum_{i \in F} x_{ij} \geq 1 \quad , \forall j \in C$$
$$y_i - x_{ij} \geq 0 \quad , \forall i \in F, \forall j \in C$$
$$x_{ij} \in \{0, 1\} \quad , \forall i \in F, \forall j \in C$$
$$y_i \in \{0, 1\} \quad , \forall i \in F$$

where:

- the first constraint represents the fact that each city must be connected to a facility;
- the second constraint represents the fact that the facility is surely opened if it's connected to a city;
- each variable can assume only 1 or 0 values.

Thanks to the last couple of constraints, we can clearly see that we have an integer linear problem. Therefore, we have to make a relaxation of the constraints of the previous formulation to be able to solve it.

This is achieved by imposing that $x_{i,j}$ and y_i should not be limited to assume only 1 or 0 value:

$$\min \sum_{i \in F, j \in C} d_j c_{ij} x_{ij} + \sum_{i \in F} f_i y_i$$

$$\sum_{i \in F} x_{ij} \geq 1 \quad , \forall j \in C$$
$$y_i - x_{ij} \geq 0 \quad , \forall i \in F, \forall j \in C$$
$$x_{ij} \geq 0 \quad , \forall i \in F, \forall j \in C$$
$$y_i \geq 0 \quad , \forall i \in F$$

We can apply a primal-dual approximation algorithm to obtain an approximation to the integer linear problem. Indeed, the dual problem it's used to obtain a bound to the optimal value of the primal problem. The dual will be:

$$\max \sum_{j \in C} \alpha_j$$

$$\alpha_j - \beta_{ij} \leq d_j c_{ij} \quad , \forall i \in F, \forall j \in C$$
$$\sum_{j \in C} \beta_{ij} \leq f_i \quad , \forall i \in F$$
$$\beta_{ij} \geq 0 \quad , \forall i \in F, \forall j \in C$$
$$\alpha \geq 0 \quad , \forall j \in C$$

where:

- α_j is the total cost that each city should pay and it is most $d_j c_{ij} + \beta_{ij}$ thanks to the first constraint;
- β_{ij} is the contribution paid by the city j to open the facility i.
- the second constraint represents that the cost to open a facility i is at least equal to the sum of all the contributions β_{ij} , $\forall j \in C$.

primal-dual 3-approximation algorithm

The following algorithm give an approximate solution to the integer linear problem that is at most 3 times the optimal one.

The algorithm follows two phases:

Phase 1

At the starting point: (1) all the variables are inizialized ($\alpha_j = 0$ and $\beta_{ij} = 0$),(2) there is no connection between any city j and facility i and (3) each facility is not opened.
From that, we increment α_j of d_j factor, until for some cities j we obtain that $\alpha_j = d_j c_{ij}$. This means that the city j has paid enough to reach the facility i and the edge (i, j) is called *tight*.
After that, we continue to increase α_j for each city , but for those cities with tight edges we increase also β_{ij} to not violate the constraint of the problem ($\alpha_j - \beta_{ij} \leq d_j c_{ij}$).
When for some facility i, $\sum_{j \in C} \beta_{ij} = f_i$, then (1) the facility i is declared *temporarily opened* and (2) all the cities that have the tight edge with the facility i will be *connected* to i - moreover, their α_j is not increased anymore.
This is done until all the cities are *connected*.

Phase 2

F_t is the set of facilities that were temporarily opened in the phase 1. Considering how we handled phase 1, can happen that some cities may have partecipate to open more than one facility. If we consider two facilities i and i' opened by a city j ($i, i' \in F_t$ and $\beta_{ij}\beta_{i'j} > 0$), those facilities are *in conflict*. If two facilities are in conflict, we can't permanently open both because this will generate a cost of the integer solution that may be too high w.r.t. the optimal one.
Let G be a graph where (1) the vertices are the facilities in F_t (2) and the edges appear only between conflicting facilities.
To understand which facilities should be opened, we consider the maximal indipendent set of the graph G; taking the vertices in the same order by which they were declared temporary open and avoiding to take nodes belonging to the same conflicting pairs.

Therefore:

- all the cities that were connected to a temporarily facility i that now is permanently opened are now declared *directly connected* to facility i;
- all the cities that were connected to a temporarily facility i' that now is not opened are now declared *indirectly connected* to facility i.

Proof

We want to prove that:

$$cost(ILP) \leq 3 \cdot cost(OPT), \text{ or}$$
$$\sum_{i \in F, j \in C} d_j c_{ij} x_{ij} + \sum_{i \in F} f_i y_i \leq 3 \cdot \sum_{j \in C} \alpha_j$$

Since that in the Phase 2 we excluded any conflicting facilities, each city is connected to only one facility. For any city j directly connected to i - we call this set as C' - we have that $\beta_{ij} = \alpha_j - d_j c_{ij}$. But this can be considered like the sum of all the contributions of the cities connected to the facility i as:

$$f_i = \sum_{j \in C'} \alpha_j - \sum_{j \in C'} d_j c_{ij}$$

For any city j indirectly connected to i we have that (1) j was previously connected to an other temporarily opened facility i', (2) i was in conflict to i' and (3) i became temporarily opened before i'. Therefore, there exists at least one city j' s.t. $\beta_{ij'}\beta_{i'j'} > 0$.
Moreover, we can say that $\frac{\beta_{ij'}}{d_{j'}} > 0$ and $\frac{\beta_{i'j'}}{d_{j'}} > 0$ (because the demand $d_{j'} > 0$).
Since that j' is contributing to both i and i', and j is contributing to i', we can say the following:

$$d_{j'} c_{ij'} \leq \alpha_{j'}$$
$$d_{j'} c_{i'j'} \leq \alpha_{j'}$$
$$d_j c_{i'j} \leq \alpha_j$$

The dual variable α_j stops increasing when the facility i' temporarily opens and $\alpha_{j'}$ stops increasing when i temporarily opens. Since α_j and $\alpha_{j'}$ are raising at same rate, we can say that $\alpha_{j'} \leq \alpha_j$.
Since that c_{ij} respects the triangle inequality we can say that:

$$d_j c_{ij} \leq d_{j'} c_{ij'} + d_{j'} c_{i'j'} + d_j c_{i'j} \leq \alpha_{j'} + \alpha_{j'} + \alpha_j \leq \alpha_j + \alpha_j + \alpha_j \leq 3 \cdot \alpha_j$$
$$d_j c_{ij} \leq 3 \cdot \alpha_j$$

Moreover:

- considering I as the subset of facilities that are permanently opened, then $\sum_{i \in I} f_i = \sum_{j \in C} \beta_{ij}$. So, the property still holds: $\sum_{i \in F} f_i y_i \leq 3 \cdot \sum_{j \in C} \beta_{ij}$
- considering the directly connected cities, we can easily prove that the same property holds: $d_j c_{ij} \leq 3 \cdot d_j c_{ij}$. So, combining the results obtained from the two subsets of cities, we can conclude that: $\sum_{i \in F, j \in C} d_j c_{ij} x_{ij} \leq 3 \cdot \sum_{j \in C} d_j c_{ij}$.

At the end, we can conclude that:

$$\sum_{i \in F, j \in C} d_j c_{ij} x_{ij} + \sum_{i \in F} f_i y_i \leq \sum_{i \in F, j \in C} d_j c_{ij} x_{ij} + 3 \sum_{i \in F} f_i y_i \leq 3 \cdot \sum_{j \in C} \alpha_j$$

Running time

The 3-approximation algorithm can find a solution in $O(m\ log\ m)$, where m is the number of edges in the bipartite graph. $m = |C| \cdot |F|$, where (1) $|C|$ is the number of cities and (2) $|F|$ is the number of facilities. First of all, we consider the edges in increasing order w.r.t. their weight $d_j c_{ij}$ using a heap data structure to know the next edge of minimum weight. The cost of updating it is $O(log\ m)$.
Notice that, each edge will be considered at most twice because (1) the first time it's considered when it becomes tight and (2) the second time when the city j is declared connected to the facility i. In each time, we have to make an update to the heap. We'll make these updates m times. So, $O(m\ log\ m)$.

Exercise 3.2

Exercise 4

Exercise 4.1

We have to prove that starting from any solution it is possible to converge to a pure Nash Equilibrium.
Note that the potential function is $\Phi = \sum_{j=1}^{m} \frac{1}{2} L_j^2$ where m are the machines and L_j is the load of the machine j. Remember that $L_j = \sum_{i=1}^{k} w_k$, where w_k is the weight w of the agent k in the machine j.

The individual goal of every agent is to minimize the load of his machine. For each agent i, the cost function is:

$$c_i(S) = (\tfrac{1}{2} L_j^2)' = L_{m(i)}(S), \text{ from the definition of potential function}$$

where (1) $S = (S_1, S_2, ..., S_n)$ is the set of strategies of all n agents and (2) $m(i)$ is the machine chosen by the agent i with strategy $S_i \in S$.

Starting from S - that represents any solution - we consider a scenario in which an agent i with weight w_i chooses to switch from a machine i to a machine k. This new strategy will be $S' = (S_1, S_2, ..., S_i', ..., S_n)$.

From S to S', the load on machines i and k changes. In particular, $L_i' = L_i - w_i$ and $L_k' = L_k + w_i$. Note that, if the agent i decided to move from i to k means that $L_k' = L_k + w_i < L_i$; otherwise he will not gain any utility in doing that. Let's consider the potential function in the two scenarios:

- $\Phi(S) = \sum_{j=1}^{m} \frac{1}{2} L_j^2 = \sum_{j=1}^{m-2} \frac{1}{2} L_j^2 + \frac{1}{2} L_i^2 + \frac{1}{2} L_k^2$
- $\Phi(S') = \sum_{j=1}^{m} \frac{1}{2} L_j^2 = \sum_{j=1}^{m-2} \frac{1}{2} L_j^2 + \frac{1}{2} (L_i')^2 + \frac{1}{2} (L_k')^2$

Considering the difference of the two potential functions:

$$\Phi(S) - \Phi(S') = \tfrac{1}{2} L_i^2 + \tfrac{1}{2} L_k^2 - \tfrac{1}{2} (L_i')^2 - \tfrac{1}{2} (L_k')^2$$

Substituing the definition of L_i' and L_k' in the previous equation, we have that:

$$\Phi(S) - \Phi(S') = -w_i^2 + w_i(L_i - L_k) = -w_i^2 + w_i L_i - w_i L_k = w_i(-w_i + L_i - L_k) = w_i(L_i - (L_k + w_i)) = w_i(L_i - L_k') = w_i \Delta c_i$$

So, we can see that the difference of the potential functions based on two different strategies S and S' is equal to the difference between the cost functions of the agent i based on the same two different strategies S and

S' multiplied by a factor (that is the weight w of the agent i). Since that (1) $w_i > 0$ and (2) $L_i - L_k - w_i > 0$ (because, $L_k + w_i < L_i$) we can say that $w_i \Delta c_i > 0$.

In conclusion, we can see that if the agent i decides to move from a machine to an other - according to his will to minimize the load of his machine - it's equal to minimize the potential function Φ. Moreover, since that the number of possible combinations is finite (because we have a finite number of machines and jobs) we can conclude that it's possible to converge to a pure Nash Equilibrium, where no more assignment can make Φ smaller.

Exercise 4.2

The global objective of the problem is to minimize the makespan. Remember that the makespan is defined as the greatest load among all the machines ($max_j L_j$).
We want to prove that the makespan of the pure Nash Equilibrium is at most twice the minimum makespan.

Given a set of machines m and agents n, each having a job with weight w_n; the optimal makespan OPT has a lower bound equal to the average load all over machines.

$$OPT \geq \frac{\sum_{j=1}^{n} w_j}{m}$$

Let's consider an instance of pure Nash Equilibrium, where m_j is the machine with the highest load L_j. It's clear to see that L_j is the makespan of the current assignement S, where S is the set of strategies.
If we consider the case in which m_j has only one job, the makespan is equal to the optimal . So, let's consider the case in which m_j has at least two jobs with weights w_i and w_k. For example, the agent i is the one with the smaller job ($w_i \leq w_k$). Therefore $w_i \leq \frac{1}{2} L_j$.

Since that we are considering a pure Nash Equilibrium, we have that $L_k \geq L_j - w_i \; \forall \; k = 1, ..., m$ machines (without j). Since that we can write $w_i \leq \frac{1}{2} L_j$ as $-w_i \geq -\frac{1}{2} L_j$, we can do the following:

$$L_k \geq L_j - w_i \geq L_j - \tfrac{1}{2} L_j = \tfrac{1}{2} L_j$$
$$\Rightarrow L_k \geq \tfrac{1}{2} L_j$$

Considering the definition of the OPT w.r.t. the loads:

$$OPT \geq \frac{\sum_{k=1}^{m} L_k}{m} = \frac{L_j}{m} + \frac{\sum_{k=1}^{m-1} L_k}{m} = \frac{L_j}{m} + \frac{(m-1)\frac{1}{2} L_j}{m} = \frac{L_j (m+1)}{2m}$$

So:

$$L_j \leq \frac{2m \cdot OPT}{m+1}$$

Since that we want an approximation of factor 2 of the makespan, we want to set:

$$2 - x = \frac{2m}{m+1} \Rightarrow x = \frac{2}{m+1}$$

Therefore:

$$L_j \leq (2 - \frac{2}{m+1}) OPT$$

We can conclude that the makespan of the pure Nash Equilibrium is at most twice the minimum makespan.

Exercise 5

Since the coin used by the elves is biased, we have that:

$$P(Head) = p \text{ and } P(Tail) = 1 - p, \text{ where } p \neq 1 - p$$

Our goal is to find an algorithm that simulates a fair coin toss while having access to this unfair coin.

A fair coin is one that: given n tosses the probability to obtain $Head$ and $Tail$ is the same ($P(Head) = P(Tail)$).

Let's consider a game with 2 tosses:

- $P(Head, Head) = p \cdot p$;
- $P(Head, Tail) = p \cdot (1 - p)$;
- $P(Tail, Head) = (1 - p) \cdot p$;
- $P(Tail, Tail) = (1 - p) \cdot (1 - p)$;

In this scenario, we can clearly see that two combinations of tosses have the same probability $P(Tail, Head) = P(Head, Tail)$. If we can isolate these combinations, we can achieve the property of a fair coin.

The algorithm to simulate a fair coin using an unfair coin is the following:

- toss the unfair coin twice;
- if we get (Head,Tail), then the result will be Head;
- if we get (Tail, Head), then the result will be Tail;
- in the other two cases, we repeat the process (tossing the unfair coin twice again).

Considering how it works, we have two possible outcomes for each double toss:

- we get a result, where $P(Result) = P(Head, Tail) + P(Tail, Head) = 2(p \cdot (1 - p))$;
- we have to repeate the double toss, where $P(Repeat) = 1 - P(Result) = 1 - 2(p \cdot (1 - p))$.

Now, we have to be sure that the algorithm does not take too long to get one of the two good results. Consider the aleatory variable X that represents the number of double tosses and $E[X]$ as the expected number of double tosses that we have to perform to get a good result, we have the following:

$$E[X] = \sum_{i=0}^{\infty} i \cdot P(X = i) = \sum_{i=0}^{\infty} i \cdot (1 - 2(p \cdot (1 - p)))^{i-1} \cdot 2(p \cdot (1 - p)) =$$
$$\frac{2(p \cdot (1-p))}{1-2(p \cdot (1-p))} \cdot \sum_{i=0}^{\infty} i \cdot (1 - 2(p \cdot (1 - p)))^{i} = \frac{2(p \cdot (1-p))}{1-2(p \cdot (1-p))} \cdot \frac{1-2(p \cdot (1-p))}{(2(p \cdot (1-p)))^2} = \frac{1}{2(p \cdot (1-p))}$$

where $2(p \cdot (1 - p))$ is $P(Head, Tail) + P(Tail, Head) = P(Result)$.

References

[1] Vijay V. Vazirani, Approximation Algorithms, 2001 (Chapter 24)