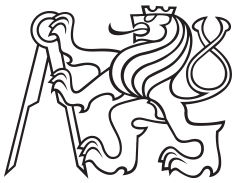
Bachelor thesis



Czech
Technical
University
in Prague

F3

Faculty of Electrical Engineering
Department of computer science

Convergence of Double Oracle algorithm

Vojtěch Outrata

Supervisor: Ing. Rostislav Horčík, Ph.D
Field of study: Game theory
May 2020

BACHELOR'S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Outrata Vojtěch** Personal ID number: **474721**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Control Engineering**

Study program: **Cybernetics and Robotics**

II. Bachelor's thesis details

Bachelor's thesis title in English:

Convergence of Double Oracle Algorithm

Bachelor's thesis title in Czech:

Konvergence Double Oracle algoritmu

Guidelines:

1) Familiarize yourself with game theory, namely with notions of normal-form game, zero-sum game, and Nash equilibrium [3, 4].
2) Get acquainted with Double Oracle Algorithm [1] allowing to find a Nash equilibrium in zero-sum games with large sets of strategies. Study the proof of its convergence. More detailed information on this algorithm can be found in [2].
3) Implement Double Oracle Algorithm on the domain from [1] or a similar one.
4) The proof of convergence is based on the ability to compute a best response on opponent's fixed strategy. However, several applications of this algorithm compute only an approximation of the best response. Analyze experimentally/theoretically how it influences the convergence of Double Oracle Algorithm.

Bibliography / sources:

[1] H.B. McMahan, G. Gordon, and A. Blum. Planning in the presence of cost functions controlled by an adversary. In Proceedings of the Twentieth International Conference on Machine Learning, 2003.
[2] Hugh Brendan Mcmahan. Robust Planning in Domains with Stochastic Outcomes, Adversaries, and Partial Observability. Ph.D. Dissertation. Carnegie Mellon Univ., Pittsburgh, PA, USA, 2006.
[3] Yoav Shoham and Kevin Leyton-Brown. Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations. Cambridge University Press, New York, NY, USA, 2008.
[4] Jiří Matoušek, Bernd Gärtner. Understanding and Using Linear Programming, 2007

Name and workplace of bachelor's thesis supervisor:

Ing. Rostislav Horčík, Ph.D., Department of Computer Science, FEE

Name and workplace of second bachelor's thesis supervisor or consultant:

Date of bachelor's thesis assignment: _____ Deadline for bachelor thesis submission: **22.05.2020**

Assignment valid until:
by the end of summer semester 2020/2021

Ing. Rostislav Horčík, Ph.D.	prof. Ing. Michael Šebek, DrSc.	prof. Mgr. Petr Páta, Ph.D.
Supervisor's signature	Head of department's signature	Dean's signature

III. Assignment receipt

The student acknowledges that the bachelor's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the bachelor's thesis, the author must state the names of consultants and include a list of references.

_____.	_____
Date of assignment receipt	Student's signature

Acknowledgements

I would like to thank my supervisor Ing. Rostislav Horčík, Ph.D for his most welcomed advices, willingness, patience and guidance during our consultations and also to my friends and family who supported me throughout my studies.

Declaration

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

Prague, 5. May 2020

Abstract

My thesis is focused on part of game theory called the zero-sum games, especially on the Double Oracle algorithm developed for these games. The Double Oracle algorithm tackles the problem of finding a Nash equilibrium in large zero-sum games of two players. This algorithm capitalizes on oracles producing a best response for each player. It is frequently used with suboptimal oracles and in my thesis I analyze the impact of the suboptimal oracles on the result of this algorithm. I implemented this algorithm on a zero-sum game of two players and furthermore I present theorems, proofs and carried out experiments.

Keywords: Game theory, Nash equilibrium, Double Oracle algorithm, Linear programming, Zero-sum games

Supervisor: Ing. Rostislav Horčík, Ph.D

Abstrakt

Má bakalářská práce je soustředěna na specifickou sekci teorie her zvanou hry s nulovým součtem. Zaměřuji se na Double Oracle algoritmus vyvinutý pro hledání Nashova ekvilibria ve hrách s nulovým součtem, kde jeden nebo oba hráči mají velký počet možných akcí. Tento algoritmus využívá možnost získat nejlepší možnou odpověď pro každého hráče ale je často používán s dobrou, ale né tou nejlepší odpovědí. V této práci analyzuji důsledek takového použití Double Oracle algoritmu. Algoritmus jsem naimplementoval na hře dvou hráčů s nulovým součtem a prezentuji věty, důkazy a experimenty popisující chování algoritmu za dané situace.

Klíčová slova: Teorie her, Linearní programování, Hry s nulovým součtem, Double Oracle algoritmus, Nashovo ekvilibrium

Překlad názvu: Konvergence Double Oracle algoritmu

Contents

Figures

Tables

Chapter 1

Introduction

Game theory is a mathematical discipline focusing on interaction of rational agents in a set environment. I focus on a part of game theory describing agents competing against each other. These types of games where two agents compete against one another are called zero-sum games in a normal-form. In such games, there always exists a strategy profile ensuring that each player has the best possible strategy against strategies of other players. Such a strategy profile is called Nash equilibrium of a game. For the zero-sum games, there is a method, described later in the thesis, that is able to find the Nash equilibrium given that a game matrix is provided. However, if this matrix is simply too large, or if one or more players have a finite but vast amount of possible strategies, then this method fails to compute the Nash-equilibrium due to a long computing time or a lack of memory required for representing pure strategies. In this thesis, I am studying the behaviour of the Double Oracle algorithm, which was invented for tackling this problem. It is an algorithm that ensures reaching previously mentioned Nash equilibrium of the zero-sum games with two agents provided that we have two oracles that produce the best response for each agent given the current strategy of an opponent. This algorithm is frequently used in an environment, in which the oracles can only approximate the best responses. My goal is to implement this algorithm on the domain from [1] and analyze how it behaves with a suboptimal oracle. I implemented both the Double Oracle algorithm and Single Oracle algorithm, which is very similar and described later in this thesis. I implemented both algorithms with oracles producing best responses and then I altered these oracles to produce suboptimal responses. I also analyzed theoretically impact of having suboptimal responses and then I compared them to the results obtained from the experiments. To study these results, we have to build up necessary theory and terminology and that is the main goal of next part in this thesis.

Part I

Game theory, Double Oracle and Single Oracle algorithms

Chapter 2

Game theory

In this chapter, I present a part of the game theory with definitions and theorems needed for this thesis. All definitions and theorems from Sections 2.2-2.4 were taken from [2].

2.1 Introduction

Game theory is a study of mathematical models of strategic interaction among rational decision-makers, called players or agents. As mentioned before, my setup and implementation of the algorithm takes place in a deterministic environment with noncooperative agents, so further in this chapter I will define needed terminology along with required theorems.

2.2 Utility theory

Definition 2.1. Agent: An individual, which is involved in a game and chooses strategies to play is an agent.

Agents in an environment are choosing strategies based on their preferences of states of the world. They are so-called self-interested agents.

Prior to developing the theory we have to define a way to interpret the agent's interests and preferences regarding states of the world he lives in. One of the most used approaches to do so, is called the utility theory. This theory represents an agent's interests in states of the world with a function mapping states of the world to real numbers.
Since in this thesis we will be operating in a deterministic environment I identify an action of a player with the outcome of this action. Let $\mathscr{A}$ denote the set of actions available to an agent. Suppose that the agent has a preference relation over the actions $a_1, a_2 \in \mathscr{A}$ as follows:

- $a_1 \succeq a_2$ - Agent weakly prefers a_1 to a_2
- $a_1 \succ a_2$ - Agent strictly prefers a_1 to a_2
- $a_1 \sim a_2$ - Agent does not have a preference between a_1 and a_2

The term lottery also needs to be defined as a probability distribution over actions $l = [p_1 : a_1, ..., p_k : a_k]$, where $a_i \in \mathscr{A}$ and each $p_i \geq 0$ and $\sum_{i=1}^{k} p_i = 1$. Let $p_l(a_i)$ denote the probability of choosing the action a_i in the lottery l and let $\mathscr{L}$ be the set of all lotteries. Suppose that the agent has the same preference relation over the lotteries as over the actions.
Utility function associates each action of a player with a real number. This utility function exists, if the above-mentioned preference relation satisfies the following axioms:

- **Completness**: $\forall a_1, a_2,\ a_1 \succ a_2$ or $a_2 \succ a_1$ or $a_1 \sim a_2$.
- **Transitivity**: If $a_1 \succeq a_2$ and $a_2 \succeq a_3$, then $a_1 \succeq a_3$
- **Substitutability**: If $a_1 \sim a_2$, then for all sequences of one or more outcomes $a_3, \ldots, a_k$ and sets of probabilities $p, p_3, ..., p_k$ for which $p + \sum_{i=3}^{k} p_i = 1$, $[p : a_1, p_3 : a_3, \ldots, p_k : a_k] \sim [p : a_2, p_3 : a_3, \ldots, p_k : a_k]$.
- **Decomposability**: If $\forall a_i \in \mathscr{A}, p_{l_1}(a_i) = p_{l_2}(a_i)$ then $l_1 \sim l_2$
- **Monotonicity**: If $a_1 \succ a_2$ and $p > q$ then $[p : a_1, 1 - p : a_2] \succ [q : a_1, 1 - q : a_2]$
- **Continuity**: If $a_1 \succ a_2$ and $a_2 \succ a_3$, then $\exists p \in [0, 1]$ such that $a_2 \sim [p : a_1, 1 - p : a_3]$.

After defining these axioms we can establish the theorem that describes that if a preference fulfills these axioms, there exists a utility function describing these preferences.

Theorem 2.2. ***(von Neumann and Morgenstern, 1944)*** *If a preference relation $\succeq$ satisfies the axioms completeness, transitivity, substitutability, decomposability, monotonicity, and continuity, then there exists a function $u : \mathscr{L} \mapsto [0, 1]$ with the properties that:*

- $u(a_1) \geq u(a_2) \iff a_1 \succeq a_2$
- $u([p_1 : a_1, \ldots, p_k : a_k]) = \sum_{i=1}^{k} p_i u(a_i)$

Furthermore, von Neumann and Morgenstern also showed in [3] that every positive affine transformation of a utility function is also a utility function satisfying the same preferences, in other words, the absolute magnitude of this function does not matter.

2.3 Games in normal form

In the last section, we have formed a utility theory, that allows using utility function for representing an agent's preferences. As we would expect, a reasonable agent will want to reach the best possible outcome or state of the world and if we take into consideration the utility function, then it comes to maximizing this function by choosing the best possible action. Now, if we consider more than one player each with his own utility function, then it is clear, that actions of one player have an impact on the other players along with their utility functions. Generally, these agents are trying to maximize their utility functions and it does not have to be the case of competing against one another.
The normal, or strategic, form of a game is the most used representation of a strategic interaction in the game theory. This representation of a game assumes that the state of the world only depends on players' actions. This is a very limiting assumption, but many other forms of games can be reduced to the normal-form game.

Definition 2.3. Normal-form game: A (finite, n-person) normal-form game is a tuple (N, A, u), where:

- N is a finite set of n players (agents)
- $A = A_1 \times \cdots \times A_n$, where A_i is a finite set of actions available to player i. Each vector $a = (a_1, \ldots, a_n)$ is called an action profile.
- $u = (u_1, \ldots, u_n)$ where $u_i : A \mapsto R$ is the utility function for a player i.

The usual way of representing normal-form games is through an n-dimensional matrix, where every item in the matrix is a list of agents' payoffs corresponding to agents' actions that lead to this item in the matrix. This representation is very important in the zero-sum games of two players, which we will discuss further in this thesis.
We also have to define strategies in the normal-form games. We have defined the actions available to each player and strategies are choices over these actions. The first strategy that comes to mind is choosing an action and playing it, this is called a pure-strategy, and if every agent follows this, then we obtain a pure-strategy profile. But another option occurs and that is randomizing over the set of actions available to a player following a probability distribution, called a mixed strategy.

Definition 2.4. Mixed strategy: Let (N, A, u) be a normal-form game, and for any set X let $\Pi(X)$ be the set of all probability distribution over X. Then the set of mixed strategies for player i is $S_i = \Pi(A_i)$.

Further on, if not specified, a strategy will always carry a meaning of the mixed strategy.
For any mixed strategy $s_i \in S_i$ of player i, let $s_i(a_i)$ denote probability of choosing the action a_i in the mixed strategy s_i.

Definition 2.5. Support: The support of a mixed strategy s_i for a player i is the set of actions $\{a_i | s_i(a_i) > 0\}$.

Definition 2.6. Mixed-strategy profile: The set of mixed-strategy profiles is the Cartesian product of the individual mixed-strategy sets, $S_1 \times \cdots \times S_n$.

2.4 Best response and Nash equilibrium

First, we define the best response of an agent to a given opponent's strategy. Prior to that let $s_{-i} = (s_1, \ldots, s_{i-1}, s_{i+1}, \ldots, s_n)$. It is a strategy profile s without the strategy of the agent i. Now we can define the best response of an agent i to s_{-i}.

Definition 2.7. Best response: The player i's best response to the strategy profile s_{-i} is a mixed strategy $s_i^* \in S_i$ such that $u_i(s_i^*, s_{-i}) \geq u_i(s_i, s_{-i})$ for all strategies $s_i \in S_i$.

We also have to define ϵ-best response for later use in this thesis.

Definition 2.8. ϵ-best response: Let $\epsilon > 0$. The player i's ϵ-best response to the strategy profile s_{-i} is a mixed strategy $\bar{s}_i \in S_i$ such that $u_i(\bar{s}_i, s_{-i}) \geq u_i(s_i, s_{-i}) - \epsilon$ for all strategies $s_i \in S_i$.

Now that we have defined the best response of an agent, we can move forward to defining the main term in this thesis, the Nash equilibrium. Nash equilibrium is a concept of a solution for a particular game in a sense, that in Nash equilibrium, no player would change their strategy even if they knew strategies of other players.

Definition 2.9. Nash equilibrium: A strategy profile $s = (s_1, \ldots, s_n)$ is a Nash equilibrium if, for all agents i, s_i is a best response to s_{-i}.

We also have to define following term of ϵ-Nash equilibrium, because it is used later on in thesis for proving convergence of the investigated algorithms.

Definition 2.10. ϵ-Nash equilibrium: Let $\epsilon > 0$. A strategy profile $s = (s_1, \ldots, s_n)$ is a ϵ-Nash equilibrium if, for all agents i, s_i is a ϵ-best response to s_{-i}.

2.5 Zero-sum games

Zero-sum games, also called constant-sum games, are a special class of the normal-form games, that we define for two players playing against each other.

Definition 2.11. Constant-sum game: A two-player normal-form game is constant-sum if there exists a constant c such that for each action profile $a \in A_1 \times A_2$ it is the case that $u_1(a) + u_2(a) = c$.

Thanks to the previously mentioned fact, that any positive affine transformation will not alter preferences, we can assume without loss of generality that $c = 0$ in every case and we obtain the term zero-sum game.
Zero-sum games are games of pure competition between two players, because increasing one player's utility function necessarily decreases other player's utility function and therefore they are competing against each other.
For representing a zero-sum game we can construct a 2-dimensional matrix M in the following fashion. Let the first agent with the set of actions A_1 be the row player and second agent with the set of actions A_2 be the column player. Every row of matrix M corresponds to one certain action from A_1 of the first agent and every column of M corresponds to an action from A_2 of the second agent. The number in matrix M on position i, j corresponds to a first agent's payoff (utility function u_1) corresponding to action i for the first agent and action j for the second agent, and therefore loss for the second agent. Let p be a mixed strategy for the row player and q be a mixed strategy for the column player. Value of the game under these strategies is:

$$V(p, q) = p^T M q$$

We will make use of this representation later for constructing a linear program solving a zero-sum game.

2.6 Finding Nash equilibrium of zero-sum games

Throughout my thesis I was confronted with solving zero-sum games efficiently as a part of the implemented algorithm and in this section, I will describe how to tackle this issue with linear programming. A derived linear program for this purpose was taken from [4].
In this section I refer to Section 2.5 for definition of matrix $M \in R^{m \times n}$ as game matrix of zero-sum game. Before continuing, I will establish the notion of **worst-case optimal mixed strategy** for both row and column player. Let there be functions:

$$\beta(\mathbf{x}) = \min_{\mathbf{y}} \mathbf{x}^T M \mathbf{y}, \;\; \alpha(\mathbf{y}) = \max_{\mathbf{x}} \mathbf{x}^T M \mathbf{y}$$

Now we can establish, that $\mathbf{x}^*$ is worst-case optimal, if $\beta(\mathbf{x}^*) = \max_{\mathbf{x}} \beta(\mathbf{x})$ and symmetrically that $\mathbf{y}^*$ is worst-case optimal, if $\alpha(\mathbf{y}^*) = \min_{\mathbf{y}} \alpha(\mathbf{y})$.
Firstly, we need to establish, that there exists a Nash equilibrium to be found and computed. I was in luck with my setup, because thanks to next theorem from [4] we can see that every zero-sum game has at least one Nash equilibrium:

Theorem 2.12. ***(Minimax theorem for zero-sum games)*** *For every zero-sum game, worst-case optimal mixed strategies for both players exist and can be efficiently computed by linear programming. If* $\mathbf{x}^*$ *is a worst- case optimal mixed strategy of row player and* $\mathbf{y}^*$ *is a worst-case optimal mixed strategy of the column player, then* $(\mathbf{x}^*, \mathbf{y}^*)$ *is a mixed Nash equilibrium, and the number* $\beta(\mathbf{x}^*) = max_{\mathbf{x}} min_{\mathbf{y}} \mathbf{x}^T \cdot M \cdot \mathbf{y} = \mathbf{x}^{*T} \cdot M \cdot \mathbf{y}^* = min_{\mathbf{y}} max_{\mathbf{x}} \mathbf{x}^T \cdot M \cdot \mathbf{y} = \alpha(\mathbf{y}^*)$ *is the same for all possible worst-case optimal mixed strategies* $\mathbf{x}^*$ *and* $\mathbf{y}^*$.

Proof of this theorem is presented in [4]. It is based on the duality in linear programming.
We will recall here the construction of linear programs from the proof allowing to compute the Nash equilibrium.

We have to start looking at it from perspective of column player trying to minimaze first player's payoff given he knows his strategy $\mathbf{x}$. According to these settings, variables are $\mathbf{y} = (y_1, \ldots, y_n)$ and $\mathbf{x} = (x_1, \ldots, x_m)$ are known numbers:

$$\text{Minimize } \mathbf{x}^T M \mathbf{y}$$
$$\text{subject to } \sum_{j=1}^{n} y_j = 1, \mathbf{y} \geq \mathbf{0}$$

Now taking the dual of this linear program results in:

$$\text{Maximize } x_0$$
$$\text{subject to } M^T \mathbf{x} - \mathbf{1} x_0 \geq 0$$

Where only x_0 is a variable. Now we construct the same linear program with two more constrains and the most important change, that $x_1, \ldots, x_n$ are regarded as variables.

$$\text{Maximize } x_0$$
$$\text{subject to } M^T \mathbf{x} - \mathbf{1} x_0 \geq 0$$
$$\sum_{i=1}^{m} x_i = 1, \mathbf{x} \geq \mathbf{0}$$

By solving this linear program we obtain the optimal solution $(x_0, \mathbf{x}^*)$.

We can derive the linear program for obtaining the optimal probability distribution of the column player in a similar way. And since both $\mathbf{x}^*$ and $\mathbf{y}^*$ are worst-case optimal, this set of strategies is a Nash equilibrium.

Chapter 3

Double Oracle and Single Oracle algorithms

3.1 Problem solved by these algorithms

Further described algorithms were developed to tackle the problem of large zero-sum games, where one or more players have a vast amount of strategies to choose from and therefore the usual method of finding Nash equilibrium with linear programming fails due to long computing time. The Double Oracle algorithm capitalizes on the existence of two oracles, each one providing the best response for one agent given a strategy of the other agent. This allows reducing computing time significantly for reaching the Nash equilibrium, provided that support of mixed strategies in Nash equilibrium is significantly smaller than number of actions for each player. The Single Oracle algorithm is a version of the Double Oracle algorithm, where one of the oracles is not needed. It is also important to note, that in this section and further on, the values in game matrix, denoted M, carry meaning of cost to the row player as opposed to the payoff for row player as presented in Section 2.5, this context was introduced in [1] and I keep rest of my thesis consistent with it. I would also like to mention that since the value of the game under the strategies p and q is $V(p,q) = p^T Mq$, it now carries meaning of a cost to the row player so he wants to minimaze this value and the column player wants to maximize this value. The value of the game in Nash equilibrium is $V_G = \min_p \max_q V(p,q) = \max_q \min_p V(p,q)$.

3.2 Single Oracle algorithm

3.2.1 Steps of the algorithm

This is a version of the Double Oracle algorithm, where one of the players has a reasonably small set of pure strategies and therefore there is no need for an oracle to produce the best responses for this player. This algorithm was also implemented with a lower number of possible camera positions as compared to the implementation of Double Oracle algorithm.
Let column player have a set of actions $\mathscr{C}$ and let $\mathscr{R}_i$ be the set of found actions for row player on the iteration i. Let q_0 be the initial arbitrary probability distribution over $\mathscr{C}$ (for example the uniform distribution).

The algorithm performs these steps on every iteration i:

1. Use the row oracle to find best pure strategy $R(q_i) = r_i$ as a response to q_i. If $r_i \in \mathscr{R}_i$, terminate the algorithm, otherwise add r_i to $\mathscr{R}_i$.
2. Solve the game with linear program as described in Section 2.6 to obtain optimal strategies $(p_i,\ q_i)$ for row and column player.

This is the algorithm as described in [1]. In the next chapter, I will present my proposal to alter terminating condition of this algorithm and show a proof of it converging to ϵ-Nash equilibrium.

3.2.2 Modified Single Oracle algorithm

The modification consists mainly in a different condition of terminating the algorithm. Let $\mathscr{C}$ be the action set of column player and $\mathscr{R}_i$ be the action set for the row player on the iteration i. Initialize the algorithm with an arbitrary pure strategy r_0, so $\mathscr{R}_0 = \{r_0\}$. On the iteration i, algorithm performs these following steps:

1. Solve the curent game with matrix M and sets $\mathscr{R}_i, \mathscr{C}$. That gives us Nash equilibrium $(p_i,\ q_i)$ of restricted game to $\mathscr{R}_i$. Let $u_i = p_i^T M q_i$.

This value u_i is upper bound on V_G, because $\mathscr{R}_i \subseteq \mathscr{R}$, where $\mathscr{R}$ are all pure stragies for the row player.

2. Produce best response $R(q_i) = r_i$ for q_i. Note that this is a lower bound on V_G, because

$$l_i = V(r_i, q_i) = \min_p p^T M q_i \leq \max_q \min_p p^T M q = V_G$$

3. If $u_i - l_i \leq \epsilon$, finish the algorithm, otherwise add r_i to $\mathscr{R}_i$.

Now it remains to prove, that this algorithm converges to ϵ-Nash equilibrium.

Theorem 3.1. *Let $\epsilon > 0$. The pair of strategies $(p_i,\ q_i)$ obtained by the modified Single Oracle algorithm forms an ϵ-Nash equilibrium.*

Proof. First, q_i is a best response to p_i, because $(p_i,\ q_i)$ forms a Nash equilibrium of a restricted game over $\mathscr{R}_i, \mathscr{C}$. As for p_i, we know from terminating condition that $u_i - l_i \leq \epsilon$, so $u_i - V_G \leq \epsilon$ must also be true, since l_i is a lower bound to V_G. From that follows

$$u_i = p_i^T M q_i \leq V_G + \epsilon = \min_p V(p, q_i) + \epsilon$$

□

Next, I want to present a proof of convergence for a suboptimal oracle. Let the row oracle produce a suboptimal response r_i to q_i:

$$\min_p V(p, q_i) \geq V(r_i, q_i) - \delta,\ \delta > 0$$

Theorem 3.2. *Let $\epsilon > 0,\ \delta > 0$. The pair of strategies obtained by the modified Single Oracle algorithm forms an $(\delta + \epsilon)$-Nash equilibrium.*

Proof. The strategy q_i is the best response to p_i, since $p_i,\ q_i$ form a Nash equilibrium of a game restricted to sets of strategies $\mathscr{R}_i,\ \mathscr{C}$. It follows from terminating condition that $u_i - l_i < \epsilon$, so

$$u_i \leq V_G + \delta + \epsilon$$

$$p_i^T M q_i = u_i \leq V_G + \delta + \epsilon$$

From last statement it is clear, that p_i is a $(\delta + \epsilon)$-best response. □

3.3 Double Oracle algorithm

3.3.1 Steps of the algorithm

The Double Oracle algorithm is an iterative algorithm, which performs three steps in each iteration. As I mentioned earlier, it is used for zero-sum games, where one or more agents have too many strategies, so on each iteration the current game matrix is updated with the pure strategies produced by the oracles. Let R be the oracle for row player and C be the oracle for the column player.
On the iteration i, we have a set of found pure strategies for both players, $\mathscr{R}_i$ for the row player, and $\mathscr{C}_i$ for the column player. The current game matrix consists of rows associated with $\mathscr{R}_i$ and columns associated with $\mathscr{C}_i$. We start the algorithm with an arbitrary row and column and then on each iteration i we perform these three steps:

1. Solving the current game (computing Nash equilibrium) with strategies restricted to the $\mathscr{R}_i$ for the row player and to $\mathscr{C}_i$ for the the column player. In this step we obtain a mixed strategy for the row player p_i and mixed strategy for the column player q_i.
2. The row player reacts to mixed strategy q_i : $R(q_i) = r_i$, finds an optimal pure strategy r_i and we add it to $\mathscr{R}_i$.
3. The column player reacts to mixed strategy p_i : $C(p_i) = c_i$, finds the optimal pure strategy c_i and we add it to $\mathscr{C}_i$.

The algorithm is terminated, when on iteration i, both produced best reponses r_i and c_i are already present in the current set of found strategies $\mathscr{R}_i$ and $\mathscr{C}_i$. In practise it is sometimes better to terminate in case that $V(p_i, c_i) - V(r_i, q_i) < \epsilon$, where ϵ is a parameter. This alternative terminating condition is investigated later in this thesis.

3.3.2 Convergence and correctness proof

Let q_i be a probability distribution over strategies of column player on the iteration i, then we use a row oracle R that provides best response in form of

pure strategy r_i, to current q_i:

$$R(q_i) = r_i, \ \min_p V(p, q_i) \geq V(r_i, q_i)$$

It is important to notice that $V(r_i, q_i)$ is also a lower bound for value of the game $V_G = \max_q \min_p V(p, q) \geq V(r_i, q_i)$. The column oracle C has to work very similarly, it has to produce a best response c_i, in form of a pure strategy, to a row distribution p_i on the iteration i:

$$C(p_i) = c_i, \ \max_q V(p_i, q) \leq V(p_i, c_i)$$

Symmetrically, $V(p_i, c_i)$ is an upper bound to the value of the game $V_G = \max_q \min_p V(p, q) = \min_p \max_q V(p, q) \leq V(p_i, c_i)$.
If we have these optimal oracles, the Double Oracle algorithm will converge to a Nash equilibrium of the game as stated in the following theorem and proof from [1].

Theorem 3.3. *The Double Oracle algorithm converges to a minimax equilibrium.*

Minimax equilibrium means that we have set of worst-case optimal mixed strategies forming a Nash equilibrium.

Proof. Thanks to assuming players having finite sets of strategies and therefore there being finite numbers of rows and columns, eventually $\mathscr{R}$ and $\mathscr{C}$ include all rows and columns and linear program is solving for the whole game. To prove correctness, we have to look at iteration i, where we do not add new column or row. If this happens, then necessarily the lower bound will have the same value as the upper bound and it remains to show, that p_i is minimax solution and q_i is maxmin solution. We will only show proof of p_i being a minimax solution, because proof of q_i being a maxmin solution is analogous. Let $v = V(p_i, q_i)$, from $\forall p \ V(p, q_i) \geq v$ follows $\forall p \ \max_q V(p, q) \geq v$. From $\forall q \ V(p_i, q) \leq v$ follows $\max_q V(p_i, q) \leq v$. These facts combined give us $\forall p, \ \max_q \ V(p_i, q) \leq \max_q \ V(p, q)$, which shows that p_i is minimax optimal. □

3.3.3 Modified terminating condition

Here in this section I show analytical proof of convergence to ϵ-Nash equilibrium under altered runs of the algorithm. As mentioned in Section 3.3.1 the terminating condition can be altered to difference of the lower and the upper bound obtained on each iteration. On iteration i let $u_i = V(p_i, c_i)$, $l_i = V(r_i, q_i)$ be the upper and lower bound on the value of the game V_G. As

mentioned in the same section, it is very practical to change the terminating condition to $u_i - l_i \leq \epsilon$, where ϵ is a parameter. The following theorem states that if we terminate the algorithm with this condition, it converges to a ϵ-Nash equilibrium.

Theorem 3.4. *Let $\epsilon > 0$. The pair of strategies (p_i, q_i) obtained by Double Oracle algorithm forms a mixed ϵ-Nash equilibrium.*

Proof. Following the Definition 2.10 of the ϵ-Nash equilibrium, we prove that both p_i and q_i are ϵ-best responses to each other.
First, we show that p_i is ϵ-best response to q_i. From terminating condition, we know that $u_i - l_i \leq \epsilon$. We further know, that

$$V(p_i, q_i) \leq u_i$$

And from terminating condition we can write:

$$V(p_i, q_i) \leq u_i \leq l_i + \epsilon$$

It is clear from these inequalities, that p_i is a ϵ-best response to q_i.
As for q_i, we can state, that:

$$V(p_i, q_i) \geq l_i$$

That combined with terminating condition gives us:

$$V(p_i, q_i) \geq l_i \geq u_i - \epsilon$$

Again, this proves that q_i is a ϵ-best response to p_i. □

It can be further proved, that $V(p_i, q_i)$ is not further than ϵ from value of the game V_G.
From terminating condition, we know that $u_i - l_i \leq \epsilon$. Because r_i is the best response, it follows $\min_p V(p, q_i) \geq l_i$ and from there $\max_q \min_p V(p, q) \geq l_i$. Combining this with terminating condition we get:

$$\max_q \min_p V(p, q) \geq u_i - \epsilon$$

$$\max_q \min_p V(p, q) + \epsilon \geq u_i$$

From c_i being the best response, we obtain $\max_q V(p_i, q) \leq u_i$. Two previous statements give us:

$$\max_q V(p_i, q) \leq \max_q \min_p V(p, q) + \epsilon$$

Strategy p_i ensures that $V(p_i, q_i)$ is not higher than ϵ above the V_G.
Due to c_i being the best response, we get:

$$\min_p \max_q V(p, q) \leq u_i$$

From r_i being the best response we obtain

$$\min_p V(p, q_i) \geq l_i \geq u_i - \epsilon$$

$$\min_p V(p, q_i) + \epsilon \geq u_i$$

These inequalities combined give us

$$\min_p \max_q V(p, q) \leq u_i \leq \min_p V(p, q_i) + \epsilon$$

$$\min_p \max_q V(p, q) - \epsilon \leq \min_p V(p, q_i)$$

Strategy q_i ensures, that $V(p_i, q_i)$ is not below the V_G more than ϵ.

3.3.4 Analytical convergence of Double Oracle algorithm with suboptimal oracles

In this section I present the analytical proof of convergence for Double Oracle algorithm with suboptimal oracles and the terminating condition described in the previous Section 3.3.3. Let the row oracle produce a suboptimal response r_i to q_i:

$$\min_p V(p, q_i) \geq V(r_i, q_i) - \delta,\ \delta > 0$$

And column oracle produce suboptimal response c_i to p_i:

$$\max_q V(p_i, q) \leq V(p_i, c_i) + \rho,\ \rho > 0$$

Let the terminating condition of the algorithm be again $u_i - l_i \leq \epsilon$ as described in Section 3.3.3.

Theorem 3.5. *Let $\epsilon > 0,\ \delta > 0, \rho > 0$ and $\omega = \max\{\delta, \rho\}$. The set of mixed strategies, provided by Double Oracle algorithm with the suboptimal oracles, $(p_i,\ q_i)$ forms a mixed $(\epsilon + \omega)$-Nash equilibrium.*

Proof. Following the Definition 2.10 of the ϵ-Nash equilibrium, we prove that both p_i and q_i are $(\epsilon + \omega)$-best responses to each other.
First, we show that p_i is $(\epsilon + \omega)$-best response to q_i. From terminating condition, we know that $u_i - l_i \leq \epsilon$. We further know, that

$$V(p_i, q_i) \leq u_i + \rho$$

And from terminating condition we can write:

$$V(p_i, q_i) \leq u_i + \rho \leq l_i + \epsilon + \rho$$

Since $\omega = \max\{\delta, \rho\}$, it also follows:

$$V(p_i, q_i) \leq l_i + \epsilon + \omega$$

It is clear from these inequalities, that p_i is a $(\epsilon + \omega)$-best response to q_i.
As for q_i, we can state, that:

$$V(p_i, q_i) \geq l_i - \delta$$

$$V(p_i, q_i) \geq l_i - \omega$$

That combined with the terminating condition gives us:

$$V(p_i, q_i) \geq l_i - \omega \geq u_i - (\epsilon + \omega)$$

This proves that q_i is a $(\epsilon + \omega)$-best response to p_i. □

And as in the last section, we can go further and prove that the value of the game $V(p_i, q_i)$ is not further than $(\epsilon + \rho + \delta)$ from the V_G.
The terminating condition gives us:

$$u_i - l_i \leq \epsilon$$

From $\min p\ V(p, q_i) \geq l_i - \delta$ it follows

$$\max_q \min_p V(p, q) \geq l_i - \delta$$

$$\max_q \min_p V(p, q) + \delta \geq l_i$$

And from $\forall q\ V(p_i, q) \leq u_i + \rho$ it follows

$$\max_q V(p_i, q) \leq u_i + \rho$$

The terminating condition shows

$$u_i + \rho \leq \epsilon + l_i + \rho$$

These two combined give us

$$\max_q V(p_i, q) - \epsilon - \rho \leq l_i$$

And the following inequalities occur

$$\max_q V(p_i, q) - \epsilon - \rho \leq l_i \leq \max_q \min_p V(p, q) + \delta$$

$$\max_q V(p_i, q) \leq \max_q \min_p V(p, q) + \delta + \epsilon + \rho$$

And therefore p_i assures that $V(p_i, q_i)$ is not higher than $(\epsilon + \rho + \delta)$ above the V_G.
In similar fashion we derive following inequalities for q_i

$$\min_p V(p, q_i) + \delta + \epsilon \geq u_i \geq \min_p \max_q V(p, q) - \rho$$

$$\min_p V(p, q_i) \geq \min_p \max_q V(p, q) - (\rho + \epsilon + \delta)$$

It follows that q_i assures the value $V(p_i, q_i)$ is not lower than $(\epsilon + \rho + \delta)$ below the V_G.

Part II

Implementation and convergence experiments of Double Oracle and Single Oracle algorithms

Chapter 4

Environment setup and implementation of algorithms

4.1 Game setup

I implemented this algorithm on the game presented in [1]. The goal is to find the Nash equilibrium of the following game. We have two players playing a zero-sum game. The first player is a potential thief starting from the red state in the following map, shown in Figure 4.1, trying to get to one of the three goals marked green. The second player is a protector of those goals and he is trying to capture the thief on his camera. The first player, the thief, does not know where the camera is, he only knows potential places, where it could be and his goal is to minimize the time he is being observed by the camera. The second player is trying to maximize the thief's time being observed by his camera. The following map was created to be similar with the map in [1]:

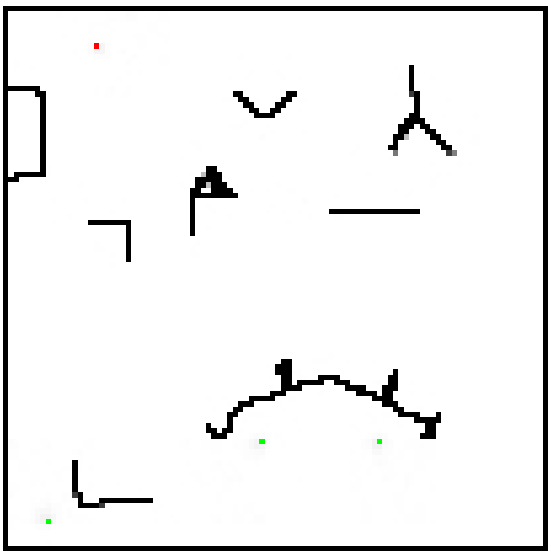


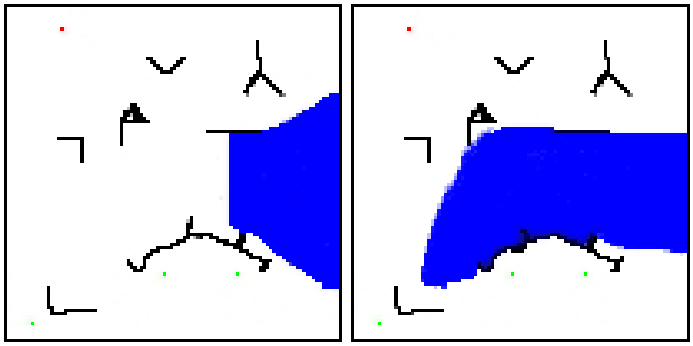
Figure 4.1: A map of the environment.

The size of the map is 100×100 pixels. Pixels also serve as states of the world, where the agent can go. Obstacles or unreachable states of the world have black color. The used environment is deterministic, which means that the actions of the planning agent are associated with an outcome of this action. For example, if an agent decides that he wants to go to his left, choosing this action will result in him going left every time.

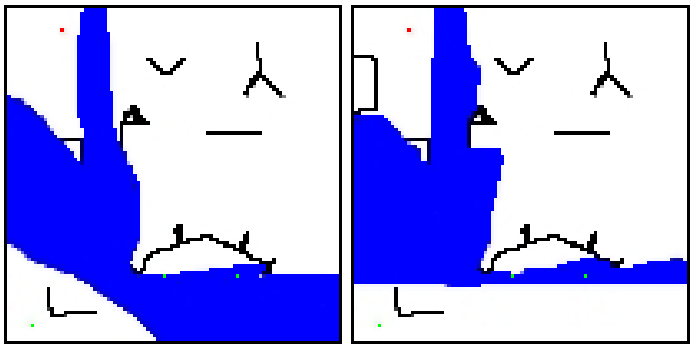
4.2 Implementation of Single Oracle algorithm

In this implementation I have used only 4 cameras, which can be seen in [1] and therefore there were only four columns in the game matrix M describing this game on every iteration. Due to this fact, I didn't have to implement the column oracle and the computation time of the linear program in the first step of the algorithm is greatly reduced.

The implementation was done in coding language Python. The second step of the algorithm from Section 3.2.1 was computed by linear programming as described in Section 2.6 with a call to the scipy library [5]. The method used to compute the optimum of this linear program is the Interior-point method as outlined in [6]. My implementation also relies heavily on NumPy library - [7] for quicker computational time and [8] for graphic visualization.
All of the possible camera locations can be seen in Figure 4.2, where blue color means, that the camera can observe that place in the map.



(a) : Camera location 1 **(b) :** Camera location 2



(c) : Camera location 3 **(d) :** Camera location 4

Figure 4.2: Possible camera locations

It remains to present the implementation of the row oracle producing the best possible route on the iteration i given a probability distribution q_i over the set of these 4 possible camera locations.

4.2.1 Row oracle

Implementation of the row oracle has to produce the best possible response for the current situation and for this task A^* algorithm was selected. This algorithm was presented in [9]. A^* is a route planning algorithm, which can be used in a deterministic environment for reaching the agent's goal.

Pseudocode for this algorithm is:

Algorithm 1: A^* algorithm

```
Function Main(start, goal, h, q_i):
    openSet := {start}
    cameFrom := an empty map
    gScore := map with default value of ∞
    gScore[start] := 0
    fScore := map with default value of ∞
    fScore[start] := 0
    while openSet is not empty do
        current := the node in openSet having the lowest fScore[] value
        if current = goal then
            return reconstructpath(cameFrom, current)
        remove current from openSet
        for each neighbor of current do
            tentativegScore := gScore[current] + d(current, neighbor) +
             camera_cost(neighbor)
            if tentativegScore < gScore[neighbor] then
                cameFrom[neighbor] := current
                gScore[neighbor] := tentativegScore
                fScore[neighbor] := gScore[neighbor] + h(neighbor)
                if neighbor not in openSet then
                    add neighbor to openSet
    return Failure

Function reconstructpath(cameFrom, current):
    totalpath := current
    while current in cameFrom.keys do
        current := cameFrom[current]
        add current to totalpath
    return totalpath
```

The A^* algorithm provides the best route from a start to a goal given that the heuristic function h is consistent. For this reason, I implemented a heuristic, which assigns a node value of euclidian distance to the closest goal, which is a consistent heuristic:

$$h(node) = \min_{goals} distance(node, goal)$$

$$distance(node, goal) = \sqrt{(node_x - goal_x)^2 + (node_y - goal_y)^2}$$

Thanks to this implementation of the heuristic function, the algorithm

always outputs the best route to one of the three goals. The function $d(current,\ neighbor)$ outputs cost of an edge from one state to its neighbor: the cost is 1 for the 4 cardinal directions and $\sqrt{2}$ for diagonal movement. Let $\mathscr{C}_i$ include k cameras on the iteration i, the function $camera\_cost(node)$ outputs:

$$camera\_cost(neighbor) = \sum_{j=1}^{k} q_{i,j} \cdot \delta(neighbor, j) \cdot 15$$

The number 15 carries meaning of a cost to the row player for stepping on a place, that is observed by the camera. This number determinates how much we value a longer path with less observation as compared to a shorter path with more observation from the camera and the absolute value of this number will not have any effect on convergence of the algorithms, so it is not an important parameter of my implementation. Variable $q_{i,j}$ is a probability of choosing camera j under probability distribution q_i on iteration i of the algorithm and $\delta()$ is a function:

$$\delta(neighbor, j) = \begin{cases} 1, & \text{if neighbor is observed by the camera } j \\ 0, & \text{if neighbor is not observed by the camera } j \end{cases}$$

The A^* algorithm with presented heuristic function and costs influenced by camera observation as defined above returns the path with minimal cost. In usual implementation function $camera\_cost(node)$ is not present, I added it to include camera observation into the path planning problem.

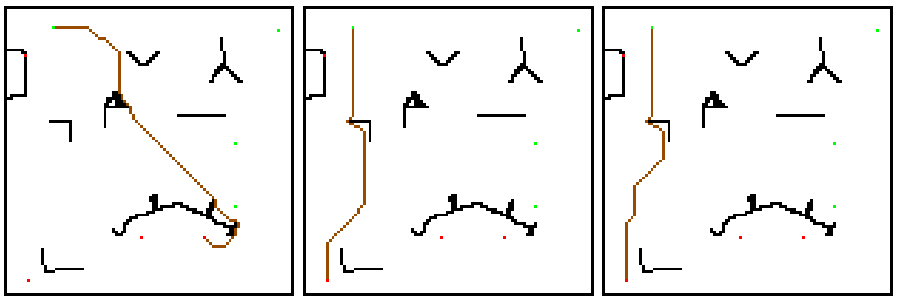
4.2.2 Results with optimal row oracle

If we have optimal oracles we can use the algorithm and terminating condition described in Section 3.2.1. My implementation of this algorithm with the previously shown row oracle reached following results. My implementation found the optimal probability distributions over cameras from Figure 4.2:

camera location	1	2	3	4
probability	0.0	0.388	0.0	0.612

Table 4.1: Probability distribution over camera locations

In the next figure, I will show the paths that were assigned probability bigger than zero under mixed strategy in Nash equilibrium.



(a) : Path 1 **(b) :** Path 2 **(c) :** Path 3

Figure 4.3: Paths chosen by the algorithm

The following table shows a probability distribution among these paths:

path number	1	2	3
probability	0.727	0.159	0.114

Table 4.2: Probability distribution over paths

The following graph shows the convergence of the Single Oracle algorithm with optimal row oracle used.

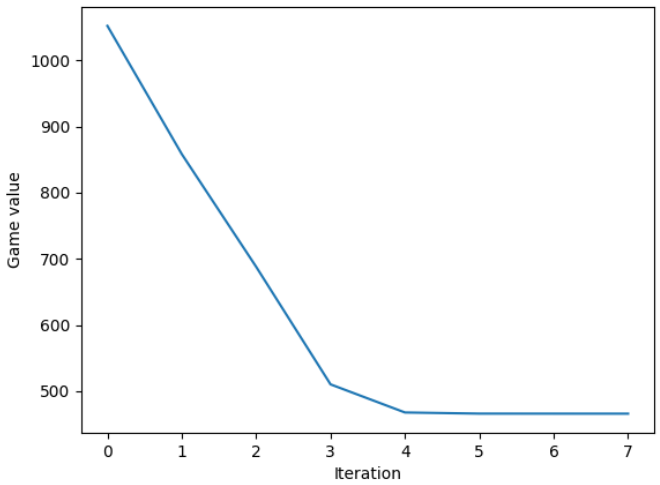


Figure 4.4: Convergence with respect to iterations

The value of the game converges at a value of 465. Thanks to there being only 4 possible camera locations used in this implementation, the algorithm converges very quickly with respect to iterations and there is no need for column oracle.

4.2.3 Convergence of Single Oracle algorithm with suboptimal oracle

For obtaining a suboptimal row oracle I altered run of the Single Oracle algorithm in the following fashion. In each iteration i, before letting row oracle produce a best response to current q_i, I generate a 100×100 array with values from a uniform distribution in interval $[0, c]$, where c is a constant from range $[0, 15]$. This random array serves as an additional cost to each state in the map. With cost altered as previously described, the row oracle reacts to different information in camera observation and due to this fact does not produce the best response available. A constant c served as a parameter of how much the oracle is altered. The bigger the c, the bigger the influence of random array on a row oracle. With this randomness introduced, I had to use proposed version of this algorithm described in Section 3.2.2. The Figure 4.5 shows the value of the game after termination of the algorithm for every $c \in [0, 15],\ c \in Z$. The parameter ϵ was set to 3.

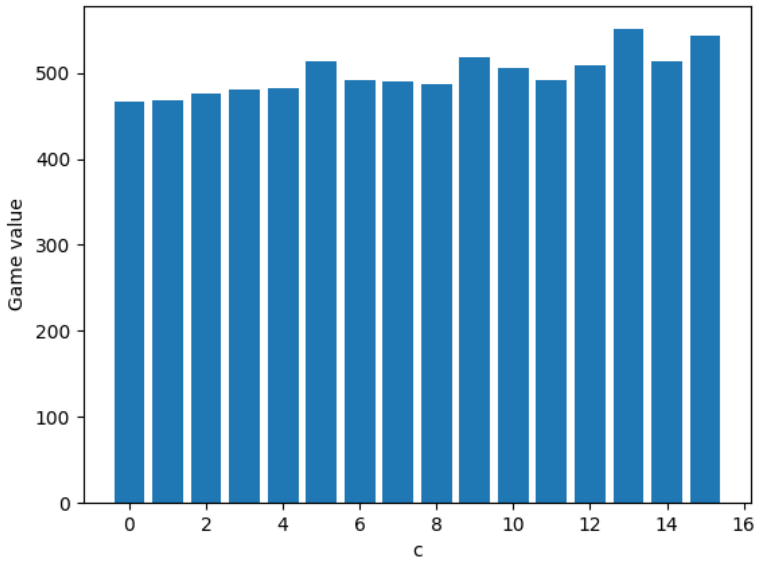
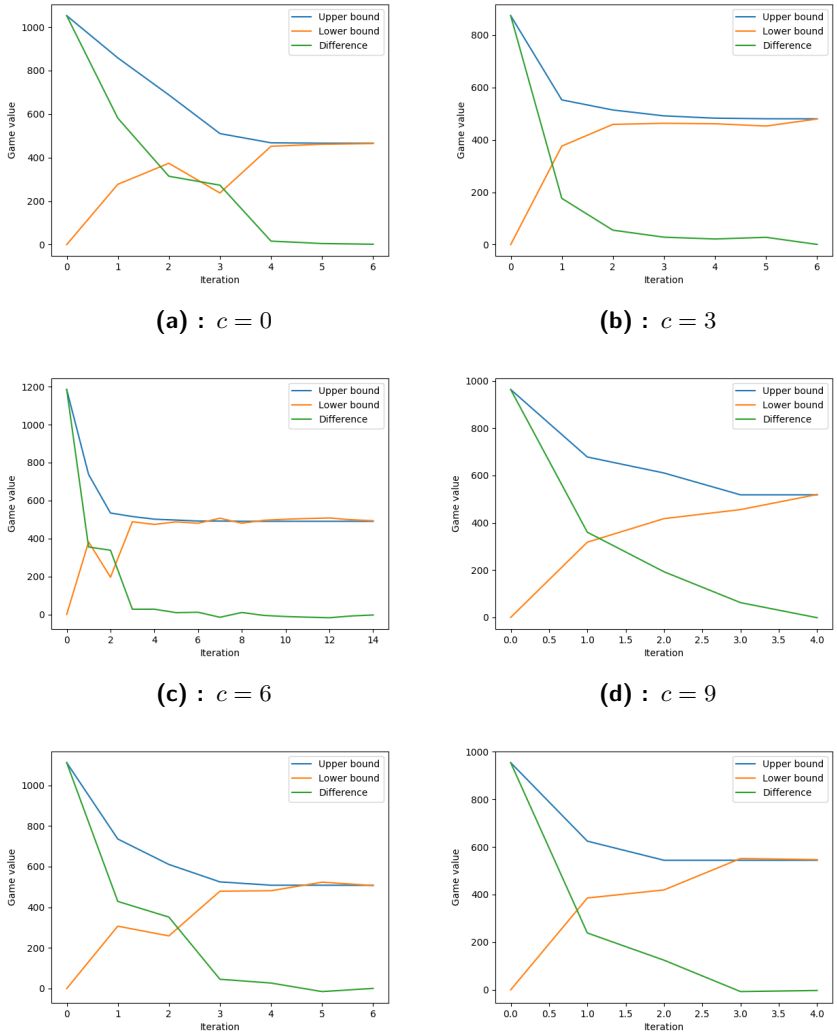


Figure 4.5: Game value with respect to c

As we can see from Figure 4.5 the value of the game at the end of the algorithm has an increasing tendency with increasing c. This tendency is what

we expected as the quality of the row oracle is decreasing with increasing c. The reached mixed strategy profile is not a Nash equilibrium due to the fact, that in every Nash equilibrium, the value of the game has to be the same. In the following Figure 4.6, we can look at convergence graphs of lower and upper bounds for $c = \{0, 3, 6, 9, 12, 15\}$:



(a) : $c = 0$ **(b) :** $c = 3$

(c) : $c = 6$ **(d) :** $c = 9$

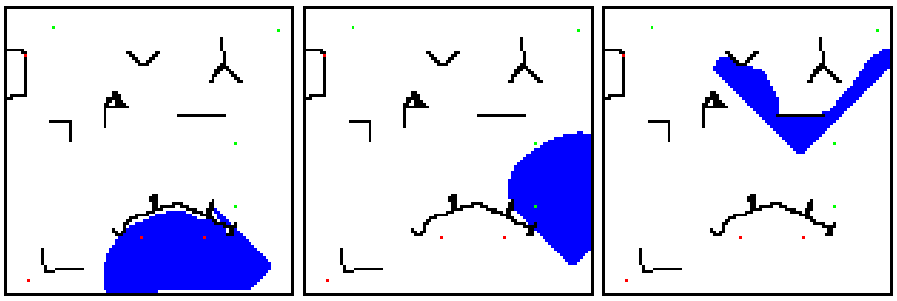
(e) : $c = 12$ **(f) :** $c = 15$

Figure 4.6: Convergence graphs

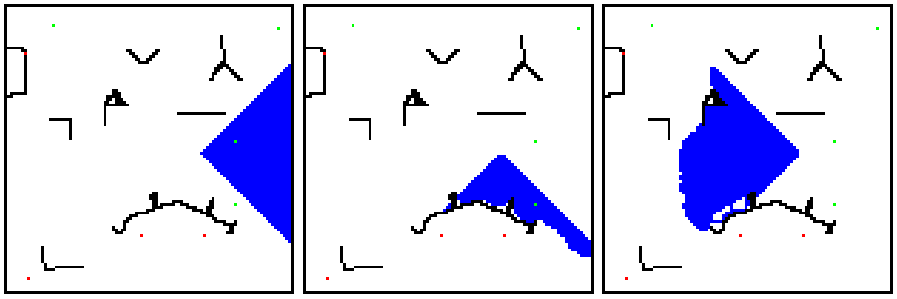
From previous Figures 4.6 and 4.5 we can see that there is not a noticeable difference in iterations the algorithm requires to terminate, but the value of the game it converges to is not the V_G computed in the previous section, and reached mixed strategies do not form a Nash equilibrium, but they do form a ϵ-Nash equilibrium as proved in Section 3.2.2.

4.3 Implementation of Double Oracle algorithm

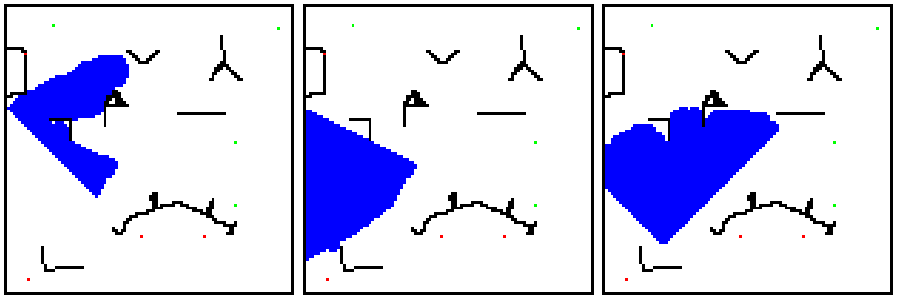
For implementing the column oracle I have added more options for camera locations to the same map. The camera locations are shown in Figure 4.7. The 12 possible positions of the camera were selected to effectively cover the map in Figure 4.1.



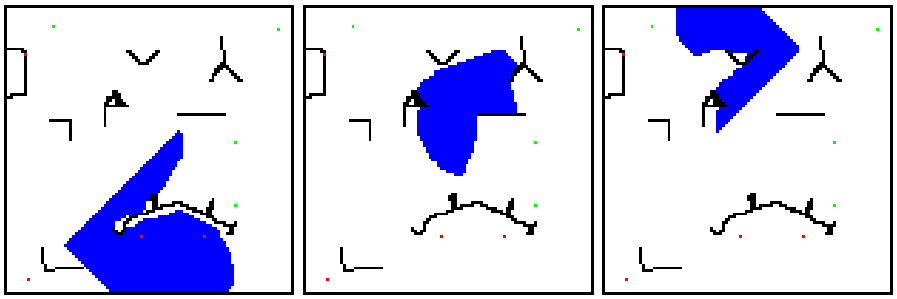
(a) : Camera location 1 **(b) :** Camera location 2 **(c) :** Camera location 3



(d) : Camera location 4 **(e) :** Camera location 5 **(f) :** Camera location 6



(g) : Camera location 7 **(h) :** Camera location 8 **(i) :** Camera location 9



(j) : Camera location 10 **(k) :** Camera location 11 **(l) :** Camera location 12

Figure 4.7: Camera locations in the map

The implementation of row oracle is the same as described in the Section 4.2.1.

4.3.1 Implementation of Column Oracle

The task for column oracle in each iteration is to pick the best camera location out of those in Figure 4.7 to a current strategy of the row player. Since there are 12 camera locations to choose from, every camera locations's result is computed and then I choose the best one to ensure having an optimal oracle providing the best response. Let there be k paths in $\mathscr{R}_i$ and p_i be the computed probability distribution over them in first step of this algorithm on the iteration i. Then the column oracle C chooses the best camera location by following formula:

$$C(p_i) = \arg\max_{cameras} \sum_{j=1}^{k} p_{i,j} c(path_j, camera)$$

Where $p_{i,j}$ is probability of choosing path j under probability distribution p_i and $c(path, camera)$ is a function returning cost of path while being observed by input camera.

4.3.2 Result of Double Oracle algorithm with optimal oracles

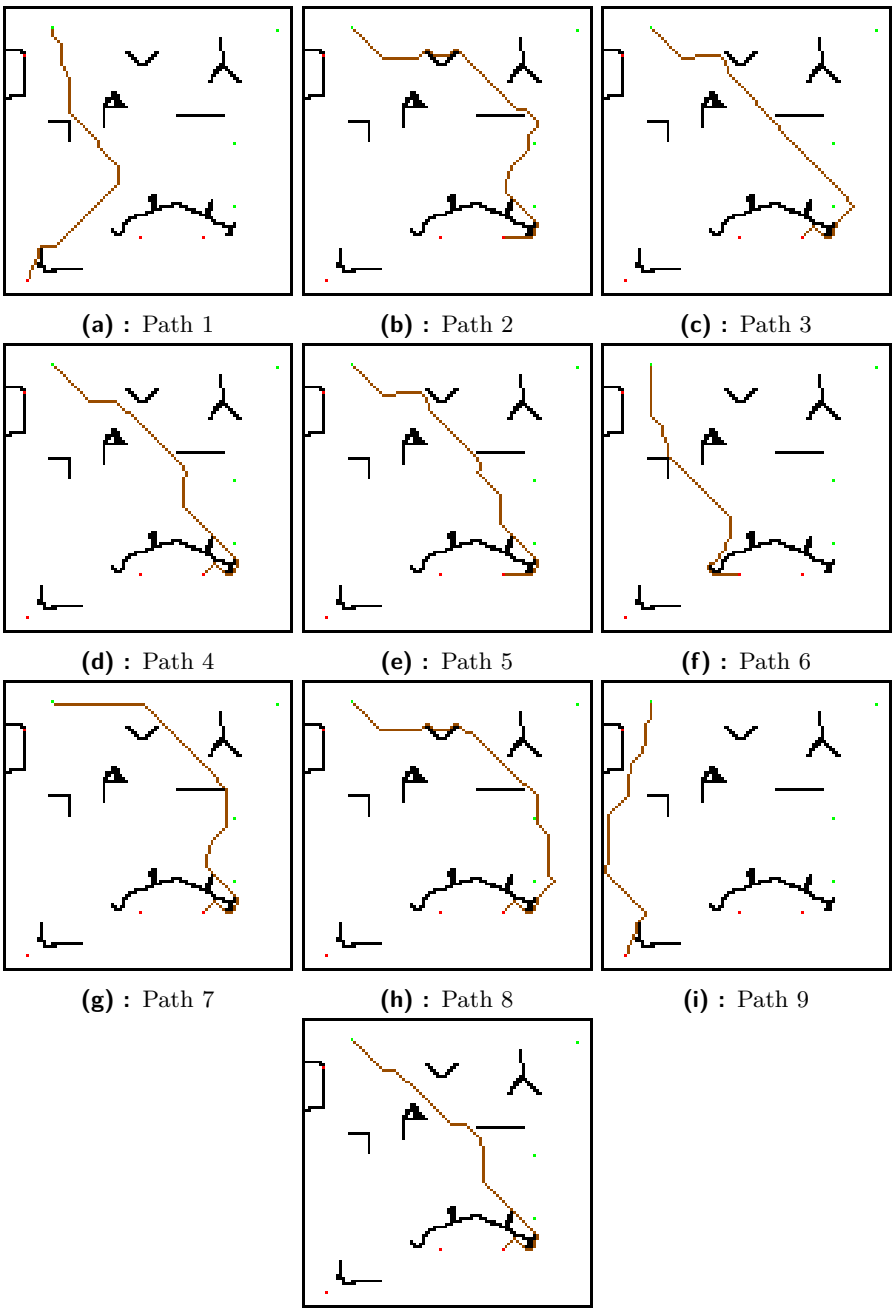
With use of optimal oracles, described in Sections 4.3.1 and 4.2.1, I could use the algorithm as described in Section 3.3.1. My implementation of the algorithm reached Nash equilibrium with following mixed strategies. The column mixed strategy is:

camera location	1	2	3	4	5	6
probability	0.165	0.158	0.167	0.043	0.053	0.018

camera location	7	8	9	10	11	12
probability	0.025	0.133	0.112	0.084	0.044	0.0

Table 4.3: Probability distribution

The following figure shows the paths with significant probability mass assigned to them in the Nash equilibrium computed by the Double Oracle algorithm.



(a) : Path 1 **(b) :** Path 2 **(c) :** Path 3

(d) : Path 4 **(e) :** Path 5 **(f) :** Path 6

(g) : Path 7 **(h) :** Path 8 **(i) :** Path 9

(j) : Path 10

Figure 4.8: Paths found by optimal oracle

And the following table shows probability mass assigned to each path in Figure 4.8 in the obtained Nash equilibrium:

path number	1	2	3	4	5
probability	0.080	0.227	0.223	0.021	0.038
path number	6	7	8	9	10
probability	0.051	0.102	0.043	0.005	0.205

Table 4.4: Probability distribution over paths

The algorithm converged to a Nash equilibrium and lower and upper bounds given by responses of oracles converged as shown in the following graph.

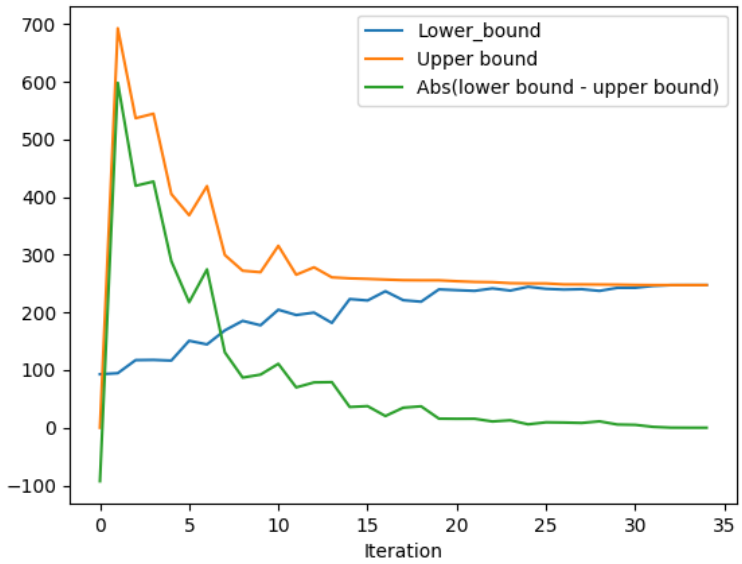


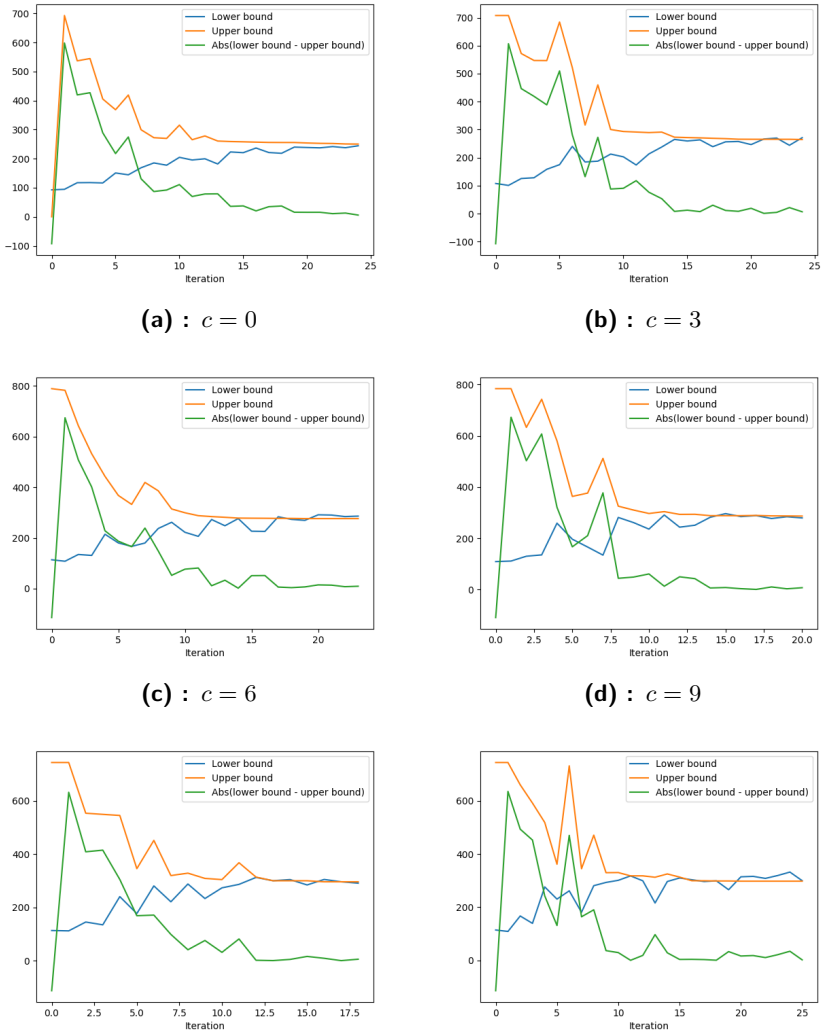
Figure 4.9: Convergence with respect to iterations

As seen in Figure 4.9 the upper and lower bounds on V_G converged to the value of the game or close to it in approximately 30 iterations of the algorithm and the value of the game it converged to is approximately 247.

4.3.3 Convergence of Double Oracle algorithm with suboptimal row oracle

Following experiment is the same experiment as in the Section 4.2.3 with prices randomly increased, causing the row oracle to produce a suboptimal

response. This experiment was done for constants $c \in Z$ from interval $[0, 15]$. Since we are using the suboptimal oracle and we are introducing randomness to the path planning, we have to alter the terminating condition to the one presented and investigated in the Section 3.3.4. The following figure shows the convergence graphs for $c = \{0, 3, 6, 9, 12, 15\}$.



(a) : $c = 0$ **(b) :** $c = 3$

(c) : $c = 6$ **(d) :** $c = 9$

(e) : $c = 12$ **(f) :** $c = 15$

Figure 4.10: Convergence for various constants c

The game values this algorithm converged to can be seen in the next bar graph.

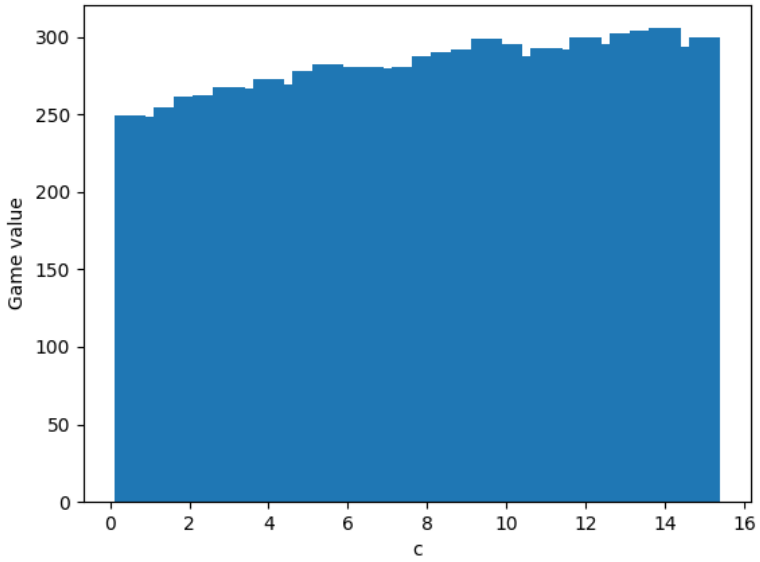
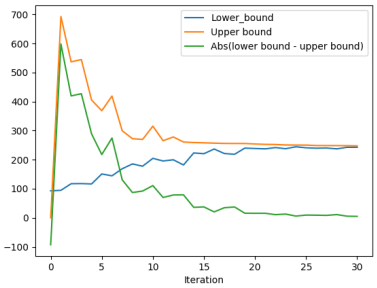
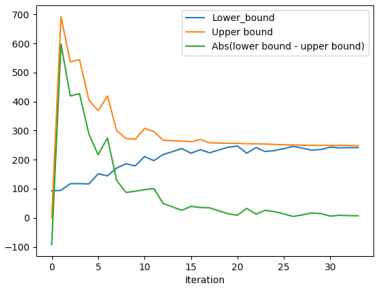


Figure 4.11: Final game values with respect to c

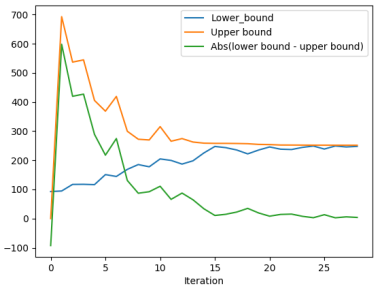
And as we can see the final value of the game is increasing along with the parameter c as expected and seen in the Section 4.2.3.
This constant c and randomness it introduces is supposed to represent a certain unknowledge of the environment or not precisely knowing the opponent's current strategy. The following experiment is representing a situation, where the agent with a optimal oracle, but an incomplete information is gaining a full view of the game as the algorithm continues, so randomness introduced is lowering. This is captured by the constant c starting at a certain value and decreasing with iterations.
This test was also done for constants $c \in Z$ from the interval $[0, 15]$ and the following figure shows results.



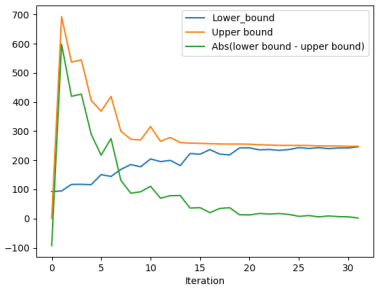
(a) : $c = 1$



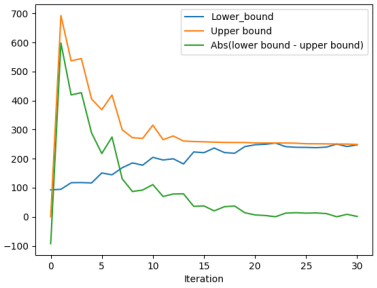
(b) : $c = 3$



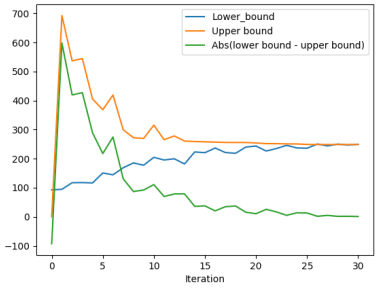
(c) : $c = 6$



(d) : $c = 9$



(e) : $c = 12$



(f) : $c = 15$

Figure 4.12: Convergence for various constants c

And final game values it converged to can be seen in the next bar graph:

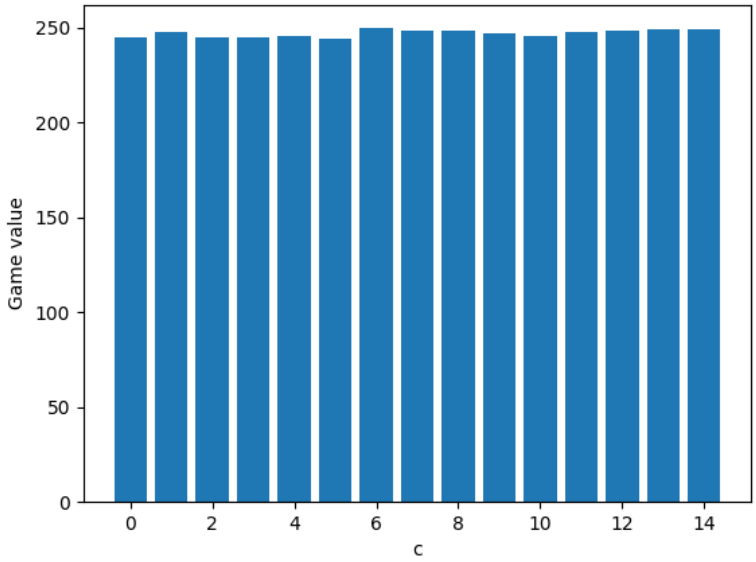


Figure 4.13: Final game values with respect to c

So as we can see in the Figures 4.12 and 4.13 in a situation, where constant c is decreasing with each iteration by $\frac{c}{30}$ it converged in approximately same count of iteration as algorithm with optimal oracle. This experiment was carried out to show, that even if we start with a suboptimal oracle, that is producing better and better responses on each iteration, we still can reach the ϵ-Nash equilibrium this algorithm converges to with the optimal oracles.

Chapter 5

Discussion

In this chapter, I would like to discuss a few points worth mentioning.
My first point is that my theorems and proofs for the convergence of the investigated algorithms with suboptimal oracles were made for suboptimal oracles, which consistently produce a response, which is worse than the best response by a fixed value. The randomness introduced to the oracle was causing this difference to not be consistent, although we could see that the oracle was consistently bad enough to produce suboptimal results. For this purpose I carried out an experiment in the Section 4.3.3, where the oracle got better throughout the iteations and we could see, that the performance was in most cases the same as with the optimal oracle.
I would also like to mentionion my choice of ϵ, I kept value of this parameter for most of my experiments at a fixed value as it does not have much effect on the run of the algorithm provided that the parameter has a small value as compared to the absolute values of the lower and the upper bounds.
The next point I would like to discuss is the absolute value of the penalty for stepping on a location observed by the camera, which was set to 15. As mentioned before in the Section 4.2.1. This value carries a meaning of how much the thief values shorter paths with a bit more observation as compared to a longer paths with less observation by the camera, because if this parameter has a high value, the oracle will produce a longer paths with as few observed locations as possible and on the contrary, if this parameter has a low value, the oracle will produce a shorter paths with more observation by the camera. So this value will have effect on the proportions of the produced paths and the cost of these paths, but it certainly does not have any effect on convergence of investigated algorithms, so it is not an important parameter for my experiments.
I would also like to mention the fact that the values V_G was different for Single Oracle and Double Oracle algorithms. It happened simply because

the game setup was different for each game and as we would expect, the V_G of game played out by Single Oracle algorithm was higher than in game for Double Oracle algorithm. I would expect this, because the camera locations in first Single Oracle experiment cover much more ground than the camera locations in Double Oracle experiment.
At last I would like to mention that for altering the row oracle to produce a suboptimal response, I also tried altering the heuristic function to a non consistent one, but results from these experiments were not clear as it was difficult to interpret the influence of this changed heuristic to the produced paths and their optimality.

Chapter 6

Conclusion

The four main goals of my thesis as described in thesis specification were all fulfilled. The main goal of the thesis was to implement the Double Oracle algorithm and I have successfully implemented this algorithm on similar problem as in [1]. Furthermore, my goal was to study convergence of this algorithm with suboptimal oracles and I have presented the theorems and proofs in Chapter 3 describing the algorithm's behaviour in these situations. After presenting these theoretical results, I carried out the experiments in Chapter 4, which correspond to these theoretical results. The experiments confirmed, that by using the suboptimal oracles, the algorithm converges to an ϵ-Nash equilibrium and not the optimal Nash equilibrium. The main contribution of my thesis is in the presented theoretical and experimental results, which together describe, how the algorithm converges under various circumstances and terminating conditions.

Bibliography

1. MCMAHAN, H. Brendan; GORDON, Geoffrey J.; BLUM, Avrim. Planning in the Presence of Cost Functions Controlled by an Adversary. 2018. Available from DOI: 10.1184/R1/6608441.v1.
2. SHOHAM, Yoav; LEYTON-BROWN, Kevin. *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*. USA: Cambridge University Press, 2008. ISBN 0521899435.
3. VON NEUMANN, J.; MORGENSTERN, O. *Theory of Games and Economic Behavior*. Princeton University Press, 1944. Science Editions.
4. MATOUŠEK, Jiří; GÄRTNER, Bernd. *Understanding and Using Linear Programming (Universitext)*. Berlin, Heidelberg: Springer-Verlag, 2006. ISBN 3540306978.
5. VIRTANEN, Pauli et al. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*. 2020, vol. 17, pp. 261–272. Available from DOI: https://doi.org/10.1038/s41592-019-0686-2.
6. ANDERSEN, Erling D.; ANDERSEN, Knud D. The Mosek Interior Point Optimizer for Linear Programming: An Implementation of the Homogeneous Algorithm. In: *High Performance Optimization*. Ed. by FRENK, Hans; ROOS, Kees; TERLAKY, Tamás; ZHANG, Shuzhong. Boston, MA: Springer US, 2000, pp. 197–232. ISBN 978-1-4757-3216-0. Available from DOI: 10.1007/978-1-4757-3216-0_8.
7. VAN DER WALT, S.; COLBERT, S. C.; VAROQUAUX, G. The NumPy Array: A Structure for Efficient Numerical Computation. *Computing in Science Engineering*. 2011, vol. 13, no. 2, pp. 22–30.
8. HUNTER, J. D. Matplotlib: A 2D Graphics Environment. *Computing in Science Engineering*. 2007, vol. 9, no. 3, pp. 90–95.

9. HART, Peter; NILSSON, Nils; RAPHAEL, Bertram. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics*. 1968, vol. 4, no. 2, pp. 100–107. Available from DOI: 10.1109/tssc.1968.300136.