CS 598RM: Algorithmic Game Theory, Fall 2020
HW 3 Solutions

1. (a) (5 points) Prove that if $\mathcal{C}$ is the set of cost functions of the form $c(x) = ax^2 + bx + c$ with $a, b, c \geq 0$, then the Piguo bound $\alpha(\mathcal{C})$ is $\frac{3\sqrt{3}}{3\sqrt{3}-2}$.

   (b) (5 points) Give example of a potential game where the strategy profile achieving minimum potential does not give the minimum cost NE.
   [Hint: Try cost-sharing game.]

(a) Let $f(x) := ax^2 + bx + c$. We wish to find

$$\sup_{r,x,a,b,c \geq 0} \quad \frac{r \cdot f(r)}{x \cdot f(x) + (r-x) \cdot f(r)} \;=\; \frac{1}{\frac{x \cdot f(x)}{r \cdot f(r)} + 1 - \frac{x}{r}} \;=\; \frac{1}{\frac{x}{r}\left(\frac{f(x)}{f(r)} - 1\right) + 1}$$

Note that, since $a, b, c \geq 0$, $f(x)$ must be monotone non-decreasing for $x \geq 0$. Thus, for $x > r$,

$$\frac{x}{r}\left(\frac{f(x)}{f(r)} - 1\right) + 1 \;>\; 1 \cdot (1-1) + 1 = 1$$

However, for $x = r$, this evaluates exactly to 1. Thus, we may restrict ourselves to the case $0 \leq x \leq r$.

Now, we wish to show that for $0 \leq x \leq r$, it is best to set $b = c = 0$. From the equation above, it should be clear that to maximize $\frac{r \cdot f(r)}{x \cdot f(x) + (r-x) \cdot f(r)}$ for $x \leq r$ fixed, it suffices to minimize $f(x)/f(r) = (ax^2 + bx + c)/(ar^2 + br + c)$. Now, this quotient takes the form $(A + c)/(B + c)$ for $0 \leq A < B$, and so the value increases, tending to 1, as c increases. Thus, it is beneficial to restrict to the $c = 0$ case. A similar argument, given $c = 0$, says we can restrict ourselves to the $b = 0$ case, since $f(x)/f(r) = (x/r) \cdot (ax + b)/(ar + b)$. Thus, $f(x) = ax^2$, and

$$\frac{r \cdot f(r)}{x \cdot f(x) + (r-x) \cdot f(r)} \;=\; \frac{ar^3}{ax^3 + (r-x)ar^2} \;=\; \frac{1}{(x/r)^3 + 1 - (x/r)}$$

Setting $\lambda = (x/r)$, it suffices to maximize $\phi(\lambda) = 1/(1 - \lambda + \lambda^3)$. We have

$$\phi'(\lambda) = \frac{1 - 3\lambda^2}{(1 - \lambda + \lambda^3)^2}$$

Setting $\phi'(\lambda) = 0$, and using the fact that $\lambda > 0$, we get

$$\phi'(\lambda) = 0 \implies 1 - 3\lambda^2 = 0 \implies \lambda = \frac{1}{\sqrt{3}}$$

(b) Consider a cost sharing game with 3 players over a directed graph, with a distinct source for each, denoted by s_1, s_2 and s_3, and one sink t. There is another vertex v. There are edges from every source to the sink t, every source to v, and $v \to t$. The costs of edges

are: $\gamma_{s_i \to t} = 1$ and $\gamma_{s_i \to v} = 0$ for every source s_i, and $\gamma_{v \to t} = 1 - \epsilon$, for a small $\epsilon > 0$. For every player, there are two $s \to t$ paths, either the direct edge $s_i \to t$ or the path $s_i \to v \to t$. We call the former the direct path and the latter the indirect.

This game has two Nash equilibria, where either all players take the direct path or all go via the indirect path. For any other strategy, there are two players both choosing the direct (or indirect) paths, and the third choosing the indirect (resp. direct). Then the player choosing the other type of path increases payoff by deviating to the type of path chosen by others. Indeed, if two players go through the direct paths, then the third is paying $1 - \epsilon$ to take the indirect path, and can improve by paying $1/3$ and taking the direct path instead (for this to work we want $\epsilon < 2/3$). Similarly, if two are choosing the indirect path, then the third player, currently paying $1/3$, can pay $(1-\epsilon)/3$ by deviating to the indirect path. For the same reason, no player has reason to deviate if all have chosen the same type of paths, hence these strategies are Nash equilibria.

Consider the NE where all choose the direct path. The cost of each player is $1/3$. The (Rosenthal) potential value of this strategy vector is $\phi = \sum_e \sum_{i=1}^{f_e} \gamma_e / i = 1/3 \cdot 3 = 1$.

Now consider the NE where all players choose the indirect paths. The cost to each player is $(1 - \epsilon)/3$. The potential value is $\phi = \sum_e \sum_{i=1}^{f_e} \gamma_e / i = 0 \cdot 3 + (1 - \epsilon) \cdot (1 + 1/2 + 1/3) = (1 - \epsilon) \cdot 11/6 > 1$, when $\epsilon < 5/11$.

Hence, the second NE has smaller cost, and for any $\epsilon \in (0, 5/11)$, higher potential than the first.

2. (10 points) This problem develops some theory about potential games; we talked about these while discussing selfish routing. We consider an abstract finite game with n players with finite strategy sets $S_1, \ldots, S_n$. Each player has a payoff function π_i mapping outcomes (elements of $S_1 \times \cdots \times S_n$) to real numbers. Recall that a potential function for such a game is defined by the following property: for every outcome $\boldsymbol{s} \in S_1 \times \cdots \times S_n$, every player i, and every deviation $s_i' \in S_i$.

$$\pi_i(s_i', \boldsymbol{s}_{-i}) - \pi_i(s_i, \boldsymbol{s}_{-i}) = \Phi(s_i', \boldsymbol{s}_{-i}) - \Phi(s_i, \boldsymbol{s}_{-i}).$$

A team game is a game in which all players have the same payoff function: $\pi_1(\boldsymbol{s}) = \cdots = \pi_n(\boldsymbol{s})$ for every outcome $\boldsymbol{s}$. In a dummy game, the payoff of every player i is independent of its strategy: $\pi_i(s_i, \boldsymbol{s}_{-i}) = \pi_i(s_i', \boldsymbol{s}_{-i})$ for every $\boldsymbol{s}_{-i}$ and every $s_i, s_i' \in S_i$.

Prove that a game with payoffs $\pi_1, \ldots, \pi_n$ is a potential game (i.e., admits a potential function) if and only if it is the sum of a team game $\pi_1^t, \ldots, \pi_n^t$ and a dummy game $\pi_1^d, \ldots, \pi_n^d$ (*i.e.,* $\pi_i(\boldsymbol{s}) = \pi_i^t(\boldsymbol{s}) + \pi_i^d(\boldsymbol{s})$ for every i and $\boldsymbol{s}$.)

We prove both ways, that a game is a potential game if and only if it is the sum of a team game and a dummy game separately.

Given game is the sum of a team game and a dummy Game ⇒ it is a potential game. The payoff of the given game at any strategy vector $\mathbf{s}$ is the sum of the payoffs from the team game and the dummy game, i.e., $\pi(\mathbf{s}) = \pi^t(\mathbf{s}) + \pi^d(\mathbf{s})$.

Define a function as the payoff of the team game. That is, $\phi(\mathbf{s}) = \pi^t(\mathbf{s})$. We will prove that this function is a valid potential function for the given game, by showing the difference in payoffs of any player by unilaterally deviating from a strategy vector $\mathbf{s}$, is the difference in the function value $\phi(.)$ at these strategy vectors.

$$\begin{aligned}
\pi_i(s_i', \mathbf{s}_{-i}) - \pi_i(s_i, \mathbf{s}_{-i}) &= \pi_i^t(s_i', \mathbf{s}_{-i}) + \pi_i^d(s_i', \mathbf{s}_{-i}) - \pi_i^t(s_i, \mathbf{s}_{-i}) - \pi_i^d(s_i, \mathbf{s}_{-i}) \\
&\quad \ldots \text{ (as game is sum of a team and a dummy game)} \\
&= \pi_i^t(s_i', \mathbf{s}_{-i}) - \pi_i^t(s_i, \mathbf{s}_{-i}) \\
&\quad \ldots \text{(as dummy game has same payoff at every strategy)} \\
&= \phi(s_i', \mathbf{s}_{-i}) - \phi(s_i, \mathbf{s}_{-i}) \quad \ldots \text{(by definition of } \phi).
\end{aligned}$$

Given game is a potential game ⇒ it is the sum of a team game and a dummy game. First, define a team game with payoff of every player i for strategy $\mathbf{s}$ as $\pi_i^t(\mathbf{s}) = \phi(\mathbf{s})$.

Define another game that gives each player payoff equal to the difference in payoffs from the given and the team games. That is, $\pi_i^d(\mathbf{s}) = \pi_i(\mathbf{s}) - \pi_i^t(\mathbf{s})$. If we show this game is a dummy game, we are done. For this, we show the definition of the dummy game is satisfied, i.e., we have $\pi_i^d(s_i, \mathbf{s}_{-i}) = \pi_i^d(s_i', \mathbf{s}_{-i})$.

$$\begin{aligned}
\pi_i(s_i', \mathbf{s}_{-i}) - \pi_i(s_i, \mathbf{s}_{-i}) &= \pi_i^t(s_i', \mathbf{s}_{-i}) + \pi_i^d(s_i', \mathbf{s}_{-i}) - \pi_i^t(s_i, \mathbf{s}_{-i}) - \pi_i^d(s_i, \mathbf{s}_{-i}) \\
&= \phi(s_i', \mathbf{s}_{-i}) - \phi(s_i, \mathbf{s}_{-i}) + \pi_i^d(s_i', \mathbf{s}_{-i}) - \pi_i^d(s_i, \mathbf{s}_{-i}).
\end{aligned}$$

But we know from the definition of the given game that $\pi_i(s_i', \mathbf{s}_{-i}) - \pi_i(s_i, \mathbf{s}_{-i}) = \phi(s_i', \mathbf{s}_{-i}) - \phi(s_i, \mathbf{s}_{-i})$. Hence, $\pi_i^d(s_i', \mathbf{s}_{-i}) - \pi_i^d(s_i, \mathbf{s}_{-i}) = 0$, completing the proof.

3. Consider a combinatorial auction with n bidders and n items where each bidder i has a unit-demand valuation v_i. This means that $v_i(S) = \max_{j \in S} v_{i,j}$ for every subset S of items. We assume that $v_{i,j} > 0$ for all i, j.

   In this auction, each bidder i submits one bid $b_{i,j}$ for each item j, and each item is sold separately using a second-price single-item auction. Assume that $b_{i,j} \in (0, v_{i,j})$ for all i, j. The utility of a bidder is her value for the items won, minus her total payment. For example, if bidder i has values v_{i1} and v_{i2} for two items, and wins both items when the second-highest bids are p_1 and p_2, then her utility is $\max\{v_{i1}, v_{i2}\} - (p_1 + p_2)$. Let $G = (A, B)$ be a bipartite graph where A is the set of bidders and B is the set of items.

   - (7 points) Show that every allocation π of items to bidders that maximizes the Social Welfare $\left(\sum_i v_i(\pi(i))\right)$ induces a matching on G.
   - (8 points) Show that the PoA of PNE in such a game can be at most 2.

In this problem, we want to show that the PoA of PNE is at most 2. The main idea is to show that, since bidders are unit-demand, every allocation which maximizes the sum of valuations will induce a matching on G, as otherwise some bidders gets two items but only receives value from one of them and some other bidder gets no items when they could've gotten one. Afterwards, the idea is to look at a bidder i and at bidder i^*, who gets i's item at the optimal allocation, and use the equilibrium properties to relate their valuations.

(a) Let OPT denote an optimal allocation, and suppose OPT does not induce a perfect matching on G. Then, there exist a bidder i that is not assigned any item. Furthermore, there exists a bidder i' who is assigned at least two items. Consider two of the items assigned to i', say items j_1 and j_2, and let, without loss of generality, $j_1 = \arg\max\{v_{i',j_1}, v_{i',j_2}\}$. Then, i' receives no utility from being assigned item j_2. Also, we know that $v_{i,j_2} > 0$. Therefore

$$u_{i'}(\{j_1, j_2\}) + u_i(\emptyset) = v_{i',j_1} < v_{i',j_1} + v_{i,j_2} = u_{i'}(\{j_1\}) + u_i(\{j_2\})$$

and we contradict the fact that the social welfare is maximized in OPT. Thus, OPT has to be a perfect matching in G.

(b) Consider a bidder i who, at equilibrium is assigned item j. Suppose that, at the optimal solution, i was assigned item j^*. Let $i^* = \arg\max_{i'} b_{i',j^*}$ be the agent that is assigned j^* at equilibrium. Then, since we are at equilibrium, we have

$$v_{i,j} - p_i \geq v_{i,j^*} - b_{i^*,j^*}$$

Since $b_{i,j} \leq v_{i,j}$, for all i, j, we get

$$v_{i,j} - p_i \geq v_{i,j^*} - v_{i^*,j^*} \iff v_{i,j} + v_{i^*,j^*} - p_i \geq v_{i,j^*}$$

Summing up over all i we get

$$\sum_i \left(v_{i,j} + v_{i^*,j^*} - p_i\right) \geq \sum_i v_{i,j^*}$$

Notice that $\sum_i v_{i,j} = SW_{EQ}$, but also $\sum_i v_{i^*,j^*} = SW_{EQ}$, as we're summing up the valuation of the bidder who gets i's optimal item, at equilibrium, for all bidders i. Therefore, this sum is equal to the sum of all bidders' valuations at equilibrium. Also notice that $\sum_i v_{i,j^*} = SW_{OPT}$, and let P_{EQ} denote the sum of prices at equilibrium. Then, we have

$$SW_{EQ} + SW_{EQ} - P_{EQ} \geq SW_{OPT} \implies SW_{EQ} \geq \frac{1}{2} SW_{OPT}$$

and thus

$$PoA = \frac{SW_{OPT}}{SW_{EQ}} \leq 2$$

4. Consider n identical machines and m selfish jobs (the players). Each job j has a processing time p_j. Once jobs have chosen machines, the jobs on each machine are processed serially from shortest to longest. (You can assume that the p_j's are distinct.) For example, if jobs with processing times $1, 3$, and 5 are scheduled on a common machine, then they will complete at times $1, 4$, and 9, respectively. The following questions concern the game in which players choose machines in order to minimize their completion times. The objective function as a planner is to minimize the total completion time $\sum_{j=1}^{m} C_j$, where C_j is the completion time job j.

   (a) (4 points) Define the rank R_j of job j in a schedule as the number of jobs on j's machine with processing time at least p_j (including j itself). For example, if jobs with processing times $1, 3$, and 5 are scheduled on a common machine, then they have ranks $3, 2$, and 1, respectively.

   Prove that in these scheduling games, the objective function value of an outcome can also be written as $\sum_{j=1}^{m} p_j R_j$.

   (b) (4 points) Prove that the following algorithm produces an optimal outcome: (i) sort the jobs from largest to smallest; (ii) for $i = 1, 2, \ldots, m$, assign the i^{th} job in this ordering to machine i mod n (where machine 0 means machine n).

   (c) (7 points) Prove that for every such scheduling game, the expected objective function value of every coarse correlated equilibrium is at most twice that of an optimal outcome.

   Hint: *The (λ, μ)-smoothness condition (see notes provided) was required for all pairs $\boldsymbol{s}^*, \boldsymbol{s}$ of outcomes. Weaken the definition so that this condition only needs to hold for some optimal outcome $\boldsymbol{s}^*$ and all outcomes $\boldsymbol{s}$. Observe that PoA of coarse correlated equilibria remains at most $\frac{\lambda}{1-\mu}$ assuming only this weaker condition (with the same proof as before). Prove that this scheduling game satisfies this weaker condition for $\lambda = 2$ and $\mu = 0$.*

   (a) Fix some machine, and suppose the jobs being run on this machine have processing time $p_1 \geq p_2 \geq \cdots \geq p_\ell$. Since the cost of a job is its completion time, and it is being processed from shortest to longest, we have total cost

$$(p_\ell) + (p_\ell + p_{\ell-1}) + (p_\ell + p_{\ell-1} + p_{\ell-2}) + \cdots \quad = \ \ell p_\ell + (\ell - 1)p_{\ell-1} + \cdots + 2p_2 + p_1$$

   Since the rank of the job with length p_j is j, this gives us the desired sum for one fixed machine. Taking the sum over all machines gives us the desired bound.

   (b) Since there are n machines, there are at most n jobs with rank 1 on their machine, at most n jobs with rank 2, and so on. We wish to choose values of R_j for $j = 1, 2, \ldots, m$ to minimize $\sum_{j=1}^{m} R_j p_j$. (We will later worry about whether this is feasible.) It should be clear that the best thing to do is to give rank 1 to the n longest jobs, rank 2 to the next n longest jobs, etc. This allocation of ranks to jobs is exactly what is attained by the algorithm described in the statement.

   (c) Assume, as in part (a), that $p_1 \geq p_2 \geq \cdots \geq p_m$. We wish to compare $\sum_{j=1}^{m} C_j(s_j^*, \boldsymbol{s}_{-j})$ to $\sum_{j=1}^{m} C(\boldsymbol{s}^*)$. Fix a machine i, and we will restrict our attention to this machine. Let

$J_i := \{j : s_j = i\}$, and $J_i^* := \{j : s_j^* = i\}$. Then, $C_j(s_j^*, \boldsymbol{s}_{-j}) = p_j + \sum_{k \in J_i : k > j} p_k$. Thus,

$$\sum_{j \in J_i^*} C_j(s_j^*, \boldsymbol{s}_{-j}) = \sum_{j \in J_i^*} \left(p_j + \sum_{k > j \in J_i} p_k \right) = \left(\sum_{j \in J_i^*} p_j \right) + \left(\sum_{k \in J_i} p_k \cdot |\{j \in J_i^* : j < k\}| \right) \quad (1)$$

where the second inequality holds because each term p_k appears in the middle sum whenever some $j \in J_i^*$ has index less than k. We will refer to this value as $n_i(k) := |\{j \in J_i^* : j < k\}|$. This gives us

$$\sum_{j=1}^{m} C_j(s_j^*, \boldsymbol{s}_{-j}) \;=\; \sum_{i=1}^{n} \sum_{j \in J_i^*} C_j(s_j^*, \boldsymbol{s}_{-j}) \;=\; \sum_{j=1}^{m} p_j + \sum_{j=1}^{m} p_j \cdot n_{s_j}(j) \quad (2)$$

Consider the output of the algorithm in part (b), when each job is placed, the number of longer jobs in each machine is the same, possibly off by 1. Thus, for any $i \neq i'$, we must have $n_i(j) \leq n_{i'}(j) + 1$. Now, it should be easy to see from (1) that

$$\sum_{j \in J_i^*} C_j(\boldsymbol{s}^*) \;=\; \sum_{j \in J_i^*} p_j + \sum_{k \in J_i^*} p_k \cdot n_{s_j^*}(k) \;=\; \sum_{j \in J_i^*} (n_{s_j^*}(j) + 1) \cdot p_j$$

and so, with (2), and $n_{s_j}(j) \leq n_{s_j^*}(j) + 1$, we have

$$\sum_{j=1}^{m} C_j(s_j^*, \boldsymbol{s}_{-j}) \leq \left(\sum_{j=1}^{m} p_j \right) + \left(\sum_{j=1}^{m} C_j(\boldsymbol{s}^*) \right) \leq 2 \sum_{j=1}^{m} C_j(\boldsymbol{s}^*)$$

5. (a) (2 points) Give an example of a game with 2 players that admits a PNE, but the best-response dynamics cycles.

   (b) This problem studies a scenario with n agents, where agent i has a positive weight $w_i > 0$. There are m identical machines. Each agent chooses a machine, and wants to minimize the load of her machine, defined as the sum of the weights of the agents who choose it. A pure Nash equilibrium in this game is an assignment of agents to machines so that no agent can unilaterally switch machines and decrease the load she experiences. Consider the following restriction of best-response dynamics:

Algorithm 1: Maximum Weight Best-Response Dynamics

While the current outcome $\boldsymbol{s}$ is not a PNE:
  among all agents with a beneficial deviation, let i denote an agent with the largest weight w_i and s_i' a best response to $\boldsymbol{s}_{-i}$
  update the outcome to $(s_i', \boldsymbol{s}_{-i})$

   i. (3 points) Show that, starting from the outcome $\boldsymbol{s}_0$ where no agent has selected any machines (all machines have load 0), the Maximum Weight Best-Response Dynamics converges to a PNE in exactly n iterations.

   ii. (5 points) Show that, starting from any outcome $\boldsymbol{s}$, the Maximum Weight Best-Response Dynamics converges to a PNE in at most n iterations.

(a) Consider the following bimatrix game

	c_1	c_2	c_3
r_1	(5, 5)	(0, 0)	(0, 0)
r_2	(0, 0)	(1, 0)	(0, 1)
r_3	(0, 0)	(0, 1)	(1, 0)

Let R denote the row player and C denote the column player. It is not difficult to see that the upper-left entry $(5, 5)$ is a PNE of the game (it is both players' dominant strategy).

However, suppose the game's initial state is the middle entry $(1, 0)$. Notice that, while R has no incentive to deviate from r_2, C can gain by deviating to c_3. Therefore, after C plays their best-response strategy, we arrive at the middle-right entry $(0, 1)$. Now, however, R can gain by deviating from r_2 to r_3. Therefore, after R plays their best-response strategy, we arrive at the bottom-right entry $(1, 0)$. Once again, C can gain by deviating from c_3 to c_2. Therefore, after C plays their best-response strategy, we arrive at the bottom-middle entry $(0, 1)$. Now, however, R can gain by deviating from r_3 to r_2. Therefore, after R plays their best-response strategy, we arrive at the initial state, the middle entry $(1, 0)$, and the best-response dynamics cycles.

(b) i. Without loss of generality, suppose that the agents are sorted in non-decreasing order of their weights, i.e. $w_1 \geq w_2 \geq \cdots \geq w_n$. For the Maximum Weight Best-Response Dynamics to converge to a PNE in exactly n iterations, it has to be the case that after agent i selects a machine, no agent in $\{1, \ldots, i\}$ wants to deviate from their strategy and select a different machine.

To show this, we use induction on the number of iterations. For the base case, agent 1 selects any machine (without loss of generality, we can arbitrarily break ties). This is their best-response, as there are no agents besides agent 1 on any machine, and thus, trivially, no agent before 1 wants to deviate from their strategy.
Next, assume that we are at the i-th iteration of the Maximum Weight Best-Response Dynamics, right before agent i plays their best-response, and no agent in $\{1, \ldots, i-1\}$ wants to deviate from their strategy and select a different machine. For the induction step, we want to show that, after i selects a machine that is their best-response strategy, no agent in $\{1, \ldots, i\}$ will want to deviate from their strategy and select a different machine. Suppose i plays their best-response strategy and selects machine m_i. Clearly, i will not want to deviate from their strategy and select a different machine, as i just played their best-response strategy.
Assume, towards contradiction, that there exists an agent $j \in \{1, \ldots, i-1\}$ that is assigned to machine m_1 and that, after i plays their best-response strategy, wants to deviate from their strategy and select a different machine, say machine m_2. Let M_1, M_2 and M_i denote the set of agents who have selected machine m_1, m_2 and m_i, respectively. There are two possible cases:

- $m_2 = m_i$: In this case, j wants to switch to m_i because they experience a smaller load on that machine. This implies that

  $$\sum_{k \in M_1} w_k > \sum_{k \in M_i} w_k$$

  But we know that

  $$\sum_{k \in M_i} w_k = \sum_{k \neq i, k \in M_i} w_k + w_i$$

  and thus

  $$\sum_{k \in M_1} w_k > \sum_{k \neq i, k \in M_i} w_k$$

  where the right-hand side is the load on machine m_i before i played their best-response strategy. Therefore, j would want to deviate to m_i before i played their best-response strategy, contradicting our inductive hypothesis.
- $m_2 \neq m_i$: Again, j wants to switch to m_2 because they experience a smaller load on that machine. This implies that

  $$\sum_{k \in M_1} w_k > \sum_{k \in M_2} w_k$$

  But the right-hand side is the load on machine m_2 before i played their best-response strategy, as i did not select m_2. Therefore, j would want to deviate to m_2 before i played their best-response strategy, again contradicting our inductive hypothesis.

Therefore, after all n agents have played their best-response strategy, no agent will want to deviate from their strategy. Thus, the Maximum Weight Best-Response Dynamics converges to a PNE in exactly n iterations.

ii. We follow the reasoning above, being more careful now, as when an agent plays their best-response strategy, the machine they were previously assigned to has smaller load. Again, for the Maximum Weight Best-Response Dynamics to converge to a PNE in at most n iterations, it has to be the case that after agent i selects a machine, no agent in $\{1, \ldots, i\}$ wants to deviate from their strategy and select a different machine. We show this via induction on the number of iterations. For the base case, after agent 1 plays their best-response, clearly agent 1 does not want to deviate from their strategy, as they just played their best-response strategy. Therefore, the base case trivially holds.
Next, assume that we are at the i-th iteration of the Maximum Weight Best-Response Dynamics. Right before agent i plays their best-response, no agent in $\{1, \ldots, i-1\}$ wants to deviate from their strategy and select a different machine. For the induction step, we want to show that, after i selects a machine that is their best-response strategy, no agent in $\{1, \ldots, i\}$ will want to deviate from their strategy and select a different machine. Clearly, i will not want to deviate from their strategy and select a different machine, as i just played their best-response strategy. Also, if i's best-response strategy was to not select a different machine, no agent in $\{1, \ldots, i-1\}$ will want to deviate from their strategy and select a different machine, as there was no change in the i-th iteration. Therefore, assume that i's best-response strategy is to switch from machine m_i to machine m_i'.
Assume, towards contradiction, that there exists an agent $j \in \{1, \ldots, i-1\}$ that is assigned to machine m_1 and that, after i switches from m_i to m_i', wants to deviate from their strategy and select a different machine, say machine m_2. Again, let M_1, M_2, M_i and M_i' denote the set of agents who have selected machine m_1, m_2, m_i and m_i', respectively. There are two possible cases:

- $m_2 \neq m_i$: In this case, the load of m_2 did not decrease in the i-th iteration. Therefore, j would want to deviate to m_2 even before i played their best-response strategy. Using our induction hypothesis that j did not want to deviate from their strategy before i played their best-response strategy, we arrive at a contradiction.
- $m_2 = m_i$: In this case, j wants to switch to m_i because they experience a smaller load on that machine. This could potentially be that the load of m_i decreased after i switched machines, and now it could be that j's best-response strategy is to switch to m_i. We show that this could never happen. Since i's best-response was to deviate from m_i and select a different machine m_i', we have

$$w_i + \sum_{k \neq i, k \in M_i} w_k > w_i + \sum_{k \neq i, k \in M_i'} w_k \implies$$
$$\sum_{k \neq i, k \in M_i} w_k > \sum_{k \neq i, k \in M_i'} w_k \tag{3}$$

Also, from the fact that j wants to deviate from m_1 to m_i, we know that

$$w_j + \sum_{k \neq j, k \in M_1} w_k > w_j + \sum_{k \neq j, k \in M_i} w_k \implies$$
$$\sum_{k \neq j, k \in M_1} w_k > \sum_{k \neq j, k \in M_i} w_k \tag{4}$$

Combining (3) and (4), we get

$$\sum_{k \neq j, k \in M_1} w_k > \sum_{k \neq i, k \in M_i'} w_k$$

where the right-hand side is exactly the load of m_i' before i played their best-response strategy. Therefore, j would want to deviate from their strategy and select machine m_i' even before i played their best-response strategy, as then they would get a smaller load, contradicting our induction hypothesis.

Therefore, after all n agents have played their best-response strategy, no agent will want to deviate from their strategy. Thus, the Maximum Weight Best-Response Dynamics converges to a PNE in at most n iterations.