Algorithm Design - Homework 2

Academic year 2017/2018

Tufa Alexandru Daniel 1628927

15 January 2018

1 Exercise 1

Given a set A of n 2-dimensional points and knowing that the diameter of A, which we'll call OPT, is the maximum Euclidean distance between any two points in A, our objective is to design an algorithm that returns a $\sqrt{2}$-approximate diameter, which we'll call $\widetilde{\Delta}$, such that

$$\frac{OPT}{\widetilde{\Delta}} \leq \sqrt{2}$$

A proper construction that will allow us to return this approximate diameter is the minimum bounding rectangle [1]. In order to build it, by iterating once on our set of points, we will get the minimum and maximum values of the x and y coordinates. The width and height of the rectangle are defined as

$$w = xMax - xMin \quad h = yMax - yMin$$

We will return as $\sqrt{2}$ approximation the maximum between these two values. To prove the correctness of this statement we need to find some bounds for our diameter.

Theorem 1.1. $OPT \leq \sqrt{w^2 + h^2}$

Proof. Let's suppose, by contradiction, that the diameter is bigger than the diagonal of the minimum bounding rectangle ($OPT > \sqrt{w^2 + h^2}$), that we can call d_{MBR}. If so, by taking the two corresponding points of the diameter, we can construct a second bounding rectangle, that we'll call BR. Because $OPT >$ diagonal of MBR, since we supposed this by contradiction, it follows that the diagonal of the new bounding rectangle is greater than the diagonal of the minimum bounding rectangle ($d_{BR} > d_{MBR}$), but this is a contradiction since we assumed that the original bounding rectangle was the minimum one. □

Theorem 1.2. $max\{w, h\} \leq OPT$

Proof. Let's suppose, by contradiction, that $max\{w, h\} > OPT$. If so, given $max\{w, h\}$ we can take the extreme points of this segment and build a new diameter, OPT'. Since $max\{w, h\} = OPT'$ and $max\{w, h\} > OPT$, that we assumed by contradiction, it follows that $OPT' > OPT$. This is a contradiction given that we assumed OPT being the maximum Euclidean distance between any two points. □

Theorem 1.3. $\sqrt{w^2 + h^2} \leq \sqrt{2} max\{w, h\}$

Proof. Let's suppose w = $max\{w, h\}$. If this is the case then, by definition of w, there exists 2 points, A and B, that have coordinates xMin and xMax. The distance between the lines passing through these points parallel to the y axis can be at most w, otherwise I would have chosen that distance as the maximum. If so, the distance between these points can be at most $\sqrt{2}w$, which is equal to $\sqrt{2} max\{w, h\}$. □

Theorem 1.4. *The maximum betwen w and h is a $\sqrt{2}$-approximation of the diameter of a set of points. ($\widetilde{\Delta} = max\{w, h\}$)*

Proof. To prove this we need to show that

$$\frac{OPT}{\widetilde{\Delta}} \leq \sqrt{2}$$

By using theorems 1.1, 1.2 and 1.3 we have proven that

$$max\{w,h\} \leq OPT \leq \sqrt{w^2+h^2} \leq \sqrt{2}max\{w,h\}$$

which gives us our desired outcome, more precisely

$$\frac{OPT}{max\{w,h\}} \leq \sqrt{2}$$

□

```
Algorithm:
    input: set of points A(x,y)

1. if length of input is < 2 then return 0
2. xMin = x coordinate of the first point, xMax = x coordinate of the first point
   yMin = y coordinate of the first point, yMax = y coordinate of the first point
3. for all points p_i in the input
        if x coordinate of p_i < xMin then xMin = x coordinate of p_i
        if x coordinate of p_i > xMax then xMax = x coordinate of p_i
        if y coordinate of p_i < yMin then yMin = y coordinate of p_i
        if y coordinate of p_i > yMax then yMax = y coordinate of p_i
4. w = xMax - xMin   h = yMax - yMin
5. return max(w, h)
```

Theorem 1.5. *Algorithm returns a $\sqrt{2}$-approximate diameter and is linear with respect to the input.*

Proof. Given that we have proven that $max\{w,h\}$ is a $\sqrt{2}$-approximate diameter and w and h are calculated as we have specified, the algorithm is correct. Also because every point is seen only once by the algorithm, we have also proven that it's linear. □

A Python implementation has been included where we consider a random set of points. Here we calculate the diameter of this set of points using the classic $O(n^2)$ algorithm and also the linear algorithm that offers the $\sqrt{2}$-approximate diameter.

2 Exercise 2

Given an unweighted and undirected graph G(V,E) and choosing a set S of $\frac{24n}{\epsilon^2}$ edges sampled uniformly at random without replacement, we will estimate the weight of the max-cut $w(A,B) = |(A \times B) \cap E|$. We will provide an algorithm that computes the weight of the max-cut in the graph generated by the sampled edges and we'll prove that $|(A \times B) \cap S| \cdot \frac{|E|}{|S|}$ is a $(1 \pm \epsilon)$ approximate estimate of $|(A \times B) \cap E|$.

Theorem 2.1. *$|(A \times B) \cap S|$ is a $(1 \pm \epsilon)$ approximate estimate of $|(A \times B) \cap E|$ with probability at least $(1 - e^{-4n})(1 - e^{-6n})$.*

Proof. To prove this we will use the Chernoff bounds. The Chernoff bounds state that, given a set of $|S|$ independent and identically distributed Bernoulli random variables $X_1, \dots X_{|S|}$ with expected value μ, we have that

$$\mathbb{P}\left[\sum_{i=1}^{|S|} X_i > (1+\epsilon)|S| \cdot \mu\right] < exp\left(-\frac{\epsilon^2 \cdot |S| \cdot \mu}{3}\right) \quad and$$

$$\mathbb{P}\left[\sum_{i=1}^{|S|} X_i < (1-\epsilon)|S|\cdot\mu\right] < exp\left(-\frac{\epsilon^2\cdot|S|\cdot\mu}{2}\right)$$

Let's take as random variable

$$X_i = \begin{cases} 1, & \text{if the i-th edge in S belongs to the max-cut} \\ 0, & \text{otherwise} \end{cases}$$

Given this assumption we can say that

$$\mathbb{P}\left[X_i = 1\right] = \frac{|(A\times B)\cap E)|}{|E|}$$

since we can express it as $\frac{\#\text{of desired cases}}{\#\text{ of possible cases}}$. Nevertheless we can say that that

$$\mu = \mathbb{E}(X_i) = 0\cdot\mathbb{P}\left[X_i = 0\right] + 1\cdot\mathbb{P}\left[X_i = 1\right] = \frac{|(A\times B)\cap E)|}{|E|}$$

By using these ingredients and knowing that $\sum_{i=1}^{|S|} X_i = |(A\times B)\cap S|$ we can say that

$$\mathbb{P}\left[\sum_{i=1}^{|S|} X_i > (1+\epsilon)|S|\cdot\mu\right] < exp\left(-\frac{\epsilon^2\cdot|S|\cdot\mu}{3}\right)$$

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

$$\mathbb{P}\left[|(A\times B)\cap S|\cdot\frac{|E|}{|S|} > (1+\epsilon)\cdot|(A\times B)\cap E)|\right] < exp\left(-\frac{\epsilon^2\cdot\frac{24n}{\epsilon^2}\cdot\frac{|(A\times B\cap E)|}{|E|}}{3}\right)$$

Since we can assume that the max-cut is at least $\frac{|E|}{2}$, and this gives us a lower bound on the probability[1], we can conclude that

$$\mathbb{P}\left[|(A\times B)\cap S|\cdot\frac{|E|}{|S|} > (1+\epsilon)\cdot|(A\times B)\cap E)|\right] < exp\left(-4n\right)$$

By iterating the same reasoning on the second Chernoff bound we can say that

$$\mathbb{P}\left[|(A\times B)\cap S|\cdot\frac{|E|}{|S|} < (1-\epsilon)\cdot|(A\times B)\cap E)|\right] < exp\left(-6n\right)$$

Given that, for each event, we consider its complementary, and we'd like the intersection of these events, since they are independent we can use the fact that

$$\mathbb{P}(A\cap B) = \mathbb{P}(A)\mathbb{P}(B)$$

These considerations allows us to write

$$\mathbb{P}\left[(1-\epsilon)\cdot|(A\times B)\cap E)| \le |(A\times B)\cap S|\cdot\frac{|E|}{|S|} \ge (1+\epsilon)\cdot|(A\times B)\cap E)|\right] \le (1-e^{-4n})(1-e^{-6n})$$

Since $n > 0$ this probability is at least 0.97925 which is greater than $\frac{1}{2}$.

□

Now let's prove that this holds for all cuts. Let's consider again

$$X_i = \begin{cases} 1, & \text{if the i-th edge in S belongs to the cut (A',B')} \\ 0, & \text{otherwise} \end{cases}$$

[1] Choosing a value greater than $\frac{|E|}{2}$ will give us a better probability.

We can say that

$$\mathbb{P}\left[X_i = 1\right] = \frac{|(A' \times B') \cap E)|}{|E|}$$

and

$$\mu = \frac{|(A' \times B') \cap E)|}{|E|}$$

The resulting modified Chernoff bound is

$$\mathbb{P}\left[|(A' \times B') \cap S| \cdot \frac{|E|}{|S|} > (1+\epsilon) \cdot |(A' \times B') \cap E)|\right] < exp\left(-\frac{\epsilon^2 \cdot \frac{24n}{\epsilon^2} \cdot \frac{|(A' \times B') \cap E)|}{|E|}}{3}\right)$$

$$\mathbb{P}\left[|(A' \times B') \cap S| \cdot \frac{|E|}{|S|} > (1+\epsilon) \cdot |(A' \times B') \cap E)|\right] < exp\left(-8n \cdot \frac{|(A' \times B') \cap E)|}{|E|}\right)$$

But this is not enough to prove what we want. By using union bound, we can say that the probability that this holds for every cut is

$$\mathbb{P}\left[\cup \text{ Chernoff bound holds for cut i}\right] \leq \sum_{i=1}^{2^n} exp\left(-8n \cdot \mu_i\right)$$

Since we want a lower bound for this probability we will consider $\mu = \frac{c}{|E|}$, where c is the minimum global cut.[2] Therefore we can say that this is equal to

$$2^n \cdot exp\left(-8n \cdot \frac{c}{|E|}\right)$$

and since $c \leq |(A \times B) \cap E|$, otherwise the statement is already proven, we can conclude that

$$2^n \cdot exp\left(-8n \cdot \frac{c}{|E|}\right) \leq 2^n \cdot exp\left(-4n\right)$$

Nevertheless, the probability of the complementary event which tells us which is the probability that $|(A' \times B') \cap S| \cdot \frac{|E|}{|S|} < (1+\epsilon) \cdot |(A' \times B') \cap E)|$ holds for all cuts is at least $1 - 2^n e^{-4n}$. Given that

$$\lim_{n \to \infty} \frac{2^n}{e^{4n}} = 0$$

we can confidently state that this probability is small.

Since we return $|(A \times B) \cap S| \cdot \frac{|E|}{|S|}$ as an estimation for the real max-cut, we still provide an algorithm for computing $(A \times B) \cap S|$.

```
Algorithm:
1. For i in range 1 .. |V|/2:
2.      Build all possible cuts (A,B) where |A|=i and |B|=|V|-i
3.      For each cut j where |A|=i and |B|=|V|-i
4.          current_cut = Find |(A × B) ∩ S|
5.          If current_cut > max_cut then max_cut = current_cut
6. Return max_cut
```

Theorem 2.2. *Algorithm has running time $O(|E| \cdot 2^n)$.*

Proof. Since there are 2^n possible cuts in a graph and that, given (A, B), we can find $|(A \times B) \cap S|$ in $O(|E|)$ because for each edge we control if its vertices are respectively in A and B, the total running time is $|E| \cdot 2^n$. □

[2] Considering a value higher than the minimum cut would only give us a higher probability.

3 Exercise 3

3.1

Let's see how the primal and dual change according to this request. Given that the cost of connecting city j to facility i is equal now to $d_j c_{ij}$ we will see a slightly modified objective function. LP-relaxation of the modified problem:

$$\begin{aligned} \text{minimize} \quad & \sum_{i \in F, j \in C} d_j c_{ij} x_{ij} + \sum_{i \in F} f_i y_i \\ \text{subject to} \quad & \sum_{i \in F} x_{ij} \geq 1, \forall j \in C \\ & y_i - x_{ij} \geq 1, \forall i \in F, j \in C \\ & x_{ij} \geq 0, \forall i \in F, j \in C \\ & y_i \geq 0, \forall i \in F \end{aligned} \tag{1}$$

The dual becomes:

$$\begin{aligned} \text{maximize} \quad & \sum_{j \in C} \alpha_j \\ \text{subject to} \quad & \alpha_i - \beta_{ij} \leq d_j c_{ij}, \forall i \in F, j \in C \\ & \sum_{i \in F} \beta_{ij} \leq f_i, \forall i \in F \\ & \alpha_j \geq 0, \forall j \in C \\ & \beta_{ij} \geq 0, \forall i \in F, j \in C \end{aligned} \tag{2}$$

Like in the original algorithm of Vazirani, we will partition the cities in two sets, directly and indirectly connected cities, where only the directly connected cities will pay for opening facilities, which means that only if j is a directly connected city and i = $\phi(j)$ then $\beta_{ij} > 0$. By using this distinction, only for an indirectly connected city j the primal condition is relaxed:

$$\frac{1}{3} d_j c_{\phi(j)j} \leq \alpha_j \leq d_j c_{\phi(j)j}$$

For the directly connected cities the primal conditions are maintained:

$$\alpha_j - \beta_{\phi(j)j} = d_j c_{\phi(j)j}$$

$$\sum_{j : \phi(j) = i} \beta_{ij} = f_i$$

Let's analyze the slightly modified algorithm. Like before, the algorithm is divided in two phases:
Phase 1: the algorithm finds a dual feasible solution and determines a set of tight edges and temporarily open facilities.
Phase 2: given the set of temporarily open facilities, F_t, we choose a subset to open, I of F_t, and find a mapping ϕ from cities to I.

Phase 1

The objective is to find a dual solution as large as possible so we keep raising α_j of each city until it gets connected to an open facility. We modify all other primal and dual variables according to this change, but still being careful at maintaining feasibility or satisfying complementary slackness conditions. We also define the notion of time so that we can associate each event with the time at which it happened. Initially all cities are not connected to any facility. We will raise the dual variable α_j for each not connected city j uniformly not at unit rate, but at rate d_j. When $\alpha_j = d_j c_{ij}$ for some edge (i, j) we will declare this edge to be tight. We should notice that, given that we raise at rate d_j and because the edge becomes tight when $\alpha_j = d_j c_{ij}$, the edge (i, j) still goes tight at time c_{ij}. From now on also β_{ij} will be raised uniformly still at rate d_j so facility i may get opened earlier than in the unit demands case. This is so because every β_{ij} is increased more than before so it will happen earlier that $\sum_{i \in F} \beta_{ij} = f_i$. When this happens we say that facility i is paid for and we declare it temporarily open. We also call each edge (i, j) such that $\beta_{ij} > 0$ special.

When facility i is temporarily open, all unconnected cities having tight edges to this facility are declared connected and facility i is called connecting witness for each of these cities. If it happens that another unconnected city j gets a tight edge to i then also j is declared connected and i it's connecting witness, but this edge is not special since $\beta_{i,j} = 0$ given that we can't raise it anymore for that connection because the sum is already equal to f_i. This phase terminates when all cities are connected. At the end of this phase it may be that a city paid towards temporarily opening several facilities, but we want to ensure that it pays only for the facility that will be eventually connected to.

Phase 2

We shall call F_t the set of temporarily open facilities and T the subgraph of G consisting of all special edges, edges (i, j) such that $\beta_{i,j} > 0$. We define T' the graph that has edge (u, v) iff there is a path of length at most 2 between u and v in T and H the subgraph of T' induced on F_t. Any maximal independent set in H, that we call I, will contain the facilities that are to be opened. For each city j, let's define $\psi = \{i \in F_t | \text{(i,j) is special}\}$ which means that in ψ we have all edges that have contributed to opening a facility ($\beta_{i,j} > 0$). Since I is an independent set, at most one of the facilities in ψ is opened. Three cases can happen:

1. If there is a facility $i \in \psi$ that is opened, then set $\phi(j) = i$ and declare city j directly connected.
2. If this didn't happen then consider tight edge (i',j) (such that $\alpha_j = d_j c_{i'j}$) where i' was the connecting witness for j. If $i' \in I$, then set $\phi(j) = i'$ and declare city j directly connected, but in this case $\beta_{i'j} = 0$.
3. If $i' \notin I$, then consider i" neighbor of i' in H such that $i'' \in I$. We can set $\phi(j) = i''$, but declare city j indirectly connected.

By doing this we have defined a primal integral solution. Nevertheless $x_{ij} = 1$ iff $\phi(j) = i$ and $y_i = 1$ iff $i \in I$, while the obtained α_j and β_{ij} form a dual feasible solution.

Given the way in which we relaxed the primal complementary slackness condition for the indirectly connected cities, the cost of the integral solution found would be withing thrice the dual found which leads us to a factor 3 approximation algorithm. We will prove this by using the following theorem

Theorem 3.1. *The primal and dual solutions satisfy*

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

Proof. To prove this we will need a few instruments. Firstly we know that α_j pays for the primal costs of opening facilities and connecting cities to facilities so let's consider

$$\alpha_j = \alpha_j^f + \alpha_j^e$$

such costs. By how we defined directly and indirectly connected cities, it holds that:

- if j is indirectly connected: $\alpha_j^f = 0$ and $\alpha_j^e = \alpha_j$ since this city doesn't pay for opening facility i
- if j is directly connected: $\alpha_j = \beta_{ij} + d_j c_{ij}$

Standing to this definitions, if we let $\alpha_j^f = \beta_{ij}$ and $\alpha_j^e = d_j c_{ij}$, if i ∈ I and since it was temporarily open at the end of Phase 1, then we can state that

$$\sum_{j : \phi(j) = i} \alpha_j^f = f_i$$

and given that indirectly connected cities don't participate in opening facilities in I, it follows that

Remark 3.1.

$$\sum_{j \in C} \alpha_i^f = \sum_{i \in I} f_i$$

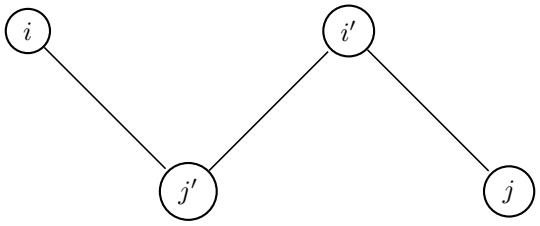
The final intermediate step that we have to do is prove that, for an indirectly connected city j,

$$d_j c_{ij} \leq 3\alpha_j^e \text{ where } \phi(j) = i$$

Let's define i, i', j, j' as:

- i' the connecting witness for city j
- (i,i') must be an edge in H since j is indirectly connected to i, which means that the path between the two is 2
- given this conditions, there must exist a city j' such that (i,j') and (i',j') are both special edges, which means that j' participated in both opening i and i'

Furthermore let's consider t_1 and t_2 the times at which i and i' were declared temporarily open during Phase 1.



Since (i', j) is tight, it follows that $\alpha_j \geq d_j c_{i'j}$. To prove that $d_j c_{ij} \leq 3\alpha_j^e$ we will show that

$$\alpha_j \geq d_{j'} c_{ij'} \quad and \quad \alpha_j \geq d_{j'} c_{i'j'}$$

Once this is proven, our claim follows by using the triangle inequality. Since we know that (i', j') and (i, j') are tight, it follows directly that $\alpha_{j'} \geq d_{j'} c_{ij'}$ and $\alpha_{j'} \geq c_{i'j'}$. We know that during Phase 1, $\alpha_{j'}$ stops growing as soon as one of the facilities that j' has a tight edge to opens. This means that $\alpha_{j'} \leq min(t_1, t_2)$. Since i' is the connecting witness for j, we know that $\alpha_j \leq t_2$. We can conclude that $\alpha_j \leq \alpha_{j'}$ which means that our inequalities follow.

After all these needed remarks, we can finally prove our initial statement of the theorem. Given that, for a directly connected city j, $d_j c_{ij} = \alpha_j^e \leq 3\alpha_j^e$, if we combine it with $d_j c_{ij} \leq 3\alpha_j^e$ as we have proved above for an indirectly connected city j, then

$$\sum_{i \in F, j \in C} d_j c_{ij} x_{ij} \leq 3 \sum_{j \in C} \alpha_j e$$

Let's add Remark 3.1. multiplied by 3 and we'll obtain the wanted disequality

$$\sum_{i \in F, j \in C} d_j c_{ij} x_{ij} + 3 \sum_{i \in I} f_i \leq 3 \sum_{j \in C} \alpha_j e + 3 \sum_{j \in C} \alpha_i^f$$

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

□

For proving the following theorem, we will use n_f to define the number of facilities, n_c to define the number of cities and $m = n_f \times n_c$ will be the total number of edges.

Theorem 3.2. *This algorithm that achieves a 3-factor approximation has a running time of* $O(m \log m)$.

Proof. To prove this, let's sort all the edges by increasing cost, so that we can have the order and the times at which the edges go tight. Once we have done this, we will store for each facility i the number of cities that are currently contributing towards it together with the anticipated time t_i at which it would be completely paid for if no other event happens on the way. We will initialize all t_i with the value ∞ and each facility will naturally have 0 cities contributing to it. The best way in which we can update each one and find che current minimum is by using a priority queue, that we will implement using a binary heap. This will grant us a cost of $O(\log n_f)$ for finding the minimum. In order for an update to happen two events can occur:

1. An edge (i, j) goes tight, which means that a city starts contributing to opening that particular facility.

2. Facility i is completely paid for.

Let's discuss what happens at each event.
1.1. We know that if facility i is not temporarily open, then one more city will pay for it. We can compute the amount that city j will contribute towards opening it in constant time. This allows us to also recompute the anticipated time at which facility i will be paid for and so we can update the heap in $O(\log n_f)$ time.
1.2. If facility i is already temporarily open, then city j is declared connected and α_j is not raised anymore. For each facility i' that was counting j as a contributor, we need to decrease the number of contributors by 1, and recompute the anticipated time at which it gets paid for and still update the heap in $O(\log n_f)$.
2. If facility i is completely paid for, then i will be declared temporarily open and all cities contributing to i will be declared connected. For each of these city j, for each facility i' that was counting it as a contributor, we will decrease the number of contributors by 1 and recompute the anticipated time at which it gets paid for still in time $O(\log n_f)$.
Given that each edge (i, j) will be considered twice, once when it goes tight and once when city j is declared connected and that we will do $O(\log n_f)$ for each edge, we can conclude that this algorithm has a running time of $O(m \log m)$. □

3.2

Let's return to the basic version of the problem and assume that a facility can be opened multiple times, but each time the facility is opened, it can serve at most u_i cities. If this is the case then we need to add the following extra constraint to the primal:

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

which means that the total number of cities that can be connected to i is limited by how many times this facility opens. If we use γ_i as the dual variable for this constraint then we obtain the following dual program:

$$\begin{aligned}
\text{maximize} \quad & \sum_{j \in C} \alpha_j \\
\text{subject to} \quad & \alpha_i - \beta_{ij} - \gamma_i \leq d_j c_{ij}, \forall i \in F, j \in C \\
& u_i \gamma_i + \sum_{i \in F} \beta_{ij} \leq f_i, \forall i \in F \\
& \alpha_j \geq 0, \forall j \in C \\
& \beta_{ij} \geq 0, \forall i \in F, j \in C \\
& \gamma_i \geq 0, \forall i \in F
\end{aligned} \tag{3}$$

Even if we added this variable, a good method for solving this problem would be to eliminate it in some way and find an integral solution with our original algorithm. In order to do this let's try assigning $\gamma_i = \frac{3f_i}{4u_i}$. The dual becomes

$$\begin{aligned}
\text{maximize} \quad & \sum_{j \in C} \alpha_j \\
\text{subject to} \quad & \alpha_i - \beta_{ij} - \frac{3f_i}{4u_i} \leq c_{ij}, \forall i \in F, j \in C \\
& u_i \left(\frac{3f_i}{4u_i} \right) + \sum_{i \in F} \beta_{ij} \leq f_i, \forall i \in F \\
& \alpha_j \geq 0, \forall j \in C \\
& \beta_{ij} \geq 0, \forall i \in F, j \in C \\
& \frac{3f_i}{4u_i} \geq 0, \forall i \in F
\end{aligned} \tag{4}$$

After some slight modifications we can see that the primal becomes

$$\begin{aligned} \text{minimize} \quad & \sum_{i\in F, j\in C}\left(c_{ij}+\frac{3f_i}{4u_i}\right)x_{ij}+\sum_{i\in F}\frac{f_i}{4}Y_i \\ \text{subject to} \quad & \sum_{i\in F}x_{ij}\geq 1, \forall j\in C \\ & Y_i - x_{ij}\geq 1, \forall i\in F, j\in C \\ & x_{ij}\geq 0, \forall i\in F, j\in C \\ & Y_i\geq 0, \forall i\in F \end{aligned} \tag{5}$$

We can see that $c_{ij}+\frac{3f_i}{4u_i}$ still follows the triangle inequality and we can use our algorithm to find an integral solution. This solution, by Theorem 3.1. where we consider demand to be equal to one ($d_j = 1$) satisfies

$$\sum_{i\in F, j\in C}\left(c_{ij}+\frac{3f_i}{4u_i}\right)x_{ij}+3\sum_{i\in F}\frac{f_i}{4}Y_i\leq 3\sum_{j\in C}\alpha_j$$

We can still consider x_{ij} as previously, while y_i can be achieved if we use $y_i = \left\lceil \frac{\sum_{j\in C} x_{ij}}{u_i} \right\rceil$. By doing so, we achieve

$$y_i \leq Y_i + \frac{\sum_{j\in C} x_{ij}}{u_i}$$

If we use this inequality and the one offered by Theorem 3.1 we get that

$$\sum_{i\in F, j\in C} c_{ij}x_{ij} + \frac{3}{4}\sum_{i\in F} f_i y_i \leq 3\sum_{j\in C}\alpha_j$$

This finally implies that

$$\sum_{i\in F, j\in C} c_{ij}x_{ij} + \sum_{i\in F} f_i y_i \leq 4\sum_{j\in C}\alpha_j$$

which guarantees us a 4-factor approximation.

4 Exercise 4

4.1

To prove that starting from any solution it's possible to converge to a pure Nash equilibrium we will need to introduce the concept of potential games. Such games utilize a function Φ called potential function. Let's define in particular a weighted potential game.

Definition 4.1. A game $G = (N, A = A_1 \times \cdots \times A_N, u : A \to \mathbb{R}^N)$ is a weighted potential game if there is a function $\Phi : A \to \mathbb{R}$ and a vector $w \in \mathbb{R}^N_{++}$ such that $\forall a_{-i} \in A_{-i}, \forall a'_i, a''_i \in A_i$,

$$\Phi(a'_i, a_{-i}) - \Phi(a''_i, a_{-i}) = w_i(u_i(a'_i, a_{-i}) - u_i(a''_i, a_{-i}))$$

Once we have found the existence of this potential function Φ we can prove the following theorem.

Theorem 4.1. *Given our load balancing problem, if we have found a potential function Φ, then starting from any solution we will always converge to a pure Nash equilibrium.*

Proof. If we allow jobs to switch from machine j to machine k because it will give a better utility, then Φ will decrease until no job has the need to change it's current machine, which means that a pure Nash equilibrium has been found. Since Φ decreases in each switch and because there are only a finite number of ways of assigning the jobs to the machines, eventually we will reach a pure Nash equilibrium. □

For finalizing our solution let's prove that $\Phi = \frac{1}{2}\sum_j L_j^2$ is a potential function.

Theorem 4.2. $\Phi = \frac{1}{2}\sum_j L_j^2$ *is a potential function for our load balancing problem.*

Proof. Let's suppose that player i switches from machine j to machine k. If this happens we have that:

$$\Delta u_i(a) = u_i(k, a_{-i}) - u_i(j, a_{-i})$$
$$\Delta u_i(a) = L_k(a) + w_i - L_j(a)$$

While for the difference in the potential function we have that:

$$\Delta\Phi(a) = \Phi(k, a_{-i}) - \Phi(j, a_{-i})$$
$$\Delta\Phi(a) = \frac{1}{2}((L_k(a) + w_i)^2 + (L_j(a) - w_i)^2 - L_k^2(a) - L_j^2(a))$$
$$\Delta\Phi(a) = \frac{1}{2}(L_k^2(a) + 2L_k^2(a)w_i + w_i^2 L_j^2(a) + 2L_j^2(a)w_i + w_i^2 - L_k^2(a) - L_j^2(a))$$
$$\Delta\Phi(a) = w_i(L_k(a) + w_i - L_j(a))$$
$$\Delta\Phi(a) = w_i\Delta u_i(a)$$

□

4.2

Theorem 4.3. *The makespan of the pure Nash equilibrium*[3] *is at most twice the minimum makespan, more precisely*

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

Proof. Let's call j a machine with the highest load and i the agent with smallest weight w_i. There must be at least two tasks assigned to machine j, otherwise MNE = OPT. Given that there are at least two tasks and w_i is the smallest, then $w_i \leq \frac{1}{2}$MNE. Now let's consider a generic machine $k \neq j$ with load $< L_j - w_i$. If this would be the case, then moving task i from j to k would decrease the makespan of agent i. Since this is a pure Nash equilibrium and no agent would choose a different machine, it must happen that

$$L_k \geq L_j - w_i \geq \text{MNE} - \frac{1}{2}\text{MNE} = \frac{1}{2}\text{MNE}$$

Given that this is a load balancing problem, the minimum makespan is necessarily greater or equal than the average load over all machines. By using this claim, we get that

$$OPT \geq \frac{\sum_{j\in m} L_j}{m} \geq \frac{\text{MNE} + \frac{1}{2}\text{MNE}(m-1)}{m} = \frac{\text{MNE}(m+1)}{2m}$$

Therefore we can way that

$$\text{MNE} \leq \frac{2m}{m+1} \cdot OPT = (2 - \frac{2}{m+1}) \cdot OPT$$

□

5 Exercise 5

A good answer to this exercise is to use the algorithm offered by John von Neumann.

```
Algorithm:
1. Toss the coin twice
2. If the results match, start over, forgetting both results
3. If the results differ, use the first result, forgetting the second.
```

Theorem 5.1. *Given a biased coin, the proposed algorithm gives the same probability of an unbiased coin.*

[3] We will refer at the makespan, $max_j L_j$, of the pure Nash equilibrium as MNE.

Proof. Given the event: `2 coin tosses`, what are the probabilities of the 4 possible outcomes? If we call p the probability of Head and $q = 1 - p$ the probability of Tails we can visualize them as follows:

	Head	Tails
Head	pp	pq
Tail	qp	qq

Given that the probability of the outcome Head-Tails (pq) is the same as the probability of the outcome Tails-Head (qp), we can consider them as 2 events with the same probability. If from these 2 events we take only the first coin toss, we can distinguish between them and use them as we would have used an unbiased coin. □

Given this knowledge, can we compute the expected number of coin flips to obtain a fair result? To obtain it, let's first see what is the expected number of rounds to be played so that the algorithm finishes. It can be defined with the following equation:

$$\mathbb{E}(r) = P(result) + P(noResult)(1 + \mathbb{E}(r)) \quad \text{r=number of rounds}$$

This is because if we don't succeed in the first round, the expected number of rounds remains the same. Given that the probability of obtaining a result is $2pq$ [4] we can see that the expected number of rounds to be played is

$$\mathbb{E}(r) = 2pq + (1 - 2pq)(1 + \mathbb{E}(r))$$

$$\mathbb{E}(r) = 2pq + 1 + \mathbb{E}(r) - 2pq - 2pq\,\mathbb{E}(r)$$

$$\mathbb{E}(r) = \frac{1}{2p(1-p)}$$

So the expected number of coin flips is

$$2\,\mathbb{E}(r) = \frac{2}{2p(1-p)} = \frac{1}{p(1-p)}$$

References

[1] Minimum bounding rectangle. https://en.wikipedia.org/wiki/Minimum_bounding_rectangle.

[4] Probability of obtaining Head-Tails(pq) + the probability of obtaining Tails-Head(qp)