# Optimal route for super tankers - Bachelor Project

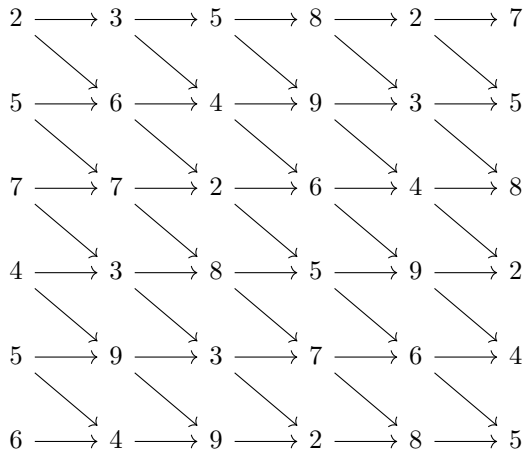
Arthur Dhonneur

Spring semester

## 1 Introduction to dynamic programming : Car commuting example

Dynamic programming stands as a powerful mathematical technique capable of solving complex problems by breaking them down into simpler subproblems. It is particularly effective in optimizing processes where decisions need to be made sequentially, and the outcome of current decisions impacts future choices.

Let's introduce our concept using the following example. Take a city, featuring nodes ( intersections ), and edges ( roads ). We have workers that have to commute, from their home to some parking lots. Homes are all located on the west of the city, and the parking lots are all located on the east of the city.
In the following diagram, we have a map connecting homes to parking lots. Values located on the nodes represent the time that we will spend in this particular intersection.




One intuitive way to solve this problem could be to list all the paths, and thus choose the shortest one. This would indeed be time consuming. We could proceed with a smarter approach.

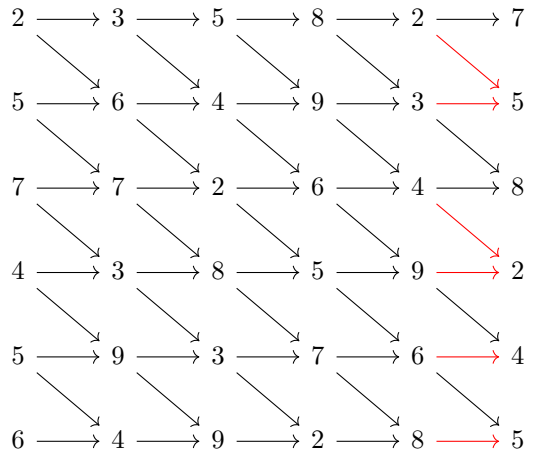
We see that our agent has to pass through 6 stages, before reaching his work place. Here we will note $n$ to be the number of stages there is to go. Let's call $s_n$ the node we are located at stage $n$, it goes from 1, in the bottom, to 6 on the top of the graph. We will call $t_n(s_n)$ the time spent in the intersection $s_n$ during the $n$ stage. We write $v_n(s_n)$ to be the value function depending on our node $s_n$

We will formulate our problem using what is called, *backward induction*

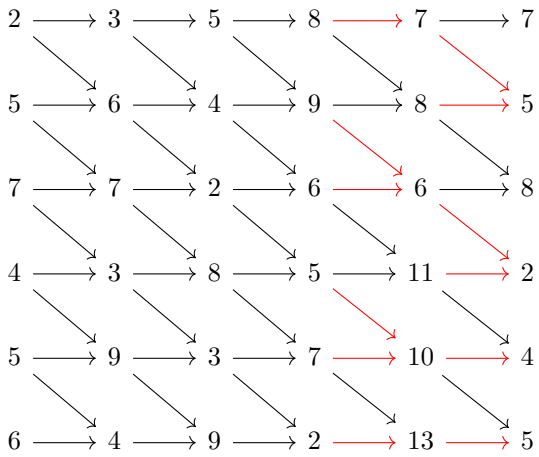
$$v_n(s_n) = \min_{s_{n-1}}\{t_n(s_n) + v_{n-1}(s_{n-1})\}$$

We want to select the best value for our value function $v_n$, given the information relative to the future stages, encoded in $v_{n-1}$. We start our induction at $n = 0$, meaning, when we have arrived to a parking lot. We then go backward, thus the term backward induction.

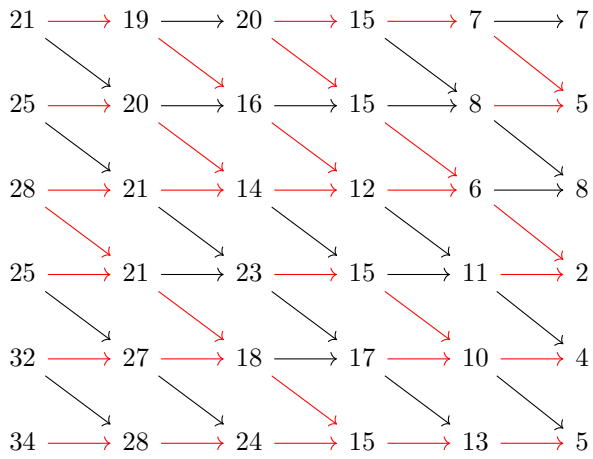
Let's now solve our problem using this particular strategy. First, let's look at the potential paths the we could be taking given that there is 1 stage to go. This gives us, for each node of the 1 stage to go column, a direction towards the 0 stage to go column. The arrows are here noted in red.




Now we will add the chosen 0 stage left nodes to the 1 stage left nodes. We operate the same red arrow process to the 2 stage left column. This is what we obtain :



We can than iterate this following process, and this would yield the following graph :



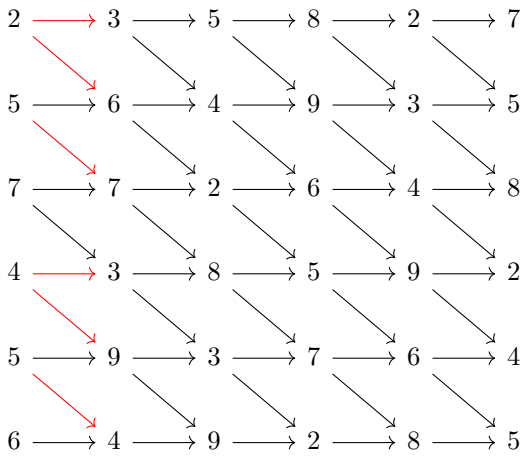

Using backward induction, we now have a map that gives us the optimal path, from any starting home to the parking lot final area.

We will now proceed with forward induction, a similar method, that will give us an optimal path from any parking lot to the home nodes.

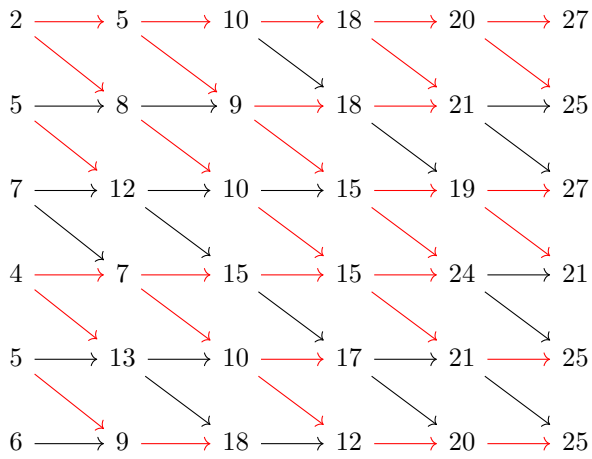
Now $n$ will represent the number of completed stages. On the left column we will have 0 stages completed, and on the right column we will have 6 stages completed. Let's note $u_n$ to be the value function, and $s_n$ to be the node where we are located after $n$ stages have been completed. The forward induction is directed by the following equation

$$u_{n-1}(s_{n-1}) = \min_{s_n}\{t_{n-1}(s_{n-1}) + u_n(s_n)\}$$

Once again we will draw in red the path that we should take given 0 stages completed, towards the 1 stages completed.




We will as for backward induction, add the cumulative time to the chosen next node, and observe the final result.




Here we gave a resolution process, for our optimal travel time problem, using what is called dynamic programming.

Dynamic programming can be formalized as a methodical approach to multi-stage decision-making processes. It is grounded in the concept of the Bellman equation, which provides a recursive decomposition of decision problems. This equation reflects the principle that an optimal policy has the property that, regardless of the initial state and initial decision, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision.

Mathematically, the Bellman equation can be expressed as:

$$V(x) = \max_{u \in U(x)} \{f(x,u) + \beta \sum_{y \in S} p(y|x,u)V(y)\}$$

$V(x)$ is the value function, representing the maximum value that can be obtained from state $x$.
$U(x)$ is the set of all possible actions from state $x$.
$f(x,u)$ is the reward function, quantifying the immediate reward of taking action $u$ in state $x$.
$\beta$ is the discount factor, representing the present value of future rewards.
$p(y|x,u)$ is the transition probability, denoting the likelihood of moving to state $y$ from state $x$ after taking action $u$.
$S$ is the set of all possible states.

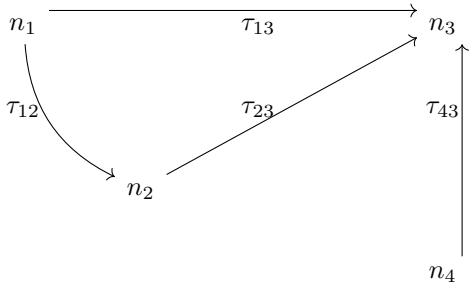
By modeling interactions and assigning probability distributions, dynamic programming accounts for uncertainty, enabling the tackling of a wide array of problems. It allows for the anticipation of future events and the formulation of strategies that optimize the expected outcome, even when the future is uncertain.

In conclusion, dynamic programming offers a robust framework for solving optimization problems that involve making a sequence of interdependent decisions. Its ability to incorporate uncertainty and to provide a clear strategy for decision-making under various scenarios makes it an indispensable tool in fields ranging from operations research to computer science and economics. As we delve deeper into the applications of dynamic programming, we will explore its efficacy in addressing real-world problems where optimal decision-making is crucial.

## 2 Dynamic Programming applied to supertankers : Finding an Optimal selling strategy

Let's now set our problem. We have a graph with $n$ nodes, each edges connected to the node $n$ is denoted by the set $\varepsilon_n$. Each node will be representing some harbours, and each edge a path between two harbours. Each edge will be given a weights, $\tau_{nm}$. These weights will represent the time to go from harbour $n$ to harbour $m$. The times $\tau_{nm}$ will follow an exponential distribution of parameter $\lambda_{nm}$. We will set the discount factor to $r$.

Here is an example with 4 nodes.




We will denote $v_n$ to be the maximum payout we can get, having our supertanker full, at the node $n$.

First of all, we assume that the cost is determined in advance, given by the quantity named $c_{nm}$, cost of going from harbour $n$ to harbour $m$.

Now let's look at the quantity $\beta_{nm}$, that represents the discount factor of our potential gain after moving from harbour $n$ to harbour $m$.

$$\begin{aligned}\beta_{nm} &= \mathbb{E}(e^{-\tau_{nm}r}) \\ &= \int_0^\infty e^{-rx}\lambda_{nm}e^{-\lambda_{nm}x}dx \\ &= \lambda_{nm}\int_0^\infty e^{-(r+\lambda_{nm})x}dx \\ &= \frac{\lambda_{nm}}{\lambda_{nm}+r}\end{aligned}$$

We see that our problem is governed by the dynamic programmation equation, and thus we get the following equation for our quantity $v_n$ :

$$v_n = \max\{p_n, \max_{k\in\varepsilon_n}\{\beta_{nk}(v_k - c_{nk})\}\} \tag{1}$$

We will now show for equation (1), the existence and the unicity of its solution.

We will write the vector $v = \begin{pmatrix} v_1 \cdots v_n \end{pmatrix}^T$, to be the vector in equation (1). We will use a sequence of functions, denoted by $f_1, \cdots, f_n$. Given a $m$, we have that :

$$f_m(x) = \left(\mathbb{1}_{k=n}p_n + \mathbb{1}_{k\in\varepsilon_n}(\beta_{nk}(x_k - c_{nk})\right)_{k=1,\cdots,n}$$

We see that $f_m(x)$ is a function from $\mathbb{R}^n$ to $\mathbb{R}^n$. Using equation (1), we observe that, taknig the max over all the components of the vector $f_m(v)$, we get :

$$v_m = \max f_m(v)$$

Now let's call $F$ the following operator :

$$F(x) = \begin{pmatrix} \max f_1(x) \\ \vdots \\ \max f_n(x) \end{pmatrix} \in \mathbb{R}^n$$

We have then transformed equation (1) into the following fixed point equation :

$$v = F(v) \tag{2}$$

We now want to use Banach's fixed point theorem in order to show the existence and unicity of the fixed point of operator $F$. We thus need to show that the map $F$ is contracting, with a $q < 1$. For this we will work with the infinity norm.

$$\|F(x) - F(y)\|_\infty = \max_{i \in \{1,\dots,N\}} |\max f_i(x) - \max f_i(y)| \tag{3}$$

$$\overset{\exists m}{=} |\max f_m(x) - \max f_m(y)| \tag{4}$$

Now, we will prove a short lemma that will be quite handy for our proof :

**Lemma 1.** *For $x, y \in \mathbb{R}^n$, we have $|\max(x) - \max(y)| \leqslant |\max(x - y)|$.*

*Proof.* First of all we will note $\max(x) = x_i, \max(y) = y_k$. Without loss of generality we can write:

$$|\max(x) - \max(y)| = |x_i - y_k| = x_i - y_k$$

Then we finish the proof with :

$$\begin{aligned} |\max(x) - \max(y)| &= x_i - y_k \\ &\leq x_i - y_i \\ &\leq \max(x - y) \\ &\leq |\max(x - y)| \end{aligned}$$

□

Using this lemma, and applying it to the equation (4), this yields :

$$\|F(x) - F(y)\|_\infty \leqslant |\max\{f_n(x) - f_n(y)\}|$$

And finally $\exists k \in \{1, \dots, N\}$, such that when evaluated to the vector $f_n(x) - f_n(y)$ it yields

$$\begin{aligned} |\max f_n(x) - f_n(y)| &= |\mathbb{1}_{\{k \in \varepsilon_n\}}(\beta_{nk} x_k - c_{nm}\tau_{nm} - \beta_{nk} y_k + c_{nm}\tau_{nm}) + \mathbb{1}_{\{k=n\}}(p_n - p_n)| \\ &= |\mathbb{1}_{\{k \in \varepsilon_n\}} \beta_{nk}(x_k - y_k)| \\ &\leqslant \beta_{nk} |x_k - y_k| \\ &\leqslant \beta \|x - y\|_\infty \end{aligned}$$

Where $\beta$ is the sup of the $\beta_{nm}$ taken over all $n, m : \beta < 1$, as there is finite number of $n, m$ and that each $\beta_{nm} < 1$.

We just proved that $F$ is a contraction, and thus the fixed point equation has a unique solution. Thus equation (1) has a unique solution, so we are now sure that there is an optimal selling strategy.

# 3 Pushing forward : Optimal mixed strategy

## 3.1 Introduction and preleminary results

Now, we will add complexity to our initial problem. We will now have the possibility to sell and to buy our commodities, and we still want to find the optimal strategy that would maximize our payoff. We will denote the quantity we hold in our boat ( full or empty ) by the binary variable $q$. We are still working on a graph with $n$ nodes, with edges given by $\tau_{nm}$, random variables that are distributed as exponential $\lambda_{nm}$ random variables. The cost between harbour $n$ and $m$ is given by the variable $c_{nm}$, the risk free asset yield is given by $r$. We will use $q$, a binary variable that will indicate wether the boat is full or empty. We also add the possibilty to stop trading, to liquidate our boat and to cash in a payoff given by $\sigma_n^q$, depending on the node and the quantity we have in the boat. We first have the following equality.

$$\sigma_n^q = \sigma_n^0 + qs_n$$

In the spirit of the dynamic programmation equation, now using the variable $q$, we would have the following equation

$$v(n,q) = \max\{\sigma_q^n; v(n,1-q) + s_n q - (1-q)b_n; \max_{m\in\varepsilon_n}\{\beta_{nm}^q(v(m,q) - c_{nm}^q\}\}$$

Let us now bring up a new variable. We will note $w(n,q)$ the optimal value given that we just moved. It is clear that now we can rewrite $v(n,q)$ using $w(n,q)$ :

$$v(n,q) = \max\{\sigma_n^q, v(n,1-q) + s_n q + b_n(q-1), w(n,q)\} \tag{5}$$

Now $w(n,q)$, is actually given by the optimal payout out of all the trading strategies given that we start at node $n$ with a quantity of $q$. We define a strategy starting at $n$ with $q$ by :

$$(N,\mathbb{S}) = \begin{cases} 1 \le N \le +\infty, \text{ number of steps before shut down} \\ \sigma_n^q = \sigma_n^0 + qs_n \\ \mathbb{S} = (n_i, q_i)_{i=0}^N \\ (n,q) = (n_0, q_0) \end{cases}$$

We denote by $S(n,q)$ the set of all strategies starting at $n$ with $q$. We now will define $w(n,q)$ to be the optimal strategy that maximizes our payoff. Formally we have

$$\begin{aligned} w(n,q) &= \max_{(N,\mathbb{S})\in S(n,q)} \Pi(N,\mathbb{S}) \\ &= \max_{(N,\mathbb{S})\in S(n,q)} \sum_{i=1}^{N} \beta_i(\mathbb{S})\{\mathbb{1}_{N>i}(s_n\Delta q_i^- - b_n\Delta q_i^+) - c_{n_{i-1}n_i}^{q_{i-1}} + \mathbb{1}_{N=i}\sigma_{n_i}^{q_{i-1}}\} \end{aligned}$$

Given that

$$\beta_i(\mathbb{S}) = \mathbb{E}(e^{-r\sum_{j=1}^i \tau_{n_{j-1}n_j}^{q_{j-1}}}) = \prod_{j=1}^{i} \frac{\lambda_{n_{j-1}n_j}^{q_{j-1}}}{r + \lambda_{n_{j-1}n_j}^{q_{j-1}}}$$

As previously explained, we want $w(n,q)$ to follow the dynamic intuitive equation :

$$\begin{aligned} w(n,q) &= \max_{m\in\varepsilon_n} \beta_{nm}^q\{-c_{nm}^q + \max\{\sigma_m^p; w(n,1-q) + qs_n - (1-q)b_n; w(m,q)\}\} \\ &=: \max_{m\in\varepsilon_n}(T_m w)(n,q) \\ &=: Tw(n,q) \end{aligned}$$

Let's write $S^*(n,q)$ to be the set of strategies where we don't liquidate nor trade at nodes where the values is $-\infty$. It is obvious that we can find a better strategy than the ones present in the complementary of $S^*(n,q)$. We can for example move to the nearest node where liquidation is not $-\infty$, and then liquidate.

Thus we see that

$$w(n,q) = \max_{(N,\mathbb{S})\in S^*(n,q)} \Pi(N,\mathbb{S})$$

We will now study the properties of $T$, we will show using the Blackwell condition that this operators has indeed unique fixed point. We recall the Blackwell condition :

$$\text{For } T: L^\infty \longrightarrow L^\infty, \quad \begin{cases} w_1 \le w_2 \implies Tw_1 \le Tw_2 \\ \text{for } c\in\mathbb{R}, \text{if } \exists \beta<1 : T(w+c) \le T(w) + c\beta \end{cases} \implies \exists! w^*, \quad T(w^*) = w^*$$

We will now show that $w \in L^\infty$. We can first give a bound to $\beta_i(\mathbb{S})$ :

$$\beta_i(\mathbb{S}) = \prod_{j=1}^{i} \frac{\lambda_{n_{j-1}n_j}^{q_{j-1}}}{r+\lambda_{n_{j-1}n_j}^{q_{j-1}}} \le \prod_{j=1}^{i} \frac{\lambda^*}{r+\lambda^*} = (\frac{\lambda^*}{r+\lambda^*})^i =: \Lambda^i, \quad \lambda^* = \sup_{n,m,q} \lambda_{nm}^q < 1$$

The first equality comes from the fact that $x \mapsto \frac{x}{r+x}$ is a stricly increasing function on $\mathbb{R}_+$. We can now bound the sum that defines $w(n,p)$. We will only look at the case $N=\infty$, because if not $w(n,p)$ is finite and thus bounded :

$$|w(n,p)| \le \max_{(N,\mathbb{S})\in S(n,q)} \sum_{i=1}^{N} \Lambda^i(|s_n\Delta q_i^- - b_n\Delta q_i^+ - c_{n_{i-1}n_i}^{q_{i-1}}|) \le \max_{(N,\mathbb{S})\in S(n,q)} \sum_{i=1}^{N} \Lambda^i(|s_n\Delta q_i^-| + |b_n\Delta q_i^+| + |c_{n_{i-1}n_i}^{q_{i-1}}|)$$

We can exclude the cases where $s_n$ or $b_n$ are negatie infinity, as it would not yield to any optimal strategy. Thus we can take the supremum over $s_{n_i}, b_{n_i}, c_{n_{i-1}n_i}^{q_{i-1}}$ and this will yield :

$$|w(n,p)| \le C^* \max_{(N,\mathbb{S})\in S(n,q)} \sum_{i=1}^{N} \Lambda^i < +\infty$$

Now let's prove that the operator $T$ satisfies the Blackwell conditions. We will start by the monoticity criterion. Assume $w_1 \le w_2$ :

$$\begin{aligned} T(w_1(n,q)) &= \max_{m\in\varepsilon_n} \beta_{nm}^q \{-c_{nm}^q + \max\{\sigma_m^p; w_1(n,1-q) + qs_n - (1-q)b_n; w_1(m,q)\}\} \\ &\le \max_{m\in\varepsilon_n} \beta_{nm}^q \{-c_{nm}^q + \max\{\sigma_m^p; w_2(n,1-q) + qs_n - (1-q)b_n; w_2(m,q)\}\} \\ &\le T(w_2(n,q)) \end{aligned}$$

Now let's show the second part of the criterion. Let's take $c\in\mathbb{R}$. This yields :

$$\begin{aligned} T(w(n,q)+c) &= \max_{m\in\varepsilon_n} \beta_{nm}^q \{-c_{nm}^q + \max\{\sigma_m^p; w(n,1-q) + c + qs_n - (1-q)b_n; w(m,q)+c\}\} \\ &\le \max_{m\in\varepsilon_n} \beta_{nm}^q \{-c_{nm}^q + c + \max\{\sigma_m^p; w(n,1-q) + qs_n - (1-q)b_n; w(m,q)\}\} \\ &\le \sup_{nm} \beta_{nm}^q c + \max_{m\in\varepsilon_n} \beta_{nm}^q \{-c_{nm}^q + \max\{\sigma_m^p; w(n,1-q) + qs_n - (1-q)b_n; w(m,q)\}\} \\ &\le \beta^* c + Tw(n,q) \end{aligned}$$

Thus $T$ satisfies the Blackwell conditions and we have the existence of a unique fixed point that we will name $w^*$. Now the point of this section is to check wether $w = w^*$.
We will prove the double equality :

$$w^*(n,q) \le w(n,q), \qquad w(n,q) \le w^*(n,q)$$

Let's prove first the second equality. We will use the fact that, given a strategy $(N, \mathbb{S}) \in S(n,q)$ the strategy after one step, that we will call $(N-1, \hat{\mathbb{S}}) \in S(n_1, q_1)$

$$\begin{aligned}
w(n,q) &= \sup_{(N,\mathbb{S})\in S(n,q)} \sum_{i=1}^{N} \beta_i(\mathbb{S})\{I(N>i)(s_n\Delta q_i^- - b_n \Delta q_i^+) - c_{n_{i-1}n_i}^{q_{i-1}} + I(N=i)\sigma_{n_i}^{q_{i-1}}\} \\
&= \sup_{(N,\mathbb{S})\in S(n,q)} \Big( \quad \beta_{nn_1}^{q}\{I(N>1)(s_{n_1}\Delta q_1^- - b_{n_1}\Delta q_1^+) - c_{n_0 n_1}^{q_0} + I(N=1)\sigma_{n_1}^{q_0} + I(N>1)\Pi(N-1,\hat{\mathbb{S}})\} \quad \Big) \\
&\leq \sup_{(N,\mathbb{S})\in S(n,q)} \Big( \quad \beta_{nn_1}^{q}\{I(N>1)(s_{n_1}\Delta q_1^- - b_{n_1}\Delta q_1^+ + w(n_1,q_1)) - c_{n_0 n_1}^{q_0} + I(N=1)\sigma_{n_1}^{q_0}\} \quad \Big) \\
&\leq Tw(n,q) \\
&\leq T^2 w(n,q) \\
&\quad \vdots \\
&\leq T^\infty w(n,q) = w^*(n,q)
\end{aligned}$$

Now let's go ahead and prove the second inequality given by :

$$w^*(n,q) \leqslant w(n,q)$$

Let us consider the feasible strategy $(N^*, \mathbb{S}^*) \in S(n,q)$ given by the following :

$$\begin{aligned}
(n_0, q_0) &= (n,q) \\
n_i &= \underset{m\in\varepsilon_{n_{i-1}}}{\arg\max} (T_m w^*)(n_{i-1}, q_{i-1}) \\
q_i &= \begin{cases} 1-q_{i-1} & \text{if } (T_{n_i} w^*)(n_{i-1},q_{i-1}) = w^*(n_i, 1-q_{i-1}) + s_{n_i} q_{i-1} + b_{n_i}(q_{i-1}-1) - c_{n_{i-1}n_i}^{q_{i-1}} \\ q_{i-1} & \text{otherwise} \end{cases}
\end{aligned}$$

and

$$N^* = \inf\{i \geq 1 : w^*(n_{i-1}, q_{i-1}) = \beta_{n_{i-1}n_i}^{q_{i-1}}(\sigma_{n_i}^{q_{i-1}} - c_{n_{i-1}n_i}^{q_{i-1}})\}$$

Using this strategy, and the fact that $w^*$ is a fixed point of $T$ implies that :

$$w^*(n_{i-1}, q_{i-1}) = \beta_{n_{i-1}n_i}^{q_{i-1}} \left( \mathbb{1}_{\{N^*=i\}} \sigma_{n_i}^{q_{i-1}} - c_{n_{i-1}n_i}^{q_{i-1}} + \mathbb{1}_{\{N^*>i\}}(s_{n_i}\Delta q_i^- - b_{n_i}\Delta q_i^+ + w^*(n_i, q_i)) \right)$$

Now we can iterate this process through all the $i \leq N^*$, starting from $i = 1$, and we will get that :

$$w^*(n,q) = w^*(n_0, q_0) = \Pi(N^*, \mathbb{S}^*) \leq w(n,q)$$

This inequality is true as $w(n,q)$ is defined as the maximum of the payouts, thus it is larger than $w^*(n,q)$ equal to the payout of a certain strategy.

$$w(n,q) = w^*(n,q)$$

### 3.2 Final result

We now want to show that the dynamic programation equation has a solution and that this solution is unique

$$v(n,q) = \max_{m\in\varepsilon_n}\{\sigma_n^q, v(n,1-q) + s_n q + b_n(q-1), \beta_{nm}^q(v(m,q) - c_{nm}^q)\} \tag{6}$$

We will show that this unique solution is given by

$$v^*(n,q) = \max\{\sigma_n^q, w^*(n,1-q) + s_n q + b_n(q-1), w^*(n,q)\} \tag{7}$$

Let's prove it now. By definition of $T$, we have that :

$$w^*(n,q) = \max_{m\in\varepsilon_n} \beta_{nm}^q(v^*(m,q) - c_{nm}^q)$$

On the other hand, definition of $v^*$ implies that

$$v^*(n,1-q) \geq w^*(n,1-q)$$

Which yields to

$$\begin{aligned} v^*(n,q) &= \max\{\sigma_n^q, w^*(n,1-q) + s_n q + b_n(q-1), w^*(n,q)\} \\ &\leq \max\{\sigma_n^q, v^*(n,1-q) + s_n q + b_n(q-1), w^*(n,q)\} \end{aligned}$$

Let's suppose by contradiciton that the inequality above is strict for a given pair $(n,q)$

$$v^*(n,q) < \max\{\sigma_n^q, v^*(n,1-q) + s_n q + b_n(q-1), w^*(n,q)\}$$

Since we have that

$$v^*(n,q) \geq \max\{\sigma_n^q, w^*(n,q)\}$$

We have that

$$v^*(n,1-q) + s_n q + b_n(q-1) > \max\{\sigma_n^q, w^*(n,q)\}$$

And this finally gives, by putting everything to the RHS of the inequatio, and using that $b_n - s_n > 0$

$$\begin{aligned} v^*(n,1-q) &> \max\{\sigma_n^{1-q} + (b_n - s_n)(1-q), w^*(n,q) + s_n(1-q) - b_n q + b_n - s_n\} \\ &> \max\{\sigma_n^{1-q}, w^*(n,q) + s_n(1-q) - b_n q\} \end{aligned}$$

This now shows that $v^*(n,1-q) = w^*(n,1-q)$, getting us a contradiction with our assumption.

$$v^*(n,q) < \max\{\sigma_n^q, w^*(n,1-q) + s_n q + b_n(q-1), w^*(n,q)\} = v^*(n,q)$$
$$v^*(n,q) < v^*(n,q)$$

Thus we have the following equality

$$v^*(n,q) = \max\{\sigma_n^q, v^*(n,1-q) + s_n q + b_n(q-1), w^*(n,q)\}$$

This shows that $v^*(n,q)$ is a solution to our dynamic programtion equation ( remember rewriting in equation (5) )

Now let's show the uniqueness of our solution. Let's assume another solution to the goal equation, and call it $\hat{v}$. Let's consider the following function

$$\hat{w}(n,q) = \max_{m \in \varepsilon_n} \{\beta_{nm}^q \left(\hat{v}(m,q) - c_{nm}^q\right)\}$$

Rewriting the dynamic programation equation yields to

$$\begin{aligned}\hat{v}(m,1-q) &= \max\{\sigma_m^{1-q}, \hat{v}(m,q) + (1-q)s_m - qb_m, \hat{w}(m,1-q)\} \\ &\geq \hat{w}(m,1-q)\end{aligned}$$

and, therefore,

$$\begin{aligned}\hat{w}(n,q) &= \max_{m \in \varepsilon_n} \{\beta_{nm}^q \left(\max\{\sigma_m^{1-q}, \hat{v}(m,q) + qs_m + (q-1)b_m, \hat{w}(m,1-q)\} - c_{nm}^q\right)\} \\ &\geq \max_{m \in \varepsilon_n} \{\beta_{nm}^q \left(\max\{\sigma_m^{1-q}, \hat{w}(m,q) + qs_m + (q-1)b_m, \hat{w}(m,1-q)\} - c_{nm}^q\right)\} \\ &= (T\hat{w})(n,q)\end{aligned}$$

We then get that, by iteration and using properties of $T$ as in the previous parts

$$\hat{w}(n,q) \geq w^*(n,q)$$

Using the similar arguments, and showing that indeed $\hat{w}(n,q)$ is indeed a payoff strategy, constructed below and that we will call $(\hat{N}, \hat{\mathbb{S}})$

$$\begin{aligned}(n_0, q_0) &= (n,q) \\ n_i &= \arg\max_{m \in \varepsilon_{n_{i-1}}} (T_m w^*)(n_{i-1}, q_{i-1}) \\ q_i &= \begin{cases} 1 - q_{i-1} & \text{if } (T_{n_i} w^*)(n_{i-1}, q_{i-1}) = w^*(n_i, 1-q_{i-1}) + s_{n_i} q_{i-1} + b_{n_i}(q_{i-1} - 1) - c_{n_{i-1}n_i}^{q_{i-1}} \\ q_{i-1} & \text{otherwise} \end{cases}\end{aligned}$$

And

$$N^* = \inf\{i \geq 1 : w^*(n_{i-1}, q_{i-1}) = \beta_{n_{i-1}n_i}^{q_{i-1}} (\sigma_{n_i}^{q_{i-1}} - c_{n_{i-1}n_i}^{q_{i-1}})\}$$

Using the fact that $w^*$ is the maximum payoff strategy, we have

$$\hat{w}(n,q) \leq w^*(n,q)$$

Thus we have that

$$\hat{w}(n,q) = w^*(n,q)$$

Now we can use it to show the following

$$\begin{aligned}\hat{v}(n,q) &= \max\{\sigma_n^q, \hat{v}(n,1-q) + s_n q + b_n(q-1), w^*(n,q)\} \\ &\geq \max\{\sigma_n^q, \hat{w}(n,1-q) + s_n q + b_n(q-1), w^*(n,q)\} \\ &\geq \max\{\sigma_n^q, w^*(n,1-q) + s_n q + b_n(q-1), w^*(n,q)\} \\ &= v^*(n,q)\end{aligned}$$

Let's finish the proof by assuming by conntradiction that we have a pair $(n,q)$ satisfying

$$\hat{v}(n,q) > v^*(n,q)$$

Using the fact that

$$v^*(n,q) \geq \max\{\sigma_n^q, w^*(n,q)\}$$

It shows that

$$\hat{v}(n,q) = \hat{v}(n,1-q) + qs_n + (q-1)b_n > \max\{\sigma_n^q, w^*(n,q)\}$$

And it yields that

$$\hat{v}(n,1-q) > \max\{\sigma_n^{1-q}, w^*(n,q) + (1-q)s_n - qb_n\}$$

This then implies that

$$\hat{v}(n,1-q) = w^*(n,1-q) = \hat{w}(n,1-q)$$

and substituting this we get our final contradiction that will prove the uniqueness of the solution

$$\begin{aligned}
\hat{v}(n,q) &= \max\{\sigma_n^q, \hat{v}(n,1-q) + s_n q + b_n(q-1_)), w^*(n,q)\} \\
&= \max\{\sigma_n^q, w^*(n,1-q) + s_n q + b_n(q-1_)), w^*(n,q)\} \\
&= v^*(n,q)
\end{aligned}$$