Problem 1. Computing shortest paths

Let G be an undirected graph with N nodes $\{s_1, \dots, s_N\}$. Suppose G is complete, i.e., every pair of nodes is connected by an edge. For $i \neq j$, let $a_{ij} \in \mathbb{R}_{>0}$ be the cost of the edge connecting s_i and s_j. For $i = j$, we set $a_{ij} = 0$. By interpreting the nodes as states and the edges as actions, we may directly apply the dynamic programming method seen in lecture to compute shortest paths in G.

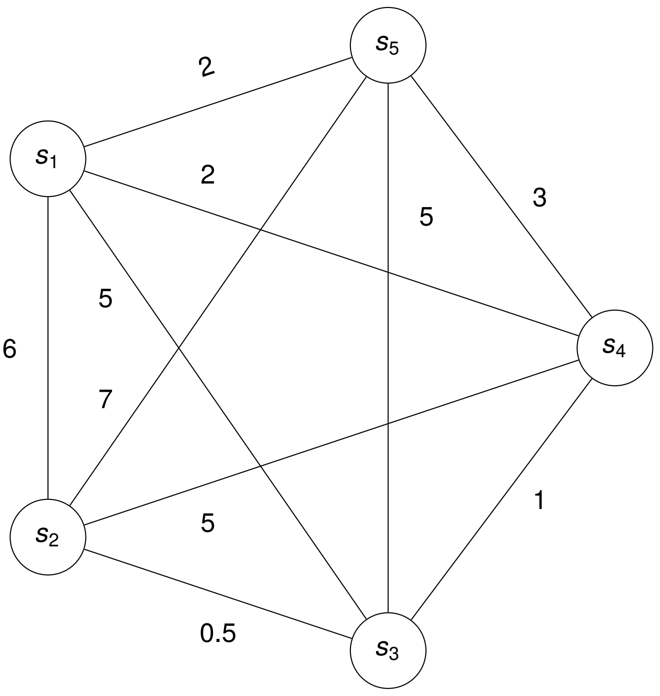

Figure 1: The graph G.

a) For the graph in Figure 1, use dynamic programming to compute the cost of the shortest path from each s_i, $i \in \{1, \dots, 4\}$, to s_5.

Hint: For each $k \in \{1, \dots, N\}$, and node s_i, let $V_k(s_i)$ be the optimal cost, i.e. shortest path, to s_N when starting from s_i with $N - k$ steps remaining. Clearly, $V_N(s_N) = 0$ and $V_N(s_j) = \infty$ for all $j \neq N$. Based on this, compute $V_{N-1}(s_i)$ for all s_i, then $V_{N-2}(s_i)$, and so on, until $V_1(s_i)$.

Solution: Observe that for all $k \in \{1, \dots, N-1\}$, and states s_i,

$$\begin{aligned} V_k(s_i) &= \min_{j \in \{1,\dots,N\}} \left\{ a_{ij} + \big(\text{optimal cost to go from } s_j \text{ to } s_N \text{ in } N-k-1 \text{ steps}\big) \right\} \\ &= \min_{j \in \{1,\dots,N\}} \left\{ a_{ij} + V_{k+1}(s_j) \right\}. \end{aligned}$$

As noted above, $V_5(s_5) = 0$ and $V_5(s_j) = \infty$ for all $j \neq 5$. Next, we compute

$$\begin{aligned} V_4(s_1) &= \min_{j \in \{1,\dots,5\}} \left\{ a_{1j} + V_5(s_j) \right\} = 2 + V_5(s_5) = 2, \\ V_4(s_2) &= \min_{j \in \{1,\dots,5\}} \left\{ a_{2j} + V_5(s_j) \right\} = 7 + V_5(s_5) = 7, \\ V_4(s_3) &= \min_{j \in \{1,\dots,5\}} \left\{ a_{3j} + V_5(s_j) \right\} = 5 + V_5(s_5) = 5, \\ V_4(s_4) &= \min_{j \in \{1,\dots,5\}} \left\{ a_{4j} + V_5(s_j) \right\} = 3 + V_5(s_5) = 3, \\ V_4(s_5) &= \min_{j \in \{1,\dots,5\}} \left\{ a_{4j} + V_5(s_j) \right\} = 0 + V_5(s_5) = 0. \end{aligned}$$

Based on this, the next step gives

$$\begin{aligned}
V_3(s_1) &= \min_{j\in\{1,\dots,5\}} \left\{a_{1j} + V_4(s_j)\right\} = 2 + V_4(s_5) = 2,\\
V_3(s_2) &= \min_{j\in\{1,\dots,5\}} \left\{a_{2j} + V_4(s_j)\right\} = 0.5 + V_4(s_3) = 5.5,\\
V_3(s_3) &= \min_{j\in\{1,\dots,5\}} \left\{a_{3j} + V_4(s_j)\right\} = 1 + V_4(s_4) = 4,\\
V_3(s_4) &= \min_{j\in\{1,\dots,5\}} \left\{a_{4j} + V_4(s_j)\right\} = 3 + V_4(s_5) = 3,\\
V_3(s_5) &= \min_{j\in\{1,\dots,5\}} \left\{a_{4j} + V_5(s_j)\right\} = 0 + V_4(s_5) = 0.
\end{aligned}$$

Iterating again, we get

$$\begin{aligned}
V_2(s_1) &= \min_{j\in\{1,\dots,5\}} \left\{a_{1j} + V_3(s_j)\right\} = 2 + V_3(s_5) = 2,\\
V_2(s_2) &= \min_{j\in\{1,\dots,5\}} \left\{a_{2j} + V_3(s_j)\right\} = .5 + V_3(s_3) = 4.5,\\
V_2(s_3) &= \min_{j\in\{1,\dots,5\}} \left\{a_{3j} + V_3(s_j)\right\} = 1 + V_3(s_4) = 4,\\
V_2(s_4) &= \min_{j\in\{1,\dots,5\}} \left\{a_{4j} + V_3(s_j)\right\} = 3 + V_3(s_5) = 3,\\
V_2(s_5) &= \min_{j\in\{1,\dots,5\}} \left\{a_{4j} + V_3(s_j)\right\} = 0 + V_3(s_5) = 3.
\end{aligned}$$

And, finally,

$$\begin{aligned}
V_1(s_1) &= \min_{j\in\{1,\dots,5\}} \left\{a_{1j} + V_2(s_j)\right\} = 2 + V_2(s_5) = 2,\\
V_1(s_2) &= \min_{j\in\{1,\dots,5\}} \left\{a_{2j} + V_2(s_j)\right\} = .5 + V_2(s_3) = 4.5,\\
V_1(s_3) &= \min_{j\in\{1,\dots,5\}} \left\{a_{3j} + V_2(s_j)\right\} = 1 + V_2(s_4) = 4,\\
V_1(s_4) &= \min_{j\in\{1,\dots,5\}} \left\{a_{4j} + V_2(s_j)\right\} = 3 + V_2(s_5) = 3,\\
V_1(s_5) &= \min_{j\in\{1,\dots,5\}} \left\{a_{4j} + V_2(s_j)\right\} = 0 + V_2(s_5) = 0.
\end{aligned}$$

Since edge weights are positive, a path with more than 4 steps cannot be optimal, as it would visit some node twice (meaning we can remove a cycle and make it shorter). Therefore the shortest path length from each node s_i to s_N is given by $V_1(s_i)$ as computed above.

b) Use the computed values to reconstruct the shortest paths (i.e. the sequence of nodes that gives optimal cost) from each s_i, $i \in \{1, \dots, 4\}$, to s_5.

Solution: As an example, we determine the shortest path from s_2 to s_5. We know $V_1(s_2) = 4.5$. To find the next vertex along a path that has total cost 4.5, we look for $i \in \{1, \dots, 5\}$ such that $a_{2i} + V_2(s_i) = V_1(s_2)$. This is satisfied by $i = 2$, meaning we first stay at s_2. Next, we again find $i \in \{1, \dots, 5\}$ such that $a_{2i} + V_3(s_i) = V_2(s_2)$, which is satisfied by $i = 3$. Then, we look for $i \in \{1, \dots, 5\}$ such that $a_{3i} + V_4(s_i) = V_3(s_3)$, such as $i = 4$. Finally, $a_{4i} + V_5(s_i) = V_4(s_4)$ is satisfied by $i = 5$. We can check that indeed, the path $s_2 \to s_3 \to s_4 \to s_5$ has the optimal cost of 4.5.

c) A naive way to compute shortest paths is to enumerate and compare all paths with at most $N - 1$ steps from a given node s_i, $i \in \{1, \dots, N-1\}$, to s_N. How many such paths[1] are there for each s_i in a complete graph with N vertices? How does the number of computations compare (asymptotically) to the dynamics programming method above?

Solution: We know that the first node in the path is s_i, and the last is s_N. This leaves $N - 2$ intermediate nodes, for each of which there are N possible choices. Hence we get N^{N-2} paths. Note that as the graph is complete any such sequence will be a valid path.

[1] To simplify counting, you may count e.g. $s_1s_1s_1s_5s_5$ and $s_1s_5s_5s_5s_5$ as distinct paths.

For the dynamics programming method, we had $N-1$ rounds of updates. In each round, we had to compute N variables $V_k(s_i)$, for $i \in \{1, \dots, N\}$. Each such computation involved taking a minimum over N options. Therefore, asymptotically, the number of computations is on the order of N^3.

This means, as N increases, the naive method quickly becomes too expensive, while the dynamic programming approach scales significantly better.

Problem 2. Escape game

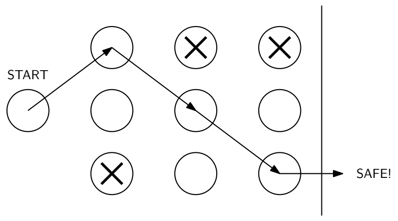

Figure 2: The escape game.

Player 1 (Alice) is trying to escape, going from the **start node** to the **safe zone** without being intercepted by Player 2 (Eve) who is trying to stop her. We model this as a dynamic game with stages $k = 1, \dots, K$ and states $\mathcal{S} = \{s_U, s_M, s_D, s_E\}$. States s_U,s_M,s_D represent Alice's current position (up, middle, or down), and state s_E is entered (and never left) once Eve has caught her.

In Figure 2, each row corresponds to a stage of the game. At each stage, Alice can decide to continue on the same row, or instead move diagonally one row up or one row down. Eve is aware of the current state, i.e. Alice's position, and she is allowed to block one of the three rows in the next stage, taking her decision simultaneously with Alice. If she selects the row corresponding to Alice's next move, the game enters state s_E which will result in cost 1 for Alice at the final stage. Otherwise, if Alice survives, i.e. the game is not in state s_E at the final stage, Alice gets cost -1. Eve's costs are the costs of Alice multiplied by -1.

The actions available to Alice depend on the current state s, namely,

$$\mathcal{U}_s = \begin{cases} \{U, M, D\}, & \text{if } s \in \{s_M, s_E\}; \\ \{U, M\}, & \text{if } s = s_U; \\ \{M, D\}, & \text{if } s = s_D. \end{cases}$$

For Eve, the action set is state-independent and given by $\mathcal{V} = \{U, M, D\}$.

a) Based on the game description, define its evolution map $f(s, u, v)$ for each s and $u \in \mathcal{U}_s$, $v \in \mathcal{V}$.

Solution: The evolution map should be defined as follows

$$f(s, u, v) = \begin{cases} s_U, & \text{if } s \neq s_E \text{ and } u = U \text{ and } v \neq U; \\ s_M, & \text{if } s \neq s_E \text{ and } u = M \text{ and } v \neq M; \\ s_D, & \text{if } s \neq s_E \text{ and } u = D \text{ and } v \neq D; \\ s_E, & \text{if } s = s_E \text{ or } u = v. \end{cases}$$

b) Additionally, we define a stage cost function $g_k(s, u, v)$ representing the cost for Alice, by setting, for any $u \in \mathcal{U}_s$, $v \in \mathcal{V}$,

$$g_k(s, u, v) = \begin{cases} 1, & \text{if } k = K \text{ and } s = s_E; \\ -1, & \text{if } k = K \text{ and } s \neq s_E; \\ 0, & \text{otherwise.} \end{cases}$$

Determine $V_K(s)$ for all $s \in \mathcal{S}$.

Solution: Following from the definition of g_k, we have

$$V_K(s) = \begin{cases} 1, & \text{if } s = s_E; \\ -1, & \text{else.} \end{cases}$$

c) Observe that for $k \in \{1, \dots, K-1\}$ and $s \in \mathcal{S}$, we have $V_k(s) = \min_{u \in \Delta(\mathcal{U}_s)} \max_{v \in \Delta(\mathcal{V})} V_{k+1}(f(s,u,v))$. Using dynamic programming, for all $s \in \mathcal{S}$, determine the values of $V_{K-1}(s)$, and then $V_{K-2}(s)$.

Hint: For each s $\in \mathcal{S}$, observe that any $u \in \mathcal{U}_s$, $v \in \mathcal{V}$ uniquely determine the next state through the evolution map. Hence, given s, you can write down the matrix representation of the respective normal-form game and determine the value of its Nash equilibrium, i.e., a saddle point of $\min_{u \in \Delta(\mathcal{U}_s)} \max_{v \in \Delta(\mathcal{V})} V_{k+1}(f(s,u,v))$.

Solution: We determine $V_{K-1}(s)$ for each state s:

- For $s = s_M$: We represent $V_K(f(s,u,v))$ as a matrix in which rows correspond to the min-player and columns to the max-player.

	U	M	D
U	1	−1	−1
M	−1	1	−1
D	−1	−1	1

The Nash equilibrium of this matrix game is $u^\star_{K-1}(s_M) = v^\star_{K-1}(s_M) = [\frac{1}{3}\ \frac{1}{3}\ \frac{1}{3}]$ and therefore $V_{K-1}(s_M) = -\frac{1}{3}$.

- For $s = s_U$: Similarly, We represent $V_K(f(s,u,v))$ as the matrix

	U	M	D
U	1	−1	−1
M	−1	1	−1

The Nash equilibrium of this matrix game is $u^\star_{K-1}(s_U) = [\frac{1}{2}\ \frac{1}{2}]$, $v^\star_{K-1}(s_U) = [\frac{1}{2}\ \frac{1}{2}\ 0]$, and therefore we have $V_{K-1}(s_U) = 0$.

- For $s = s_D$, the computations are symmetric to the case $s = s_U$ and one obtains $V_{K-1}(s_U) = 0$.
- For $s = s_E$, we get $V_{K-1}(s_E) = V_K(s_E) = 1$.

Next, we determine $V_{K-2}(s)$ for each state s based on the above computed results for V_{K-1}

- For $s = s_M$: We represent $V_{K-1}(f(s,u,v))$ as a matrix in which rows correspond to the min-player and columns to the max-player.

	U	M	D
U	1	0	0
M	$-1/3$	1	$-1/3$
D	0	0	1

The Nash equilibrium of this matrix game is $u^\star_{K-2}(s_M) = [\frac{4}{11}\ \frac{3}{11}\ \frac{4}{11}]$, $v^\star_{K-2}(s_M) = [\frac{3}{11}\ \frac{5}{11}\ \frac{3}{11}]$ and therefore $V_{K-2}(s_M) = \frac{3}{11}$.

- For $s = s_U$: Similarly, We represent $V_{K-1}(f(s,u,v))$ as the matrix

	U	M	D
U	1	0	0
M	$-1/3$	1	$-1/3$

One can find the following Nash equilibrium: $u^\star_{K-2}(s_U) = [1\ 0]$, $v^\star_{K-2}(s_U) = [0\ 1\ 0]$, and therefore we have $V_{K-2}(s_U) = 0$.

- For $s = s_D$, the computations are again symmetric to the case $s = s_U$ and one obtains $V_{K-2}(s_U) = 0$.

- For $s = s_E$, we get $V_{K-2}(s_E) = V_{K-1}(s_E) = 1$.

d) As $K \to \infty$, if the backward iteration converges, we would have $V(s) = \min_{u \in \Delta(\mathcal{U}_s)} \max_{v \in \Delta(\mathcal{V})} V(f(s,u,v))$, and we would obtain stationary policies $u^\star(s)$, $v^\star(s)$. Give a set of equations $V(s)$ would have to satisfy for such stationary policies.

Solution: The following is a necessary condition for $u^\star(s)$, $v^\star(s)$ to be optimal stationary policies: For all $s \in \mathcal{S}$,

$$V(s) = V(f(s, u^\star(s), v^\star(s))).$$