Search and Rescue Games

Games on Trees and Graphs

Deon Ha

Search and Rescue Games

Games on Trees and Graphs

by

Deon Ha

to obtain the degree of

Master of Science

in

Applied Mathematics

at the Delft University of Technology,
to be defended publicly on Friday August 27, 2021 at 15:00 AM.

Student number:	4469445	
Project duration:	December 2, 2020 – August 27, 2021	
Thesis committee:	Dr. ir. R. J. Fokkink	TU Delft, Supervisor
	Msc. J. Brethouwer,	Supervisor
	Prof. dr. F. H. J. Redig,	TU Delft
	Dr. J. A. M. de Groot,	TU Delft

Delft University of Technology
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft Institute of Applied Mathematics

Preface

This thesis is written as part of my master in Applied Mathematics.

In my years as a bachelor and master student I always enjoyed game theory related topics. Therefore, I reached out to one of my supervisors, Dr. ir. R.J. Fokkink, to write a thesis about a game theory related problem. He introduced me to *Search and Rescue Games* and encouraged me to write a thesis about it. During the course of the project I learned a lot about mathematics and game theory in particular. I would like to thank Dr. ir. R.J. Fokkink and Msc. J. Brethouwer for their continued support and guidance during the course of the project.

On top of that, I would also like to thank the thesis committee: Dr. ir. R.J. Fokkink, Msc. J. Brethouwer, Prof. dr. F. H. J. Redig and Dr. J. A. M. de Groot for their time and interest to evaluate my work. Finally, I would also like to thank my family and friends for their support during the course of the project.

Deon Ha
Delft, August 2021

Abstract

In this thesis report we consider a search and rescue problem in which one or multiple targets/objects are hidden in some playing field, and must be rescued/found by a searcher. The targets are for example: earthquake survivors, lost hikers or prisoners held by an adversary, and are hidden in some section of the play field. Searching any of the sections of the play field has a certain probability of failing; the searcher might get lost, trapped or captured herself. The goal is to find the search that maximises the probability of finding all targets, and to find the hiding spot that minimises it. We define and solve the search and rescue game on a play field for which movement between any two section of the play field is always possible. We also define and solve the game played on a tree for which movement is limited by the tree structure. Due to the complexity of this game, we restrict ourselves to one target. Finally, we extend the game by replacing the play field with specific types of graphs that are not trees. For some of these graphs, we have found (partial) solutions.

Contents

1 Introduction

Many search and rescue operations can be extremely dangerous for the searcher. For example, a firefighter who has to rescue someone from a burning building. At any moment, the firefighter could be trapped himself/herself resulting in the loss of two lives. It is therefore useful to model such rescue operations which has been done in this report. Alpern and Gal [1] and Hohzaki [7] provide a general overview of the literature on search games. Lidbetter [9] is, as far as we know, the only one to consider a model in which the searcher has a probability of being trapped/captured herself.

We start by defining the basic game in chapter 2. In this game, the searcher has to find one or more targets hidden in some play field. The play field is divided into multiple sections in which the hider needs to hide one or multiple targets. In this chapter, movement between any two sections of the play field is always possible.

In Chapter 3, we consider the search and rescue game played on a tree. This is, for most part, the same game, but now played on a tree. Movement between any two sections is now limited by the structure of the tree. Due to the complexity of the game, we only consider the case in which only one target is hidden and needs to be found. Chapter 2 and 3 is mostly written using the literature from Lidbetter [9].

Finally, we extend the game on a tree to a game on a graph in chapter 4. Due to the complexity of this game, we only consider specific types of graphs with one target. In this game, movement between any two sections is limited by the structure of the graph. However, unlike the previous game it is now possible to reach a certain section of the graph in multiple ways.

The goal of the report is to investigate how the search and rescue game is played on a graph.

2 Search and Rescue Games

Assume that there is a burglar (searcher) and a wealthy man (hider) who owns three houses. The wealthy man is currently enjoying his vacation abroad leaving his three houses unoccupied. He also likes to keep all his riches in one of the three houses. To prevent his houses of being burgled, every house has an alarm system. However, because some alarm systems are old, the burglar can sometimes bypass it and rob the house. In which house should the wealthy man hide his riches and in what order should the burglar burgle the houses?

The game described above is an example of a search and rescue game where the burglar is the searcher and the wealthy man is the hider. There are many more examples and adaptations of search and rescue games that usually have a more darker tone. Think about a kidnapper hiding its victims in one or multiple locations, a mountain hiker going missing in a dangerous area or someone trapped in a burning building. Because lives are at stake, time and efficiency are of the essence. It is therefore useful to understand how these type of games are played and what strategies are optimal for both players.

2.1. Basic Rules

A search and rescue game is a game played between two players: the searcher and the hider. The game is played on a board/grid of size n, say $S \equiv \{1,2,\dots n\}$ which denotes the hiding locations. The hider needs to hide $k \in \{1,2,\dots n-1\}$ objects/targets in the set of hiding locations S. In the case that $k = 1$, the hider hides a single object in one of the n possible locations. The searcher needs to search the hiding locations in some order until all hidden objects are found. When searching a location i, there is a probability $p_i \in (0,1)$ that the search will be successful and all objects hidden in location i will be found. On the other hand, $1 - p_i$ is the probability that the search in location i will be unsuccessful. When this happens, the searcher loses the game and gains nothing (+0 units) from the hider. However, if the searcher manages to find all hidden objects, she wins the game and gains +1 unit from the hider.

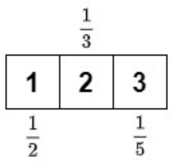


Figure 2.1: Search and Rescue Game with n = 3

2.2. Strategies and Pay-off

The hider needs to hide k objects in the set of hiding locations S. It is obvious that hiding more than one object in the same hiding location is non-optimal (and dominated by other strategies). The searcher needs to search through all hiding locations in some order. Every order in which the hiding locations can be searched is a permutation of the set S.

Definition 2.2.1 (**Hider's Pure Strategy**). *A pure strategy h for the hider is a subset $A \subseteq S$ of size k, i.e.*

$$h \in S^{(k)} \equiv \{A \subseteq S : |A| = k\}.$$

Definition 2.2.2 (**Seacher's Pure Strategy**). *A pure strategy σ for the searcher is a permutation*

$$\sigma : S \to S$$

of S such that $\sigma(i)$ is the i-th location searched in the permutation σ for $i = 1, \ldots, n$.

If the hider and searcher choose their pure strategies h and σ respectively, the pay-off for the searcher is defined as the probability of successfully finding all hidden objects.

Definition 2.2.3 (**Total success probability**). *The probability that the searcher searches a set $A \subseteq S$ successfully is given by*

$$f(A) = \prod_{i \in A} p_i. \tag{2.1}$$

Definition 2.2.4. *The set*

$$S_i^\sigma = \bigcup \{j \in S : \sigma^{-1}(j) \leq i\} \tag{2.2}$$

is the set containing the first i locations searched by some permutation σ.

Definition 2.2.5 (**Pay-off**). *The pay-off for the searcher when she uses σ and the hider uses h is given by*

$$P(h, \sigma) = f(S_i^\sigma) = \prod_{i \in S_i^\sigma} p_i \tag{2.3}$$

where i is minimal such that $h \subseteq S_i^\sigma$. Similarly, the pay-off for the hider is

$$-P(h, \sigma) = -\prod_{i \in S_i^\sigma} p_i. \tag{2.4}$$

Note that the pay-off for the hider is derived from the fact that he gains $+0$ when he wins and -1 when he loses. This makes the game a finite-zero-sum game which has several useful properties that will be used later on. Arguably, the pay-off for the hider could also be defined as

$$1 - P(h, \sigma) = 1 - \prod_{i \in S_i^\sigma} p_i, \tag{2.5}$$

which is the total winning probability of the hider. This makes the game not a finite-zero-sum game but a constant-sum game. However, subtracting a constant from the pay-off of a constant-sum game does not change the optimal strategies. Hence, we can subtract 1 from the pay-off in 2.5 to obtain 2.4 which is the pay-off of the finite-zero-sum game. Therefore, both games have the same optimal strategies. By defining the hider's pay-off as in 2.4, the properties of a finite-zero-sum game can directly be used.

Definition 2.2.6 (**Search and Rescue Game**). *A search and rescue game with n hiding locations and k objects is a game in which the hider chooses a hiding strategy h as defined in definition 2.2.1, the searcher chooses a searching strategy σ as defined in definition 2.2.2 and the pay-off for the searcher and hider is given by $P(h, \sigma)$ and $-P(h, \sigma)$ which are defined in definition 2.3 and 2.4 respectively. We denote this game by Γ_f where f is the function defined in 2.1.*

Example 2.2.1. *Consider the search and rescue game depicted in figure 2.1 where $k = 1$. The number of search locations is $n = 3$, and therefore the hiding locations are $S = \{1, 2, 3\}$. Finally, the success probabilities of searching the hiding locations is given by $\underline{p} = (p_1, p_2, p_3) = \left(\frac{1}{2}, \frac{1}{3}, \frac{1}{5}\right)$.*

The hider has three pure strategies, he can hide the object in either location 1, 2 or 3. The pure strategies of the searcher are all permutations σ of the set $S = \{1, 2, 3\}$, e.g. $\sigma_1 : 123 \to 123$, $\sigma_2 : 123 \to 132$, etc. The search strategy σ_2 tells the searcher to search the hiding locations in the order $(1, 3, 2)$ which we will simply abbreviate by $\sigma_2 = 132$. It is obvious that the search stops midway when the object has been found.

Assume the hider plays $h = 3$ (i.e. he hides the object in location 3) and the searcher plays σ_2. In this case, the searcher will first search location 1 which will be searched successfully with probability $p_1 = \frac{1}{2}$. Because no object is hidden in location 1, the search continues. The next location that will be searched in σ_2 is location

3 which will be searched successfully with probability $\frac{1}{5}$. Combining both probabilities, the probability that both locations 1 and 3 will be searched successfully is given by $p_1 \cdot p_3$. Since the hider hid the object in location 3, the search has come to an end. The pay-off for the strategies $h = 3$ and σ_2 is then given by

$$P(h,s) = f(S_i^{\sigma_2}) = f(S_2^{\sigma_2}) = p_1 p_3 = \frac{1}{10}$$

where $i = 2$ is minimal such that $h \subseteq S_i^{\sigma_1}$ and

$$S_2^{\sigma_2} = \bigcup \{j \in S : \sigma^{-1}(j) \le 2\} = \{1\} \cup \{3\} = \{1,3\}.$$

However, if the searcher would have started her search in location 3, the pay-off would have been $\frac{1}{5}$. Hence, the hiding strategy $h = 3$ can guarantee that the pay-off is at most $\frac{1}{5}$. With the help of mixed strategies, i.e. a probability distribution over the pure strategies, the hider can further guarantee that the pay-off and also the value of the game are even lower.

The searcher seeks to maximize the pay-off, whereas the hider seeks to minimise it. The searcher has exactly six pure strategies which is equal to the total number of permutation of the set S. On the other hand, the hider has exactly three pure strategies which are all hiding locations. Therefore, the game can be seen as a simple 6×3-matrix game. The corresponding pay-off matrix is given as follows

$$\begin{array}{cc} & \text{H} \\ \text{S} & \begin{array}{c|ccc} & 1 & 2 & 3 \\ \hline (123) & \frac{1}{2} & \frac{1}{6} & \frac{1}{30} \\ (132) & \frac{1}{2} & \frac{1}{30} & \frac{1}{10} \\ (213) & \frac{1}{6} & \frac{1}{3} & \frac{1}{30} \\ (231) & \frac{1}{30} & \frac{1}{3} & \frac{1}{15} \\ (312) & \frac{1}{10} & \frac{1}{30} & \frac{1}{5} \\ (321) & \frac{1}{30} & \frac{1}{15} & \frac{1}{5} \end{array} \end{array}. \tag{2.6}$$

Since the matrix game is a finite-zero-sum game by definition 2.2.5 of the pay-off, the game has a value and optimial strategies exist by the Min-Max Theorem of von Neumann [8] page 13. Since the matrix does not have a saddle point, the optimal strategy is a mixed strategy. A mixed strategy for the hider is a probability distribution over all possible locations to hide the object in, i.e. S, and for the searcher it is a probability distribution over all permutations σ of the set S. The game can now easily be solved using existing matrix game solvers.

However, these matrix game solvers usually have a limitation to the size of the matrix. The number of rows of the pay-off matrix correspond to the number of pure strategies of the searcher, which is equal to the total number of permutations of S, that is $n!$. This means that the size of the matrix grows rapidly when n becomes large. Because of this, there is a need to find a more general method to compute the optimal strategies.

2.3. Optimal Strategies using Indexibility

It turns out that a more general optimal strategy can be found without solving a large matrix game, but rather with the help of term indexibility. The term indexibility comes from the optimisation branch of mathematics, more specifically in scheduling problems. Bertsimas and Nino-Mora [4] use it to describe dynamic and stochastic scheduling problems. In these type of scheduling problems, jobs come into a server that need to be processed in some order. A solution is found by assigning an index to each job and to process the job with the highest index at every stage. The general solution to the search and rescue game tries to replicate this.

We briefly sketch the idea of the new solution method, a more detailed formulation including proof will follow later on in the chapter. The idea is that every hiding location i will be assigned an index z_i. Then for a subset $A \subseteq S$ of size k, we define the index of the set A as

$$z_A = \prod_{i \in A} z_i. \tag{2.7}$$

It is optimal for the hider to play the mixed strategy $\underline{q}$ which plays a subset $A \subseteq S$ of size k with some probability proportional to z_A. We denote q_A as the probability of playing the subset A in $\underline{q}$.

The searcher's optimal strategy is similar to that of the hider. It is optimal for the searcher to play the mixed strategy $\underline{s}$ which first searches a subset $A \subseteq S$ of size k before searching the remaining locations in a uniformly random order with some probability proportional to z_A. The order in which the first k locations are searched in A is irrelevant. We denote $s(A)$ as the strategy that first searches A before searching the remaining locations in a uniformly random order, and we denote s_A as the probability of playing $s(A)$ in $\underline{s}$.

In order to define the new solution method, we first need to define the indices z_i. These indices follow directly from the definition of indexibility.

Definition 2.3.1. *For $i \in S$ and $A \subseteq S$, define*

$$f_A(i) = f(A \cup \{i\}) - f(A) = f(A \cup i) - f(A). \tag{2.8}$$

Definition 2.3.2 (**Indexible**). *Let $f : 2^S \to \mathbb{R}$ be a positive function that satisfies $f(A) < f(B)$ when $B \subset A$. Then f is z-indexable if there exists a $\underline{z} = (z_1, ..., z_n) \in \mathbb{R}^n$ with $z_k > 0$ for all k such that*

$$\frac{f_{A \cup j}(i)}{f_{A \cup i}(j)} = \frac{z_i}{z_j}, \quad i, j \notin A. \tag{2.9}$$

where z_k is the index of location k.

Note that the total success probability function f defined in 2.1 is indeed a positive function that satisfies $f(A) < f(B)$ when $B \subset A$. The second part follows from

$$\frac{f_{A \cup j}(i)}{f_{A \cup i}(j)} = \frac{f(A \cup i \cup j) - f(A \cup j)}{f(A \cup i \cup j) - f(A \cup i)} = \frac{p_i p_j \prod_{s \in A} p_s - p_j \prod_{s \in A} p_s}{p_i p_j \prod_{s \in A} p_s - p_i \prod_{s \in A} p_s} = \frac{p_j(1-p_i)}{p_i(1-p_j)} = \frac{z_i}{z_j}.$$

Hence, the index of location i is given by

$$z_i = \frac{1-p_i}{p_i} \tag{2.10}$$

which is equal to the inverse of the odds. Furthermore, since $p_i \in (0,1)$ the indices also satisfy $z_i > 0$ for all i. Therefore, the total success probability function f of the search and rescue game Γ_f is indeed z-indexable. As a result, every hiding location will be assigned a value given by the index in 2.10. This index will be used to define the probabilities q_A and s_A.

Before defining the probabilities q_A and s_A and proving the optimality of the strategies $\underline{q}$ and $\underline{s}$, we first state and prove two important properties of the game Γ_f. The following lemmas tell us that in the game Γ_f where we restrict the searcher to strategies of the form $s(A)$, the game becomes symmetrical. Let us first consider the case $k = 1$, i.e. the hider hides a single object.

Lemma 1. *Let $S = \{1, ..., n\}$, $k = 1$ and consider the search and rescue game Γ_f for an arbitrary set function $f : 2^S \to \mathbb{R}$. Then for any two hiding locations $i, j \in S$ we have $P_f\big(i, s(j)\big) = P_f\big(j, s(i)\big)$.*

Proof. First notice that since $s(j)$ and $s(i)$ are mixed strategies, $P_f\big(i, s(j)\big)$ and $P_f\big(j, s(i)\big)$ are expected pay-offs. In the case where $i = j$, the lemma trivially holds. Therefore, consider the case in which $i \neq j$. Let $Z = S \setminus \{i, j\}$ and denote $\mathcal{P}(Z)$ as the power set of Z.

Assume that the hider and searcher play the strategies i and $s(j)$ respectively. The strategy $s(j)$ always starts with searching location j. After searching location j, the remaining locations $X = S \setminus \{j\}$ are searched in a uniformly random order until location i has been searched. Hence, note that location i and j must always be searched. Of course, the search always starts in j and ends in i.

Since $X = S \setminus \{j\} = Z \cup \{i\}$ is searched in a uniformly random order until location i has been searched, there is a possibility that a random subset $R \subseteq Z$ will be searched before i. To further clarify, the uniformly random search of X can be thought of as uniformly sampling from X without replacements until location i has been sampled. The subset R would then be all elements of Z sampled before i. This random subset R can be any subset of Z which is equivalent to saying that $R \in \mathcal{P}(Z)$. Let $Z_1, ..., Z_k$ be all the elements of $\mathcal{P}(Z)$ such that $\mathcal{P}(Z) = \{Z_1, ..., Z_k\}$, and let $p(Z_l)$ be the probability that the random set R is given by Z_l. Then it follows that with probability $p(Z_l)$, the set Z_l will be searched after j and before i. Therefore, with probability $p(Z_l)$ the pay-off is equal to

$$f(j \cup Z_l \cup i)$$

for all $l = 1, ..., k$. Hence, the expected pay-off is given by

$$P_f\big(i, s(j)\big) = \sum_{l=1}^{k} f(j \cup Z_l \cup i) \cdot p(z_l).$$

Now assume the hider and searcher play the strategies j and $s(i)$ respectively. Location i will be searched first, after which the remaining locations $Y = S \setminus \{i\}$ are searched in a uniformly random order until j has been searched. Note that similar as before, location i and j must always be searched.

Now $Y = S \setminus \{i\} = Z \cup \{j\}$ is searched in a uniformly random order until j has been searched. Similar as before, there is a possibility that a random subset $R \subseteq Z$ will be searched before i as a result of the uniformly random search of Y. Because the random subset R can be any subset of Z, it is equivalent to say that $R \in \mathcal{P}(Z)$. Let $Z_1, ..., Z_k$ and $p(Z_l)$ be defined as before. Then it follows that with probability $p(Z_l)$, the set Z_l will be searched after i and before j. Therefore, with probability $p(Z_l)$ the pay-off is equal to

$$f(i \cup Z_l \cup j)$$

for all $l = 1, ..., k$. Hence, the expected pay-off is given by

$$\begin{aligned} P_f\big(j, s(i)\big) &= \sum_{l=1}^{k} f(i \cup Z_l \cup j) \cdot p(Z_l) \\ &= \sum_{l=1}^{k} f(j \cup Z_l \cup i) \cdot p(Z_l) = P_f\big(i, s(j)\big). \end{aligned}$$

□

Lemma 1 states the simplest case of a more general lemma. The lemma tells us that for $k = 1$, the strategy pair i and $s(j)$ yield the same expected pay-off as the pair j and $s(i)$. A similar lemma can be stated and proven for general k where the proof is almost identical.

Lemma 2. *Let $S = \{1, ..., n\}$, k be a positive integer and consider the search and rescue game Γ_f for an arbitrary set function $f : 2^S \to \mathbb{R}$. Then for any $A, B \in S^{(k)}$ we have $P_f\big(A, s(B)\big) = P_f\big(B, s(A)\big)$.*

Proof. First notice that since $s(B)$ and $s(A)$ are mixed strategies, $P_f\big(A, s(B)\big)$ and $P_f\big(B, s(A)\big)$ are expected pay-offs. In the trivial case where $A = B$, the lemma follows immediately. Therefore, consider the non-trivial case in which $A \neq B$. Let $Z = S \setminus (A \cup B)$ be the complement of the union of A and B, and denote $\mathcal{P}(Z)$ as the power set of Z.

Assume the hider and searcher use the strategies A and $s(B)$ respectively. The strategy $s(B)$ will first search the set B after which it will search the remaining locations $X = S \setminus B$ in a uniformly random order until all locations of A have been searched. Hence, note that the sets A and B must always be searched. This follows from the fact that the search always starts with searching all locations of B and ends when all locations of A are searched.

Since $X = S \setminus B = Z \cup A$ is searched in a uniformly random order until all locations of A have been searched, there is a possibility that a random subset $R \subseteq Z$ will be searched before all locations of A have been searched. Because R can be any subset of Z, it is equivalent to saying that $R \in \mathcal{P}(Z)$. Let $Z_1, ..., Z_k$ be all the elements of $\mathcal{P}(Z)$ such that $\mathcal{P}(Z) = \{Z_1, ..., Z_k\}$, and let $p(Z_l)$ be the probability that the random set R is given by Z_l. Then it follows that with probability $p(Z_l)$, the set Z_l will be searched in some order after A and before the last element of B. Therefore, with probability $p(Z_l)$ the pay-off is equal to

$$f(B \cup Z_l \cup A)$$

for all $l = 1, ..., k$. Hence, the expected pay-off is given by

$$P_f\big(A, s(B)\big) = \sum_{l=1}^{k} f(B \cup Z_l \cup A) \cdot p(Z_l).$$

Now assume the hider and searcher play B and $s(A)$ respectively. All locations in A will be searched first, after which the remaining locations $Y = S \setminus A$ are searched in a uniformly random order until all locations of B have been searched. Once again, the sets A and B must always be searched.

Since $Y = S \setminus A = Z \cup B$ is searched in a uniformly random order until all locations of B have been searched, there is a possibility that a random subset $R \subseteq Z$ will be searched before all locations of B have been searched. Because R can be any subset of Z, it is equivalent to say that $R \in \mathcal{P}(Z)$. Therefore, let $Z_1, ..., Z_k$ and $p(Z_l)$ be defined as before. Then it follows that with probability $p(Z_l)$, the set Z_l will be searched in some order after A and before the last element of B. Therefore, with probability $p(Z_l)$ the pay-off is equal to

$$f(A \cup Z_l \cup B)$$

for all $l = 1, ..., k$. Hence, the expected pay-off is given by

$$\begin{aligned} P_f\big(B, s(A)\big) &= \sum_{l=1}^{k} f(A \cup Z_l \cup B) \cdot p(Z_l) \\ &= \sum_{l=1}^{k} f(B \cup Z_l \cup A) \cdot p(Z_l) = P_f(A, s(B)). \end{aligned}$$

□

Note that for every pure strategy A of the hider there is exactly one matching mixed strategy $s(A)$ for the searcher and vice versa. And so by restricting the searcher to strategies of the form $s(A)$, both players have the same number of strategies. As a result of lemma 2, the strategy pairs $\big(A, s(B)\big)$ and $\big(B, s(A)\big)$ have the same expected pay-off for all $A, B \in S^{(k)}$. Therefore, the game Γ_f can be made symmetric by restricting the searcher to strategies of the form $s(A)$. Moreover, the restricted game is then a finite-zero-sum symmetric matrix game.

To show that the restricted game is indeed symmetric, consider the game in example 2.2.1 with pay-off matrix given in 2.6. Notice that the first and second strategy of the searcher can be combined into the strategy $s(1)$ by playing both strategies with equal probability. Similarly, the third and fourth, and fifth and sixth strategy can be combined into the strategies $s(2)$ and $s(3)$ respectively. By restricting the searcher to the strategies $s(1)$, $s(2)$ and $s(3)$, the pay-off matrix becomes

$$\begin{array}{ccc} & & \text{H} \\ & & \begin{array}{ccc} 1 & 2 & 3 \end{array} \\ \text{S} & \begin{array}{c} s_1 \\ s_2 \\ s_3 \end{array} & \begin{pmatrix} \frac{1}{2} & \frac{1}{10} & \frac{1}{15} \\ \frac{1}{10} & \frac{1}{3} & \frac{1}{20} \\ \frac{1}{15} & \frac{1}{20} & \frac{1}{5} \end{pmatrix} \end{array} \tag{2.11}$$

which is indeed symmetrical. The second lemma continues from the restricted symmetrical game and allows us to define an optimal strategy for both players. It tells us that if a player has a strategy that makes the opponent indifferent, then this strategy is optimal for both players.

Lemma 3. *Consider a zero-sum game with symmetric pay-off matrix in which player one has a mixed strategy $\underline{x}$ which makes player two indifferent between all her pure strategies. Then the strategy $\underline{x}$ is optimal for both players.*

Proof. Let $\underline{x}$ be the mixed strategy for player one that makes player two indifferent, $\underline{y}$ be an arbitrary pure strategy for player two and A the symmetric pay-off matrix. Since $\underline{x}$ makes player two indifferent, the pay-off is given by

$$P(\underline{x}, \underline{y}) = \underline{x}^T A \underline{y} = V$$

for all pure strategies $\underline{y}$ of player two. Now if player two plays the strategy $\underline{x}$ and player one plays any pure strategy $\underline{y}$, then

$$\begin{aligned} P(\underline{y}, \underline{x}) = \underline{y}^T A \underline{x} &= \underline{y}^T A^T \underline{x} \\ &= \big(\underline{x}^T A \underline{y}\big)^T \\ &= V^T = V = P(\underline{x}, \underline{y}). \end{aligned}$$

Here we use that $A = A^T$ since A is symmetric. Since $\underline{y}$ is taken arbitrarily for player one, player two can make player one indifferent between all his pure strategies. Hence, player one can guarantee a pay-off of at least V with $\underline{x}$ whereas player two can hold the pay-off down to at most V with $\underline{x}$. Therefore, the strategy $\underline{x}$ is optimal for both players. □

The original game Γ_f can now be solved using a more general method which does not involve solving a large matrix game. The following theorem states a general solution to the game Γ_f.

Theorem 1. *Consider the search and rescue game Γ_f and suppose f is z-indexable. Then it is optimal for the hider to use the mixed strategy $\underline{q}$ which chooses a set $A \in S^{(k)}$ with probability q_A given by*

$$q_A = \frac{\prod_{i \in A} z_i}{T_k(S)}, \quad \forall A \in S^{(k)} \tag{2.12}$$

with

$$T_k(S) = \sum_{B \in S^{(k)}} \prod_{i \in B} z_i. \tag{2.13}$$

It is optimal for the searcher to use the strategy $\underline{s}$ that plays the strategy $s(A)$ with probability proportional to q_A.

The term $T_k(S)$ is the product of all z-indices in a set B summed over all possible sets $B \in S^{(k)}$. It can be thought of as the combined total index of all possible hiding strategies. Let $i \notin A \subset S$ be a hiding location in S. The fraction

$$\frac{T_{k-1}(A) z_i}{T_k(S)}$$

denotes the probability that $k-1$ objects are hidden in a set $A \subset S$ with the last object being hidden in location i.

Proof. Assume the hider plays the strategy $\underline{q}$ as defined in the theorem. We will show that this strategy makes the searcher indifferent between all her pure strategies. Let σ_1 be an arbitrary permutation of S and let σ_2 be the permutation σ_1 in which two adjacent elements are transposed. Note that any permutation σ can be obtained from σ_1 by iteratively transposing two adjacent elements. The idea is to show that by transposing two adjacent elements of a permutation, the expected pay-off remains unchanged. As a result, all permutations must have the same expected pay-off. To that end, it is sufficient to prove that σ_1 and σ_2 have the same expected pay-off.

Suppose that the element j comes immediately after i in σ_1 and that the elements i and j are transposed in σ_2. Clearly, if element j in σ_1 comes in position k or earlier, then transposing i and j leaves the expected pay-off unchanged. Of course, at the bare minimum k locations need to be searched and it does not matter in which order these k locations will be searched. Hence, suppose this is not the case and let A be the set of locations searched before i in σ_1 (or before j in σ_2). Then the difference in expected pay-off is given by

$$\begin{aligned} P(\underline{q},\sigma_1) - P(\underline{q},\sigma_2) &= \left(\frac{T_{k-1}(A)z_i}{T_k(S)} f(A \cup i) + \frac{T_{k-1}(A \cup i)z_j}{T_k(S)} f(A \cup i \cup j)\right) \\ &\quad - \left(\frac{T_{k-1}(A)z_j}{T_k(S)} f(A \cup j) + \frac{T_{k-1}(A \cup j)z_i}{T_k(S)} f(A \cup i \cup j)\right) \\ &= \frac{T_{k-1}(A)z_i}{T_k(S)} f(A \cup i) - \frac{T_{k-1}(A)z_j}{T_k(S)} f(A \cup j) \\ &\quad + \frac{T_{k-1}(A \cup i)z_j - T_{k-1}(A \cup j)z_i}{T_k(S)} f(A \cup i \cup j) \end{aligned} \tag{2.14a}$$

where f is the total success probability as defined in definition 2.2.3.

Claim. *The following equality holds*

$$T_{k-1}(A \cup i)z_j - T_{k-1}(A \cup j)z_i = z_j T_{k-1}(A) - z_i T_{k-1}(A). \tag{2.15}$$

Proof of claim. *See appendix A.2.1.*

As a result of the claim and by using 2.8 and 2.9, it follows that

$$\begin{aligned}
P(\underline{q},\sigma_1) - P(\underline{q},\sigma_2) &= \frac{T_{k-1}(A)z_i}{T_k(S)} f(A\cup i) - \frac{T_{k-1}(A)z_j}{T_k(S)} f(A\cup j) + \frac{T_{k-1}(A)z_j}{T_k(S)} f(A\cup i\cup j) \\
&\quad - \frac{T_{k-1}(A)z_i}{T_k(S)} f(A\cup i\cup j) \\
&= \frac{T_{k-1}(A)z_j}{T_k(S)} f_{A\cup j}(i) - \frac{T_{k-1}(A)z_i}{T_k(S)} f_{A\cup i}(j) \\
&= \frac{T_{k-1}(A)}{T_k(S)} \big(z_j f_{A\cup j}(i) - z_i f_{A\cup i}(j)\big) \\
&= \frac{T_{k-1}(A)}{T_k(S)} \Big(z_j f_{A\cup j}(i) - \frac{f_{A\cup j}(i) z_j}{f_{A\cup i}(j)} f_{A\cup i}(j)\Big) = 0.
\end{aligned}$$

Thus, the strategy $\underline{q}$ makes the searcher indifferent between all her pure strategies and $V = P(\underline{q},\sigma_1)$ is an upper bound to the value of the game. Now if we restrict the searcher to strategies of the form $s(A)$, the restricted game is symmetric by lemma 2 and the strategies $\underline{q}$ and $\underline{s}$ are optimal by lemma 3. The value of the restricted game is therefore equal to V, which is also a lower bound for the value of the unrestricted game. Hence, the value of the unrestricted game is V and the strategies $\underline{q}$ and $\underline{s}$ are indeed optimal. □

Theorem 1 allows us define more general optimal strategies for the search and rescue game Γ_f. Furthermore, it has been shown that the total success probability function f defined in 2.1 is indexible and that the index is given by 2.10. As a result, the probability q_A (and also s_A) in theorem 1 can be further simplified.

Remark 1. *Define the set function*

$$o(A) = \prod_{i\in A} \frac{1-p_i}{p_i}, \quad \forall A \in S^{(k)}. \tag{2.17}$$

Then the probability q_A in theorem 1 is proportional to $o(A)$ and given by

$$q_A = \Big(\sum_{B\in S^{(k)}} o(B)\Big)^{-1} o(A), \quad \forall A\in S^{(k)}. \tag{2.18}$$

For $k=1$, the set function and probability simplify to

$$o(i) = \frac{1-p_i}{p_i} \quad \textit{and} \quad q_i = \Big(\sum_{j\in S} o(j)\Big)^{-1} o(i), \quad \forall i \in S. \tag{2.19}$$

The value V of the game for $k=1$ follows from the fact that the hiding strategy $\underline{q}$ makes the searcher indifferent. Hence, let σ be the permutation that searches the hiding locations in increasing order from 1 to n. Then the value of the game is

$$\begin{aligned}
V = P(\underline{q},\sigma) &= \sum_{i=1}^{n} q_i \prod_{j\le i} p_j \\
&= \Big(\sum_{j\in S} \frac{1-p_j}{p_j}\Big)^{-1} \sum_{i=1}^{n} \frac{1-p_i}{p_i} \prod_{j\le i} p_j = \Big(\sum_{j\in S} \frac{1-p_j}{p_j}\Big)^{-1} \sum_{i=1}^{n} (1-p_i) \prod_{j<i} p_j \qquad (2.20a) \\
&= \Big(\sum_{j\in S} \frac{1-p_j}{p_j}\Big)^{-1} \Big(1-\prod_{i\in S} p_i\Big) = \Big(\sum_{j\in S} o(j)\Big)^{-1} \Big(1-\prod_{i\in S} p_i\Big), \qquad (2.20b)
\end{aligned}$$

where we use that the second sum of 2.20a is telescopic.

Finally, example 2.2.1 can now be solved using 2.19 and it follows that $\underline{q} = \underline{s} = (\frac{1}{7}, \frac{2}{7}, \frac{4}{7})$ are the optimal strategies. Furthermore, by 2.20b the value of the game is $V = \frac{29}{210}$. Since the game is relatively small, the optimality of the strategies and the value of the game can easily be verified using the pay-off matrix M in 2.11. Solving either $\underline{q}^T M$ or $M\underline{s}$ shows that both strategies make the opponent indifferent between all his/her pure strategies. The value of the game can be verified by solving $\underline{q}^T M \underline{s}$.

2.4. Alternative Optimal Strategies for k = 1

Theorem 1 of the previous section describes how the game Γ_f with general k can be solved without solving a potentially large matrix game. In the case where $k = 1$, it is optimal for the hider to play a mixed strategy that mixes all of his n pure strategies, namely $h = 1, 2, \ldots, n$. The searcher on the other hand is restricted to the strategies $s(1), \ldots, s(n)$ which are mixed strategies.

Recall that the strategy $s(i)$ searches location i first before randomizing between all other locations. Therefore, every strategy $s(i)$ is a mix of several other pure strategies. It is equivalent to the mixed strategy in which all pure strategies that start in location i are played with equal probability. Hence, every mixed strategy $s(i)$ is constructed from $(n-1)!$ pure strategies so that the optimal strategy $\underline{s}$ is a mix of $n!$ pure strategies.

In general, the solution of a matrix game are mixed strategies in which both players mix an equal number of pure strategies. In the case where $k = 1$, the hider is mixing n pure strategies, but the searcher is mixing $n!$. Naturally, the question arises whether there exist something similar for the searcher. To be more concrete, does there exist an optimal strategy for the searcher that mixes n pure strategies and can these n pure strategies always form an optimal strategy for any probabilities $\underline{p}$. To investigate the existence of such an optimal strategy, we will make use of numerical results simulated in `Matlab`. We will restrict ourselves to the case in which $k = 1$.

It is numerically possible to solve small search and rescue games by solving the corresponding matrix game. The first script that is implemented is a very simple matrix game solver, see appendix A.1.1. Given the success probabilities $\underline{p}$ of the game Γ_f, the pay-off matrix A is first constructed. Every column of the matrix A corresponds to a hiding locations whereas every row to a permutation of S. Using a standard matrix game solver, the matrix game can be solved to obtain an optimal strategy for both players and also the value of the game.

Running multiple instances of the first script where the probabilities $\underline{p}$ are taken arbitrarily, it follows that both the hider and searcher always mix an equal number of n pure strategies. A downside of the standard matrix game solver used in the script is that it only returns one of the possibly many optimal strategies. Therefore, we use a second matrix game solver from Avis et al. [2] which is available online. For arbitrary probabilities $\underline{p}$, both solvers show that there exist optimal strategies for the searcher in which she mixes an equal number of n pure strategies. Moreover, in multiple instances we found that the same n pure strategies of the searcher formed an optimal strategy. To investigate if such a statement holds for all probabilities $\underline{p}$, we need to verify whether there exist a combination of n pure strategies for the searcher such that they can always form an optimal strategy.

Let A be the pay-off matrix of size $m \times n$ where $m = n!$. The values of A are determined by the probabilities $\underline{p}$ and the hider always plays an optimal mixed strategy in which he mixes all of his n pure strategies. If the searcher always has an optimal strategy that mixes the same n pure strategies, then we can always restrict the searcher to these n pure strategies in A without changing the value of the game. The resulting pay-off matrix A' is then a submatrix of A of size $n \times n$ and should have the same value as the game with pay-off matrix A. To further illustrate, take the pay-off matrix in 2.6 for example. The hider has three pure strategies and therefore his optimal strategy is a mix of three pure strategies. Assume for the moment that we think that the first, fourth and fifth row strategies of the searcher can always form an optimal strategy. We then restrict the searcher to these pure row strategy, and the resulting submatrix becomes

$$A' = \begin{array}{c} \\ (123) \\ (231) \\ (312) \end{array} \begin{array}{c} \begin{array}{ccc} 1 & 2 & 3 \end{array} \\ \begin{pmatrix} \frac{1}{2} & \frac{1}{6} & \frac{1}{30} \\ \frac{1}{30} & \frac{1}{3} & \frac{1}{15} \\ \frac{1}{10} & \frac{1}{30} & \frac{1}{5} \end{pmatrix} \end{array}. \tag{2.21}$$

Now we solve the restricted game with pay-off matrix A' and compare the value of the restricted game with the value of the original game. If the value of both games are the same, then there exists an optimal strategy for the searcher that is a mix of the three pure row strategies. However, note that the original pay-off matrix A in 2.6 is only defined for a certain $\underline{p}$. Hence, to show that the three pure row strategies of the searcher can always form an optimal strategy, we need to iterate over all possible $\underline{p}$ and show that the value of the restricted matrix game is identical to that of the original matrix. To iterate over all possible $\underline{p}$, we define a n−dimensional grid on the interval $(0,1)^n$ with a variable number of grid points. Every grid point corresponds to a probability $\underline{p}$ and we iterate over all grid points and do the following:

1. Construct the pay-off matrix A from $\underline{p}$.

2. Construct the restricted pay-off matrix A' from A using the n predetermined pure row strategies.
3. Solve both matrix games and compare the value of both games
4. Group the grid points depending on whether the values in the previous step agree or disagree.

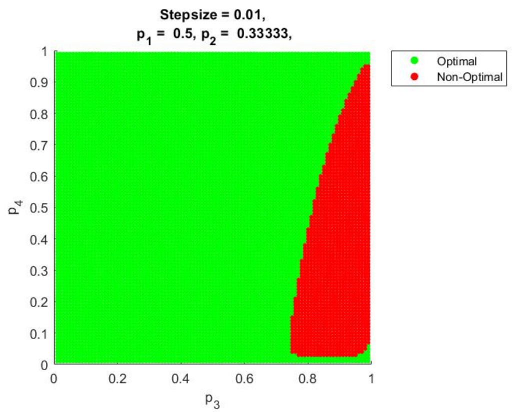
Experiment 2.4.1. *Let $n = 4$ and $k = 1$ such that $S = \{1,2,3,4\}$. The hider and searcher have $n = 4$ and $n! = 24$ pure strategies respectively. Fix $p_1 = \frac{1}{2}$, $p_2 = \frac{1}{3}$ and let p_3 and p_4 be the points defined in a two-dimensional grid on the interval $(0,1)\times(0,1)$. We consider two smaller experiments that differ in which row strategies the searcher is restricted to. In the first experiment, we restrict the searcher to the pure strategies:*

$$\sigma_1 = 1234,\ \sigma_2 = 2341,\ \sigma_3 = 3412 \quad \textit{and} \quad \sigma_4 = 4132.$$

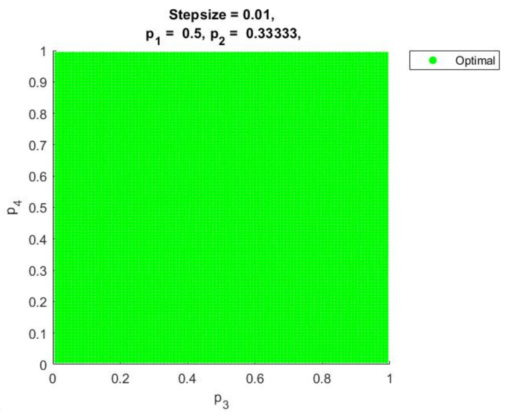
In the second experiment, we restrict the searcher to the following pure strategies:

$$\sigma'_1 = 1234,\ \sigma'_2 = 2341,\ \sigma'_3 = 3412 \quad \textit{and} \quad \sigma'_4 = 4123.$$

The results of experiment 2.4.1 are shown in figure 2.2. Figure 2.2a shows that there are probabilities p_3 and p_4 such that the value of the restricted and unrestricted game do not agree. Here, the searcher restricts herself to the pure strategies $\sigma_1, \sigma_2, \sigma_3$, and σ_4. Since the values do not agree, the pure strategies do not always form an optimal strategy. On the other hand in figure 2.2b, the searcher is restricted to the pure strategies $\sigma'_1, \sigma'_2, \sigma'_3$, and σ'_4. As shown in the figure, every grid point always has an optimal strategy consisting of the four pure strategies for the fixed p_1 and p_2. This is due to the fact that the combination of the pure strategies satisfy certain properties.



(a) Row strategies: $\sigma_1, \sigma_2, \sigma_3$, and σ_4



(b) Row strategies: $\sigma'_1, \sigma'_2, \sigma'_3$, and σ'_4

Figure 2.2: Grid points (p_3, p_4) for experiment 2.4.1

The difference between the two experiments is in the last pure strategy. Notice that in the second experiment, the pure strategies are chosen such that every hiding location is searched first exactly once. Furthermore, the pure strategies also have the property that after searching the first location, the remaining locations are searched in an increasing order. More specifically, after searching location i, location $(i \mod n) + 1$ is searched next. As a result, for every hiding location i and position j there is exactly one pure strategy that searches location i in position j. Hence, every hiding location is searched an equal amount of times on every position. It is because of this additional structure that an optimal strategy exist for all grid points and fixed p_1 and p_2.

Definition 2.4.1. *Define the set σ^+ which contains the searcher's pure strategies σ for which the following holds:*

1. *The set σ^+ has size n (which is equal to the number of pure strategies of the hider).*
2. *For every hiding location i, there exist exactly one pure strategy $\sigma \in \sigma^+$ such that σ searches location i first.*
3. *For every pure strategy $\sigma \in \sigma^+$, location $(i \mod n) + 1$ is searched after i for all locations i.*

Further analysis where p_1, p_2, p_3 and p_4 are all taken from a $4-$dimensional grid show that there always exists an optimal strategy if the searcher is restricted to the strategies σ^+. Moreover, varying the number of hiding locations n also has no impact on the existence of such an optimal strategy. What's more is that by restricting the searcher to the pure strategies in σ^+, the game does not necessarily become symmetric as opposed to the strategy $\underline{s}$ in theorem 1. An example of this is the matrix in 2.21 where the searcher is restricted to σ^+ and the matrix is clearly not symmetric. However, numerically solving the game shows that the probability of hiding the object in location $i = 1$ is equal to the probability of the searcher playing the pure strategy $\sigma \in \sigma^+$ that starts in location $i = 1$. Moreover, these probabilities are exactly the probabilities as defined in theorem 1. Hence, it looks as if there exist even stronger version of theorem 1 for $k = 1$ in which both players mix n pure strategies with similar probabilities. This brings us to the following proposition.

Proposition 2.4.1. *Consider the search and rescue game Γ_f for $k = 1$ and suppose f is z-indexable. Then it is optimal for the hider to use the mixed strategy $\underline{q}$ which plays a hiding location $i \in S$ with probability q_i given by*

$$q_i = \frac{1-p_i}{p_i}\Big(\sum_{j\in S}\frac{1-p_j}{p_j}\Big)^{-1}. \tag{2.22}$$

Let $s'(i) \in \sigma^+$ be the pure strategy that searches location i first. Then it is optimal for the searcher to use the strategy $\underline{s'}$ that plays the strategies $s'(i)$ with probability q_i.

Proof. From the proof of theorem 1, it follows that the hiding strategy $\underline{q}$ makes the searcher indifferent between all her pure strategies. Therefore, the pay-off is equal to $V = P(\underline{q},\sigma)$ for any pure strategy σ of the searcher. If the strategy $\underline{s'}$ makes the hider indifferent between all his pure strategies, then we have that the pay-off is equal to $V = P(h,\underline{s'})$ for any pure hiding strategy h. Hence, we must have that $V = P(\underline{q},\sigma) = P(h,\underline{s'})$ is the value of the game and that both $\underline{q}$ and $\underline{s'}$ are optimal (otherwise one of the two strategies does not make the opponent indifferent). Therefore, it is sufficient to show that the strategy $\underline{s'}$ makes the hider indifferent between all his pure strategies.

Assume the searcher plays the strategy $\underline{s'}$ and let i and $i+1$ be two arbitrary and consecutive pure strategies of the hider. If the expected pay-off for both pure hiding strategies is identical, it must be that all pure hiding strategies give the same expected pay-off against $\underline{s'}$. This follows from the fact that i is taken arbitrarily. Hence, it is sufficient to show that both pure strategies give the same expected pay-off. The expected pay-offs for the hiding strategies i and $i+1$ are given by

$$\begin{aligned}
P(i,\underline{s'}) &= s'_1 p_1....p_i + s'_2 p_2....p_i + ... + s'_i p_i \\
&\quad + s'_{i+1} p_{i+1}....p_n p_1....p_i + s'_{i+2} p_{i+2}....p_n p_1....p_i + ... + s_n p_n p_1....p_i \\
&= \sum_{x=1}^{i} s'_x \prod_{y=x}^{i} p_y + \sum_{x=i+1}^{n} s'_x \prod_{y=x}^{n} p_y \prod_{z=1}^{i} p_z \\
P(i+1,\underline{s'}) &= \sum_{x=1}^{i+1} s'_x \prod_{y=x}^{i+1} p_y + \sum_{x=i+2}^{n} s'_x \prod_{y=x}^{n} p_y \prod_{z=1}^{i+1} p_z.
\end{aligned}$$

The difference in expected pay-off is found by subtracting the former from the latter.

$$\begin{aligned}
P(i,\underline{s'}) - P(i+1,\underline{s'}) &= \sum_{x=1}^{i} s'_x \prod_{y=x}^{i} p_y + \sum_{x=i+1}^{n} s'_x \prod_{y=x}^{n} p_y \prod_{z=1}^{i} p_z - \sum_{x=1}^{i+1} s'_x \prod_{y=x}^{i+1} p_y \\
&\quad - \sum_{x=i+2}^{n} s'_x \prod_{y=x}^{n} p_y \prod_{z=1}^{i+1} p_z \\
&= (1-p_{i+1})\Big(\sum_{x=1}^{i} s'_x \prod_{y=x}^{i} p_y\Big) - s'_{i+1} p_{i+1} \\
&\quad + (1-p_{i+1})\Big(\sum_{x=i+2}^{n} s'_x \prod_{y=x}^{n} p_y \prod_{z=1}^{i} p_z\Big) + s'_{i+1} p_{i+1}....p_n p_1....p_i \\
&= (1-p_{i+1})\Bigg(\Big(\sum_{x=1}^{i} s'_x \prod_{y=x}^{i} p_y\Big) + \Big(\sum_{x=i+2}^{n} s'_x \prod_{y=x}^{n} p_y \prod_{z=1}^{i} p_z\Big)\Bigg) \\
&\quad + s'_{i+1} p_{i+1}(p_{i+2}....p_n p_1....p_i - 1)
\end{aligned}$$

Now we fill in the probabilities s_i' which are given in 2.22 and simplify the expected pay-off to

$$
\begin{aligned}
P(i,\underline{s}') - P(i+1,\underline{s}') &= \Big(\sum_{j=1}^{n}\frac{1-p_j}{p_j}\Big)^{-1}\cdot(1-p_{i+1})\Big(\Big(\sum_{x=1}^{i}\frac{1-p_x}{p_x}\prod_{y=x}^{i}p_y\Big)+\Big(\sum_{x=i+2}^{n}\frac{1-p_x}{p_x}\prod_{y=x}^{n}p_y\prod_{z=1}^{i}p_z\Big)\Big)\\
&\quad+\Big(\sum_{j=1}^{n}\frac{1-p_j}{p_j}\Big)^{-1}\cdot\frac{1-p_{i+1}}{p_{i+1}}p_{i+1}(p_{i+2}....p_np_1....p_i-1)\\
&=\Big(\sum_{j=1}^{n}\frac{1-p_j}{p_j}\Big)^{-1}\cdot(1-p_{i+1})\\
&\quad\cdot\Big(\Big(\sum_{x=1}^{i}(1-p_x)\prod_{y=x+1}^{i}p_y\Big)+\Big(\sum_{x=i+2}^{n}(1-p_x)\prod_{y=x+1}^{n}p_y\prod_{z=1}^{i}p_z\Big)+\Big(\prod_{y=1}^{i}p_y\prod_{y=i+2}^{n}p_y\Big)-1\Big) \qquad (2.23)
\end{aligned}
$$

Observe that 2.23 contains two telescopic terms, see appendix A.2.2. Consequently, the difference in expected pay-off simplifies to

$$
\begin{aligned}
P(i,\underline{s}') - P(i+1,\underline{s}') &= \Big(\sum_{j=1}^{n}\frac{1-p_j}{p_j}\Big)^{-1}\cdot(1-p_{i+1})\cdot \qquad (2.24)\\
&\quad\Big(1-\prod_{y=1}^{i}p_y+\Big(\prod_{y=1}^{i}p_y\Big)\Big(1-\prod_{y=i+2}^{n}p_y\Big)+\Big(\prod_{y=1}^{i}p_y\prod_{y=i+2}^{n}p_y\Big)-1\Big)\\
&=0. \qquad (2.25)
\end{aligned}
$$

□

Proposition 2.22 allows us to define optimal strategies that only mix n pure strategies for $k=1$. Furthermore, the pure strategies that are mixed are fixed and do not change when $\underline{p}$ changes. They also satisfy certain properties which are: every location i is searched first exactly once and after searching the first location, the remain locations are searched in an increasing order. We can also introduce a slightly different set of pure strategies, say σ^-, such that it has the same size as σ^+ and also contains pure strategies such that every location is searched as first exactly once. The only difference is that after searching a location i, the remaining locations are searched in a decreasing order. Such a set σ^- also satisfies the property that every location is searched in every position exactly once. Hence, it is very likely that a similar proposition holds for the pure strategies in σ^- and that the proof is found in a similar way.

2.5. Alternative Optimal Strategies for k > 1

Proposition 2.4.1 shows that for $k=1$, there exist optimal strategies for the game Γ_f such that both players mix an equal number of n pure strategies. Moreover, for every pure strategy mixed in the hider's optimal strategy there is exactly one pure strategy mixed in the searcher's optimal strategy such that both are played with equal probability q_i defined in 2.22. Unfortunately, the proposition and optimal strategies are only defined for $k=1$. Therefore, we will investigate whether a generalization of proposition 2.4.1 exists for $k>1$ in this section.

For the sake of our analysis, we will restrict ourselves to the case where $n=5$ and $k=2$ so that the hiding locations are given by the set $S=\{1,2,3,4,5\}$. The hider hides two objects in any pair of hiding locations of S independent of their order, i.e. the pure hiding strategies

$$h=\{1,2\}=12 \quad\text{and}\quad h=\{2,1\}=21$$

are identical. Similarly, notice that the pure searching strategies

$$\sigma_1=12345 \quad\text{and}\quad \sigma_2=21345$$

are also identical. Since the hider hides two objects, it does not matter in what order the first two hiding locations are searched. As a result, the hider and searcher have

$$\binom{n}{k}=\frac{n!}{k!(n-k)!}=\frac{n(n-1)}{2}=10 \quad\text{and}\quad \frac{n!}{k!}=60$$

pure strategies respectively. Numerical results show that by solving the matrix game for abitrary values of $\underline{p}$, there are optimal strategies for the searcher that mix exactly ten pure strategies. Hence we ask ourselves if we can restrict the searcher to ten pure strategies $\sigma_1, ..., \sigma_{10}$ such that for any success probability $\underline{p} = (p_1, ..., p_5)$ there exist a mix of these ten pure strategies that is optimal?

In the case $k = 1$, we defined a set of pure strategies σ^+ as in definition 2.4.1. This pure strategy set satisfied certain properties and as a result was able to always form an optimal strategy. For $k > 1$ we define to define a slightly different set, say σ^*. Since the hider has $\binom{n}{k} = 10$ pure strategies, the set σ^* needs to contain the same number of pure strategies. Furthermore, because the hider's pure strategies consist of pairs of hiding locations, we want every pair of hiding locations to be searched first exactly once. Hence, the pure strategies in σ^* must be chosen such that for every pair of hiding locations i and j there is exactly one pure strategy $\sigma \in \sigma^*$ such that σ searches i and j first (in any order). These are essentially the first and second property of definition 2.4.1 for $k > 1$.

Finally, the third property is where we try out and test different pure strategy sets. We define multiple sets of σ^* that each have a different third property. For each of these sets, we check whether the pure strategies in this set can form an optimal strategy for general $\underline{p}$. This has been done in the following experiment that is similar to experiment 2.4.1.

Experiment 2.5.1. *Let $n = 5$ and $k = 2$ such that $S = \{1,2,3,4,5\}$. Fix $p_1 = \frac{1}{2}$, $p_2 = \frac{1}{3}$, $p_3 = \frac{1}{4}$ and let p_4 and p_5 be the points in a two-dimensional grid on the interval $(0,1) \times (0,1)$. We consider the following four smaller experiments in which we restrict the searcher to ten row strategies such that for every pair of hiding locations i and j with i ≠ j there is exactly one pure strategies that searches i and j first. The experiments differ from one another by the following final property:*

Exp. 1 After searching the first two locations, the remaining locations are searched in an increasing order starting from the maximum of the first and second location searched, i.e. if location 1 *and* 3 *are searched first, then the order of the remaining locations is* $\sigma = 13452$.

Exp. 2 After searching the first two locations, the remaining locations are searched in an decreasing order starting from the maximum of the first and second location searched, i.e. if location 1 *and* 3 *are searched first, then the order of the remaining locations is* $\sigma = 13254$.

Exp. 3 After searching the first two locations, the remaining locations are searched in an increasing order starting from the lowest of the remaining locations, i.e. if location 1 *and* 3 *are searched first, then the order of the remaining locations is* $\sigma = 13245$.

Exp. 4 For every pair of hiding locations i and j, if a pure strategy σ searches i and j in position x and y respectively then there is no other pure strategy that does the same nor is there a pure strategy that searches i and j in position y and x respectively, i.e. if σ searches location 1 *and* 2 *as first and third respectively, then there is no other pure strategy that does the same nor is there a pure strategy that searches location* 2 *and* 1 *as first and third respectively.*

Table 2.1 shows four pure strategy sets of size ten to which the searcher is restricted to in experiment 2.5.1. The first three pure strategy sets focus on searching the locations in either an increasing or decreasing order similar as in experiment 2.4.1. However, in doing so the strategy sets become less symmetric in the sense that a location i might be searched in position j more often than location $k \neq i$ in j. This is not the case in the fourth strategy set which is inspired by the Kirkman's schoolgirl problem [3].

Table 2.1: Reduced Row Strategies for experiment 2.5.1

	σ_1	σ_2	σ_3	σ_4	σ_5	σ_6	σ_7	σ_8	σ_9	σ_{10}
Exp. 1	12345	13452	14523	15234	23451	24513	25134	34512	35124	45123
Exp. 2	12543	13254	14325	15432	23154	24315	25431	34215	35421	45321
Exp. 3	12345	13245	14235	15234	23145	24135	25134	34125	35124	45123
Exp. 4	12345	23451	34512	45123	51234	24135	41352	13524	35241	52413

The Kirkman's schoolgirl problem is a well-known problem in the optimization field of mathematics and is stated as follows: *Fifteen girls have to walk in five rows of three to school for seven days. Arrange them daily* *such that no two girls walk abreast twice.* It is a very interesting problem which I previously had not heard of. A brute force approach would result in checking $(15!)^7$ number of combinations which is enormous. A solution to this problem has since been found by either combining backtracking and constraint checking [11] or by constructing a Kirkman triple system [5]. Figure 2.3 shows one of the possible solutions to the schoolgirl problem. Every numbered small circle represents a schoolgirl which is connected to a coloured line. The coloured lines connect exactly three girls which represents a grouping of three girls. By rotating the girls in the outer an inner circle (in the same direction), the grouping of the next days can be found. Although the exact solution of the Kirkman's schoolgirl problem is not that important for us, it is still an interesting problem. See Barnier and Brisset [3], Eric W. [5] and Toal [11] for more about the Kirkman's schoolgirl problem.

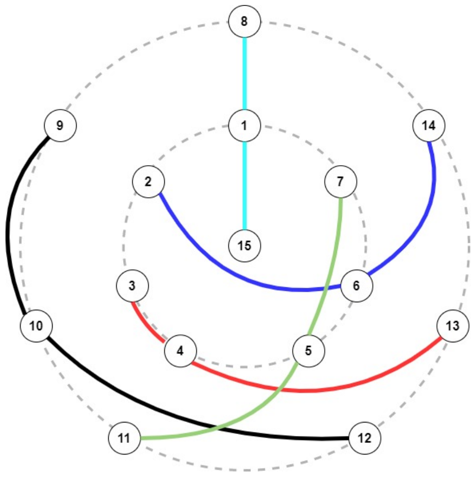
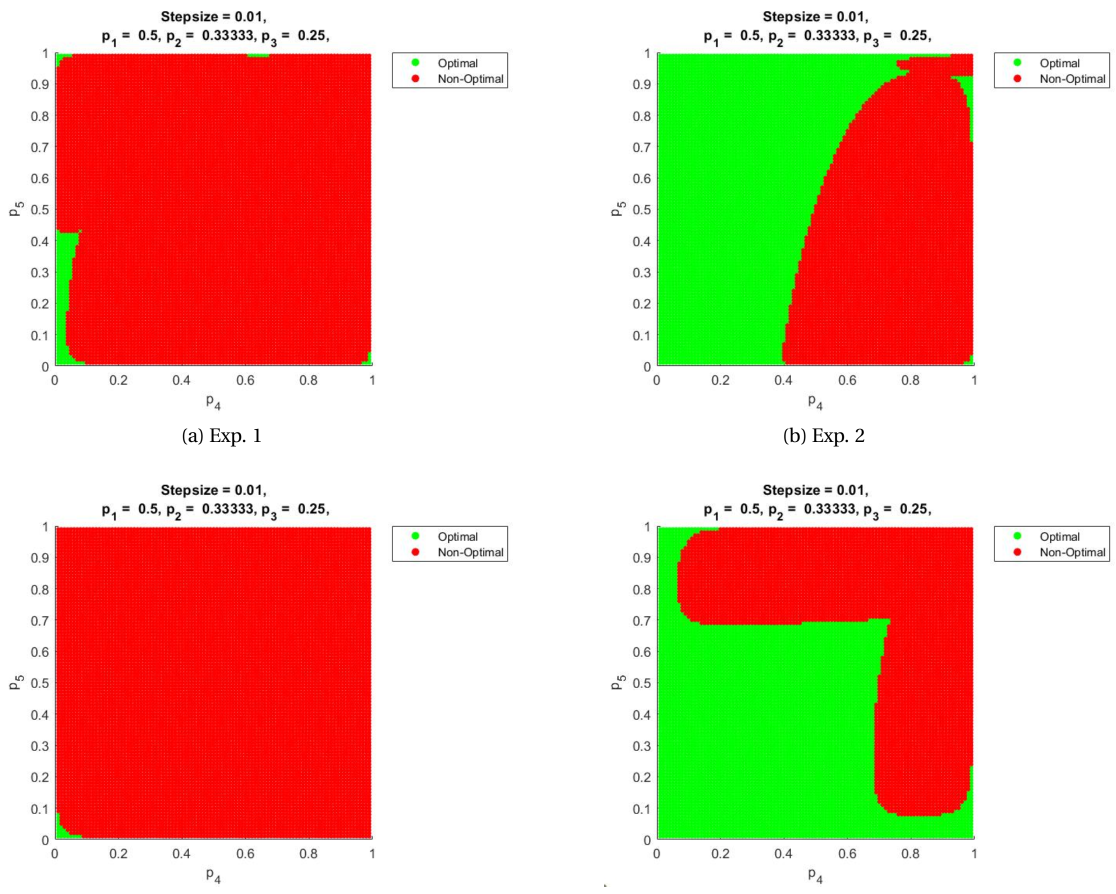


Figure 2.3: Kirkman's schoolgirl solution redrawn from Toal [11]

Our problem is comparable to the Kirkman's schoolgirl problem, we have $n = 5$ girls and $\frac{n(n-1)}{2} = 10$ days. The goal is to order them such that no two girls walk on the same pair of positions twice. The fourth pure strategy set in table 2.1 is found by shifting a permutation, i.e. moving all elements of the permutation to the left and moving the first element to the end. The first five pure strategies are found by shifting the trivial permutation four times. The last five permutations are found by element-wise multiplying the trivial permutation by two and then working modulo five (where 0 corresponds to 5). This permutation can also be shifted four times, which gives us exactly ten pure strategies. Notice that this strategy set also has the property that for every two hiding locations i and j and position k, i and j are searched an equal amount of times in position j. This extra property was also satisfied by the strategy set defined in definition 2.4.1 for the case $k = 1$.

The results of experiment 2.5.1 can be found in figure 2.4. The figure shows that for every experiment and pure strategy set there exist a value $\underline{p}$ such that the value of the restricted and unrestricted game are not equal. Interestingly, figure 2.4c and 2.4d show a symmetric result in $y = x$ whereas figure 2.4a and 2.4b do not. We expected figure 2.4d to show a symmetric result, because it has more symmetrical properties. The remaining results being symmetric or asymmetric might be due to the fixed p_1, p_2 and p_3. By fixing these three values, we focus on a very small subset of $(0,1)^5$ in which certain solution can be symmetric or asymmetric.

Because for every pure strategy set there exist a $\underline{p}$ such that the value of the restricted and unrestricted game are not equal, the four pure strategy sets can not form an optimal strategy for general $\underline{p}$ and proposition 2.22 does not hold for $k > 1$. This suggests that an optimal strategy for the searcher consisting of the same $\binom{n}{k} = 10$ pure strategies might not exist after all. To look deeper into the last statement, we need to iterate over all possible combinations of ten row strategies and verify whether or not they can form an optimal strategy for general $\underline{p}$. The hider has 60 pure strategies of which we need to take a subset of size 10, hence there are a total of $\binom{60}{10} \approx 7.5\text{E}10$ possible subsets which is impossible to run with our current hardware. Therefore, we will investigate the case where $n = 4$ and $k = 2$ in the following experiment.



(a) Exp. 1

(b) Exp. 2

(c) Exp. 3

(d) Exp. 4

Figure 2.4: Grid points (p_4, p_5) for experiment 2.5.1

Experiment 2.5.2. *Let $n = 4$ and $k = 2$ such that the hider and searcher have six and twelve pure strategies respectively. We restrict the searcher to a set of six pure strategies. Since the searcher has twelve pure strategies, there are a total of $\binom{12}{6} = 924$ possible combinations. Let p_1, p_2, p_3 and p_4 be points taken from a four-dimensional grid on the interval $(0,1)^4$. We iterative check every combination of pure strategies to see if it can form an optimal strategy for every $\underline{p}$.*

The runtime of experiment 2.5.2 is mostly dependent on the number of grind points. By taking only a few number of grid points, the experiment shows that there does not exist a subset of six pure strategies that can always form an optimal strategy for any $\underline{p}$. Moreover, this subset also does not exist when we allow it to have seven or eight pure strategies. These subsets only exist when they are of size equal or greater than nine. To conclude, there does not exist a subset of six pure strategies such that a mix of these six can always form an optimal strategy for any $\underline{p}$. Hence, there does not exist a generalisation of proposition 2.22.

The results of experiment 2.5.2 show that the probabilities $\underline{p}$ influence which pure strategies the searcher should mix in her optimal strategy. Let x be the number of pure strategies that the hider mixes in his optimal strategy. We know that for some sub interval of $\underline{p}$, there must exist a subset of x pure strategies for the searcher such that a mix of these x pure strategies is optimal. It might be an idea to divide the interval $\underline{p}$ into multiple smaller intervals. On every smaller interval, we could find an optimal strategy for the searcher that consists of x pure strategies. By combing the solutions of all sub intervals, we would have somewhat of a piecewise solution for the searcher that always mixes x pure strategies. The problem is in how we divide the interval $\underline{p}$ and which pure strategies are optimal on what sub interval of $\underline{p}$. We leave these problems to our successor.

3
Search and Rescue Games on a Tree

In the previous chapter, we considered the search and rescue game in which the success probabilities $\underline{p} = (p_1, ..., p_n)$ were independent. That is to say if i and j are two hiding locations and p_i and p_j are the probabilities of successfully searching i and j respectively, then the total success probability function f is defined as the product

$$f(i \cup j) = p_i p_j.$$

In this chapter, we consider the game in which the probabilities are dependent. So now the total success probability of searching i and j is not necessarily equal to the product of the two individual probabilities. Following Lidbetter [9], the game is now played on a graph G or more specific, a tree. The hiding locations are the vertices of the tree which are connected with one another by the edges. The total success probability function f is now defined by a tree G. Due to the complexity of the game played on a tree, the hider now hides a single object in any of the vertices of the tree which the search has to find. The searcher has to move from one vertex to another using the edges, and find the hidden object. She wins if she succeeds and loses otherwise.

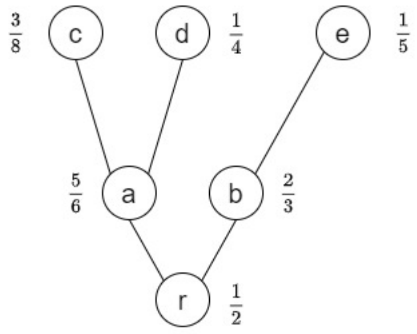


Figure 3.1: Search and Rescue Game on a Tree with Root r

3.1. Definitions and Rules

We start the chapter by first introducing some basic definitions and notation related to graphs and trees.

Definition 3.1.1 (**Graph and Tree**). *A graph $G = (V, E)$ consists of a vertex set $V(G)$ and an edge set $E(G)$ which we usually denote by V and E respectively. A tree T is a graph such that any two vertices are connected by exactly one path, i.e. one finite sequence of edges.*

By definition, a tree is always a graph, but a graph is not necessarily a tree. In the remainder of the chapter, we refer to G as the graph and the tree on which the game is played on.

Definition 3.1.2 (**Neighbour Vertices**). *Two vertices x and y are called neighbours if the edge $(x, y) \in E(G)$ exists.*

Definition 3.1.3 (**Root and Branches**)**.** *A root vertex r of a tree is a designated vertex $r \in V(G)$ which is the starting vertex of the tree. The root vertex will be the starting point of the game for the searcher. The branches of a tree with root r are the connected components when the root vertex and its ingoing/outgoing edges are removed. The roots of the branches are the neighbours of r in G.*

Definition 3.1.4 (**Degree and Branch Vertices**)**.** *The degree of a vertex v is the number of neighbours of v. The root r is a branch vertex if it has degree 2 and non root vertices are called branch vertices if they have degree 3. In general, vertices of a tree can have any degree and are therefore not necessarily branch vertices.*

Definition 3.1.5 (**Leaf**)**.** *A leaf is a vertex of degree 1 (excluding the root vertex). We denote $L(G)$ or L as the set containing all leaf vertices of a tree G.*

Definition 3.1.6 (**Subtree**)**.** *A subtree $G' = (V', E')$ of a tree $G = (V, E)$ is a subset of vertices $V' \subseteq V$ and edges $E' \subseteq E$ such that G' is also a tree. For any vertex $v \in V$ we denote $G(v)$ as the subtree of G containing all vertices and edges whose path to r contain v.*

Definition 3.1.7 (**Binary Tree**)**.** *A tree G is a binary tree if every non leaf vertex is a branch vertex, i.e. the root has degree 2 and all non leaf (and non root) vertices have degree 3. We call the tree with only one (root) vertex r the trivial binary tree.*

We are now able to define the search and rescue game on a tree. The set of hiding locations $S = \{1, 2, ..., n\}$ now corresponds to the vertex set $V(G)$ of a tree G, i.e. every vertex of the tree is a hiding location. The edge set $E(G)$ is a collection of unordered pairs of vertices (i, j) with $i, j \in V(G)$ and $i \neq j$. The searcher has to start the search at a pre-determined root vertex $r \in V(G)$ and can only go from vertex a directly to vertex b if the edge $(a, b) \in E(G)$ exists. Because the searcher starts the game at r, the root vertex is always searched first. When searching vertex i there is a probability $p_i \in (0, 1]$ that the search will be successful and that the hidden object in i will be found. The probability p_i is, unlike the previous search and rescue game, allowed to take on the value 1 which makes it possible for some hiding locations to always be searched successfully.

The hider has to hide one object in any vertex $v \in V(G)$. However, hiding an object in a non-leaf vertex is a bad choice and being dominated. Of course, the leaf vertices can be thought of as the end points of the tree. If the hider hides the object in a non-leaf vertex, then there is a leaf vertex that is even further away from the root such that the searcher has to search even more vertices before finding the object.

For example, consider the search and rescue game on the tree depicted in figure 3.1. It is clear that if the searcher starts at r, hiding the object in location b is being dominated by hiding in e. If the hider hides the object in b, he might as well hide it in e such that the searcher needs to search an extra vertex. Hence, the hider always hides the object in one of the leaf vertices.

Definition 3.1.8 (**Hider's Pure Strategy**)**.** *A pure strategy h for the hider is a leaf vertex of G, i.e. $h \in L(G)$.*

The searcher has to search the vertices of the tree in some order. However, she is now restricted by the fact that she has to start the search at the root r and can only search the remaining vertices using the tree structure. This means that after searching r, the searcher is only allowed to search the neighbours of r. Recall that for the regular search and rescue game, the searcher chooses a permutation σ of the hiding locations S and that the set S_i^σ contains the first i locations searched by σ. Using the same methodology, we define the search of a tree as follows.

Definition 3.1.9 (**Search of a Tree**)**.** *A search of a tree G is a permutation*

$$\sigma : S \to V(G)$$

such that

- *The root r is searched first, i.e. $\sigma(1) = r$.*
- *At every step i, the searcher is only allowed to search the vertices v if it is a neighbour of a previously searched vertex, i.e. $\exists x \in S_{i-1}^\sigma : (x, \sigma(i)) \in E(G) \;\; \forall i = 2, ..., n$.*

Note that by this definition, the searcher is allowed to freely move between vertices that have already been searched without additional penalty. Since the hider only hides the object in leaf vertices, the searcher wants to search as few non leaf vertices as possible before searching any leaf vertex. Of course, if the hider only hides in the leaf vertices, searching additional non-leaf vertices before searching a leaf vertex is a bad strategy. To illustrate, consider the tree in figure 3.1. Let

$$\sigma = \{r, a, c, d, b, e\} \quad \text{and} \quad \sigma' = \{r, a, b, c, d, e\}$$

be two searches of G. Both strategies search the leaf vertices in the order c, d and e. The strategy σ does so by using the shortest path to any leaf vertex, whereas the strategy σ' does not. Observe that both strategies are equally good when the object is hidden in e and that σ is a better strategy than σ' when it is hidden in either c or d. This is because σ' will search an extra vertex b in the latter case, thus decreasing the total success probability (if we define the total success probability similar to definition 2.2.3 for the moment). Therefore, σ dominates σ'.

Consequently, the searcher's pure strategies are the permutations of the vertices S such that the leaf vertices are searched using the shortest paths. In other words, the searcher's pure strategies are the permutations of the leaf vertices L in which the leaf vertices are searched using the shortest paths.

Definition 3.1.10 (**Searcher's Pure Strategy**). *Let $m = |L|$ and let $X = \{1, \ldots, m\}$ be an ordering of the leaf vertices. A pure strategy σ for the searcher is a permutation*

$$\sigma : X \to L$$

of L such that

- *$\sigma(i)$ is the i-th leaf vertex searched in the permutation σ for $i = 1, \ldots, m$.*
- *Every leaf vertex $\sigma(i)$ is searched by searching all vertices on the shortest path from r to $\sigma(i)$ that have not been searched already.*

Every searcher's pure strategy corresponds to exactly one search of the tree as defined in definition 3.1.9, with the only difference being in notation. Lidbetter [9] defines the searcher's pure strategies as all possible searches of a tree, which is the same as definition 3.1.9 (which is also the same as definition 3.1.10 plus all dominated searching strategies). It is unimportant which definition will be used for the proofs later on, we have chosen to stick with Lidbetter [9] for the proofs.

Let G be a tree that has $m = |L(G)|$ leaf vertices. Then the hider has m pure strategies which are all leaf vertices. The searcher on the other hand has $m!$ pure strategies, namely all possible permutations of the m leaf vertices. Here the leaf vertices are searched using the shortest paths. The total success probability function f is now defined on the leaf vertices. If A is a set of leaf vertices that is searched, then all vertices that are on the path from r to any $x \in A$ should also be searched.

Definition 3.1.11 (**Total Success Probability**). *Let $A \subseteq L$ be a subset of leaf vertices and let $A' \supseteq A$ be the vertex set of the subtree spanned by A in G. Then the total success probability function $f : 2^L \to (0, 1)$ is given by*

$$f(A) = \prod_{i \in A'} p_i \quad \text{and} \quad f(\emptyset) = 1. \tag{3.1}$$

In the remainder of the chapter, we will denote A' as the set defined above for any set of leaf vertices A.

The pay-off is defined similar as in the regular search and rescue game, albeit with the total success probability function as in 3.1.

Definition 3.1.12 (**Pay-off**). *The pay-off for the searcher when she plays σ (as in definition 3.1.10) and the hider plays h is given by*

$$P(h, \sigma) = f(S_i^\sigma) = \prod_{i \in (S_i^\sigma)'} p_i \tag{3.2}$$

where i is minimal such that $h \in S_i^\sigma$. Similarly, the pay-off for the hider is

$$-P(h, \sigma) = -\prod_{i \in (S_i^\sigma)'} p_i. \tag{3.3}$$

Definition 3.1.13 (**Search and Rescue Game on a Tree**). *A search and rescue game on a tree G with m leaf vertices and one hidden object is a game in which the hider chooses a hiding strategy h as defined in definition 3.1.8, the searcher chooses a searching strategy σ as defined in definition 3.1.10 and the pay-off for the searcher and hider is given by $P(h, \sigma)$ and $-P(h, \sigma)$ which are defined in 3.2 and 3.3 respectively.*

We will now define a small example of the search and rescue game on a tree.

Example 3.1.1 (Search and Rescue Game on a Tree: Figure 3.1). *Consider the search and rescue game on the tree in figure 3.1 where the success probabilities are given next to the vertices. Let r be the root and starting vertex of the searcher.*

The hider can hide the object in one of the leaf vertices: c, d or e, which are his pure strategies. The searcher needs to choose in what order she will search the leaf vertices. Every order is a permutation of the leaf vertices, and there are exactly $3! = 6$ permutations, namely

$$\sigma_1 = \{c,d,e\}, \quad \sigma_2 = \{c,e,d\}, \quad \sigma_3 = \{d,c,e\}, \quad \sigma_4 = \{d,e,c\}, \quad \sigma_5 = \{e,c,d\}, \quad \sigma_6 = \{e,d,c\}.$$

Recall that these permutations search the leaf vertices using the shortest paths. For example, the pure strategy σ_1 is equivalent to a search that searches the vertices in the order $\{r,a,c,d,b,e\}$.

Assume the hider and searcher play the pure strategies $h = c$ and σ_2 respectively. The strategy σ_2 will first search the leaf vertex c by searching all vertices on the shortest path from r to c. The shortest path from r to c is by going from r to a to c. Since the hider hid the object in c, the pay-off is given by

$$P(h,\sigma_2) = f(S_1^{\sigma_1}) = f(\{c\}) = \prod_{i\in\{c\}'} p_i = \prod_{i\in\{r,a,c\}} p_i = p_r p_a p_c = \frac{5}{32}.$$

Now assume that the hider plays the pure strategy $h = e$ instead. The strategy σ_2 will first search the leaf vertex c before searching e. To be able to search c, the vertices r and a need to be searched prior to c. Since nothing is found in c, the searcher can move back to the root r (without additional penalty). From r, the searcher will now make its way to e. In order to reach e, the vertex b has to be searched first. Since the object is hidden in e, the pay-off is given by the product of the five searched vertices.

Notice that by searching e after c, the searcher is going from one branch of the tree (the branch with root a) to the other (the branch with root b) without fully searching the former branch (d has not been searched). For the optimal strategy defined later on, this property is undesirable. That is, partially searching a branch of a tree. This means that in the example, the leaf vertices c and d should always be searched consecutively in any order. These type of searches are called depth-first searches, and it will be shown later on that the searcher's optimal strategy consists of depth-first searches.

Definition 3.1.14 (Subsearch). *A subsearch α of a search σ is the restriction of σ to some set $\{i, i+1, ..., j-1, j\}$, i.e. α describes the sequence of vertices searched in step i to j.*

Definition 3.1.15 (Depth-first search). *A depth-first search σ of a tree G is a search such that $\forall\, v \in V(G)$, all vertices of $G(v)$ are searched immediately and without interruption after v in σ.*

By restricting the searcher to strategies that are depth-first searches, the searcher has even less pure strategies. In example 3.1.1, the searcher only has four pure strategies namely:

$$\sigma_1 = \{c,d,e\}, \quad \sigma_3 = \{d,c,e\}, \quad \sigma_5 = \{e,c,d\}, \quad \text{and} \quad \sigma_6 = \{e,d,c\}. \tag{3.4}$$

The example can now be solved by first constructing the pay-off matrix

$$A = \begin{array}{c} \\ \sigma^1 \\ \sigma^2 \\ \sigma^3 \\ \sigma^4 \end{array} \begin{array}{c} \begin{array}{ccc} c & d & e \end{array} \\ \begin{pmatrix} \frac{5}{32} & \frac{5}{128} & \frac{1}{192} \\ \frac{5}{128} & \frac{5}{48} & \frac{1}{192} \\ \frac{1}{48} & \frac{1}{192} & \frac{1}{15} \\ \frac{1}{192} & \frac{1}{72} & \frac{1}{15} \end{pmatrix} \end{array} \tag{3.5}$$

which can then be solved using matrix-game solvers. However, this method is once again not feasible for very large and complicated trees. Therefore, a more general solution method is needed.

3.2. Depth-first Optimal Strategy

In this section, we will introduce a more general method to compute the optimal strategies for the search and rescue game on a tree. These optimal strategies are defined recursively for both players. The idea is that for every vertex v we consider the subtree $G(v)$ and its branches. Each branch is assigned a probability of being played over the others. By multiplying the probabilities, we eventually get a probability distribution over the leaf vertices for the hider and a probability distribution over all depth-first searches for the searcher, which are mixed strategies. These strategies are also known as behavioral strategies [6].

Definition 3.2.1 (Behavioural Strategy). *A behavioural strategy is a probability distribution over the set of all possible actions at every point in a game.*

The optimal strategies make use of an important property of the tree, namely that it is binary. Recall that a binary tree is a tree in which the root has degree 2 and all other non leaf vertices have degree 3. In general, search and rescue games can be played on any tree which do not necessarily need to be binary. However, it is possible to transform any arbitrary tree into a binary tree such that the game played on both trees are equivalent. That is to say, both games have the same set of leaf vertices and for any set of leaf vertices A, the total success probability $f(A)$ is the same in both games. Binary trees have the property that every non leaf vertex has exactly two branches. Therefore, games played on binary trees are easier to analyse. We will now show how arbitrary non binary trees can be transformed into binary trees.

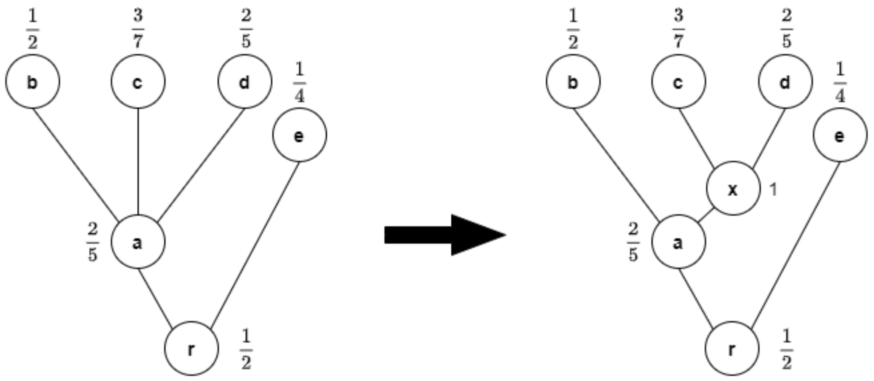


Figure 3.2: Transforming vertex a of degree 4

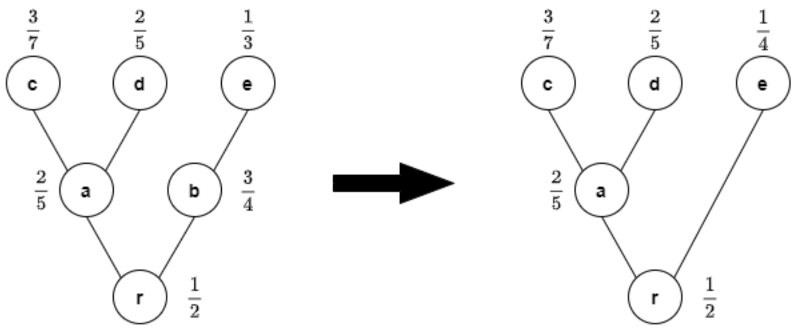


Figure 3.3: Removing vertex b of degree 2

To transform an arbitray non binary tree into a binary tree, there are two operations that are repeatedly applied. The first operation transforms a non leaf vertices of degree greater than 3 (greater than 2 for the root) into a branch vertex, i.e. a vertex of degree 3 (or 2 for the root), without changing the game. An example of such a transformation is given in figure 3.2.

Observe that vertex a in the left tree has degree $d(a) = 4$. By adding vertex x with $p_x = 1$ and connecting it to a and all branches of a except for one, a now has degree 3 and x has degree $d(a) - 1$, which is one less than what a previously had. We continue to add vertices x until a and all added vertices x have degree of exactly 3. Note that since all added vertices x have $p_x = 1$, the new game is equivalent to the old game. Finally, by repeating this for all non leaf vertices that have degree greater than 3 (or 2 for the root vertex), it follows that every non leaf vertex now has degree less than or equal to 3.

The second operation removes all non leaf vertices of degree less than 3 (less than 2 for the root) without changing the game. An example is shown in figure 3.3. First observe that vertex b in the left tree has degree $d(b) = 2$. By removing b and connecting the old neighbours of b by a direct edge, the new tree has one less non leaf vertex that is not a branch vertex. By updating the success probability of the neighbour that is furthest away from the root as the product of its own success probability and the success probability of the removed vertex, the new and old game are equivalent. In our example, we remove b and thus we update $p_e^{\text{new}} = p_e^{\text{old}} \cdot p_b = \frac{1}{3} \cdot \frac{3}{4} = \frac{1}{4}$. Repeating this operation for all non leaf vertices of degree less than 3 (or 2 for the root vertex), it follows that the root vertex has at least degree 2 and all other non leaf vertices have degree at least 3.

By combining both operations, any tree can be transformed such that the root has degree 2 and all other non leaf vertices have degree 3. Furthermore, both operations do not change the game (i.e. the game played on the resulting tree is equivalent to the game played on the original tree). Hence, every tree can be transformed into a game equivalent binary tree. Thus, we may restrict our analysis to search and rescue games played on binary trees. This makes our analysis somewhat easier, because we are now able to use that the tree only consists of branch vertices (including the root vertex) and leaf vertices.

Before defining the optimal strategy for the hider, we introduce some more notation. For a subset $A \subseteq V(G)$, we denote

$$\pi(A) = \prod_{i \in A} p_i. \tag{3.6}$$

For a graph G we write $\pi(G)$ to express $\pi(V(G))$. Denote h_G as the hider's strategy on the tree G and $h_G(v)$ as the probability of hiding the object in vertex v when the game is played on the tree G. Finally, we denote V_G as the value of the game played on the tree G. The optimal hiding strategy can now be defined as follows.

Definition 3.2.2 (**Optimal Tree Hiding Strategy**). *Let G be a binary tree. If G is the trivial binary tree, i.e. it only has one root vertex r, then trivially*

$$h_G(r) = 1 \quad \text{and} \quad V_G = p_r. \tag{3.7}$$

If G is not the trivial binary tree, then by definition the root r has two branches, say G_1 and G_2 with roots r_1 and r_2 respectively. The probability of hiding the object in branch G_i when the game is played on G is then given by

$$h_G(G_i) = \lambda_G \left(\frac{1 - \pi(G_i)}{V_{G_i}} \right), \quad i = 1,2 \tag{3.8}$$

with λ_G a normalizing factor given by

$$\lambda_G = \left(\frac{1 - \pi(G_1)}{V_{G_1}} + \frac{1 - \pi(G_2)}{V_{G_2}} \right)^{-1}. \tag{3.9}$$

Let $v \in V(G_i)$ be a leaf vertex for $i = 1,2$. Then the probability of hiding the object in v when the game is played on G is given by a conditional probability on G_i, i.e.

$$h_G(v) = h_{G_i}(v) h_G(G_i). \tag{3.10}$$

The formula for the value of the game is given by

$$V_G = p_r \lambda_G \big(1 - \pi(G_1)\pi(G_2)\big). \tag{3.11}$$

The optimal tree searching strategy will be defined later on. We will focus on the optimal tree hiding strategy h_G for now. Before proving the optimality of the tree hiding strategy and that V_G defined in 3.11 is indeed the value of the game, we give an example of how to apply the tree hiding strategy to the tree depicted in figure 3.1.1.

Example 3.2.1 (**Optimal Tree Hiding Strategy for Example 3.1.1**). *It is easier to execute the tree hiding strategy starting from the leaf vertices, rather than from the root. This allows all variables to be filled in immediately. First observe that the tree in figure 3.1.1 is not binary, but can be made binary by the same procedure as in figure 3.3. Then by 3.7, it follows that for the corresponding binary tree*

$$V_{G(c)} = \frac{3}{8}, \quad V_{G(d)} = \frac{1}{4}, \quad V_{G(e)} = \frac{2}{15}$$

and

$$h_{G(c)}(c) = h_{G(d)}(d) = h_{G(e)}(e) = 1.$$

The tree has two branch vertices, namely a and r. For branch vertex a, it follows from 3.9 that

$$\lambda_{G(a)} = \left(\frac{1-\pi(G(c))}{V_{G(c)}} + \frac{1-\pi(G(d))}{V_{G(d)}}\right)^{-1} = \left(\frac{1-\frac{3}{8}}{\frac{3}{8}} + \frac{1-\frac{1}{4}}{\frac{1}{4}}\right)^{-1} = \frac{3}{14}.$$

Then by filling in 3.8 and 3.11 we find

$$h_{G(a)}(G(c)) = h_{G(a)}(c) = \lambda_{G(a)}\left(\frac{1-\pi(G(c))}{V_{G(c)}}\right) = \frac{3}{14}\left(\frac{1-\frac{3}{8}}{\frac{3}{8}}\right) = \frac{5}{14}$$

$$h_{G(a)}(G(d)) = h_{G(a)}(d) = \lambda_{G(a)}\left(\frac{1-\pi(G(d))}{V_{G(d)}}\right) = \frac{3}{14}\left(\frac{1-\frac{1}{4}}{\frac{1}{4}}\right) = \frac{9}{14}.$$

and

$$V_{G(a)} = p_a\lambda_{G(a)}\Big(1-\pi\big(G(c)\big)\pi\big(G(d)\big)\Big) = \frac{5}{6}\cdot\frac{3}{14}\left(1-\frac{3}{8}\cdot\frac{1}{4}\right) = \frac{145}{896}.$$

For the root vertex r, it follows that

$$\lambda_G = \left(\frac{1-\pi(G(a))}{V_{G(a)}} + \frac{1-\pi(G(e))}{V_{G(e)}}\right)^{-1} = \left(\frac{1-\frac{5}{6}\cdot\frac{3}{8}\cdot\frac{1}{4}}{\frac{145}{896}} + \frac{1-\frac{2}{15}}{\frac{2}{15}}\right)^{-1} = \frac{290}{3537}.$$

Therefore by 3.8 and 3.11

$$h_G(G(a)) = \lambda_G\left(\frac{1-\pi(G(a))}{V_{G(a)}}\right) = \frac{290}{3537}\left(\frac{1-\frac{5}{64}}{\frac{145}{896}}\right) = \frac{1652}{3537}$$

$$h_G(G(e)) = \lambda_G\left(\frac{1-\pi(G(e))}{V_{G(e)}}\right) = \frac{290}{3537}\left(\frac{1-\frac{2}{15}}{\frac{2}{15}}\right) = \frac{1885}{3537}$$

and the value of the game is

$$V_G = p_r\lambda_G\Big(1-\pi\big(G(a)\big)\pi\big(G(e)\big)\Big) = \frac{1}{2}\cdot\frac{290}{3537}\left(1-\frac{5}{64}\cdot\frac{2}{15}\right) = \frac{13775}{339552}.$$

To conclude, by 3.10 it is optimal for the hider to hide the object in c, d or e with probability

$$h_G(c) = h_G(G(a))h_{G(a)}(c) = \frac{1652}{3537}\cdot\frac{5}{14} = \frac{590}{3537}$$

$$h_G(d) = h_G(G(a))h_{G(a)}(d) = \frac{1652}{3537}\cdot\frac{9}{14} = \frac{118}{393}$$

$$h_G(e) = h_G(G(e))h_{G(e)}(e) = \frac{1885}{3537}\cdot 1 = \frac{1885}{3537}$$

which agrees with the optimal hiding strategy found when solving the matrix game in 3.5. This concludes the optimal tree hiding strategy example.

To show that the formula for the value V_G given in 3.11 is indeed the value of the game, we will first show that h_G guarantees a pay-off of at most V_G. Then by introducing the optimal tree searching strategy s_G and showing that the searcher can guarantee a pay-off of at least V_G, it follows that V_G is indeed the value of the game and that both strategies are optimal. To that end, we will first show that any depth-first search of G has expected pay-off V_G against the hiding strategy h_G.

Recall that the hider's pure strategies are all leaf vertices, hence a mixed strategy for the hider is a probability distribution over the leaf vertices. This is the same as a probability distribution over all vertices where the non leaf vertices are played with probability 0. Let $\underline{x}$ be such a mixed strategy for the hider and let σ be a search of G. Then the expected pay-off is given by

$$P(\underline{x},\sigma) = P(\sigma) = x_{\sigma(1)}p_{\sigma(1)} + x_{\sigma(2)}p_{\sigma(2)}p_{\sigma(1)} + \ldots + x_{\sigma(n)}p_{\sigma(n)}\ldots p_{\sigma(1)}. \tag{3.12}$$

Here, we will drop the $\underline{x}$ in $P(\underline{x},\alpha)$ and instead write $P(\sigma)$ when there is no ambiguity. Similarly, the expected pay-off for some subsearch α of vertices $\sigma(\{i,i+1,\ldots,j\})$ is given by

$$P(\underline{x},\alpha) = P(\alpha) = x_{\alpha(i)}p_{\alpha(i)} + x_{\alpha(i+1)}p_{\alpha(i+1)}p_{\alpha(i)} + \ldots + x_{\alpha(j)}p_{\alpha(j)}\ldots p_{\alpha(i)}. \tag{3.13}$$

Lemma 4. *Let G be a binary tree and assume the hider plays the hiding strategy h_G as defined in definition 3.2.2. Then any depth-first search σ of G has expected pay-off $P(\sigma) = V_G$.*

Proof. Let σ be depth-first search of G. We prove the lemma by induction on the number of vertices of G. If G only has one vertex r, then trivially by 3.7

$$P(\sigma) = p_r = V_G.$$

Now assume the lemma holds for trees with fewer than n vertices. Let G be a binary tree with n vertices including root vertex r. Since G is a binary tree, the root r is a branch vertex and has two branches. Let G_1 and G_2 be the branches of G. By the induction hypothesis, any depth-first search of G_i has expected pay-off V_{G_i} for $i = 1, 2$. Let σ_1 and σ_2 be depth-first searches of G_1 and G_2 respectively. Recall that with probability $h_G(G_i)$ the hider hides the object in G_i. Hence with the induction hypothesis, it follows that

$$P(\sigma_1) = h_G(G_1)V_{G_1} \quad \text{and} \quad P(\sigma_2) = h_G(G_2)V_{G_2}.$$

Assume without loss of generality that σ searches r, σ_1 and σ_2 in that order. Then σ is a depth-first search of G and the expected pay-off is

$$\begin{aligned} P(\sigma) &= p_r\big(P(\sigma_1) + \pi(G_1)P(\sigma_2)\big) \\ &= p_r\big(h_G(G_1)V_{G_1} + \pi(G_1)h_G(G_2)V_{G_2}\big). \end{aligned}$$

Finally, by 3.8 and 3.11 it follows that

$$\begin{aligned} P(\sigma) &= p_r\Big(\lambda_G\big(1 - \pi(G_1)\big) + \pi(G_1)\lambda_G\big(1 - \pi(G_2)\big)\Big) \\ &= p_r\lambda_G\Big(1 - \pi(G_1)\pi(G_2)\Big) = V_G. \end{aligned}$$

□

Lemma 4 shows that any depth-first search has an expected pay-off V_G against the hiding strategy h_G. Hence, to show that V_G is an upper bound to the value of the game it remains to show that depth-first searches are best responses against h_G. Here we once again make use of the term indexibility similar to section 2.3. Previously, the index was assigned to every hiding location such that a higher index indicates a higher probability of being searched first. For the game played on a tree, the index is now assigned to a subsearch rather than to a hiding location such that if two disjoint subsearches can be executed consecutively, then the subsearch with the higher index should be executed first. For a fixed hider strategy $\underline{x}$ and subsearch α of vertices A with $\pi(A) \neq 1$, the index is defined as

$$I_x(\alpha) = I(\alpha) = \frac{P(\alpha)}{1 - \pi(A)}. \tag{3.14}$$

Note that the index in 3.14 is well-defined since $\pi(A) \neq 1$. The following lemma shows that if two disjoint subsearches can be executed consecutively, then the subsearch with the highest index has a higher expected pay-off and hence should be executed first.

Lemma 5. *Let $\underline{x}$ be a fixed hider strategy and let σ be a search. Suppose a subsearch α of σ searches a subset $A \subseteq V(G)$ immediately before a subsearch β of σ searches a subset $B \subseteq V(G)$ disjoint from A with $\pi(A), \pi(B) \neq 1$. Let σ' be the search identical to σ where only the order of α and β are interchanged. Then $P(\sigma) \leq P(\sigma') \iff I(\alpha) \leq I(\beta)$. Furthermore, $P(\sigma) = P(\sigma') \iff I(\alpha) = I(\beta)$*

Proof. Let C be the vertices searched before α in σ (and β in σ'). Then, the difference in expected pay-off is given by

$$\begin{aligned} P(\sigma) - P(\sigma') &= \big(\pi(C)P(\alpha) + \pi(C)\pi(A)P(\beta)\big) - \big(\pi(C)P(\beta) + \pi(C)\pi(B)P(\alpha)\big) \\ &= \pi(C)\Big(P(\alpha)\big(1 - \pi(B)\big) - P(\beta)\big(1 - \pi(A)\big)\Big) \\ &= \pi(C)\Big(I(\alpha)\big(1 - \pi(A)\big)\big(1 - \pi(B)\big) - I(\beta)\big(1 - \pi(B)\big)\big(1 - \pi(A)\big)\Big) \\ &= \pi(C)\big(1 - \pi(A)\big)\big(1 - \pi(B)\big)\big(I(\alpha) - I(\beta)\big). \end{aligned}$$

The lemma follows immediately from the fact that the first three terms are positive. □

Lemma 5 will now be used to prove that depth-first searches are best responses to the tree hiding strategy h_G. This will be done by defining an iterative method such that any best response can be transformed to a depth-first search without changing the expected pay-off. Then by lemma 4, every depth-first search has the same expected pay-off. Hence, every depth-first search is a best response to h_G. This proves that V_G is indeed an upper bound to the value of the game.

Lemma 6. *Let G be a binary tree and assume the hider plays the tree hiding strategy h_G as in definition 3.2.2. Then the following statements hold:*

1. *If v is a branch vertex and σ_1 and σ_2 are depth-first searches of the branches G_1 and G_2 of $G(v)$ respectively, then $I(\sigma_1) = I(\sigma_2)$.*

2. *Any depth-first search σ is a best response to the tree hiding strategy h_G and h_G ensures that the expected pay-off is at most V_G.*

Proof. First, note that $G_1, G_2 \subset G(v) \subseteq G$. In the remainder of the proof, we let $i = 1,2$ when we mention G_i and the like. Since $G_i \subset G(v)$, it follows for the first statement that

$$h_G(G_i) = h_G\big(G(v)\big) \cdot h_{G(v)}(G_i) \quad \text{for } i = 1,2.$$

Recall that $h_G(G_i)$ is the probability of hiding the object in the branch G_i when the game is played on the tree G. Therefore, with lemma 4 it follows that

$$P(\sigma_i) = h_G(G_i) \cdot V_{G_i} = h_G\big(G(v)\big) \cdot h_{G(v)}(G_i) \cdot V_{G_i}.$$

Since G_i are the branches of $G(v)$, we apply 3.8 to the term $h_{G(v)}(G_i)$ to obtain

$$P(\sigma_i) = h_G\big(G(v)\big) \cdot \lambda_{G(v)} \cdot \big(1 - \pi(G_i)\big).$$

Finally, dividing both sides by $\big(1-\pi(G_i)\big)$ it follows that

$$I(\sigma_i) = \frac{P(\sigma_i)}{\big(1-\pi(G_i)\big)} = h_G\big(G(v)\big) \cdot \lambda_{G(v)}. \tag{3.15}$$

Note that the right-hand side of 3.15 is independent of i, hence $I(\sigma_1) = I(\sigma_2)$ which proves the first statement.

For the second statement, let σ be a best response to h_G that is not depth-first. We will show that we can iteratively change the order of subsearches in σ such that we obtain a depth-first search σ' that has the same expected pay-off as σ. First note that since G is a binary tree, there must exist a branch vertex v such that the branches of $G(v)$ are both given by a single leaf vertex. The subsearch of these branches in σ are trivially depth-first. Let v be such a branch vertex, G_i be its branches and σ_i be their respective subsearches in σ. Then σ searches the branches G_i in a depth-first order, but not necessarily consecutively. Assume w.l.o.g. that σ searches the branches of $G(v)$ in the order σ_1, α, σ_2, where α is the subsearch of some subset $A \subset V(G)$ disjoint from $G(v)$. Since σ is a best response, it follows by lemma 5 that

$$I(\sigma_1) \geq I(\alpha) \geq I(\sigma_2).$$

But by part 1, we have that

$$I(\sigma_1) = I(\sigma_2) = I(\alpha).$$

Therefore, swapping the order in which σ_1 and α are searched in σ leaves the expected pay-off unchanged. After the swap, the order in which $G(v)$ is searched is: α, σ_1, σ_2. Furthermore, note that vertex v must be searched before σ_1 and that α does not search v. Let β be the subsearch of the vertices searched after v and before α in σ such that the order of the search is given by: v, β, α, σ_1, σ_2. Notice that changing the order of the search such that v is searched immediately before σ_1 does not change the expected pay-off. By doing so, the only thing that changes is that the subsearches β and α are now searched one step earlier. Hence, the expected pay-off does not decrease. It also does not increase, otherwise σ would not be a best response.

The subtree $G(v)$ is now searched in a depth-first order in σ without changing the expected pay-off. By repeating these steps for all v, it follows that all subtrees $G(v)$ are searched in a depth-first order. Now consider the branch vertices $w \neq v$ such that the branches of $G(w)$ are either leaf vertices and/or the branches $G(v)$. Then the branches of $G(w)$ are searched in a depth first order, but not necessarily consecutively. Therefore, we can again repeat the previous steps to eventually obtain a depth-first search σ' of $G(r) = G$ that has the same expected pay-off as σ. Hence, σ' is also a best response to h_G. By lemma 4, any depth-first search has the same expected pay-off V_G against h_G. Therefore, any depth-first search is a best response to h_G and h_G ensures an expected pay-off of at most V_G. □

In order to show that V_G is the value of the game, it remains to show that there exist a searching strategy s_G such that the searcher can guarantee a pay-off of at least V_G. This also directly proves that the strategies h_G and s_G are optimal. Before defining the optimal tree searching strategy s_G, the following lemma will be introduced to ensure that the optimal searching strategy s_G is well-defined.

Lemma 7. *Let G be a binary tree, then the following inequality holds*

$$\pi(G) \le V_G \le 1, \tag{3.16}$$

where V_G is defined as in 3.11.

Proof. The proof is by induction on the number of vertices. If G has only vertex r, then

$$\pi(G) = p_r = V_G \le 1.$$

Assume the lemma holds for trees with fewer than n vertices. Let G be a binary tree with n vertices, r be the root vertex of G and let G_1 and G_2 be the branches of G. Since G_1 and G_2 have fewer than n vertices, it follows by the induction hypothesis that

$$\pi(G_1) \le V_{G_1} \le 1$$
$$\pi(G_2) \le V_{G_2} \le 1.$$

It follows that the normalizing factor in 3.9 satisfies

$$\begin{aligned}\lambda_G &= \left(\frac{1-\pi(G_1)}{V_{G_1}} + \frac{1-\pi(G_2)}{V_{G_2}}\right)^{-1} \ge \left(\frac{1-\pi(G_1)}{\pi(G_1)} + \frac{1-\pi(G_2)}{\pi(G_2)}\right)^{-1} \\ &= \frac{\pi(G_1)\pi(G_2)}{\pi(G_2)\big(1-\pi(G_1)\big) + \pi(G_1)\big(1-\pi(G_2)\big)} \\ &\ge \frac{\pi(G_1)\pi(G_2)}{\big(1-\pi(G_1)\big) + \pi(G_1)\big(1-\pi(G_2)\big)} \\ &= \frac{\pi(G_1)\pi(G_2)}{1-\pi(G_1)\pi(G_2)}.\end{aligned}$$

It follows from 3.11 that

$$V_G = p_r\lambda_G\big(1-\pi(G_1)\pi(G_2)\big) \ge p_r\pi(G_1)\pi(G_2) = \pi(G).$$

Furthermore, the normalizing factor in 3.9 also satisfies

$$\begin{aligned}\lambda_G &= \left(\frac{1-\pi(G_1)}{V_{G_1}} + \frac{1-\pi(G_2)}{V_{G_2}}\right)^{-1} \le \big(1-\pi(G_1) + 1-\pi(G_2)\big)^{-1} \\ &\le \Big(1-\pi(G_1) + \pi(G_1)\big(1-\pi(G_2)\big)\Big)^{-1} \\ &= \big(1-\pi(G_1)\pi(G_2)\big)^{-1}.\end{aligned}$$

Therefore, it follows that 3.11 also satisfies

$$V_G = p_r\lambda_G\big(1-\pi(G_1)\pi(G_2)\big) \le p_r \le 1.$$

□

It follows from lemma 7 that the formula for the value V_G defined in 3.11 is bounded. These bounds are very intuitive if we think of the pay-offs as probabilities and V_G as the value of the game. In the worst case, all locations need to be searched which corresponds to the lower bound $\pi(G)$. In the proof of the lemma, it is shown that p_r is an even stricter upper bound to V_G. In the best case, only the root vertex r needs to be searched which is an upper bound. The optimal searching strategy s_G can now be defined as follows.

Definition 3.2.3 (Optimal Tree Searching Strategy). *Let G be a binary tree. If G is the trivial binary tree with root r, then trivially*

$$q_r = 1. \tag{3.17}$$

If G is not the trivial binary tree, then the tree searching strategy s_G is a choice of depth-first searches of G and described by specifying which branch of the tree is searched first at every branch vertex. Let v be a branch vertex with branches G_1 and G_2. The branch G_1 is searched first with probability

$$q_{G_1} = \lambda_G\left(\frac{1}{V_{G_1}} - \frac{\pi(G_2)}{V_{G_2}}\right) \tag{3.18}$$

where $\lambda(G)$ and V_G are defined as in the optimal tree hiding strategy in definition 3.2.2. Trivially, G_2 will be searched first with probability $1 - q_{G_1}$.

The optimal tree searching strategy is a depth-first search in which the probability of searching one branch before the other is given by q_{G_1} as in 3.18. Let us first verify that the probabilities q_{G_1} and q_{G_2} are well-defined. It is trivial that both probabilities add up to 1. Hence, we will show that q_{G_1} and q_{G_2} are non-negative. By lemma 7, it follows that

$$\lambda(G) > 0, \quad \frac{1}{V_{G_1}} \geq 1 \quad \text{and} \quad \frac{\pi(G_2)}{V_{G_2}} \leq 1.$$

Hence, q_{G_1} is indeed non-negative. A similar argument shows that q_{G_2} is non-negative, and therefore both probabilities are well-defined. Let us now demonstrate how the optimal searching strategy can be applied to example 3.1.1.

Example 3.2.2 (Optimal Tree Searching Strategy for Example 3.1.1). *Recall that the pure strategies for the searcher are given in 3.4 which are all depth-first searches of G. We will show that the optimal tree searching strategy is a mixed strategy that mixes between the four pure strategies in 3.4. First note that*

$$V_{G(c)} = \frac{3}{8}, \quad V_{G(d)} = \frac{1}{4}, \quad V_{G(e)} = \frac{2}{15}, \quad V_{G(a)} = \frac{145}{896}$$

and

$$\lambda_{G(a)} = \frac{3}{14}, \quad \lambda_G = \frac{290}{3537}$$

which were previously calculated in example 3.2.1. For the root vertex r, it follows that

$$q_{G(a)} = \lambda_G\left(\frac{1}{V_{G(a)}} - \frac{\pi(G(e))}{V_{G(e)}}\right) = \frac{290}{3537}\left(\frac{896}{145} - 1\right) = \frac{1502}{3537}$$
$$q_{G(e)} = \lambda_G\left(\frac{1}{V_{G(e)}} - \frac{\pi(G(a))}{V_{G(a)}}\right) = \frac{290}{3537}\left(\frac{15}{2} - \frac{14}{29}\right) = \frac{2035}{3537}.$$

For branch vertex a, it follows that

$$q_{G(c)} = \lambda_{G(a)}\left(\frac{1}{V_{G(c)}} - \frac{\pi(G(d))}{V_{G(d)}}\right) = \frac{3}{14}\left(\frac{8}{3} - 1\right) = \frac{5}{14}$$
$$q_{G(d)} = \lambda_{G(a)}\left(\frac{1}{V_{G(d)}} - \frac{\pi(G(c))}{V_{G(c)}}\right) = \frac{3}{14}\left(4 - 1\right) = \frac{9}{14}.$$

To conclude, the first pure strategy $\sigma^1 = \{c, d, e\}$ corresponds to searching vertex a over b with probability $q_{G(a)}$ and also searching vertex c over d with probability $q_{G(c)}$. Hence, the optimal tree searching strategy plays σ^1 with probability

$$s_{\sigma^1} = q_{G(a)} \cdot q_{G(c)} = \frac{3755}{24759}.$$

The probabilities of playing the remaining three pure strategies are obtained in a similar way. It follows that the optimal tree searching strategy s_G is a mixed strategy in which the pure strategies of 3.4 are played according to the following probabilities

$$s_G = (s_{\sigma^1}, s_{\sigma^3}, s_{\sigma^5}, s_{\sigma^6}) = \left(\frac{3755}{24759}, \frac{751}{2751}, \frac{10175}{49518}, \frac{2035}{5502}\right).$$

This concludes the optimal tree searching strategy example.

The following lemma shows that the tree searching strategy s_G ensure an expected pay-off of at least V_G against any hider's pure strategy. Since we have already shown that V_G is an upper bound to the value of the game, the lemma proves that V_G is indeed the value of the game and that both strategies h_G and s_G are optimal.

Lemma 8. *Let G be a binary tree and assume the hider plays a pure hiding strategy h. Then the tree searching strategy s_G ensures an expected pay-off of at least V_G.*

Proof. The proof is yet again by induction on the number of vertices of G. Let G be a binary tree. First assume that G only has one vertex r. Then obviously

$$P(s_G) = p_r = V_G$$

by 3.7. Now assume the lemma holds for all binary trees with fewer than n vertices. Let G be a binary tree with n vertices and let r be the root of G. Since G is a binary tree we have that r is a branch vertex. Let G_1 and G_2 be the branches of G. The tree searching strategy s_G will search both G_1 and G_2 in a depth-first way after searching the root r. Assume w.l.o.g. that $h \in V(G_1)$. By applying the induction hypothesis on G_1 and G_2, and using 3.18 and 3.11, it follows that

$$\begin{aligned} P(s_G) &\geq p_r\big(q_{G_1}V_{G_1} + q_{G_2}\pi(G_2)V_{G_1}\big) \\ &= p_r\lambda_G\left(\frac{1}{V_{G_1}} - \frac{\pi(G_2)}{V_{G_2}}\right)V_{G_1} + p_r\lambda_G\left(\frac{1}{V_{G_2}} - \frac{\pi(G_1)}{V_{G_1}}\right)\pi(G_2)V_{G_1} \\ &= p_r\lambda_G\Big(1-\pi(G_1)\pi(G_2)\Big) = V_G. \end{aligned}$$

Hence, tree searching strategy s_G ensures an expected pay-off of at least V_G. □

From lemma 6 and 8, it follows that formula for the value V_G in 3.11 is indeed the value of the game. Furthermore, the tree hiding and searching strategies h_G and s_G defined in definition 3.2.2 and 3.2.3 respectively are indeed optimal strategies.

3.3. Binary Tree Reconstruction from an Oracle Function

In the previous sections, the search and rescue game on an arbitrary tree has been defined and solved. This is done by first transforming the tree into a game equivalent binary tree (if necessary), and then solving the game played on the binary tree. Every binary tree can be uniquely represented by its total success probability function f as defined in definition 3.1.11. For example, the tree depicted in figure 3.4 can be represented by a function f with values given in table 3.1.

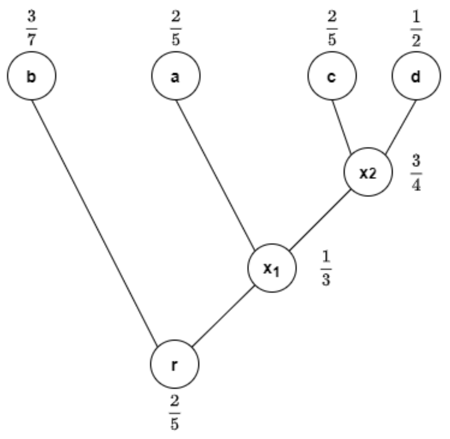


Figure 3.4: Tree Reconstruction Using Dilworth Truncation

Typically, only the values of the function f are given by some oracle. As a result, the structure of the tree and the success probabilities $\underline{p}$ remain unknown. We know from the previous section that if a function f has a tree representation, then the corresponding game can be solved. However, this solution makes use of

Table 3.1: Oracle Function f of the Tree in Figure 3.4

S	$\emptyset$	a	b	c	d	ab	ac	ad	bc	bd	cd	abc	abd	acd	bcd	abcd
f	1	$\frac{4}{75}$	$\frac{6}{35}$	$\frac{1}{25}$	$\frac{1}{20}$	$\frac{4}{175}$	$\frac{2}{125}$	$\frac{1}{50}$	$\frac{3}{175}$	$\frac{3}{140}$	$\frac{1}{50}$	$\frac{6}{875}$	$\frac{3}{350}$	$\frac{1}{125}$	$\frac{3}{350}$	$\frac{3}{875}$

the structure of the tree which is now unknown. Therefore, the goal of this section is to find a method to reconstruct a tree from an oracle function f given that it has a tree representation.

Furthermore, not all oracle functions allow a tree representation. There are certain properties that f must satisfy in order to allow a tree representation. The goal of this section is to find all necessary requirements on f such that it has a tree representation, and to find a method to reconstruct a tree from an oracle function f given that it has a tree representation.

Let us first consider the binary tree in figure 3.4 for which the values of its function f are given in table 3.1. Since f has a tree representation, we might be able to observe from table 3.1 that f satisfies certain properties. The first observation is that f is a decreasing set function.

Definition 3.3.1 (**Decreasing Set Function**). *Let L be a set. A set function $f : 2^L \to (0,1)$ is decreasing iff*

$$f(A) \geq f(B), \quad \forall A, B \in 2^L : A \subseteq B. \tag{3.19}$$

Proposition 3.3.1. *Let L be the set of leaf vertices and let $f : 2^L \to (0,1)$ be an oracle function. If f has a tree representation, then it must be decreasing.*

Proof. The proof is by contradiction. Assume that $\exists A, B \in 2^L : A \subseteq B$ and $f(A) < f(B)$. Trivially, if $A = B$ then $f(A) = f(B)$ which is a contradiction. Therefore, assume that $A \neq B$ and let $x_1, ..., x_n$ be all vertices of A and let $y_1, ..., y_m$ be all vertices of $B \setminus A$. Since $B = (B \setminus A) \cup A$ and both sets on the right-hand side are disjoint, it follows that

$$f(A) = \prod_{i=1}^{n} p_{x_i} > \prod_{i=1}^{n} p_{x_i} \prod_{j=1}^{n} p_{y_m} = f(B),$$

which is a contradiction. Hence, f must be decreasing. □

Proposition 3.3.1 states the first requirement on the oracle function f. The requirement that f is a decreasing set function is very intuitive. If searching a set of leaf vertices has a certain success probability, then by searching more vertices the total success probability can not be any greater.

The second requirement on the oracle function f is somewhat harder to derive from table 3.1. This requirement is similar to the notion of submodular and supermodular function Narayanan [10].

Definition 3.3.2 (**Submodular/Supermodular Function**). *Let L be a set. A set function $f : 2^L \to (0,1)$ is submodular iff*

$$f(A \cup B) + f(A \cap B) \leq f(A) + f(B), \quad \forall A, B \in 2^L. \tag{3.20}$$

Similarly, it is supermodular iff

$$f(A \cup B) + f(A \cap B) \geq f(A) + f(B), \quad \forall A, B \in 2^L. \tag{3.21}$$

Proposition 3.3.2. *Let L be the set of leaf vertices and let $f : 2^L \to (0,1)$ be an oracle function. If f has a tree representation, then it must be that*

$$f(A \cup B) f(A \cap B) \geq f(A) f(B), \quad \forall A, B \in 2^L. \tag{3.22}$$

We call functions that satisfy 3.22 multiplicative supermodular functions.

Proof. The proof is by contradiction. Assume $\exists A, B \in 2^L : f(A \cup B) f(A \cap B) < f(A) f(B)$. There are two cases:

1: $A \cap B = \emptyset$. First note that $(A \cup B)' = A' \cup B'$ in which the elements of $A' \cap B'$ are counted twice in the right-hand side. Hence, it follows that

$$(A \cup B)' = A' \cup (B' \setminus A')$$

in which both sets on the right-hand side are disjoint. By definition 3.1.11, it follows that

$$f(A \cup B) f(A \cap B) = \prod_{i \in (A \cup B)'} p_i = \prod_{i \in A'} p_i \prod_{j \in B' \setminus A'} p_j > \prod_{i \in A'} p_i \prod_{j \in B'} p_j = f(A) f(B).$$

2: $A \cap B \neq \emptyset$. First observe that the following relation holds

$$(A \cap B)' \subseteq (A' \cap B'), \tag{3.23}$$

see appendix A.2.3. Hence, it follows from definition 3.1.11 that

$$\begin{aligned} f(A \cup B) f(A \cap B) &= \prod_{i \in (A \cup B)'} p_i \prod_{k \in (A \cap B)'} p_k = \prod_{i \in A'} p_i \prod_{j \in B' \setminus A'} p_j \prod_{k \in (A \cap B)'} p_k \\ &\geq \prod_{i \in A'} p_i \prod_{j \in B' \setminus A'} p_j \prod_{k \in A' \cap B'} p_k = \prod_{i \in A'} p_i \prod_{j \in B'} p_j = f(A) f(B). \end{aligned}$$

Both cases lead to a contradiction which proves the proposition. □

Proposition 3.3.2 states the second requirement of the oracle function f and closely resembles the definition of supermodular functions, but now the inequality is multiplicative instead of additive. This requirement is less intuitive than the requirement of f being decreasing. It more or less says that if f has a tree representation and A and B are two sets of leaf vertices, then the product of the vertices that A' and B' have in common (leaf and branch vertices) is at most $f(A \cap B)$.

The requirement that f is decreasing and multiplicative supermodular are not sufficient to prove that f has a tree representation. Furthermore, we also can not derive any more requirements solely from looking at the values in table 3.1. However, there is a third requirement on f that we found when we tried to reconstruct a tree from a function f.

Conjecture 3.3.1. *Let L be the set of leaf vertices and let $f : 2^L \to (0,1)$ be an oracle function. If f has a tree representation, then for any subset $X \subseteq L$ there exists a split of X into two non-empty sets A and A^c, and a constant c such that*

$$c \cdot f(Y) = f(Y \cap A) f(Y \cap A^c), \quad \forall Y \subseteq X. \tag{3.24}$$

We call functions f for which this property holds totally reducible.

It becomes more evident later on (when we try to reconstruct a tree from a function f) why we think that conjecture 3.3.1 is a necessary requirement on f. For now, let us move on and try to reconstruct a tree from an oracle function f, given that f has a tree representation.

In Narayanan [10] an operation called the Dilworth Truncation is used to solve a version of the hybrid rank problem. In this problem, a graph $G = (V, E)$ needs to be partitioned or split in two sets A and A^c such that an objective function is minimized. The idea is to use an operation similar to the Dilworth Truncation to reconstruct a binary tree from an oracle function f. This operation iterative splits a set of leaf vertices into two groups (branches) by maximizing an objective function. Every split corresponds to a branch vertex that connects its two branches. Hence, the structure of the tree can be recovered by iteratively splitting the entire set of the leaf vertices.

Before defining the operation, we introduce the following notation. Let $L = \{y_1, ..., y_m\}$ be the set of leaf vertices. We denote

$$\{y_1, ..., y_k | y_{k+1}, ..., y_m\}, \quad 1 \leq k \leq m$$

as the split (or partition) of L into two sets $\{y_1, ..., y_k\}$ and $\{y_{k+1}, ..., y_m\}$. If any of the two sets is not the singleton set, i.e. the set with only one vertex, then it can be further split in two similar as before. For example, we denote

$$\{y_1, ..., y_j || y_{j+1}, ..., y_k | y_{k+1}, ..., y_l || y_{l+1}, ..., y_m\}, \quad 1 \leq j < k < l \leq m$$

as the split of L similar as before where in addition, the set $\{y_1, ..., y_k\}$ and $\{y_{k+1}, ..., y_m\}$ are both split in two once more. The Dilworth Truncation operation to reconstruct a tree from an oracle function f is stated as follows.

Definition 3.3.3 (Dilworth Truncation Operation for Binary Tree Reconstruction). *Let $L = \{y_1, ..., y_m\}$ be the set of leaf vertices and let $f : 2^L \to (0,1)$ be some oracle function for which a tree representation exists. For a subset $X \subseteq L$, we split X into two non-empty sets A and $A^c = (X \setminus A)$ such that the excess defined as*

$$e(A, X) = \frac{f(A) f(A^c)}{f(X)} \tag{3.25}$$

is maximised over A. Every split of a set X into two sets A and A^c corresponds to a branch vertex x with branches A and A^c. Let $x_1,...,x_i$ be the branch vertices of the splits that occurred prior to x, then the success probability of x is given by

$$p_x = \frac{e(A,X)}{\prod_{k=1}^{i} p_{x_k}}. \tag{3.26}$$

The tree can be reconstructed by iteratively splitting L until there are only singleton sets left. Let y_l be a leaf vertex which is connected to the branch vertices $x_1,...,x_j$. Then the success probability of y_l is given by

$$p_{y_l} = \frac{f(l)}{\prod_{k=1}^{j} p_{x_k}}. \tag{3.27}$$

Let us give a small example of how to apply the Dilworth Truncation operation to reconstruct a tree from an oracle function f, given that such a representation exists.

Example 3.3.1 (**Binary Tree Reconstruction of figure 3.4 from table 3.1**). *Let $L = \{a,b,c,d\}$ be the set of leaf vertices. We first split the entire set L until only singleton sets remain. For the first split, let $X = L$. There are a total of seven unique splits*

$$L_1^{(1)} = \{a|b,c,d\},\quad L_2^{(1)} = \{b|a,c,d\},\quad L_3^{(1)} = \{c|a,b,d\},\quad L_4^{(1)} = \{d|a,b,c\},$$

$$L_5^{(1)} = \{a,b|c,d\},\quad L_6^{(1)} = \{a,c|b,d\},\quad L_7^{(1)} = \{a,d|b,c\}.$$

For every split, the excess defined in 3.25 is given by

$$e(\{a\},\{a,b,c,d\}) = \frac{2}{15},\quad e(\{b\},\{a,b,c,d\}) = \frac{2}{5},\quad e(\{c\},\{a,b,c,d\}) = \frac{1}{10},\quad e(\{d\},\{a,b,c,d\}) = \frac{1}{10},$$

$$e(\{a,b\},\{a,b,c,d\}) = \frac{2}{15},\quad e(\{a,c\},\{a,b,c,d\}) = \frac{1}{10},\quad e(\{a,d\},\{a,b,c,d\}) = \frac{1}{10}.$$

The excess is maximised when X is split as in $L_2^{(1)}$. Hence, we first split b from the rest as follows

$$\{b|a,c,d\}.$$

This split indicates that there is a branch vertex r (the root) that connects the branches $\{b\}$ and $\{a,c,d\}$. Now we iteratively continue with the branches of r. Since $\{b\}$ is a singleton set, we can not split it further. Therefore, we continue and split the other branch $\{a,c,d\}$. Let $X = \{a,c,d\}$, then there are a total of three unique splits of X, namely

$$L_1^{(2)} = \{a||c,d\},\quad L_2^{(2)} = \{c||a,d\},\quad L_3^{(2)} = \{d||a,c\}.$$

The excess of every split is equal to

$$e(\{a\},\{a,c,d\}) = \frac{2}{15},\quad e(\{c\},\{a,c,d\}) = \frac{1}{10},\quad e(\{d\},\{a,c,d\}) = \frac{1}{10}$$

and maximised when X is split as in $L_1^{(2)}$. Hence, we split $\{a\}$ from $\{c,d\}$ as follows

$$\{b|a||c,d\}.$$

The second split tells us that there is a branch vertex x_1, that is connected to r, and connects the branches $\{a\}$ and $\{c,d\}$. Continuing iteratively, we skip the singleton set $\{a\}$ and split the branch $\{c,d\}$. Note that the branch only has one trivial split, and therefore we split the entire set L as follows

$$\{b|a||c|||d\}.$$

The last split of the set $\{c,d\}$ tells us that there is a branch vertex x_2, that is connected to x_1 which in turn is connected to r, and connects the branches $\{c\}$ and $\{d\}$. The structure of the tree has now been recovered. It remains to compute the success probabilities of the vertices.

Note that the root r is connected to x_1 which is connected to x_2. Furthermore, r splits $\{b\}$ from $\{a,c,d\}$, x_1

splits $\{a\}$ from $\{c,d\}$ and x_2 splits $\{c\}$ from $\{d\}$. Therefore, by 3.26 it follows that the success probabilities for the branch vertices are given by

$$p_r = e\big(\{b\},\{a,b,c,d\}\big) = \frac{2}{5}, \quad p_{x_1} = \frac{e\big(\{a\},\{a,c,d\}\big)}{p_r} = \frac{\frac{2}{15}}{\frac{2}{5}} = \frac{1}{3}, \quad p_{x_2} = \frac{e\big(\{c\},\{c,d\}\big)}{p_r p_{x_1}} = \frac{\frac{1}{10}}{\frac{2}{15}} = \frac{3}{4}.$$

For the leaf vertices, note that b is connected to r, a is connected to x_1 which is connected to r and both c and d are connected to x_2 which is connected to x_1 which is connected to r. Therefore, it follows from 3.27 that the success probabilities for the leaf vertices are given by

$$p_b = \frac{f(b)}{p_r} = \frac{3}{7}, \quad p_a = \frac{f(a)}{p_r p_{x_1}} = \frac{2}{5}, \quad p_c = \frac{f(c)}{p_r p_{x_1} p_{x_2}} = \frac{2}{5}, \quad p_d = \frac{f(d)}{p_r p_{x_1} p_{x_2}} = \frac{1}{2}.$$

This concludes the tree reconstruction example.

By iteratively splitting the tree as in definition 3.3.3, it is easy to verify that we eventually obtain a binary tree. Let X be the set of leaf vertices. Since f has a binary tree representation, we can split X into two sets A and A^c such that the subtrees spanned by both sets only have the root vertex in common. Such a split is found exactly when the excess defined in 3.25 is maximised.

Furthermore, since the subtrees only have the root vertex in common, the success probability of the root vertex must be equal to the excess. For subsequent splits (branch vertices), the subtrees of the sets A and A^c have one branch vertex (belonging to the current split) and multiple previously restored branch vertices (belonging to the splits prior to the current one) in common. Hence, the success probability of the branch vertex belonging to the current split must be exactly as in 3.26.

The success probability of a leaf vertex must also be given as in 3.27, because the leaf vertex is connected by a sequence of branch vertices. Hence, by dividing the total success probability of a leaf vertex with the success probabilities of all branch vertices spanned by the subtree of the leaf vertex, we obtain the success probability of the leaf vertex.

Conjecture 3.3.1 follows from the Dilworth Truncation operation that reconstructs tree from f. It says that if f has a tree representation, then for any subset of leaf vertices $X \subseteq L$ there exist a split of X into two branches A and A^c such that

$$c = \frac{f(Y \cap A) f(Y \cap A^c)}{f(Y)}, \quad \forall Y \subseteq X.$$

If we think about this in terms of a tree, it is the equivalent of saying that there exist a set of branch vertices B such that $(Y \cap A)' \cap (Y \cap A^c)' = B$ for all subsets $Y \subseteq X$. In other words, for any subset of leaf vertices $Y \subseteq X$, $f(Y)$ always contains the product of the success probabilities of the branch vertices in B.

There are two open questions in this section that we leave to our successor. The first question is whether or not it is possible to prove conjecture 3.3.1. The split X that is needed in the conjecture is most likely the split with maximal excess. With such a split and using that f has a binary tree representation, it might be possible to prove the conjecture.

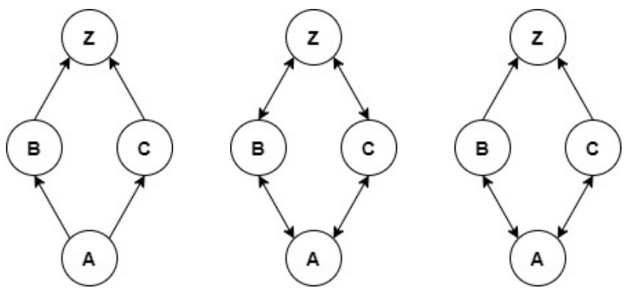
The second question relates to the requirements on the oracle function f such that it has a tree representation. In order to prove that a function f has a tree representation, is it sufficient that f is decreasing, multiplicative supermodular and totally reducible? It might be possible to prove this question by showing that the Dilworth Truncation operation in definition 3.3.3 is able to reconstruct a binary tree from f, given that f satisfies the three aforementioned requirements. However, we have yet to find an answer to both of these questions.

4

Search and Rescue Games on a Graph

In this chapter, we extend the search and rescue game on a tree that has been defined in the previous chapter. Recall that trees are graphs such that any two vertices are connected with exactly one path. In general, a graph does not necessarily need to be a tree so that there can be more than one path between any two vertices. Therefore, we are interested in how the game changes when it is played on a graph that is not a tree. By playing the search and rescue game on a graph, the game becomes more complicated. Therefore, we will restrict ourselves to very simple graphs that have a sink vertex (i.e. a vertex of no return). There might exist a general solution for graphs that have a sink vertex, this chapter is a first step towards such a solution.

Let n be the number of vertices of the graph. The graphs that we will consider in this chapter have a starting vertex A (also called a source vertex) and a vertex of no return Z (also called a sink vertex). The remaining $m = n - 2$ vertices are called the in-between vertices, these in-between vertices are all directly connected to the source and sink vertex. The edges of the graph can be both directed and/or undirected. With the addition of the sink vertex Z, it is possible that the game will end prematurely when it has been searched. This happens independent on whether or not it was searched successfully. Figure 4.1 shows three examples of the aforedescribed graphs for $n = 4$, the graphs (and also the corresponding game) differ from one another depending on the directionality of the edges.



(a) Type 1: Directed Edges

(b) Type 2: Undirected Edges

(c) Type 3: Directed & Undirected Edges

Figure 4.1: Search and Rescue Games on a Graph for $n = 4$

The game on a graph is similar to that on a tree. The hider hides one object in any of the vertices which the searcher has to find. The searcher has to start at the source vertex A. Every vertex has a certain probability of being searched successfully and at any point in the game, the searcher can only search a vertex if the structure of the graph allows it. Every section in this chapter is dedicated to the game played on one of the three graph types as in figure 4.1.

Before we jump into the analysis of the games, we define some more notation. For any vertex X of a graph, we denote $h(X)$ as the pure strategy of the hider that hides the object in vertex X. Furthermore, we denote

h_X as the probability of the hider playing $h(X)$. For any in-between vertex X of the graph, we denote $s(X)$ as the pure strategy of the searcher that searches A, X and then Z in that order. We also denote s_X as the probability of the searcher playing $s(X)$. Similarly, for two in-between vertices X_1 and X_2 we denote $s(X_1, X_2)$ as the pure strategy of the searcher that searches A, X_1, X_2 and then Z in that order, and we denote $s_{X_1X_2}$ as the probability of the searcher playing $s(X_1, X_2)$.

4.1. Game 1: Search and Rescue Game with Directed Edges

In this section, we consider the search and rescue games on the type one graphs as in figure 4.1a. All edges of the graph are either directed from the source vertex to any in-between vertex, or from any in-between vertex to the sink vertex. We will refer to the aforedescribed game as the search and rescue game of type one with n vertices. Figure 4.2 shows three examples of the search and rescue game of type one.

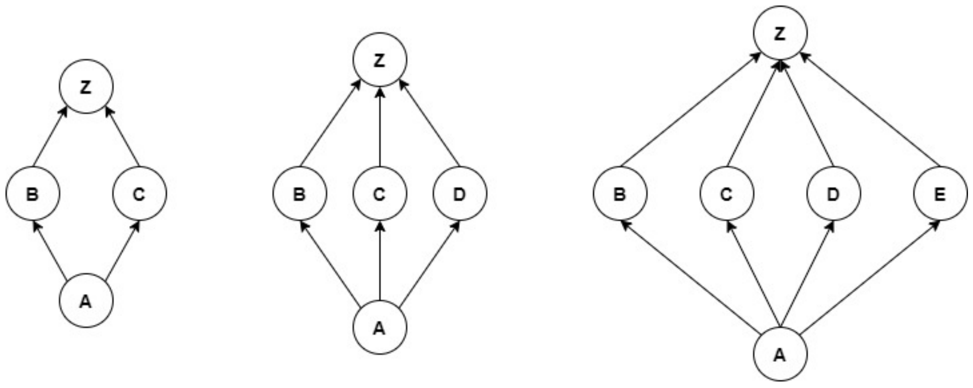


Figure 4.2: Type One Search and Rescue Game for $n = 4, 5$ and 6

The hider hides the object in any of the vertices of the graph. It is trivial that hiding the object in A is being dominated. Hence, the hider has exactly $n-1$ pure strategies. The searcher needs to choose through which in-between vertex she will traverse to the sink vertex. There are exactly $m = n-2$ in-between vertices, so that the searcher has m pure strategies. Let us first consider the simplest case where $n = 4$ (and $m = 2$) and try to solve this game. The graph belonging to this game is depicted in figure 4.2.

The searcher has two pure strategies. She can either take the left path with $s(B)$ or the right path with $s(C)$. The hider on the other hand has three pure strategies, namely $h(B)$, $h(C)$ and $h(Z)$. Since the hider and searcher only have two and three pure strategies respectively, we can easily compute the pay-off matrix for this game. In general, the hider and searcher have $n-1$ and m pure strategies respectively, so that the pay-off matrix is of size $m \times (n-1)$ which is still feasible for small n. The corresponding matrix game can now be solved using a standard matrix game solver. For the graph with $n = 4$, the pay-off matrix is given as follows

$$\begin{array}{cc} & \begin{array}{ccc} B & C & Z \end{array} \\ \begin{array}{c} (ABZ) \\ (ACZ) \end{array} & \begin{pmatrix} p_A p_B & 0 & p_A p_B p_Z \\ 0 & p_A p_C & p_A p_C p_Z \end{pmatrix} \end{array}. \tag{4.1}$$

To analyse the search and rescue games of type one, we have written a code that constructs the pay-off matrix given the success probabilities $\underline{p} = (p_A, ..., p_Z)$. The corresponding matrix game can then be solved by a matrix game solver. The code for the construction of the pay-off matrix can be found in appendix A.1.5, and the code for the matrix game solver can be found in appendix A.1.4.

The idea is to first solve the simplest case where $n = 4$ and try to generalise its solution. To that end, we will need to analyse the behaviour of both players. This is done by generating random probabilities for $\underline{p}$ and solving the corresponding matrix game. For every random probability $\underline{p}$ and corresponding solution, we observe which pure strategies both players mix in their optimal strategy. By repeating this several times, we are able to make the following observations:

- The first observation is that the searcher always mixes both of her pure strategies. In other words, the searcher always plays a mixed strategy $\underline{s} = (s_B, s_C)$ which plays $s(B)$ or $s(C)$ with some probability s_B and s_C respectively.

- The second observation is that the hider will not hide the object in Z when $p_Z \geq \frac{1}{2}$. Hence, the hider plays a mixed strategy $\underline{h} = (h_B, h_C, h_Z) = (h_B, h_C, 0)$ which plays $h(B)$ and $h(C)$ with some probability h_B and h_C respectively. Furthermore, the probabilities h_B and h_C are identical to the probabilities s_B and s_C respectively.
- The last observation is that the hider will not hide the object in either B or C when $p_Z < \frac{1}{2}$. This is dependent on the probabilities p_B and p_C, where the vertex with the higher success probability will not be played. Hence, the hider will mix $h(Z)$ with either $h(B)$ or $h(C)$.

By observation, we now know which pure strategies both players will mix in their optimal strategy. The game is now easily solved, the result has need summarised in the following theorem.

Theorem 2. *Consider the search and rescue game of type one with $n = 4$. The graph of this game is given in figure 4.2 with pay-off matrix given in 4.1. The solution of this game is dependent on the success probability p_Z.*

– *If $p_Z \geq \frac{1}{2}$, it is optimal for the searcher to use the mixed strategy $\underline{s} = (s_B, s_C)$ which plays $s(B)$ and $s(C)$ with probability s_B and s_C respectively given by*

$$\underline{s} = (s_B, s_C) = \left(\frac{1}{p_B}, \frac{1}{p_C}\right)\lambda, \quad \lambda = \frac{1}{\frac{1}{p_B} + \frac{1}{p_C}}. \tag{4.2}$$

For the hider, it is optimal to use the mixed strategy $\underline{h} = (h_B, h_C, 0)$ which plays $h(B)$ and $h(C)$ with probability $h_B = s_B$ and $h_C = s_C$ respectively as defined in 4.2. The value of the game is given by

$$V_G = \lambda p_A. \tag{4.3}$$

– *If $p_Z < \frac{1}{2}$, assume w.l.o.g. that $p_B \leq p_C$. Then it is optimal for the searcher to use the mixed strategy $\underline{s} = (s_B, s_C)$ which plays $s(B)$ and $s(C)$ with probability s_B and s_C respectively given by*

$$\underline{s} = (s_B, s_C) = \left(\frac{1}{p_B}, \frac{1}{p_C p_Z} - \frac{1}{p_C}\right)\mu, \quad \mu = \frac{1}{\frac{1}{p_B} + \frac{1}{p_C p_Z} - \frac{1}{p_C}}. \tag{4.4}$$

For the hider, it is optimal to use the mixed strategy $\underline{h} = (h_B, 0, h_Z)$ which plays $h(B)$ and $h(Z)$ with probability h_B and h_Z respectively given by

$$\underline{h} = (h_B, 0, h_Z) = \left(\frac{p_C}{p_B} - 1, 0, \frac{1}{p_Z}\right)\xi, \quad \xi = \frac{1}{\frac{p_C}{p_B} - 1 + \frac{1}{p_Z}}. \tag{4.5}$$

The value of the game is given by

$$V_G = \mu p_A. \tag{4.6}$$

In the case where $p_C \leq p_B$, the optimal strategies are defined in a similar and symmetrical way.

Proof. First observe that the probabilities defined in 4.2, 4.4 and 4.5 are all non-negative and well-defined. This follows from the fact that

- $p_B, p_C, p_Z \in (0, 1)$
- $p_B \leq p_C$, for 4.5

Let $p_Z \geq \frac{1}{2}$ and assume the searcher plays $\underline{s} = (s_B, s_C)$ as in 4.2. The expected pay-off is dependent on what hiding strategy the hider plays. The hider has three pure strategies: $h = B$, $h = C$ and $h = Z$, and the expected pay-off against any pure strategy of the hider is given by

$$\begin{array}{ccccccc} B & C & Z & & B & C & Z \\ \Big(\frac{\lambda}{p_B}\cdot p_A p_B & \frac{\lambda}{p_C}\cdot p_A p_C & \frac{\lambda}{p_B}\cdot p_A p_B p_Z + \frac{\lambda}{p_C}\cdot p_A p_C p_Z\Big) & = & (\lambda p_A & \lambda p_A & 2\lambda p_A p_Z) \\ & & & & B & C & Z \\ & & & \geq & (\lambda p_A & \lambda p_A & \lambda p_A), \end{array}$$

where we use that $p_Z \geq \frac{1}{2}$. Hence, the searcher can guarantee a pay-off of at least λp_A with $\underline{s}$ and therefore $V_G \geq \lambda p_A$. Now assume the hider plays the strategy $\underline{h} = (h_B, h_C, 0)$ with probabilities equivalent to 4.2. Then the expected pay-off is dependent on what searching strategy the searcher plays. The searcher has two pure strategies: $s = (ABZ)$ or $s = (ACZ)$, and the expected pay-off against any pure strategy of the searcher is given by

$$\begin{array}{c} (ABZ) \\ (ACZ) \end{array} \begin{pmatrix} \frac{\lambda}{p_B} \cdot p_A p_B \\ \frac{\lambda}{p_C} \cdot p_A p_C \end{pmatrix} = \begin{pmatrix} \lambda p_A \\ \lambda p_A \end{pmatrix}.$$

Hence, the hider can ensure that the pay-off is at most λp_A with $\underline{h}$ and therefore $V_G \leq \lambda p_A$. As a result, $V_G = \lambda p_A$ is the value of the game and the strategies $\underline{s}$ and $\underline{h}$ are indeed optimal.

Now let $p_Z < \frac{1}{2}$ and assume w.l.o.g. that $p_B \leq p_C$ and that the searcher plays $\underline{s} = (s_B, s_C)$ as in 4.4. It follows that the expected pay-off is given by

$$\begin{array}{ccc} B & C & Z \\ \Big(\mu p_A & \mu p_A\Big(\frac{1}{p_Z} - 1\Big) & \mu p_A p_Z + \mu p_A p_Z\Big(\frac{1}{p_Z} - 1\Big)\Big) \end{array} \geq \begin{array}{ccc} B & C & Z \\ (\mu p_A & \mu p_A & \mu p_A) \end{array},$$

where we use that $p_Z < \frac{1}{2}$. Hence, the searcher can guarantee a pay-off of at least μp_A with $\underline{s}$ and therefore $V_G \geq \mu p_A$. Now assume the hider plays the strategy $\underline{h} = (h_B, 0, h_Z)$ as in 4.5. Then the expected pay-off is given by

$$\begin{array}{c} (ABZ) \\ (ACZ) \end{array} \begin{pmatrix} \xi p_A p_B \big(\frac{p_C}{p_B} - 1\big) + \xi p_A p_B \\ \xi p_A p_C \end{pmatrix} = \begin{pmatrix} \xi p_A p_C \\ \xi p_A p_C \end{pmatrix}.$$

Hence, the hider can ensure that the pay-off is at most $\xi p_A p_C$ with $\underline{h}$ and therefore $V_G \leq \xi p_A p_C$. Finally, notice that

$$\mu p_A = \frac{p_A}{\frac{1}{p_B} + \frac{1}{p_C}\Big(\frac{1}{p_Z} - 1\Big)} = \frac{p_A p_C}{\frac{p_C}{p_B} + \frac{1}{p_Z} - 1} = \xi p_A p_C.$$

As a result, $V_G = \mu p_A$ is the value of the game and the strategies $\underline{s}$ and $\underline{h}$ are optimal. □

Theorem 2 solves the search and rescue game of type one with $n = 4$. The solution to this game is dependent on the success probabilities $\underline{p} = (p_A, p_B, p_C, p_Z)$. It is optimal for the searcher to mix all her pure strategies, whereas it is optimal for the hider to not mix $h(A)$ and one other pure strategy depending on $\underline{p}$. By adding more in-between vertices to the game and by solving the corresponding matrix game, we have discerned that the solution to the game with general n is a generalisation of theorem 2.

Theorem 3. *Consider the search and rescue game of type one with $n \geq 4$ vertices and m in-between vertices. Let $X_1, \ldots, X_m$ be all in-between vertices and let X_k be the in-between vertex with the highest success probability, i.e.*

$$p_{X_k} \geq p_{X_i}, \quad \forall i = 1, \ldots, m.$$

The solution of this game is depedent on the success probability p_Z.

- *If $p_Z \geq \frac{1}{m}$, it is optimal for the searcher to use the mixed strategy $\underline{s} = (s_{X_1}, \ldots, s_{X_m})$ which plays any $s(X_i)$ with probability s_{X_i} given by*

$$s_{X_i} = \frac{\lambda}{p_{X_i}}, \quad \lambda = \left(\sum_{i=1}^{m} \frac{1}{p_{X_i}}\right)^{-1}, \quad i = 1, \ldots, m. \tag{4.7}$$

  For the hider, it is optimal to use the mixed strategy $\underline{h} = (h_{X_1}, \ldots, h_{X_m}, h_Z) = (h_{X_1}, \ldots, h_{X_m}, 0)$ which plays any $h(X_i)$ with probability $h_{X_i} = s_{X_i}$ as defined in 4.7. The value of the game is given by

$$V_G = \lambda p_A. \tag{4.8}$$

- *If $p_Z < \frac{1}{m}$, it is optimal for the searcher to use the mixed strategy $\underline{s} = (s_{X_1}, \ldots, s_{X_m})$ which plays any $s(X_i)$ with probability s_{X_i} given by*

$$s_{X_i} = \frac{\mu}{p_{X_i}}, \quad i \in \{1, \ldots, m \mid i \neq k\}, \tag{4.9}$$

$$s_{X_k} = \frac{\mu}{p_{X_k}}\left(\frac{1}{p_Z} - m + 1\right), \quad \mu = \left(\frac{1}{p_{X_k}p_Z} - \frac{m+1}{p_{X_k}} + \sum_{\substack{i=1\\ i\neq k}}^{m} \frac{1}{p_{X_i}}\right)^{-1}. \tag{4.10}$$

For the hider, it is optimal to use the mixed strategy $\underline{h} = (h_{X_1}, ..., h_{X_m}, h_Z)$ which plays any $h(X_i)$ and $h(Z)$ with probability h_{X_i} and h_Z respectively given by

$$h_{X_i} = \left(\frac{p_{X_k}}{p_{X_i}} - 1\right)\xi, \quad i = 1, ..., m, \tag{4.11}$$

$$h_Z = \frac{\xi}{p_Z}, \quad \xi = \left(\frac{1}{p_Z} + \sum_{i=1}^{m}\left(\frac{p_{X_k}}{p_{X_i}} - 1\right)\right)^{-1}. \tag{4.12}$$

The value of the game is given by

$$V_G = \mu p_A. \tag{4.13}$$

Proof. First observe that the probabilities defined in 4.7, 4.9, 4.10, 4.11 and 4.12 are all non-negative and well-defined. This follows from the fact that

- $p_A, p_Z, p_{X_1}, ..., p_{X_m} \in (0, 1)$
- $p_Z < \frac{1}{m}$, for 4.10
- $p_{X_k} \geq p_{X_i}, \quad i = 1, ..., m$, for 4.11

Let $p_Z \geq \frac{1}{m}$ and assume the searcher plays $\underline{s} = (s_{X_1}, ..., s_{X_m})$ as in 4.7. We will show that with the searching strategy $\underline{s}$, the searcher can guarantee a pay-off of at least λp_A. There are two cases:

– The hider hides the object in X_i so that the expected pay-off is given by

$$P\big(h(X_i), \underline{s}\big) = \frac{\lambda}{p_{X_i}} \cdot p_A p_{X_i} = \lambda p_A, \quad i = 1, ..., m.$$

– The hider hides the object in Z so that the expected pay-off is given by

$$P\big(h(Z), \underline{s}\big) = \sum_{i=1}^{m}\left(\frac{\lambda}{p_{X_i}} \cdot p_A p_{X_i} p_Z\right) = \sum_{i=1}^{m} \lambda p_A p_Z = m\lambda p_A p_Z \geq \lambda p_A,$$

where we use that $p_Z \geq \frac{1}{m}$.

Hence, the searcher can guarantee a pay-off of at least λp_A with $\underline{s}$ and therefore $V_G \geq \lambda p_A$. Now assume the hider plays the strategy $\underline{h} = (h_{X_1}, ..., h_{X_m}, 0)$ with probabilities equivalent to 4.7. Then the expected pay-off against any pure strategy $s(X_i)$ of the searcher is given by

$$P\big(\underline{h}, s(X_i)\big) = \frac{\lambda}{p_{X_i}} \cdot p_A p_{X_i} + 0 \cdot p_A p_{X_i} p_Z = \lambda p_A, \quad i = 1, ..., m.$$

Hence, the hider can ensure that the pay-off is at most λp_A with $\underline{h}$ and therefore $V_G \leq \lambda p_A$. As a result, $V_G = \lambda p_A$ is the value of the game and the strategies $\underline{s}$ and $\underline{h}$ are indeed optimal.

Now let $p_Z < \frac{1}{m}$ and assume that the searcher plays $\underline{s} = (s_{X_1}, ..., s_{X_m})$ as in 4.9 and 4.10. We will show that with the searching strategy $\underline{s}$, the searcher can guarantee a pay-off of at least μp_A. There are three cases:

– The hider hides the object in X_k so that the expected pay-off is given by

$$P\big(h(X_k), \underline{s}\big) = \frac{\mu}{p_{X_k}}\left(\frac{1}{p_Z} - m + 1\right) \cdot p_A p_{X_k} = \mu p_A\left(\frac{1}{p_Z} - m + 1\right) \geq \mu p_A,$$

where we use that $p_Z < \frac{1}{m}$.

- The hider hides the object in $X_i \neq X_k$ so that the expected pay-off is given by

$$P\big(h(X_i), \underline{s}\big) = \frac{\mu}{p_{X_i}} \cdot p_A p_{X_i} = \mu p_A, \quad i \in \{1, \dots, m \mid i \neq k\}.$$

- The hider hides the object in Z so that the expected pay-off is given by

$$\begin{aligned} P\big(h(Z), \underline{s}\big) &= \frac{\mu}{p_{X_k}} \left(\frac{1}{p_Z} - m + 1\right) \cdot p_A p_{X_k} p_Z + \sum_{\substack{i=1 \\ i \neq k}}^{m} \left(\frac{\mu}{p_{X_i}} \cdot p_A p_{X_i} p_Z\right) \\ &= \mu p_A p_Z \left(\frac{1}{p_Z} - m + 1\right) + (m-1)\mu p_A p_Z = \mu p_A. \end{aligned}$$

Hence, the searcher can guarantee a pay-off of at least μp_A with $\underline{s}$ and therefore $V_G \geq \mu p_A$. Now assume the hider plays the strategy $\underline{h} = (h_{X_1}, \dots, h_{X_m}, h_Z)$ as in 4.11 and 4.12. Then the expected pay-off against any pure strategy $s(X_i)$ of the searcher is given by

$$P\big(\underline{h}, s(X_i)\big) = \left(\frac{p_{X_k}}{p_{X_i}} - 1\right)\xi \cdot p_A p_{X_i} + \frac{\xi}{p_Z} \cdot p_A p_{X_i} p_Z = \xi p_A p_{X_k}, \quad i = 1, \dots, m.$$

Hence, the hider can ensure that the pay-off is at most $\xi p_A p_{X_k}$ with $\underline{h}$ and therefore $V_G \leq \xi p_A p_{X_k}$. Finally, notice that

$$\begin{aligned} \mu p_A = p_A \left(\frac{1}{p_{X_k} p_Z} - \frac{m+1}{p_{X_k}} + \sum_{\substack{i=1 \\ i \neq k}}^{m} \frac{1}{p_{X_i}}\right)^{-1} &= p_A p_{X_k} \left(\frac{1}{p_Z} - (m-1) + p_{X_k} \sum_{\substack{i=1 \\ i \neq k}}^{m} \frac{1}{p_{X_i}}\right)^{-1} \\ = p_A p_{X_k} \left(\frac{1}{p_Z} + \left(\frac{p_{X_k}}{p_{X_k}} - 1\right) + \sum_{\substack{i=1 \\ i \neq k}}^{m} \left(\frac{p_{X_k}}{p_{X_i}} - 1\right)\right)^{-1} &= p_A p_{X_k} \left(\frac{1}{p_Z} + \sum_{i=1}^{m} \left(\frac{p_{X_k}}{p_{X_i}} - 1\right)\right)^{-1} = \xi p_A p_{X_k} \end{aligned}$$

As a result, $V_G = \mu p_A$ is the value of the game and the strategies $\underline{s}$ and $\underline{h}$ are optimal. □

Theorem 3 solves the search and rescue game of type one for general n. It is optimal for the searcher to play a mixed strategy that mixes all of her pure strategies. For the hider, it is optimal to play a mixed strategy that mixes all of his pure strategies except for $h(A)$ and one other. This is either $h(Z)$ or $h(X_k)$ depending on the success probabilities $\underline{p}$. This concludes the analysis for the search and rescue games of type one.

For future research, it would be interesting to analyse the game in which there are multiple sink vertices. These vertices could also be connected to one another so that they in a way form another layer of vertices. Furthermore, we can also add an entire new layer of in-between vertices such that the searcher needs to go from A, to a vertex of the first layer followed by a vertex of the second layer before finally reaching a sink vertex. The layers also do not necessarily need to have the same number of vertices. If a solution to this game is not found akin to theorem 3, it might be possible to find one using behavioral strategies similar to the optimal strategies found in section 3.2.

4.2. Game 2: Search and Rescue Game with Undirected Edges

In this section, we consider the search and rescue game similar to section 4.1. However, all edges are now undirected. For convenience, we call this game the search and rescue game of type two with n vertices. Figure 4.1b and 4.3 show a few examples of the type two search and rescue game.

The hider hides the object in any of the vertices where it is once again trivial that hiding in A is being dominated. Hence, the hider still has $n-1$ pure strategies. The searcher is now allowed to search multiple in-between vertices, unlike in the previous game. She is also allowed to move from the sink vertex back to an in-between vertex and the game does not end prematurely when the sink vertex has been searched. Hence, the vertex Z is not a sink vertex anymore but more like a vertex that is furthest away from the root (we will still call it a sink vertex for convenience). In order to search the sink vertex the searcher needs to have searched at least one in-between vertex. Therefore, it can be shown that the searcher has $(n-2)(n-2)!$ pure strategies.

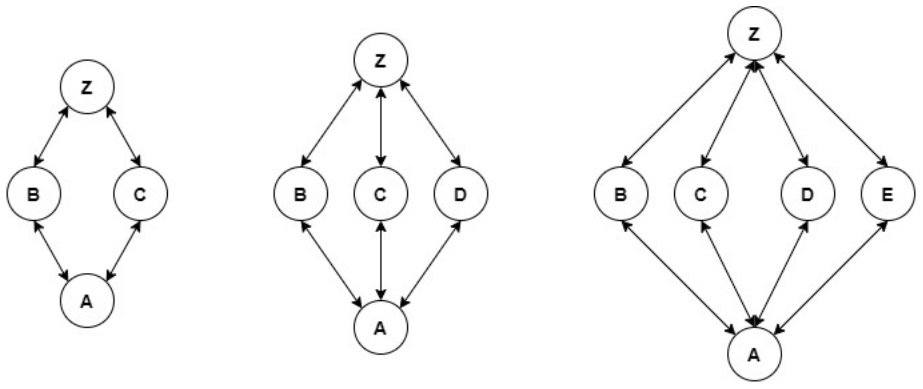


Figure 4.3: Type Two Search and Rescue Game for $n = 4, 5$ and 6

To analyse the type two games, we use the same approach as in section 4.1. The pay-off matrix for the type two games can be constructed using the code found in appendix A.1.6. We first attempt to solve the simplest case where $n = 4$ by generating random probabilities for $\underline{p}$. By solving the corresponding matrix game, we might be able to find patterns in the optimal strategies for both players. If the optimal strategies have been found, we try to generalise them for the game with general n. To that end, consider the search and rescue game of type two with $n = 4$. The graph corresponding to this game is depicted in figure 4.3 and the pay-off matrix for this game is given by

$$\begin{array}{c} \\ (ABZC) \\ (ACZB) \\ (ABCZ) \\ (ACBZ) \end{array} \begin{array}{c} \begin{array}{ccc} \quad B \quad & \quad C \quad & \quad Z \quad \end{array} \\ \begin{pmatrix} p_A p_B & p_A p_B p_C p_Z & p_A p_B p_Z \\ p_A p_B p_C p_Z & p_A p_C & p_A p_C p_Z \\ p_A p_B & p_A p_B p_C & p_A p_B p_C p_Z \\ p_A p_B p_C & p_A p_C & p_A p_B p_C p_Z \end{pmatrix} \end{array}. \tag{4.14}$$

By generating random probabilities for $\underline{p}$ and by solving the corresponding matrix game, we are able to make a few observations:

- The first observation is that there exist a value for p_Z, say x, such that the optimal strategies for both players depend on whether $p_Z \geq x$ or $p_Z < x$. A similar result was also found for the search and rescue game of type one in the previous section where $x = \frac{1}{m}$.
- The second observation is that the hider mixes all of his pure strategies when $p_Z \geq x$, i.e. the hider plays $\underline{h} = (h_B, h_C, h_Z)$. Furthermore, the optimal strategy of the hider makes the searcher indifferent between all her pure strategies. The searcher mixes three out of four of her pure strategies such that the hider is indifferent between all his pure strategies.
- The last observation is that the hider mixes $h(Z)$ and $h(B)$ or $h(C)$ (whichever vertex has a lower success probability) when $p_Z < x$. The hider's optimal strategy makes the searcher indifferent between her pure strategies $(ABZC)$ and $(ACZB)$. The remaining pure strategies of the searcher, $(ABCZ)$ and $(ACBZ)$, give her a lower pay-off than the aforementioned two. Similarly, the searcher mixes $(ABZC)$ and $(ACZB)$ in her optimal strategy which makes the hider indifferent between $h(Z)$ and $h(B)$ or $h(C)$ (again depending on whichever vertex has the lower success probability). The remaining pure strategies will give the searcher a higher expected pay-off than the aforementioned two.

Now that we know which pure strategies both players will mix in their optimal strategy, we try to solve. By first recovering the optimal strategies for both players depending on the value of p_Z, the value of x can be found by equating the expected pay-offs for both half-spaces.

However, this time the equations are more troublesome even for the simplest case where $n = 4$. It turns out that the value for x is a long square root term which is a solution to a quadratic equation. Moreover, both the optimal strategies and the value for x can not be generalised. Hence, we can not find a general solution to this game. Nevertheless, we did find the following result.

Conjecture 4.2.1. *Consider the search and rescue game of type two with $n \geq 4$ vertices and m in-between vertices. Let $X_1, ..., X_m$ be all in-between vertices and let x be the value such that the optimal strategies for both players can be separated depending on whether $p_Z \geq x$ or $p_Z < x$. If $p_Z \geq x$, then it is optimal for the hider to use the mixed strategy $\underline{h} = (h_{X_1}, ..., h_{X_m}, h_Z)$ which plays $h(X)$ with probability given by*

$$h_X = \left(\frac{1 - p_X}{p_X}\right)\lambda, \quad \lambda = \left(\frac{1 - p_Z}{p_Z}\right) + \sum_{i=1}^{m}\left(\frac{1 - p_{X_i}}{p_{X_i}}\right). \tag{4.15}$$

By generating random probabilities for $\underline{p}$ (under the constraint that $p_Z \geq x$) and solving the corresponding matrix game, we have observed that conjecture 4.2.1 indeed holds. In order to prove the conjecture, we would need to show that the searcher is indifferent between all her pure strategies so that the pay-off is at most V_G. We would also need to show that the searcher has a strategy $\underline{s}$ which can guarantee her a pay-off at least V_G. Since the searcher has way more pure strategies compared to the previous game, this can get quite tedious.

Due to the game being less practical than the previous game, we have not spend as much time analysing this game compared to the others. Equating pay-offs does not yield a solution, hence it might be possible to solve the game by solving its subgames first. At the start of the game, the searcher has exactly m options, each corresponding to searching a certain in-between vertex. Notice that after searching the first in-between vertex, the remaining game can be seen as a search and rescue game on a tree with one root vertex which is connected to m leaf vertices. The m leaf vertices are exactly the remaining $m-1$ in-between vertices and the sink vertex. By solving all subgames as done in section 3.2, it might be possible to solve the original game.

4.3. Game 3: Search and Rescue Game with Directed Endpoint

The third variation of the search and rescue game that we will consider is a mix of the type one and type two search and rescue game. The game is similar as the previous two games, but now the edges from the source vertex to any in-between vertex are undirected, and the edges from any in-between vertex to the sink vertex are directed. The resulting game will be called the search and rescue game of type three. Figure 4.1c and 4.4 show a few examples of the search and rescue game of type three.

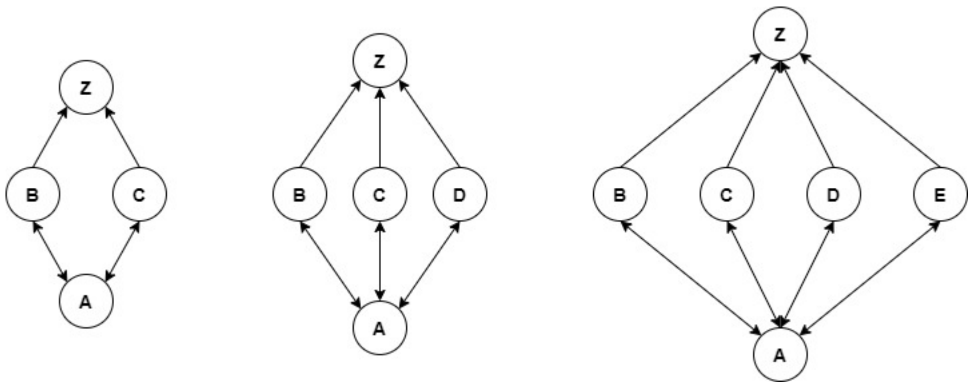


Figure 4.4: Type Three Search and Rescue Game for $m = 2, 3$ and 4

Once again, the hider has $n-1$ pure strategies corresponding to hiding the object in any vertex (except for the source vertex). The searcher is allowed to search any number of in-between vertices before searching the sink vertex. In order to search the sink vertex, the searcher needs to have searched at least one in-between vertex. Once the sink vertex has been searched, the game ends independent on whether or not it was searched successfully. It can be shown that the searcher has exactly

$$\sum_{i=1}^{m}\prod_{j=0}^{i-1}(m-j)$$

pure strategies, which is slightly less compared to the search and rescue game of type two. To analyse the game, we use the same approach as in the previous two sections. The code to construct the pay-off matrix can be found in appendix A.1.7. We first consider the game in which $n = 4$ for which the pay-off matrix is given by

$$\begin{array}{c} \\ (ABZ) \\ (ACZ) \\ (ABCZ) \\ (ACBZ) \end{array} \begin{array}{c} \begin{array}{ccc} B & C & Z \end{array} \\ \begin{pmatrix} p_A p_B & 0 & p_A p_B p_Z \\ 0 & p_A p_C & p_A p_C p_Z \\ p_A p_B & p_A p_B p_C & p_A p_B p_C p_Z \\ p_A p_B p_C & p_A p_C & p_A p_B p_C p_Z \end{pmatrix} \end{array}. \tag{4.16}$$

By solving the matrix games which belong to randomly generated values for $\underline{p}$, we observe the following:

- Similar to the search and rescue game of type one defined in section 4.1, the optimal strategies are dependent on whether $p_Z \geq \frac{1}{2}$ or $p_Z < \frac{1}{2}$.
- The hider plays a mixed strategy $\underline{h} = (h_B, h_C, h_Z)$ that mixes all of his pure strategies when $p_Z \geq \frac{1}{2}$. The searcher on the other hand plays a mixed strategy $\underline{s} = (s_B, s_C, s_{BC}, 0)$ which mixes all of her pure strategies except for $s(CB)$.
- The hider does not hide the object in either B or C when $p_Z < \frac{1}{2}$. This is dependent on the probabilities p_B and p_C, where the vertex with the higher success probability will not be played. Hence, the hider will mix $h(Z)$ with either $h(B)$ or $h(C)$. The searcher plays a mixed strategy $\underline{s} = (s_B, s_C, 0, 0)$ which mixes all pure strategies that search exactly one in-between vertex.

By equating the expected pay-offs, we are able to solve the game for $n = 4$. The following theorem summarises our results.

Theorem 4. *Consider the search and rescue game of type three with $n = 4$. The graph of this game is given in figure 4.4 with pay-off matrix given in 4.16. The solution of this game is dependent on the success probability p_Z.*

- *If $p_Z \geq \frac{1}{2}$, it is optimal for the searcher to use the mixed strategy $\underline{s} = (s_B, s_C, s_{BC}, 0)$ which plays $s(B)$, $s(C)$ and $s(BC)$ with probability s_B, s_C and s_{BC} respectively given by*

$$s_B = (1 - p_Z)\lambda, \quad s_C = \frac{p_B}{p_C}(p_C + p_Z - 2p_C p_Z)\lambda, \quad s_{BC} = (2p_Z - 1)\lambda, \tag{4.17}$$

$$\lambda = \frac{1}{p_B + p_Z - 2p_B p_Z + \frac{p_B p_Z}{p_C}}. \tag{4.18}$$

  For the hider, it is optimal to use the mixed strategy $\underline{h} = (h_B, h_C, h_Z)$ which plays $h(B)$, $h(C)$ and $h(Z)$ respectively with probability given by

$$\underline{h} = (h_B, h_C, h_Z) = \left(\frac{1 - p_B}{p_B}, \frac{1 - p_C}{p_C}, \frac{1}{p_Z}\right)\mu, \quad \mu = \frac{1}{\frac{1-p_B}{p_B} + \frac{1-p_C}{p_C} + \frac{1}{p_Z}}. \tag{4.19}$$

  The value of the game is given by

$$V_G = \mu p_A. \tag{4.20}$$

- *If $p_Z < \frac{1}{2}$, assume w.l.o.g. that $p_B \leq p_C$. Then it is optimal for the searcher to use the mixed strategy $\underline{s} = (s_B, s_C, 0, 0)$ which plays $s(B)$ and $s(C)$ with probability s_B and s_C respectively given by*

$$\underline{s} = (s_B, s_C, 0, 0) = \left(\frac{1}{p_B}, \frac{1 - p_Z}{p_C p_Z}, 0, 0\right)\xi, \quad \xi = \frac{1}{\frac{1}{p_B} + \frac{1-p_Z}{p_C p_Z}}. \tag{4.21}$$

  For the hider, it is optimal to use the mixed strategy $\underline{h} = (h_B, 0, h_Z)$ which plays $h(B)$ and $h(Z)$ with probability h_B and h_Z respectively given by

$$\underline{h} = (h_B, 0, h_Z) = \left(\frac{p_C - p_B}{p_B}, 0, \frac{1}{p_Z}\right)\eta, \quad \eta = \frac{1}{\frac{p_C - p_B}{p_B} + \frac{1}{p_Z}}. \tag{4.22}$$

  The value of the game is given by

$$V_G = \xi p_A. \tag{4.23}$$

  In the case where $p_C \leq p_B$, the optimal strategies are defined in a similar and symmetrical way.

Proof. First observe that the probabilities defined in 4.17, 4.19, 4.21 and 4.22 are non-negative and well-defined. This follows from the fact that

- $p_B, p_C, p_Z \in (0,1)$
- $p_C, p_Z \geq p_C p_Z$, for 4.17
- $p_Z \geq \frac{1}{2}$, for 4.17
- $p_B \leq p_C$, for 4.22

Let $p_Z \geq \frac{1}{2}$ and assume the searcher plays $\underline{s} = (s_B, s_C, s_{BC}, 0)$ as in 4.17 and 4.18. We will show that with this searching strategy, the hider is indifferent between all his pure strategies. There are three cases:

– The hider hides the object in B so that the expected pay-off is given by

$$P\big(h(B), \underline{s}\big) = (1-p_Z)\lambda \cdot p_A p_B + (2p_Z - 1)\lambda \cdot p_A p_B = \lambda p_A p_B p_Z.$$

– The hider hides the object in C so that the expected pay-off is given by

$$P\big(h(C), \underline{s}\big) = \frac{p_B}{p_C}(p_C + p_Z - 2p_C p_Z)\lambda \cdot p_A p_C + (2p_Z - 1)\lambda \cdot p_A p_B p_C = \lambda p_A p_B p_Z.$$

– The hider hides the object in Z so that the expected pay-off is given by

$$\begin{aligned} P\big(h(Z), \underline{s}\big) &= (1-p_Z)\lambda \cdot p_A p_B p_Z + \frac{p_B}{p_C}(p_C + p_Z - 2p_C p_Z)\lambda \cdot p_A p_C p_Z + (2p_Z - 1)\lambda \cdot p_A p_B p_C p_Z \\ &= \lambda p_A p_B p_Z. \end{aligned}$$

Hence, the searcher can guarantee a pay-off of at least $\lambda p_A p_B p_Z$ with the strategy $\underline{s}$ and therefore $V_G \geq \lambda p_A p_B p_Z$. Now assume the hider plays the strategy $\underline{h} = (h_B, h_C, h_Z)$ as in 4.19. We will show that with this hiding strategy, the searcher is indifferent between all her pure strategies. There are four cases:

– The searcher plays $s(B)$ so that the expected pay-off is given by

$$P\big(\underline{h}, s(B)\big) = \left(\frac{1-p_B}{p_B}\right)\mu \cdot p_A p_B + \left(\frac{1}{p_Z}\right)\mu \cdot p_A p_B p_Z = \mu p_A.$$

– The searcher plays $s(C)$ so that the expected pay-off is given by

$$P\big(\underline{h}, s(C)\big) = \left(\frac{1-p_C}{p_C}\right)\mu \cdot p_A p_C + \left(\frac{1}{p_Z}\right)\mu \cdot p_A p_C p_Z = \mu p_A.$$

– The searcher plays $s(BC)$ so that the expected pay-off is given by

$$P\big(\underline{h}, s(BC)\big) = \left(\frac{1-p_B}{p_B}\right)\mu \cdot p_A p_B + \left(\frac{1-p_C}{p_C}\right)\mu \cdot p_A p_B p_C + \left(\frac{1}{p_Z}\right)\mu \cdot p_A p_B p_C p_Z = \mu p_A.$$

– The searcher plays $s(CB)$ so that the expected pay-off is given by

$$P\big(\underline{h}, s(CB)\big) = \left(\frac{1-p_C}{p_C}\right)\mu \cdot p_A p_C + \left(\frac{1-p_B}{p_B}\right)\mu \cdot p_A p_B p_C + \left(\frac{1}{p_Z}\right)\mu \cdot p_A p_B p_C p_Z = \mu p_A.$$

Hence, the hider can ensure that the pay-off is at most μp_A with $\underline{h}$ and therefore $V_G \leq \mu p_A$. Notice that

$$\begin{aligned} \mu p_A &= \frac{p_A}{\frac{1-p_B}{p_B} + \frac{1-p_C}{p_C} + \frac{1}{p_Z}} &&= \frac{p_A p_B p_Z}{p_Z - p_B p_Z + \frac{p_B p_Z}{p_C} - p_B p_Z + p_B} \\ &= \frac{p_A p_B p_Z}{p_B + p_Z - 2p_B p_Z + \frac{p_B p_Z}{p_C}} = \lambda p_A p_B p_Z. \end{aligned}$$

As a result, $V_G = \mu p_A$ is indeed the value of the game and the strategies $\underline{s}$ and $\underline{h}$ are optimal.

Now let $p_Z < \frac{1}{2}$ and assume w.l.o.g. that $p_B \leq p_C$ and that the searcher plays $\underline{s} = (s_B, s_C, 0, 0)$ as in 4.21. We will show that with this searching strategy, the searcher can guarantee a pay-off of at least ξp_A. There are three cases:

– The hider hides the object in B so that the expected pay-off is given by

$$P\big(h(B),\underline{s}\big) = \frac{\xi}{p_B}\cdot p_A p_B = \xi p_A$$

– The hider hides the object in C so that the expected pay-off is given by

$$P\big(h(C),\underline{s}\big) = \left(\frac{1-p_Z}{p_C p_Z}\right)\xi\cdot p_A p_C = \xi p_A\cdot\left(\frac{1-p_Z}{p_Z}\right) \geq \xi p_A,$$

where we use that $p_Z < \frac{1}{2}$.

– The hider hides the object in Z so that the expected pay-off is given by

$$P\big(h(Z),\underline{s}\big) = \frac{\xi}{p_B}\cdot p_A p_B p_Z + \left(\frac{1-p_Z}{p_C p_Z}\right)\xi\cdot p_A p_C p_Z = \xi p_A.$$

Hence, the searcher can guarantee a pay-off of at least ξp_A with $\underline{s}$ and therefore $V_G \geq \xi p_A$. Now assume the hider plays the strategy $\underline{h} = (h_B, 0, h_Z)$ as in 4.22. We will show that with this hiding strategy, the pay-off is at most $\eta p_A p_C$. There are four cases:

– The searcher plays $s(B)$ so that the expected pay-off is given by

$$P\big(\underline{h}, s(B)\big) = \left(\frac{p_C - p_B}{p_B}\right)\eta\cdot p_A p_B + \frac{\eta}{p_Z}\cdot p_A p_B p_Z = \eta p_A p_C.$$

– The searcher plays $s(C)$ so that the expected pay-off is given by

$$P\big(\underline{h}, s(C)\big) = \frac{\eta}{p_Z}\cdot p_A p_C p_Z = \eta p_A p_C.$$

– The searcher plays $s(BC)$ so that the expected pay-off is given by

$$\begin{aligned} P\big(\underline{h}, s(BC)\big) &= \left(\frac{p_C - p_B}{p_B}\right)\eta\cdot p_A p_B + \frac{\eta}{p_Z}\cdot p_A p_B p_C p_Z \\ &= \eta(p_A p_C - p_A p_B + p_A p_B p_C) \leq \eta p_A p_C, \end{aligned}$$

where we use that $p_A p_B \geq p_A p_B p_C$.

– The searcher plays $s(CB)$ so that the expected pay-off is given by

$$\begin{aligned} P\big(\underline{h}, s(CB)\big) &= \left(\frac{p_C - p_B}{p_B}\right)\eta\cdot p_A p_B p_C + \frac{\eta}{p_Z}\cdot p_A p_B p_C p_Z \\ &= \eta(p_A p_C^2 - p_A p_B p_C + p_A p_B p_C) = \eta p_A p_C^2 \leq \eta p_A p_C, \end{aligned}$$

where we use that $p_C \in (0,1)$.

Hence, the hider can ensure that the pay-off is at most $\eta p_A p_C$ with $\underline{h}$ and therefore $V_G \leq \eta p_A p_C$. Finally, notice that

$$\xi p_A = \frac{p_A}{\frac{1}{p_B} + \frac{1-p_Z}{p_C p_Z}} = \frac{p_A p_C}{\frac{p_C}{p_B} - 1 + \frac{1}{p_Z}} = \frac{p_A p_C}{\frac{p_C - p_B}{p_B} + \frac{1}{p_Z}} = \eta p_A p_C.$$

Therefore, the strategies $\underline{s}$ and $\underline{h}$ are optimal and ξp_A is the value of the game. □

Theorem 4 solves the search and rescue game of type three for $n = 4$. The solution looks very similar to the solution found in theorem 2. It is optimal for the hider to either mix all pure strategies or all pure strategies except for the in-between vertex with the highest success probability. The searcher always mixes the pure strategies that directly go to the sink vertex, i.e. the pure strategies that search exactly one in-between vertex. Depending on the probability p_Z, the searcher may or may not mix the pure strategy that searches all in-between vertices in a chronological order.

Since the game has been solved for the simplest case where $n = 4$, we will now try to generalise its solution for general n. This is done similar to section 4.1 by adding more in-between vertices to the game, and by solving the corresponding matrix game.

Proposition 4.3.1. *Consider the search and rescue game of type three with $n \geq 4$ vertices and m in-between vertices. Let $X_1, \dots, X_m$ be all in-between vertices and let X_k be the in-between vertices with the highest success probability, i.e.*

$$p_{X_k} \geq p_{X_i}, \quad \forall i = 1, \dots, m. \tag{4.24}$$

Assume that $p_Z < \frac{1}{m}$. Then it is optimal for the hider to use the mixed strategy $\underline{h} = (h_{X_1}, \dots, h_{X_m}, h_Z)$ which plays any $h(X_i)$ and $h(Z)$ with probability h_{X_i} and h_Z respectively given by

$$h_{X_i} = \left(\frac{p_{X_k} - p_{X_i}}{p_{X_i}}\right)\lambda, \quad i = 1, \dots, m, \tag{4.25}$$

$$h_Z = \frac{\lambda}{p_Z}, \quad \lambda = \left(\frac{1}{p_Z} + \sum_{i=1}^{m}\left(\frac{p_{X_k} - p_{X_i}}{p_{X_i}}\right)\right)^{-1}. \tag{4.26}$$

It is optimal for the searcher to use the mixed strategy $\underline{s} = (s_{X_1}, \dots, s_{X_m}, 0, \dots, 0)$ which plays any $s(X_i)$ with probability s_{X_i} given by

$$s_{X_i} = \frac{\mu}{p_{X_i}}, \quad i \in \{1, \dots, m \mid i \neq k\}, \tag{4.27}$$

$$s_{X_k} = \frac{1-(m-1)p_Z}{p_{X_k}p_Z}\mu, \quad \mu = \left(\frac{1-(m-1)p_Z}{p_{X_k}p_Z} + \sum_{\substack{i=1 \\ i\neq k}}^{m}\frac{1}{p_{X_i}}\right)^{-1}. \tag{4.28}$$

The value of the game is given by

$$V_G = \mu p_A. \tag{4.29}$$

Proof. First observe that the probabilities defined in 4.25, 4.26, 4.27 and 4.28 are all non-negative and well-defined. This follows from the fact that

- $p_A, p_{X_1}, \dots, p_{X_m}, p_Z \in (0,1)$
- Inequality 4.24, for 4.25
- $p_Z < \frac{1}{m}$, for 4.28

Let $p_Z < \frac{1}{m}$ and assume the searcher plays the mixed strategy $\underline{s} = (s_{X_1}, \dots, s_{X_m}, 0, \dots, 0)$ as in 4.27 and 4.28. We will show that this searching strategy guarantees a pay-off of at least μp_A. There are three cases:

- The hider hides the object in X_k so that the expected pay-off is given by

$$P\big(h(X_k), \underline{s}\big) = \frac{1-(m-1)p_Z}{p_{X_k}p_Z}\mu \cdot p_A p_{X_k} = \frac{\mu p_A}{p_Z} - \mu(m-1)p_A = \mu p_A\left(\frac{1}{p_Z} - m + 1\right) \geq \mu p_A,$$

  where we use that $p_Z < \frac{1}{m}$.

- The hider hides the object in $X_i \neq X_k$ so that the expected pay-off is given by

$$P\big(h(X_i), \underline{s}\big) = \frac{\mu}{p_{X_i}} \cdot p_A p_{X_i} = \mu p_A, \quad \forall i \in \{1, \dots, m \mid i \neq k\}.$$

- The hider hides the object in Z so that the expected pay-off is given by

$$\begin{aligned} P\big(h(Z), \underline{s}\big) &= \frac{1-(m-1)p_Z}{p_{X_k}p_Z}\mu \cdot p_A p_{X_k} p_Z + \sum_{\substack{i=1 \\ i\neq k}}^{m}\left(\frac{\mu}{p_{X_i}} \cdot p_A p_{X_i} p_Z\right) \\ &= \mu p_A - (m-1)\mu p_A p_Z + (m-1)\mu p_A p_Z = \mu p_A. \end{aligned}$$

Hence, the searcher can guarantee a pay-off of at least μp_A with $\underline{s}$ and therefore $V_G \geq \mu p_A$. Now assume the hider plays the hiding strategy $\underline{h} = (h_{X_1}, \dots, h_{X_m}, h_Z)$ as in 4.25 and 4.26. We will show that the pay-off is at most $\lambda p_A p_{X_k}$. Let $X_{a_1}, \dots, X_{a_j}$ be some sequence of in-between vertices. There are three cases:

– The searcher plays $s(X_k)$ so that the expected pay-off is given by

$$P(\underline{h}, s(X_k)) = 0 \cdot p_A p_{X_k} + \frac{\lambda}{p_Z} \cdot p_A p_{X_k} p_Z = \lambda p_A p_{X_k}.$$

– The searcher plays $s(X_i) \neq s(X_k)$ so that the expected pay-off is given by

$$P(\underline{h}, s(X_i)) = \left(\frac{p_{X_k} - p_{X_i}}{p_{X_i}}\right)\lambda \cdot p_A p_{X_i} + \frac{\lambda}{p_Z} \cdot p_A p_{X_i} p_Z = \lambda p_A p_{X_k}, \quad \forall i \in \{1, ..., m \mid i \neq k\}.$$

– The searcher searches some sequence of in-between vertices $X_{a_1}, ..., X_{a_j}$ before Z, i.e. she plays $s(X_{a_1}, ..., X_{a_j})$ so that the expected pay-off is given by

$$\begin{aligned}
P(\underline{h}, s(X_{a_1}, ..., X_{a_j})) &= \sum_{v=1}^{j}\left(\left(\frac{p_{X_k} - p_{X_{a_v}}}{p_{X_{a_v}}}\right)\lambda \cdot p_A \prod_{w=1}^{v} p_{X_{a_w}}\right) + \frac{\lambda}{p_Z} \cdot p_A p_Z \prod_{w=1}^{j} p_{X_{a_w}} \\
&= \sum_{v=1}^{j}\left(\left(\frac{p_{X_k}}{p_{X_{a_v}}}\right)\lambda \cdot p_A \prod_{w=1}^{v} p_{X_{a_w}}\right) - \sum_{v=1}^{j}\left(\lambda \cdot p_A \prod_{w=1}^{v} p_{X_{a_w}}\right) + \lambda p_A \prod_{w=1}^{j} p_{X_{a_w}} \\
&= \sum_{v=1}^{j}\left(\lambda p_A p_{X_k} \prod_{w=1}^{v-1} p_{X_{a_w}}\right) - \sum_{v=1}^{j-1}\left(\lambda \cdot p_A \prod_{w=1}^{v} p_{X_{a_w}}\right) \\
&= \lambda p_A p_{X_k} + \sum_{v=2}^{j}\left(\lambda p_A p_{X_k} \prod_{w=1}^{v-1} p_{X_{a_w}}\right) - \sum_{v=1}^{j-1}\left(\lambda \cdot p_A \prod_{w=1}^{v} p_{X_{a_w}}\right) \\
&= \lambda p_A p_{X_k} + \sum_{v=1}^{j-1}\left(\lambda p_A p_{X_k} \prod_{w=1}^{v} p_{X_{a_w}}\right) - \sum_{v=1}^{j-1}\left(\lambda \cdot p_A \prod_{w=1}^{v} p_{X_{a_w}}\right) \\
&= \lambda p_A p_{X_k} + (p_{X_k} - 1) \sum_{v=1}^{j-1}\left(\lambda \cdot p_A \prod_{w=1}^{v} p_{X_{a_w}}\right) \leq \lambda p_A p_{X_k},
\end{aligned}$$

where we use that $p_{X_k} < 1$.

Hence, the hider can ensure that the pay-off is at most $\lambda p_A p_{X_k}$ with $\underline{h}$ and therefore $V_G \leq \lambda p_A p_{X_k}$. Finally, notice that

$$\begin{aligned}
\mu p_A = p_A\left(\frac{1-(m-1)p_Z}{p_{X_k} p_Z} + \sum_{\substack{i=1 \\ i\neq k}}^{m} \frac{1}{p_{X_i}}\right)^{-1} &= p_A p_{X_k}\left(\frac{1-(m-1)p_Z}{p_Z} + \sum_{\substack{i=1 \\ i\neq k}}^{m} \frac{p_{X_k}}{p_{X_i}}\right)^{-1} \\
= p_A p_{X_k}\left(\frac{1}{p_Z} - (m-1) + \sum_{\substack{i=1 \\ i\neq k}}^{m} \frac{p_{X_k}}{p_{X_i}}\right)^{-1} &= p_A p_{X_k}\left(\frac{1}{p_Z} + \sum_{i=1}^{m}\left(\frac{p_{X_k} - p_{X_i}}{p_{X_i}}\right)\right)^{-1} = \lambda p_A p_{X_k}.
\end{aligned}$$

Therefore, the strategies $\underline{s}$ and $\underline{h}$ are indeed optimal and the value of the game is μp_A. □

Proposition 4.3.1 solves the search and rescue game of type three for general n when $p_Z < \frac{1}{m}$. In this case, it is optimal for the hider to play a mixed strategy that hides in all vertices except for the in-between vertex with the highest success probability. It is optimal for the searcher to play a mixed strategy that mixes all pure strategies that search exactly one in-between vertex before the sink vertex.

In the case where $p_Z \geq \frac{1}{m}$, the optimal strategies are more troublesome to find. When we tried to generalise the solution found in theorem 4 for general n, we found an optimal strategy for the searcher that is only defined on a subinterval for p_Z. In other words, there is a subinterval

$$\frac{1}{m} \leq p_Z \leq x < 1$$

for some value x such that the optimal strategies are only defined when p_Z is in the specified interval. Moreover, the simulations show that when p_Z is sufficiently large (i.e. greater than x), there are many optimal strategies for the searcher which all mix a variety of pure strategies. These optimal strategies have in common that they do not mix all pure strategies of the searcher that search exactly one in-between vertex. We have yet to find a pattern for the searcher, however we did find the following for the hider.

Proposition 4.3.2. *Consider the search and rescue game of type three with $n \geq 4$ vertices and m in-between vertices. Let $X_1, \ldots, X_m$ be all in-between vertices and assume that $p_Z \geq \frac{1}{m}$. The hider can ensure that the pay-off is at most λp_A with the mixed strategy $\underline{h} = (h_{X_1}, \ldots, h_{X_m}, h_Z)$ which plays any $h(X_i)$ and $h(Z)$ with probability h_{X_i} and h_Z respectively given by*

$$h_{X_i} = \left(\frac{1-p_{X_i}}{p_{X_i}}\right)\lambda, \quad i = 1, \ldots, m, \tag{4.30}$$

$$h_Z = \frac{\lambda}{p_Z}, \quad \lambda = \left(\frac{1}{p_Z} + \sum_{i=1}^{m}\left(\frac{1-p_{X_i}}{p_{X_i}}\right)\right)^{-1}. \tag{4.31}$$

Proof. First observe that the probabilities defined in 4.30 and 4.31 are non-negative and well-defined. We will show that the searcher is indifferent between all her pure strategies. Let $X_{a_1}, \ldots, X_{a_j}$ be some sequence of in-between vertices. If the searcher plays $s(X_i)$ for some $i = 1, \ldots, m$, then the expected pay-off is given by

$$P\big(\underline{h}, s(X_i)\big) = \left(\frac{1-p_{X_i}}{p_{X_i}}\right)\lambda \cdot p_A p_{X_i} + \frac{\lambda}{p_Z} \cdot p_A p_{X_i} p_Z = \lambda p_A$$

Now if the searcher searches some sequence of in-between vertices $X_{a_1}, \ldots, X_{a_j}$ before Z, i.e. she plays $s(X_{a_1}, \ldots, X_{a_j})$, then the expected pay-off is given by

$$\begin{aligned}
P\big(\underline{h}, s(X_{a_1}, \ldots, X_{a_j})\big) &= \sum_{v=1}^{j}\left(\left(\frac{1-p_{X_{a_v}}}{p_{X_{a_v}}}\right)\lambda \cdot p_A \prod_{w=1}^{v} p_{X_{a_w}}\right) + \frac{\lambda}{p_Z} \cdot p_A p_Z \prod_{w=1}^{j} p_{X_{a_w}} \\
&= \sum_{v=1}^{j}\left(\frac{\lambda}{p_{X_{a_v}}} \cdot p_A \prod_{w=1}^{v} p_{X_{a_w}}\right) - \sum_{v=1}^{j}\left(\lambda p_A \prod_{w=1}^{v} p_{X_{a_w}}\right) + \lambda p_A \prod_{w=1}^{j} p_{X_{a_w}} \\
&= \sum_{v=1}^{j}\left(\lambda p_A \prod_{w=1}^{v-1} p_{X_{a_w}}\right) - \sum_{v=1}^{j-1}\left(\lambda p_A \prod_{w=1}^{v} p_{X_{a_w}}\right) \\
&= \lambda p_A + \sum_{v=2}^{j}\left(\lambda p_A \prod_{w=1}^{v-1} p_{X_{a_w}}\right) - \sum_{v=1}^{j-1}\left(\lambda p_A \prod_{w=1}^{v} p_{X_{a_w}}\right) \\
&= \lambda p_A + \sum_{v=1}^{j-1}\left(\lambda p_A \prod_{w=1}^{v} p_{X_{a_w}}\right) - \sum_{v=1}^{j-1}\left(\lambda p_A \prod_{w=1}^{v} p_{X_{a_w}}\right) = \lambda p_A.
\end{aligned}$$

Hence, the hider can ensure that the pay-off is at most λp_A with the hiding strategy $\underline{h}$. □

To prove that the strategy $\underline{h}$ defined in proposition 4.3.1 is optimal for the hider, we need to show that the searcher has a strategy $\underline{s}$ such that she can guarantee a pay-off of at least λp_A with λ as defined in proposition 4.3.1. For the searcher, we found the following possible optimal strategy.

Conjecture 4.3.1. *Consider the search and rescue game of type three with $n \geq 4$ vertices and m in-between vertices. Let $X_1, \ldots, X_m$ be all in-between vertices and assume that*

$$\left(m - 1 - \sum_{j=2}^{m}\prod_{i=2}^{j} p_{X_i}\right)^{-1} \geq p_Z \geq \frac{1}{m}. \tag{4.32}$$

It is optimal for the searcher to play a mixed strategy $\underline{s}$ which plays any $s(X_i)$ and $s(X_1, \ldots, X_m)$ with probability proportional to s_{X_i} and s_Ω respectively given by

$$s_{X_1} = 1 - (m-1)p_Z + p_Z \sum_{j=2}^{m-1}\prod_{i=2}^{j} p_{X_i}, \quad s_\Omega = m p_Z - 1, \tag{4.33}$$

$$s_{X_m} = \frac{p_{X_1}}{p_{X_m}}\left(p_Z + \prod_{j=2}^{m} p_{X_j} - (m+1)p_Z \prod_{j=2}^{m} p_{X_j} + p_Z \sum_{j=2}^{m}\prod_{i=2}^{j} p_{X_i}\right), \tag{4.34}$$

$$s_{X_i} = \frac{p_{X_1}}{p_{X_i}}\left(p_Z + \prod_{j=2}^{i} p_{X_j} - m p_Z \prod_{j=2}^{i} p_{X_j} + p_Z \sum_{j=2}^{m-1}\prod_{k=2}^{j} p_{X_k}\right), \quad i = 2, \ldots, m-1. \tag{4.35}$$

The inequality in 4.32 is needed to ensure that the probability s_{X_1} defined in 4.33 is non-negative and well-defined. Observe that if

$$s_{X_1} = 1 - p_Z\left((m-1) - \sum_{j=2}^{m-1}\prod_{i=2}^{j} p_{X_i}\right) \geq 0,$$

then it must be that the first inequality sign holds. Also observe that

$$0 < \sum_{j=2}^{m}\prod_{i=2}^{j} p_{X_i} < m-1$$

so that

$$0 < m-1-\sum_{j=2}^{m}\prod_{i=2}^{j} p_{X_i} < m-1$$

and finally

$$\left(m-1-\sum_{j=2}^{m}\prod_{i=2}^{j} p_{X_i}\right)^{-1} > \frac{1}{m-1} > \frac{1}{m}.$$

The non-negativity of 4.34 and 4.35 follow from the fact that

$$\begin{aligned} p_Z + \left(\prod_{j=2}^{m} p_{X_j}\right) + p_Z\left(\sum_{j=2}^{m}\prod_{i=2}^{j} p_{X_i}\right) &> p_Z\left(\prod_{j=2}^{m} p_{X_j}\right) + p_Z\left(\prod_{j=2}^{m} p_{X_j}\right) + p_Z\left(\sum_{j=2}^{m}\prod_{i=2}^{m} p_{X_i}\right) \\ &= (m+1)p_Z\left(\prod_{j=2}^{m} p_{X_j}\right) \end{aligned}$$

and

$$\begin{aligned} p_Z + \left(\prod_{j=2}^{i} p_{X_j}\right) + p_Z\left(\sum_{j=2}^{m-1}\prod_{k=2}^{j} p_{X_k}\right) &> p_Z\left(\prod_{j=2}^{i} p_{X_j}\right) + p_Z\left(\prod_{j=2}^{i} p_{X_j}\right) + p_Z\left(\sum_{j=2}^{m-1}\prod_{k=2}^{m-1} p_{X_k}\right) \\ &= mp_Z\left(\prod_{j=2}^{i} p_{X_j}\right), \end{aligned}$$

which are both positive terms. It is also easy to see that $s_{X_\Omega} > 0$, because $p_Z \geq \frac{1}{m}$. Hence, the probabilities defined in the conjecture are all well-defined. It has been shown in appendix A.2.4 that the searcher can gaurantee a pay-off of at least

$$V_G = \mu p_A p_{X_1} p_Z\left(1 + \sum_{j=2}^{m-1}\prod_{i=2}^{j} p_{X_i}\right) \tag{4.36}$$

with the searching strategy $\underline{s}$ as defined in conjecture 4.3.1. To show that this strategy and the hiding strategy defined in proposition 4.3.2 are indeed optimal, it remains to show that

$$V_G = \mu p_A p_{X_1} p_Z\left(1 + \left(\sum_{j=2}^{m-1}\prod_{i=2}^{j} p_{X_i}\right) = \lambda p_A.\right. \tag{4.37}$$

Proving this can be quite tedious because of the length of μ, but should not be impossible. If one can show that 4.37 indeed holds, then the search and rescue game of type three for general n would be solved for all

$$0 < p_Z \leq \left(m-1-\sum_{j=2}^{m}\prod_{i=2}^{j} p_{X_i}\right)^{-1}.$$

For the remaining p_Z, the optimal strategies remain a mystery. However, we have observed with simulations that the hiding strategy defined in proposition 4.3.2 is still optimal. And so, the only missing piece is the optimal strategy for the searcher.

4.4. Game 3: Behavioural Strategy

In this section, we continue from where we left off in the previous section. For the search and rescue game of type three with general n, the optimal strategy for the searcher remains unknown for sufficiently large values of p_Z. Therefore, we try a different approach and try to obtain the searcher's optimal strategy using behavioural strategies similar to section 3.2.

The idea is to assign probabilities to the edges of the graph. Each edge indicates the probability of being played over the other. It might be possible to choose the probabilities such that this behavioural strategy forms an optimal strategy. To that end, we will first try to solve the simplest case where $n = 4$. We will then try to generalise its solution. Once again consider the search and rescue game of type three with $n = 4$. Figure 4.5 shows the graph belonging to this game and the edges with their assigned probabilities.

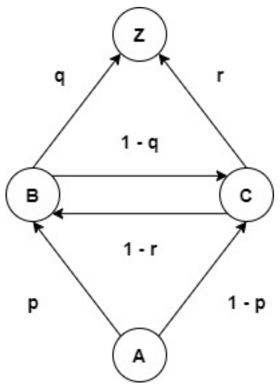


Figure 4.5: Type One Search and Rescue Game for $n = 4$,

First observe that the game on the graph in figure 4.5 is identical to that of figure 4.4. Traversing from B to C in figure 4.5 is equivalent to traversing from B to A to C in figure 4.4. A similar observation can be made when traversing from C to B. At the start of the game, the searcher needs to make a decision to either search B or C. As seen in figure 4.5, the searcher will choose B with probability p and C otherwise.

Assume for the moment that she chooses B. She will then have to make the decision to either directly search Z or to first search C before moving to Z. The former happens with probability q and the latter with probability $1-q$. Similarly, if the searcher chose C at the start instead of B, she will either directly search Z with probability r or she will first search B and then Z with probability $1-r$.

Observe that such a behavioural strategy forms a mixed strategy for the searcher. The pure strategy (ABZ) will be played when the searcher chooses B over C with probability p, and Z over C with probability q. Hence, the probability of playing (ABZ) is given by $p \cdot q$. In a similar way, the other pure strategies of the searcher are played with some probability. Hence, the searcher is in fact playing a mixed strategy $\underline{s} = (s_B, s_C, s_{BC}, s_{CB})$. The probabilities for such a strategy are given by

$$\underline{s} = (s_B, s_C, s_{BC}, s_{CB}) = \big(pq, (1-p)r, p(1-q), (1-p)(1-r)\big). \tag{4.38}$$

The difficulty lies in determining the probabilities p, q and r. From theorem 4, we know that the hider can ensure that the pay-off is at most $\lambda p_A p_B p_Z$ with λ as defined in 4.18. Therefore, we want to choose the probabilities p, q and r such that the expected pay-off is at least $\lambda p_A p_B p_Z$ against any pure strategy of the hider. By trial and error, we have found the following.

Theorem 5. *Consider the search and rescue game of type three with $n = 4$ vertices. Assume that $p_Z \geq \frac{1}{2}$. Then the behavioural strategy in which*

$$\underline{p} = (p, 1-p) = \left(\frac{1}{p_B}, \frac{1}{p_C}\right)\mu, \quad \mu = \frac{1}{\frac{1}{p_B} + \frac{1}{p_C}} \tag{4.39}$$

$$\underline{q} = (q, 1-q) = \left(1 - \frac{2p_Bp_Z - p_B}{p_Bp_C + p_Bp_Z + p_Cp_Z - 2p_Bp_Cp_Z}, \frac{2p_Bp_Z - p_B}{p_Bp_C + p_Bp_Z + p_Cp_Z - 2p_Bp_Cp_Z}\right) \tag{4.40}$$

$$\underline{r} = (r, 1-r) = \left(1 - \frac{2p_Cp_Z - p_C}{p_Bp_C + p_Bp_Z + p_Cp_Z - 2p_Bp_Cp_Z}, \frac{2p_Cp_Z - p_C}{p_Bp_C + p_Bp_Z + p_Cp_Z - 2p_Bp_Cp_Z}\right) \tag{4.41}$$

forms an optimal mixed strategy for the searcher.

Proof. First observe that the probabilities defined in 4.39 are non-negative and well-defined. For 4.40 it is sufficient to show that

$$0 \leq \frac{2p_B p_Z - p_B}{p_B p_C + p_B p_Z + p_C p_Z - 2p_B p_C p_Z} \leq 1. \tag{4.42}$$

Note that since

$$p_B p_C, p_C p_Z \geq p_B p_C p_Z$$

we have that

$$\begin{aligned}
p_B p_C + p_B p_Z + p_C p_Z - 2p_B p_C p_Z &\geq p_B p_Z, \\
\frac{1}{p_B p_C + p_B p_Z + p_C p_Z - 2p_B p_C p_Z} &\leq \frac{1}{p_B p_Z}, \\
\frac{2p_B p_Z - p_B}{p_B p_C + p_B p_Z + p_C p_Z - 2p_B p_C p_Z} &\leq \frac{2p_B p_Z - p_B}{p_B p_Z} = 2 - \frac{1}{p_Z} \leq 1,
\end{aligned}$$

where we use that $1 > p_Z \geq \frac{1}{2}$. Furthermore, since

$$2p_B p_Z - p_B \geq 0,$$

both the numerator and the denominator are positive. Hence, it must be that 4.42 is non-negative and therefore well-defined. In a similar way, we can show that 4.41 is well-defined.

Assume the searcher plays according to the behavioural strategy. This is equivalent to playing the mixed strategy $\underline{s} = (s_B, s_C, s_{BC}, s_{CB})$ which plays $s(B), s(C), s(BC)$ and $s(CB)$ with probabilities s_B, s_C, s_{BC} and s_{CB} respectively given by

$$\begin{aligned}
s_B = pq &= \frac{\mu}{p_B} - \frac{2\mu p_Z - \mu}{p_B p_C + p_B p_Z + p_C p_Z - 2p_B p_C p_Z}, \\
s_C = (1-p)r &= \frac{\mu}{p_C} - \frac{2\mu p_Z - \mu}{p_B p_C + p_B p_Z + p_C p_Z - 2p_B p_C p_Z}, \\
s_{BC} = p(1-q) &= \frac{2\mu p_Z - \mu}{p_B p_C + p_B p_Z + p_C p_Z - 2p_B p_C p_Z}, \\
s_{CB} = (1-p)(1-r) &= \frac{2\mu p_Z - \mu}{p_B p_C + p_B p_Z + p_C p_Z - 2p_B p_C p_Z}.
\end{aligned}$$

From theorem 4, we know that the value of the game is $V_G = \lambda p_A p_B p_Z$. So it remains to show that this mixed strategy can guarantee a pay-off of at least V_G. There are three cases:

- The hider hides the object in B so that the expected pay-off is given by

$$\begin{aligned}
P\big(h(B), \underline{s}\big) &= \left(\frac{\mu}{p_B} - \frac{2\mu p_Z - \mu}{p_B p_C + p_B p_Z + p_C p_Z - 2p_B p_C p_Z}\right) \cdot p_A p_B + \frac{2\mu p_Z - \mu}{p_B p_C + p_B p_Z + p_C p_Z - 2p_B p_C p_Z} \cdot p_A p_B \\
&\quad + \frac{2\mu p_Z - \mu}{p_B p_C + p_B p_Z + p_C p_Z - 2p_B p_C p_Z} \cdot p_A p_B p_C \\
&= \mu p_A + \frac{2\mu p_A p_B p_C p_Z - \mu p_A p_B p_C}{p_B p_C + p_B p_Z + p_C p_Z - 2p_B p_C p_Z}.
\end{aligned}$$

- The hider hides the object in C so that the expected pay-off is given by

$$\begin{aligned}
P\big(h(C), \underline{s}\big) &= \left(\frac{\mu}{p_C} - \frac{2\mu p_Z - \mu}{p_B p_C + p_B p_Z + p_C p_Z - 2p_B p_C p_Z}\right) \cdot p_A p_C + \frac{2\mu p_Z - \mu}{p_B p_C + p_B p_Z + p_C p_Z - 2p_B p_C p_Z} \cdot p_A p_B p_C \\
&\quad + \frac{2\mu p_Z - \mu}{p_B p_C + p_B p_Z + p_C p_Z - 2p_B p_C p_Z} \cdot p_A p_C \\
&= \mu p_A + \frac{2\mu p_A p_B p_C p_Z - \mu p_A p_B p_C}{p_B p_C + p_B p_Z + p_C p_Z - 2p_B p_C p_Z}.
\end{aligned}$$

– The hider hides the object in Z so that the expected pay-off is given by

$$P\big(h(Z), \underline{s}\big) = \mu p_A + \frac{2\mu p_A p_B p_C p_Z - \mu p_A p_B p_C}{p_B p_C + p_B p_Z + p_C p_Z - 2 p_B p_C p_Z}, \tag{4.43}$$

see appendix A.2.5.

Hence, the searcher can guarantee a pay-off of at least

$$V_G = \mu p_A + \frac{2\mu p_A p_B p_C p_Z - \mu p_A p_B p_C}{p_B p_C + p_B p_Z + p_C p_Z - 2 p_B p_C p_Z}$$

with the mixed strategy $\underline{s}$. Finally, notice that

$$\begin{aligned}
\mu p_A + \frac{2\mu p_A p_B p_C p_Z - \mu p_A p_B p_C}{p_B p_C + p_B p_Z + p_C p_Z - 2 p_B p_C p_Z} &= \frac{\mu(p_A p_B p_C + p_A p_B p_Z + p_A p_C p_Z - 2 p_A p_B p_C p_Z)}{p_B p_C + p_B p_Z + p_C p_Z - 2 p_B p_C p_Z} \\
&\quad + \frac{2\mu p_A p_B p_C p_Z - \mu p_A p_B p_C}{p_B p_C + p_B p_Z + p_C p_Z - 2 p_B p_C p_Z} \\
&= \frac{\mu(p_A p_B p_Z + p_A p_C p_Z)}{p_B p_C + p_B p_Z + p_C p_Z - 2 p_B p_C p_Z} \\
&= \frac{p_A p_B p_C p_Z}{p_B p_C + p_C p_Z - 2 p_B p_C p_Z + p_B p_Z} \\
&= \frac{p_A p_B p_Z}{p_B + p_Z - 2 p_B p_Z + \frac{p_B p_Z}{p_C}} = \lambda p_A p_B p_Z
\end{aligned}$$

so that the strategy $\underline{s}$ is indeed optimal. $\square$

For the search and rescue game of type three with $n = 4$ and $p_Z \geq \frac{1}{2}$, we have now found two optimal strategies for the searcher. Theorem 5 states an optimal strategy which originates from a behavioral strategy. This optimal strategy mixes all of the searcher's pure strategies, as opposed to theorem 4 which mixes only three out of four. Therefore, the optimal strategy that originates from the behavioural strategy is more symmetric than that of theorem 4, but also has more terms and is therefore more complex. As a result we have yet to find a generalisation.

Finding a generalisation for theorem 5 is harder than it looks. Finding a behavioural strategy that gives an optimal strategy for $n = 5$ already proves to be quite difficult. This is because the probabilities on the edges change depending on what the searcher has already searcher. To further illustrate, consider the game with $n = 5$ depicted in figure 4.4. If the searcher searches B first, she can go from B to C with some probability x. However, if the searcher goes from A to D, and then to B, she has the option to either search B or directly search Z. The probability of searching B in this case is not the same as the probability x from the previous case. To find an optimal behavioural strategy, we would need to introduce more unknown variables which we would then have to guess (since we do not have enough expressions from the pay-offs to solve for the unknown variables).

5 Conclusion

In this report we have defined and solved multiple search and rescue games. This includes the standard search and rescue game, the search and rescue game on a tree, and multiple search and rescue games on very simple graphs. The graphs that we have considered all have the same characteristics: they have a source vertex, a sink vertex and multiple in-between vertices. The graphs differ from one another depending on whether the edges are directed or undirected.

For the basic search and rescue game, the optimal strategies are found by assigning an index

$$z_i = \frac{1-p_i}{p_i}$$

to every location. It is optimal for the hider to hide the objects in some subset $A \in S^{(k)}$ with some probability proportional to the product of the indices of all locations $i \in A$ as in theorem 1. With the selfsame probability, it is optimal for the searcher to first search the locations in A before searching the remaining locations in an uniformly random order. We have also found a different optimal strategy for the searcher when $k = 1$, i.e. only one object is hidden. Instead of searching the remaining objects in an uniformly random order, the searcher can also search the remaining locations in a chronologically increasing/decreasing order.

For the search and rescue game on a tree, the optimal strategies are found similar to behavioural strategies. It is optimal for the hider to hide the object in one of the leaf vertices with some probability as in definition 3.2.2. For the searcher it is optimal to search the tree in a depth-first order, where one branch is chosen over the other with some probability as in definition 3.2.3.

Finally, we have also looked at the search and rescue game played on very simple graphs that are not trees. The graphs differ from one another depending on the directionality of the edges and we have analysed three types of graphs:

- Type 1: All edges are directed.
- Type 2: All edges are undirected.
- Type 3: The edges from the source vertex to any in-between vertex are undirected, and the edges from any in-between vertex to the sink vertex are directed.

The optimal strategies for these games depend on the success probability of the sink vertex p_Z.

For the type one graphs, the threshold for p_Z is at $\frac{1}{m}$ and a solution to the game is given in theorem 3. Depending on the value of p_Z, it is optimal for the hider to play a mixed strategy that mixes all of his pure strategies except for either the pure strategy that hides in the sink vertex, or the pure strategy that hides in the in-between vertex with the highest success probability. For the searcher, it is optimal to play a mixed strategy that mixes all of her pure strategies.

For the type two graphs, the threshold for p_Z is at some unknown value x. We have yet to find a general expression for the value of x nor have we found the optimal strategies. We do however have a hunch of what the hider's optimal strategies could be when $p_Z \geq x$. In this case, the hider will play a mixed strategy that mixes all of his pure strategies with probability proportional to the inverse of the odds as in conjecture 4.2.1.

For the type three graphs, the threshold for p_Z is once again at $\frac{1}{m}$. Proposition 4.3.1 solves the game when $p_Z < \frac{1}{m}$. In this case, it is optimal for the hider to play a mixed strategy that mixes all of his pure strategies except for the pure strategy that hides in the in-between vertex with the highest success probability. For the searcher, it is optimal to play a mixed strategy that mixes all pure strategies that search exactly one in-between vertex before the sink vertex.

In the case where $p_Z \geq \frac{1}{m}$, proposition 4.3.2 and conjecture 4.3.1 state the possible optimal strategies for both players. It is most likely optimal for the hider to play a mixed strategy that mixes all of his pure strategies. For the searcher, there is an interval on which it might be optimal to play a mixed strategy that mixes all pure strategies that search exactly one in-between vertex, and the pure strategy that searches all in-between vertices in chronological order. Outside of this interval, the searcher does not mix all of her pure strategies that search exactly one in-between vertex. We have yet to prove that these strategies are optimal nor have we found an optimal strategy for the searcher outside of this interval.

A
Appendix

A.1. Matlab Code

A.1.1. Chapter 2 - Solve a Search and Rescue Matrix Game

```
%%  Solver(Probabilities, Trip_Chance)
%   Solves a search and rescue matrix game
%   Input:  Probabilities := Array of the success probability p = (p_1, .. , p_n)
%           k := Number of objects
%           Trip_Chance = Extra fail probability in p_1 or p_2
%   Output: A = Pay-off Matrix
%           Mod_A = Row-reduced pay-off matrix
%           Saddle_Point = Boolean detecting saddle point
%           Opt_Hider = One hider's optimal strategy
%           Opt_Searcher = One searcher's optimal strategy
%           Searcher_Pures = Pure strategies mixed in Opt_Searcher
%           Value = The value of the game
%           Mod_A = Pay-off matrix reduced to rows in Opt_Searcher

function [A, Saddle_Point, Opt_Hider, Opt_Searcher, Searcher_Pures, Value, Mod_A] = Solver(Probabilities, k, Trip_Chance)

% Define rational formatting
format rat;

% Define size of pay-off matrix
n = length(Probabilities);
nrColums = nchoosek(n,k);
nrRows = factorial(n)/factorial(k);

% Define hider's pure strategies
Permutations = sortrows(perms(1:n));
Col_Permutations = unique(Permutations(:,1:k), 'rows');
ColStrats = unique(sort(Col_Permutations,2), 'rows');

% Define searcher's pure strategies
RowStrats = [];
for i = 1:length(ColStrats)
    cRow = ColStrats(i,:);
    cPermutations = Permutations(:,1:k);
    cMatchingRows = Permutations(find(ismember(cPermutations,cRow,'rows')), :);
    RowStrats = [RowStrats; cMatchingRows];
end

% Empty pay-off matrix
A = ones(nrRows,nrColums);

% Fill in pay-off matrix
for i = 1:nrRows

    % Current row-strategy
    cRowStrat = RowStrats(i,:);

    for j = 1:nrColums

        % Current column-strategy
        cColStrat = ColStrats(j,:);

        % Search_Index keeps track of which element in the permutation to search next.
        % Counter keeps track of whether all objects have been found
        Search_Index = 1;
        Counter = 0;

        % One_Gone and Two_Gone keep track of whether one of the two has been searcherd
        % Extra_Once makes sure the extra trip probability is only added once
        One_Gone = 0;
        Two_Gone = 0;
        Extra_Once = 0;

        while Counter ~= k

            % Update current searched element
            Searched_Element = cRowStrat(Search_Index);

            %Update total success probability
            A(i,j) = A(i,j) * Probabilities(Searched_Element);

            % Update whether location 1 or 2 has been searched
            if Searched_Element == 1
                One_Gone = 1;
            end
            if Searched_Element == 2
                Two_Gone=1;
            end

            % Check if one of the hidden objects has been found
            if(ismember(Searched_Element, cColStrat))
                Counter = Counter + 1;
            end

            % Add extra trip chance to the total success probability
            if One_Gone == 1 && Two_Gone == 1 && Extra_Once==0
                A(i,j) = A(i,j) * Trip_Chance;
                Extra_Once = 1;
            end

            % Update the next element to be searched
            Search_Index = Search_Index + 1;
        end
    end
end

%% Matrix game solver
Opt_Hider = [];
Opt_Searcher = [];
Searcher_Pures = [];
Mod_A = [];

r=[];s=[];[m,n]=size(A);
if min(max(A))==max(min(A'))
    b=max(A);Strategy_Ist=[];Strategy_IInd=[];ms=[];
    for i=1:n
        for j=1:m
            if isequal(b(i),A(j,i))
                if isequal(A(j,i),min(A(j,:)))
                    r(length(r)+1)=j;
                    s(length(s)+1)=i;
                end
            end
        end
    end
    if (length(r)==1 && length (s)==1)
        Answer=['The Game has a saddle point at the location :- (' int2str(r) ',' int2str(s) ')
                and value of the game is ' num2str(A(r,s),6) '. So no mixed strategy is needed.'];

        % Update Saddle_Point and Value
        Saddle_Point = true;
        Value = num2str(A(r,s),6);
    else
        for i=1:length(r)
            ms=[ms '(' int2str(r(i)) ',' int2str(s(i)) '),'];
        end
        Answer=['The Game has saddle points at the locations :-' ms '
                and value of the game is ' num2str(A(r(1),s(1)),6) '. So no mixed strategy is needed.'];

        % Update Saddle_Point and Value
        Saddle_Point = true;
        Value = num2str(A(r(1),s(1)),6);
    end
else
        X_a=linprog(-[1;zeros(m,1)],[ones(n,1) -A'],zeros(n,1),[0 ones(1,m)],[1],[-inf;zeros(m,1)]);v=X_a(1,1);X_a(1,:)=[];
        X_b=linprog([1;zeros(n,1)],[-ones(m,1) A],zeros(m,1),[0 ones(1,n)],[1],[-inf;zeros(n,1)]);X_b(1,:)=[];

    Answer=['The Game has no saddle point and value of the game is ' num2str(v,6) '
            and therefore the suggested mixed strategy is given in mixed strategy matrix.'];
    % Update Saddle_Point and Value
    Saddle_Point = false;
    Value = num2str(v,6);

    % Update optimal strategies of Searcher and Hider
    Opt_Searcher = X_a;
    Opt_Hider = X_b;

    % Find indices used in optimal strategy of searcher
    Searcher_Indices = find(Opt_Searcher);

    % Restore the used permutations in the optimal searcher strategy
    for z = 1:length(Searcher_Indices)
        Search_Permutation_Index = Searcher_Indices(z,:);
        for x = 1:length(RowStrats(Search_Permutation_Index,:))
            Searcher_Pures(z,x) = RowStrats(Search_Permutation_Index,x);
        end
    end

    % Construct Pay-off matrix using only the pure strategies played
    Row_Strats_Used = find(Opt_Searcher);
    for i = 1:length(Row_Strats_Used)
        Mod_A = [Mod_A;A(Row_Strats_Used(i),:)];
    end
end
end
```

A.1.2. Section 2.4 and 2.5 - Experiment 2.4.1 and 2.5.1

```
%% TestValue(Prob, k, Trip_Chance, Steps)
%  Test if the value of the original and row-reduced game match for fixed probabilities p_1,..,p_(n-2).
%  Input:  Prob := Array of the success probability p := (p_1, .. , p_(n-2))
%          k := Number of objects
%          Trip_Chance := Extra fail probability in p_1 or p_2
%          Steps := The stepsize of the grid

function [] = TestValue(Prob, k, Trip_Chance, Steps)

% Define rational formatting
format rat;

% Define stepsize, and the start and end of the grid
Stepsize = 1/Steps;
Grid_Start = Stepsize;
Grid_End = 1 - Stepsize;

% Define empty lists to separate grid points
Hypo1 = [];
Hypo2 = [];

% Define indices of the row strategies in the row-reduced game
n = length(Prob) + 2;

% n = 4, k = 1
%Index_Row_Strats = [1, 10, 17, 20]; %(1234 2341 3412 4132) %(Exp 2.4.1.a)
%Index_Row_Strats = [1, 10, 17, 19]; %(1234 2341 3412 4123) %(Exp 2.4.1.b)
%Index_Row_Strats = [6, 8, 15, 24]; %(1432 2143 3214 4321)

% n = 5, k = 2
%Index_Row_Strats = [1,10,17,19,28,35,37,47,49,55]; %(Exp 2.5.1.1)
%Index_Row_Strats = [6,8,15,24,26,33,42,45,54,60]; %(Exp 2.5.1.2)
%Index_Row_Strats = [1,7,13,19,25,31,37,43,49,55]; %(Exp 2.5.1.3)
%Index_Row_Strats = [1,11,28,31,47,52,55,16,19,41]; %(Exp 2.5.1.4)

% Solve the original and row-reduced game and group grid points
for x = Grid_Start:Stepsize:Grid_End
    for y = Grid_Start:Stepsize:Grid_End

        % Define probabilty p
        Probabilities = [Prob, x, y];
        A = [];

        % Solve original game for (x,y)
        [Old_A, Saddle_Point, Opt_Hider, Opt_Searcher, Searcher_Pures, Old_Value, Mod_A] = Solver(Probabilities, k, Trip_Chance);

        % Restrict pay-off matrix of original game
        for i = 1:length(Index_Row_Strats)
            A = [A;Old_A(Index_Row_Strats(i),:)];
        end

        % Solve restricted game
        r=[];s=[];[m,n]=size(A);
        if min(max(A))==max(min(A'))
```

```
            b=max(A);Strategy_Ist=[];Strategy_IInd=[];ms=[];
            for i=1:n
                for j=1:m
                    if isequal(b(i),A(j,i))
                        if isequal(A(j,i),min(A(j,:)))
                            r(length(r)+1)=j;
                            s(length(s)+1)=i;
                        end
                    end
                end
            end
            if (length(r)==1 && length (s)==1)
                Value = num2str(A(r,s),6);
            else
                for i=1:length(r)
                    ms=[ms '(' int2str(r(i)) ',' int2str(s(i)) '),'];
                end
                Value = num2str(A(r(1),s(1)),6);
            end
        else
            X_a=linprog(-[1;zeros(m,1)],[ones(n,1) -A'],zeros(n,1),[0 ones(1,m)],[1],[-inf;zeros(m,1)]);v=X_a(1,1);X_a(1,:)=[];
            X_b=linprog([1;zeros(n,1)],[-ones(m,1) A],zeros(m,1),[0 ones(1,n)],[1],[-inf;zeros(n,1)]);X_b(1,:)=[];
            Value = num2str(v,6);
        end

        % Compare values
        if(abs(str2double(Old_Value) - str2double(Value)) < 0.00001)
            Hypo1 = [Hypo1;Probabilities(length(Probabilities)-1:length(Probabilities))];
        else
            Hypo2 = [Hypo2;Probabilities(length(Probabilities)-1:length(Probabilities))];
        end
    end
end

% Generate 2d plot of all grid points
figure(1)
grid on;

% Plot Optimal points
if (~isempty(Hypo1))
    x1 = Hypo1(:,1);
    y1 = Hypo1(:,2);
    scatter(x1,y1,10,'g', 'filled');axis([0 1 0 1]);
    hold on;
end

% Plot Non-Optimal points
if (~isempty(Hypo2))
    x2 = Hypo2(:,1);
    y2 = Hypo2(:,2);
    scatter(x2,y2,10,'r', 'filled');axis([0 1 0 1]);
    hold on;
end

% Set legend
if (isempty(Hypo1))
    legend('Non-Optimal');
elseif(isempty(Hypo2))
    legend('Optimal');
else
    legend('Optimal', 'Non-Optimal');
end

% Plot titles
legend('Location','northeastoutside');
Title_String = '';
for i = 1:length(Probabilities)-2
    Title_String = [Title_String, ' p_', num2str(i), ' = ', ' ', num2str(Probabilities(i)), ','];
end
title({
    ['Stepsize = ', num2str(Stepsize), ',']
    [Title_String]
    %['Extra Trip Chance = ', num2str(Trip_Chance)]
    });
xlabel(['p_', num2str(length(Probabilities)-1)]);
ylabel(['p_', num2str(length(Probabilities))]);
```

A.1.3. Section 2.5 - Experiment 2.5.2

```
%% ReducedValue(k, Trip_Chance, Steps)
%  Test if there exist a pure strategy set that can always form an optimal strategy
%  Input:  k := Number of Objects
%          Trip_Chance:= Extra fail probability in p_1 or p_2
%          Steps := The stepsize of the grid
%   Output: Res1 := Set of permutations that do not satisfy the statement
%           Res2 := Set of permutations that satisfy the statement

function [Res1, Res2] = ReducedValue(k, Trip_Chance, Steps)

% Define rational formating
format rat;

% Define stepsize, and the start and end of the grid
Stepsize = 1/Steps;
Grid_Start = Stepsize;
Grid_End = 1 - Stepsize;

% Define all combinations of pure strategy sets
n = 4;
Num_RowStrats = factorial(n)/factorial(k);
Num_ColStrats = factorial(n)/(factorial(k) * factorial(n-k));
All_Indices = nchoosek(1:Num_RowStrats, Num_ColStrats);
Reductions_Size = size(All_Indices);
Num_Reductions = Reductions_Size(1);

% Empty pure strategy set list
Res1 = [];
Res2 = [];

for z = 1:1:Num_Reductions

    % Empty list for probabilities p
    Hypo1 = [];
    Hypo2 = [];

    % Current pure strategy combination
    Index_Row_Strats = All_Indices(z,:);
    error = 0;

    for x = Grid_Start:Stepsize:Grid_End
        for y = Grid_Start:Stepsize:Grid_End
            for x1 = Grid_Start:Stepsize:Grid_End
                for y1 = Grid_Start:Stepsize:Grid_End

                    % Define probabilty p
                    Probabilities = [x, y, x1, y1];
                    A = [];

                    % Solve original game for (x,y,x1,y1)
                    [Old_A, Saddle_Point, Opt_Hider, Opt_Searcher, Searcher_Pures, Old_Value, Mod_A] = Solver(Probabilities, k, Trip_Chance);

                    % Reduce pay-off matrix of original game
                    for i = 1:length(Index_Row_Strats)
                        A = [A;Old_A(Index_Row_Strats(i),:)];
                    end

                    % Solve restricted game
                    r=[];s=[];[m,n]=size(A);
                    if min(max(A))==max(min(A'))
                        b=max(A);Strategy_Ist=[];Strategy_IInd=[];ms=[];
                        for i=1:n
                            for j=1:m
                                if isequal(b(i),A(j,i))
                                    if isequal(A(j,i),min(A(j,:)))
                                        r(length(r)+1)=j;
                                        s(length(s)+1)=i;
                                    end
                                end
                            end
                        end
                        if (length(r)==1 && length (s)==1)
                            Value = num2str(A(r,s),6);
                        else
                            for i=1:length(r)
                                ms=[ms '(' int2str(r(i)) ',' int2str(s(i)) '),'];
                            end
                            Value = num2str(A(r(1),s(1)),6);
                        end
                    else
                        X_a=linprog(-[1;zeros(m,1)],[ones(n,1) -A'],zeros(n,1),[0 ones(1,m)],[1],[-inf;zeros(m,1)]);v=X_a(1,1);X_a(1,:)=[];
                        X_b=linprog([1;zeros(n,1)],[-ones(m,1) A],zeros(m,1),[0 ones(1,n)],[1],[-inf;zeros(n,1)]);X_b(1,:)=[];
                        Value = num2str(v,6);
                    end

                    % Compare values
                    if(abs(str2double(Old_Value) - str2double(Value)) < 0.00001)
                        Hypo1 = [Hypo1;Probabilities(length(Probabilities)-1:length(Probabilities))];
                    else
```

```
                        Hypo2 = [Hypo2; Probabilities(length(Probabilities)-1:length(Probabilities))];

                        % Current combination does not satisfy statement
                        error = 1;
                    end

                    if(error == 1)
                        break;
                    end
                end

                if(error == 1)
                    break;
                end
            end

            if(error == 1)
                break;
            end
        end

        if(error == 1)
            break;
        end
    end

    % Group pure strategy set combinations
    if(error == 1)
        Res1 = [Res1; Index_Row_Strats];
    else
        Res2 = [Res2; Index_Row_Strats];
    end
end
```

A.1.4. Chapter 4 - Matrix Game Solver

```
%%  MatrixGameSolver(A)
%   Solves a matrix game
%   Input:    A := Pay-off Matrix
%   Output:  Value := Value of the game
%            X_a := Optimal Search Strategy
%            X_b := Optimal Hiding Strategy
function[Value, X_a, X_b] = MatrixGameSolver(A)

r=[];s=[];[m,n]=size(A);
if min(max(A))==max(min(A'))
    b=max(A);Strategy_Ist=[];Strategy_IInd=[];ms=[];X_a = 0; X_b = 0;
    for i=1:n
        for j=1:m
            if isequal(b(i),A(j,i))
                if isequal(A(j,i),min(A(j,:)))
                    r(length(r)+1)=j;
                    s(length(s)+1)=i;
                end
            end
        end
    end
    if (length(r)==1 && length (s)==1)
        Value = A(r,s);
    else
        for i=1:length(r)
            ms=[ms '(' int2str(r(i)) ',' int2str(s(i)) '),'];
        end
        Value = A(r(1),s(1));
    end
else
    X_a=linprog(-[1;zeros(m,1)],[ones(n,1) -A'],zeros(n,1),[0 ones(1,m)],[1],[-inf;zeros(m,1)]);v=X_a(1,1);X_a(1,:)=[];
    X_b=linprog([1;zeros(n,1)],[-ones(m,1) A],zeros(m,1),[0 ones(1,n)],[1],[-inf;zeros(n,1)]);X_b(1,:)=[];
    Value = v;
end
end
```

A.1.5. Section 4.1 - Pay-off Matrix for Type One Games

```
%%  Construct_MatrixGame1(Probabilities)
%   Constructs the Pay-Off Matrix for Game 1
%   Input:  Probabilities := Array of the success probability p = (p_1, .. , p_n)
%   Output: A := Pay-off Matrix
%           Probabilities := Array of the success probability p = (p_1, .. , p_n)
%           ColStrats := All Hider's Pure Strategies
%           RowStrats := All Searcher's Pure Strategies
function [A, Probabilities, ColStrats, RowStrats] = Construct_MatrixGame1(Probabilities)
```

```
% Define formating of data, either rational or decimal.
format rat;

% Matrix size
n = length(Probabilities);
nrow = n - 2;
ncol = n - 1;

% Empty matrix
A = ones(nrow, ncol);

% Hider's Pure Strategies
ColStrats = 2:n;

% Searcher's Pure Strategies
InBetween = transpose(2:(n-1));
RowStrats = [ones(size(InBetween)) InBetween ones(size(InBetween)) * n];

for i = 1:nrow

    cRowStrat = RowStrats(i,:);

    for j = 1:ncol

        cColStrat = ColStrats(j);

        Search_Index = 1;
        Counter = 0;

        while Counter ~= 1

            % Update current searched element
            Searched_Element = cRowStrat(Search_Index);

            % Update total success probability
            A(i,j) = A(i,j) * Probabilities(Searched_Element);

            % Check if one of the hidden objects has been found
            if(ismember(Searched_Element, cColStrat))
                Counter = 1;
            end

            % Check if cRowStrat can not find the object
            if(Search_Index == size(RowStrats,2) && Counter ~= 1)
                Counter = 1;
                A(i,j) = A(i,j) * 0;
            else
                % Update the next element to be searched
                Search_Index = Search_Index + 1;
            end
        end
    end
end
end
```

A.1.6. Section 4.2 - Pay-off Matrix for Type Two Games

```
%% Construct Pay-Off Matrix for Game 2
function [A, Probabilities, ColStrats, RowStrats] = Construct_MatrixGame2(Probabilities)

% Define formating of data, either rational or decimal.
format rat;

% Matrix size
n = length(Probabilities);
nrow = factorial(n-1) - factorial(n-2);
ncol = n-1;

% Empty matrix
A = ones(nrow, ncol);

% Hider's Pure Strategies
ColStrats = 2:n;

% Searcher's Pure Strategies
Perms = perms(1:n);
RowStrats = sortrows(Perms(find(Perms(:,1) == 1 & Perms(:,2) ~= n),:));

for i = 1:nrow

    cRowStrat = RowStrats(i,:);

    for j = 1:ncol
```

```
        cColStrat = ColStrats(j);

        Search_Index = 1;
        Counter = 0;

        while Counter ~= 1

            % Update current searched element
            Searched_Element = cRowStrat(Search_Index);

            % Update total success probability
            A(i,j) = A(i,j) * Probabilities(Searched_Element);

            % Check if one of the hidden objects has been found
            if(ismember(Searched_Element, cColStrat))
                Counter = 1;
            else
                % Update the next element to be searched
                Search_Index = Search_Index + 1;
            end
        end
    end
end
end
```

A.1.7. Section 4.3 - Pay-off Matrix for Type Three Games

```
%% Construct Pay-Off Matrix for Game 3
function [A, Probabilities, ColStrats, RowStrats] = Construct_MatrixGame3(Probabilities)

% Define formating of data, either rational or decimal.
format rat;

% Number of vertices and inner vertices
n = length(Probabilities);
m = n-2;

% Hider's Pure Strategies
ColStrats = transpose(2:n);

% Searcher's Pure Strategies
Permutations = sortrows(perms(1:n));
nSearchedLocations = 3;
RowStrats = [];

while(nSearchedLocations ~= n+1)
    % Pure strategies of size nSearchedLocations
    Perms = unique(Permutations(Permutations(:,1) == 1 & Permutations(:,nSearchedLocations) == n, 1:nSearchedLocations), 'rows');

    % Fill remaining columns with zeros
    [zeroRows, zeroCols] = size(Perms);
    FilledPerms = [Perms zeros(zeroRows, n-zeroCols)];

    % Add the strategies to RowStrats
    RowStrats = [RowStrats; FilledPerms];
    nSearchedLocations = nSearchedLocations + 1;
end

% Empty matrix
ncol = length(ColStrats);
nrow = length(RowStrats);
A = ones(nrow, ncol);

for i = 1:nrow
    % Current Row Strategy
    cRowStrat = RowStrats(i,:);

    for j = 1:ncol
        % Current Column Strategy
        cColStrat = ColStrats(j,:);

        % Variable indicating location searched currently
        Search_Index = 1;
        Counter = 0;

        % While hidden object has not been found
        while Counter ~= 1

            % Update current searched element
            Searched_Element = cRowStrat(Search_Index);

            % The last point is searched and we can't move back
            if(Searched_Element == 0)
                A(i,j) = A(i,j) * 0;
                Counter = 1;
            else
                % Update total success probability
                A(i,j) = A(i,j) * Probabilities(Searched_Element);

                % Check if one of the hidden objects has been found
                if(ismember(Searched_Element, cColStrat))
                    Counter = 1;
                end

                % Update the next element to be searched
                Search_Index = Search_Index + 1;
            end
        end
    end
end
end
```

A.2. Proofs

A.2.1. Claim: Equality 2.15

To proof: The following equation holds

$$T_{k-1}(A\cup i)z_j - T_{k-1}(A\cup j)z_i = z_j T_{k-1}(A) - z_i T_{k-1}(A).$$

Proof. By definition, it follows that

$$T_{k-1}(A\cup i)z_j - T_{k-1}(A\cup j)z_i = z_j\cdot\Big(\sum_{B\in(A\cup i)^{(k-1)}}\prod_{l\in B} z_l\Big) - z_i\cdot\Big(\sum_{B\in(A\cup j)^{(k-1)}}\prod_{l\in B} z_l\Big).$$

Notice that

$$(A\cup i)^{(k-1)} \equiv \{B\subseteq(A\cup i): |B| = k-1\} = \{B\subseteq A: |B| = k-1\}\cup\{i\cup B: B\subseteq A\ \&\ |B| = k-2\}$$

in which both sets are disjoint. A similar result follows for the set $(A\cup j)^{(k-1)}$. Therefore, it follows that

$$\begin{aligned} T_{k-1}(A\cup i)z_j - T_{k-1}(A\cup j)z_i &= z_j\cdot\Big(\sum_{B\in A^{(k-1)}}\prod_{l\in B} z_l\Big) + z_i z_j\Big(\cdot\sum_{B\in A^{(k-2)}}\prod_{l\in B} z_l\Big)\\ &\quad - z_i\cdot\Big(\sum_{B\in A^{(k-1)}}\prod_{l\in B} z_l\Big) - z_i z_j\cdot\Big(\sum_{B\in A^{(k-2)}}\prod_{k\in B} z_k\Big)\\ &= z_j T_{k-1}(A) - z_i T_{k-1}(A). \end{aligned}$$

□

A.2.2. Claim: Equation 2.23 has two Telescopic terms

To proof: The term

$$\Big(\sum_{x=1}^{i}(1-p_x)\prod_{y=x+1}^{i} p_y\Big) + \Big(\sum_{x=i+2}^{n}(1-p_x)\prod_{y=x+1}^{n} p_y\prod_{z=1}^{i} p_z\Big) + \Big(\prod_{y=1}^{i} p_y\prod_{y=i+2}^{i} p_y\Big) - 1$$

has two telescopic terms.

Proof. Consider the terms

$$\sum_{x=1}^{i}(1-p_x)\prod_{y=x+1}^{i} p_y \quad\text{and}\quad \sum_{x=i+2}^{n}(1-p_x)\prod_{y=x+1}^{n} p_y\prod_{z=1}^{i} p_z.$$

By writing out the summations, it follows that

$$\begin{aligned} \sum_{x=1}^{i}(1-p_x)\prod_{y=x+1}^{i} p_y &= (1-p_1)p_2\ldots p_i + (1-p_2)p_3\ldots p_i + \ldots + (1-p_{i-1})p_i + (1-p_i)\\ &= 1 - p_1\ldots p_i = 1 - \prod_{y=1}^{i} p_y. \end{aligned}$$

and

$$\sum_{x=i+2}^{n}(1-p_x)\prod_{y=x+1}^{n}p_y\prod_{z=1}^{i}p_z=\left(\prod_{z=1}^{i}p_z\right)\Big((1-p_{i+2})p_{i+3}....p_n+(1-p_{i+3})p_{i+4}....p_n$$
$$+....+(1-p_{n-1})p_n+(1-p_n)\Big)$$
$$=\left(\prod_{z=1}^{i}p_z\right)(1-p_{i+2}....p_n)=\left(\prod_{z=1}^{i}p_z\right)\left(1-\prod_{y=i+2}^{n}p_y\right)$$

□

A.2.3. Claim: Relation 3.23

To proof:

$$(A\cap B)'\subseteq(A'\cap B').$$

Proof. Let $x\in(A\cap B)'$. If x is a leaf vertex, then x must be in $(A\cap B)$. Since $(A\cap B)\subset(A'\cap B')$, it follows that $x\in(A'\cap B')$. If x is not a leaf vertex, then there must exist a leaf vertex $y\in(A\cap B)$ such that the subtree spanned by y contains x. Since $y\in(A\cap B)\subset(A'\cap B')$, it follows that $y\in A'$ and $y\in B'$. Hence, all vertices that are in the subtree spanned by y must be in A' and B' which includes x, and therefore $x\in(A'\cap B')$. □

A.2.4. Conjecture 4.3.1: Equation 4.36

To proof: The searching strategy $\underline{s}$ guarantees the searcher an expected pay-off of at least

$$V_G=\mu p_A p_{X_1}p_Z\left(1+\sum_{j=2}^{m-1}\prod_{i=2}^{j}p_{X_i}\right).$$

Proof. Let μ be the normalisation constant, i.e.

$$\mu=\left(\sum_{i=1}^{m}s_{X_i}+mp_Z-1\right)^{-1}.$$

Assume the searcher plays the mixed strategy $\underline{s}$ as in conjecture 4.3.1. There are four cases:

- The hider plays $h(X_1)$ so that the expected pay-off is given by

$$P\big(h(X_1),\underline{s}\big)=\left(1-(m-1)p_Z+p_Z\sum_{j=2}^{m-1}\prod_{i=2}^{j}p_{X_i}\right)\mu\cdot p_A p_{X_1}+(mp_Z-1)\mu\cdot p_A p_{X_1}$$
$$=\underline{\mu p_A p_{X_1}}-\underline{m\mu p_A p_{X_1}p_Z}+\mu p_A p_{X_1}p_Z+\mu p_A p_{X_1}p_Z\sum_{j=2}^{m-1}\prod_{i=2}^{j}p_{X_i}+\underline{m\mu p_A p_{X_1}p_Z}-\underline{\mu p_A p_{X_1}}$$
$$=\mu p_A p_{X_1}p_Z\left(1+\sum_{j=2}^{m-1}\prod_{i=2}^{j}p_{X_i}\right).$$

– The hider plays $h(X_m)$ so that the expected pay-off is given by

$$\begin{aligned}
P\big(h(X_m),\underline{s}\big) &= \frac{p_{X_1}}{p_{X_m}}\Big(p_Z + \prod_{j=2}^{m} p_{X_j} - (m+1)p_Z \prod_{j=2}^{m} p_{X_j} + p_Z \sum_{j=2}^{m}\prod_{i=2}^{j} p_{X_i}\Big)\mu\cdot p_A p_{X_m} \\
&\qquad + (mp_Z - 1)\mu\cdot p_A \prod_{i=1}^{m} p_{X_i} \\
&= \mu p_A p_{X_1} p_Z + \underline{\mu p_A \textstyle\prod_{j=1}^{m} p_{X_j}} - \underline{m\mu p_A p_Z \textstyle\prod_{j=1}^{m} p_{X_j}} - \mu p_A p_Z \prod_{j=1}^{m} p_{X_j} \\
&\qquad + \mu p_A p_{X_1} p_Z \sum_{j=2}^{m}\prod_{i=2}^{j} p_{X_i} + \underline{m\mu p_A p_Z \textstyle\prod_{j=1}^{m} p_{X_j}} - \underline{\mu p_A \textstyle\prod_{j=1}^{m} p_{X_j}} \\
&= \mu p_A p_{X_1} p_Z - \mu p_A p_Z \prod_{j=1}^{m} p_{X_j} + \mu p_A p_{X_1} p_Z \sum_{j=2}^{m}\prod_{i=2}^{j} p_{X_i} \\
&= \mu p_A p_{X_1} p_Z \Big(1 + \sum_{j=2}^{m-1}\prod_{i=2}^{j} p_{X_i}\Big).
\end{aligned}$$

– The hider plays $h(X_i)$ with $i \neq 1, m$, so that the expected pay-off is given by

$$\begin{aligned}
P\big(h(X_i),\underline{s}\big) &= \frac{p_{X_1}}{p_{X_i}}\Big(p_Z + \prod_{j=2}^{i} p_{X_j} - mp_Z \prod_{j=2}^{i} p_{X_j} + p_Z \sum_{j=2}^{m-1}\prod_{k=2}^{j} p_{X_k}\Big)\mu\cdot p_A p_{X_i} + (mp_Z - 1)\mu\cdot p_A \prod_{j=1}^{i} p_{X_j} \\
&= \mu p_A p_{X_1} p_Z + \underline{\mu p_A p_{X_1} \textstyle\prod_{j=2}^{i} p_{X_j}} - \underline{m\mu p_A p_{X_1} p_Z \textstyle\prod_{j=2}^{i} p_{X_j}} + \mu p_A p_{X_1} p_Z \sum_{j=2}^{m-1}\prod_{k=2}^{j} p_{X_k} \\
&\qquad + \underline{m\mu p_A p_Z \textstyle\prod_{j=1}^{i} p_{X_j}} - \underline{\mu p_A \textstyle\prod_{j=1}^{i} p_{X_j}} \\
&= \mu p_A p_{X_1} p_Z \Big(1 + \sum_{j=2}^{m-1}\prod_{k=2}^{j} p_{X_k}\Big),
\end{aligned}$$

for $i = 2, \dots, m$.

– The hider plays $h(Z)$ so that the expected pay-off is given by

$$
\begin{aligned}
P\big(h(Z),\underline{s}\big) = &\left(1-(m-1)p_Z + p_Z \sum_{j=2}^{m-1}\prod_{i=2}^{j} p_{X_i}\right)\mu\cdot p_A p_{X_1} p_Z \\
&+ \frac{p_{X_1}}{p_{X_m}}\left(p_Z + \prod_{j=2}^{m} p_{X_j} - (m+1)p_Z \prod_{j=2}^{m} p_{X_j} + p_Z \sum_{j=2}^{m}\prod_{i=2}^{j} p_{X_i}\right)\mu\cdot p_A p_{X_m} p_Z \\
&+ \sum_{i=2}^{m-1}\left(\frac{p_{X_1}}{p_{X_i}}\left(p_Z + \prod_{j=2}^{i} p_{X_j} - m p_Z \prod_{j=2}^{i} p_{X_j} + p_Z \sum_{j=2}^{m-1}\prod_{k=2}^{j} p_{X_k}\right)\mu\cdot p_A p_{X_i} p_Z\right) \\
&+ (m p_Z - 1)\mu\cdot p_A p_Z \prod_{i=1}^{m} p_{X_i} \\
= &\ \mu p_A p_{X_1} p_Z - \underline{(m-1)\mu p_A p_{X_1} p_Z^2} + \mu p_A p_{X_1} p_Z^2 \sum_{j=2}^{m-1}\prod_{i=2}^{j} p_{X_i} \\
&+ \underline{\mu p_A p_{X_1} p_Z^2} + \underline{\mu p_A p_{X_1} p_Z \textstyle\prod_{j=2}^{m} p_{X_j}} - (m+1)\mu p_A p_{X_1} p_Z^2 \prod_{j=2}^{m} p_{X_j} + \mu p_A p_{X_1} p_Z^2 \sum_{j=2}^{m}\prod_{i=2}^{j} p_{X_i} \\
&+ \sum_{i=2}^{m-1}\left(\underline{\mu p_A p_{X_1} p_Z^2} + \mu p_A p_{X_1} p_Z \prod_{j=2}^{i} p_{X_j} - m\mu p_A p_{X_1} p_Z^2 \prod_{j=2}^{i} p_{X_j} + \mu p_A p_{X_1} p_Z^2 \sum_{j=2}^{m-1}\prod_{k=2}^{j} p_{X_k}\right) \\
&+ m\mu p_A p_Z^2 \prod_{i=1}^{m} p_{X_i} - \underline{\mu p_A p_Z \textstyle\prod_{i=1}^{m} p_{X_i}} \\
= &\ \mu p_A p_{X_1} p_Z + \underline{\mu p_A p_{X_1} p_Z^2 \textstyle\sum_{j=2}^{m-1}\prod_{i=2}^{j} p_{X_i}} - \underline{(m+1)\mu p_A p_Z^2 \textstyle\prod_{j=1}^{m} p_{X_j}} + \underline{\mu p_A p_{X_1} p_Z^2 \textstyle\sum_{j=2}^{m}\prod_{i=2}^{j} p_{X_i}} \\
&+ \sum_{i=2}^{m-1}\left(\mu p_A p_Z \prod_{j=1}^{i} p_{X_j} - m\mu p_A p_Z^2 \prod_{j=1}^{i} p_{X_j}\right) + (m-2)\mu p_A p_{X_1} p_Z^2 \sum_{j=2}^{m-1}\prod_{k=2}^{j} p_{X_k} \\
&+ \underline{m\mu p_A p_Z^2 \textstyle\prod_{i=1}^{m} p_{X_i}} \\
= &\ \mu p_A p_{X_1} p_Z + \underline{2\mu p_A p_{X_1} p_Z^2 \textstyle\sum_{j=2}^{m-1}\prod_{i=2}^{j} p_{X_i}} + \underline{\mu p_A p_Z^2 \textstyle\prod_{i=1}^{m} p_{X_i}} - \underline{\mu p_A p_Z^2 \textstyle\prod_{i=1}^{m} p_{X_i}} \\
&+ \sum_{i=2}^{m-1}\left(\mu p_A p_Z \prod_{j=1}^{i} p_{X_j} - m\mu p_A p_Z^2 \prod_{j=1}^{i} p_{X_j}\right) + \underline{(m-2)\mu p_A p_{X_1} p_Z^2 \textstyle\sum_{j=2}^{m-1}\prod_{k=2}^{j} p_{X_k}} \\
= &\ \mu p_A p_{X_1} p_Z + \underline{m\mu p_A p_{X_1} p_Z^2 \textstyle\sum_{j=2}^{m-1}\prod_{i=2}^{j} p_{X_i}} + \sum_{i=2}^{m-1}\mu p_A p_Z \prod_{j=1}^{i} p_{X_j} - \underline{\textstyle\sum_{i=2}^{m-1} m\mu p_A p_Z^2 \prod_{j=1}^{i} p_{X_j}} \\
= &\ \mu p_A p_{X_1} p_Z \left(1 + \sum_{j=2}^{m-1}\prod_{k=2}^{j} p_{X_k}\right).
\end{aligned}
$$

Hence, the hider is indifferent between all his pure strategies and the searching strategy $\underline{s}$ guarantees that the pay-off is at least

$$
V_G = \mu p_A p_{X_1} p_Z \left(1 + \sum_{j=2}^{m-1}\prod_{k=2}^{j} p_{X_k}\right).
$$

□

A.2.5. Theorem 5: Equation 4.43

To proof:

$$
P\big(h(Z),\underline{s}\big) = \mu p_A + \frac{2\mu p_A p_B p_C p_Z - \mu p_A p_B p_C}{p_B p_C + p_B p_Z + p_C p_Z - 2 p_B p_C p_Z}.
$$

Proof. Assume the hider plays $h(Z)$ and the searcher plays $\underline{s}$ as defined in theorem 5. Then the expected pay-off is given by

$$\begin{aligned}
P\big(h(Z),\underline{s}\big) &= \left(\frac{\mu}{p_B} - \frac{2\mu p_Z - \mu}{p_B p_C + p_B p_Z + p_C p_Z - 2p_B p_C p_Z}\right)\cdot p_A p_B p_Z \\
&\quad + \left(\frac{\mu}{p_C} - \frac{2\mu p_Z - \mu}{p_B p_C + p_B p_Z + p_C p_Z - 2p_B p_C p_Z}\right)\cdot p_A p_C p_Z \\
&\quad + \frac{2\mu p_Z - \mu}{p_B p_C + p_B p_Z + p_C p_Z - 2p_B p_C p_Z}\cdot p_A p_B p_C p_Z \\
&\quad + \frac{2\mu p_Z - \mu}{p_B p_C + p_B p_Z + p_C p_Z - 2p_B p_C p_Z}\cdot p_A p_B p_C p_Z \\
&= 2\mu p_A p_Z + \frac{4\mu p_A p_B p_C p_Z^2 - 2\mu p_A p_B p_C p_Z - 2\mu p_A p_B p_Z^2 - 2\mu p_A p_C p_Z^2 + \mu p_A p_B p_Z + \mu p_A p_C p_Z}{p_B p_C + p_B p_Z + p_C p_Z - 2p_B p_C p_Z} \\
&= 2\mu p_A p_Z - 2\mu p_A p_Z\left(\frac{p_B p_C + p_B p_Z + p_C p_Z - 2p_B p_C p_Z}{p_B p_C + p_B p_Z + p_C p_Z - 2p_B p_C p_Z}\right) + \frac{\mu p_A p_B p_Z + \mu p_A p_C p_Z}{p_B p_C + p_B p_Z + p_C p_Z - 2p_B p_C p_Z} \\
&= \mu p_A \cdot \frac{p_B p_C + p_B p_Z + p_C p_Z - 2p_B p_C p_Z - p_B p_C + 2p_B p_C p_Z}{p_B p_C + p_B p_Z + p_C p_Z - 2p_B p_C p_Z} \\
&= \mu p_A + \frac{2\mu p_A p_B p_C p_Z - \mu p_A p_B p_C}{p_B p_C + p_B p_Z + p_C p_Z - 2p_B p_C p_Z}.
\end{aligned}$$

□

Bibliography

[1] Steve Alpern and Shmuel Gal. *The theory of search games and rendezvous*, volume 55. Springer Science & Business Media, 2006.

[2] D. Avis, G. Rosenberg, R. Savani, and B. von Stengel. Enumeration of nash equilibria for two-player games. *Economic Theory*, 42(1):9–37, 2010. Online solver available at http://banach.lse.ac.uk. Accessed: 2020-12-20.

[3] Nicolas Barnier and Pascal Brisset. Solving the kirkman's schoolgirl problem in a few seconds. In *International Conference on Principles and Practice of Constraint Programming*, pages 477–491. Springer, 2002. Accessed: 2021-04-09.

[4] Dimitris Bertsimas and José Nino-Mora. Conservation laws, extended polymatroids and multiarmed bandit problems; a polyhedral approach to indexable systems. *Mathematics of Operations Research*, 21 (2):257–306, 1996. Accessed: 2021-01-13.

[5] Weisstein Eric W. Kirkman's schoolgirl problem. *From MathWorld–A Wolfram Web Resource*. https://mathworld.wolfram.com/KirkmansSchoolgirlProblem.html. Accessed: 2021-01-13.

[6] Thomas S Ferguson. *A Course in Game Theory*. World Scientific, 2020. Accessed: 2021-01-26.

[7] Ryusuke Hohzaki. Search games: Literature and survey. *Journal of the Operations Research Society of Japan*, 59(1):1–34, 2016.

[8] Samuel Karlin. *Mathematical Methods and Theory in Games, Programming, and Economics*. Addison-Wesley Publishing Company, June 2019. ISBN 978-1-483-22400-8. Accessed: 2021-02-27.

[9] Thomas Lidbetter. Search games with multiple hidden objects. *SIAM Journal on Control and Optimization*, 51(4):3056–3074, 2013. Accessed: 2020-12-02.

[10] Hariharan Narayanan. *Submodular functions and electrical networks*, volume 54. Elsevier, 1997. Accessed: 2021-04-16.

[11] Ray Toal. Solving the kirkman schoolgirl problem. Online article available at https://cs.lmu.edu/~ray/notes/kirkman/. Accessed: 2021-04-09.