The London School of Economics and Political Science

Hide-and-seek and Other Search Games

Thomas Lidbetter

A thesis submitted to the Department of Mathematics of the London School of Economics for the degree of Doctor of Philosophy, London, May 2013

Declaration

I certify that the thesis I have presented for examination for the MPhil/PhD degree of the London School of Economics and Political Science is solely my own work other than where I have clearly indicated that it is the work of others (in which case the extent of any work carried out jointly by me and any other person is clearly identified in it).

I warrant that this authorisation does not, to the best of my belief, infringe the rights of any third party.

I declare that my thesis consists of 24535 words.

Statement of conjoint work

I confirm that Chapter 2 was jointly co-authored with Prof Steve Alpern and I contributed 70% of this work.

I confirm that Chapter 3 was jointly co-authored with Prof Steve Alpern and I contributed 70% of this work.

I confirm that Chapter 6 was jointly co-authored with Prof Steve Alpern and Doctor Robbert Fokkink and I contributed 80% of this work.

Acknowledgements

First and foremost I would like to thank Steve Alpern for providing invaluable supervision of this thesis. His comments and suggestions have been consistently provocative and insightful and it has been a pleasure to work with him.

I would also like to thank Robbert Fokkink for kindly inviting me to visit him on two occasions in the Delft University of Technology; Chapter 6 stemmed from work with Steve Alpern and him. Special thanks must also go to Richard Weber for his many useful discussions which contributed to Chapters 4 and 5.

I must also acknowledge the crucial importance of the work of Shmuel Gal which played a large part in drawing me to this topic and developing my ideas.

I would also like to offer thanks to the staff of the Department of Mathematics at the London School of Economics and my fellow postgraduate students for creating a supportive atmosphere which has aided me through the course of this thesis.

Finally I am grateful to both the Department of Mathematics at the London School of Economics and the Home Office for providing me with the means to complete this thesis.

Abstract

In the game of hide-and-seek played between two players, a *Hider* picks a hiding place and a *Searcher* tries to find him in the least possible time. Since Isaacs had the idea of formulating this mathematically as a zero-sum game almost fifty years ago in his book, *Differential Games*, the theory of *search games* has been studied and developed extensively. In the classic model of search games on networks, first formalised by Gal in 1979, a Hider strategy is a point on the network and a Searcher strategy is a constant speed path starting from a designated point of the network. The Searcher wishes to minimise the time to find the Hider (the payoff), and the Hider wishes to maximise it. Gal solved this game for certain classes of networks: that is, he found optimal strategies and the payoff assuming best play on both sides. Here we study new formulations of search games, starting with a model proposed by Alpern where the speed of the Searcher depends on which direction he is travelling. We give a solution of this game on a class of networks called *trees*, generalising Gal's work. We also show how the game relates to another new model of search studied by Baston and Kikuta, where the Searcher must pay extra *search costs* to search the network's nodes (or vertices). We go on to study another new model of search called *expanding search*, which models coal mining. We solve this game on trees and also study the related problem where the Hider's strategy is known to the Searcher. We extend the expanding search game to consider what happens if there are several hidden objects and solve this game for certain classes of networks. Finally we study a game in which a squirrel hides nuts from a pilferer.

Contents

List of Figures

1 Introduction

Since the conception of *search games* almost fifty years ago, the field has expanded and developed in many different directions. In this thesis we focus in on one particular theme: that of search games with a mobile Searcher and an immobile Hider (or hidden objects). Games of this type may be described as 'hide-and-seek' games. Traditionally, a Hider picks a point or points in some search space and a Searcher moves around the space, trying to find all the points in the least possible time. The Hider wishes to maximise the time and so the problem is formulated as a zero-sum game. The main results in this field can be found (in chronological order) in Gal's book on search games [22], Garnaev's book [25], Alpern and Gal's monograph [11], and Gal's recent survey [24].

In this chapter we begin in Section 1.1 by discussing how Isaacs [28] first introduced search games of this type, and how he described strategies for both the Hider and the Searcher which would continue to be of fundamental importance in later work in the field. In Section 1.2 we then turn to the first rigorous definition, given by Gal [21], of a search game with an immobile Hider and a mobile Searcher who starts from a given point. We indicate how Gal solved his game if the search space is a tree or if it is *Eulerian* (it has an Eulerian cycle), showing that the value of the game in these cases is equal to half the time of the shortest tour of the network.

We then show in Section 1.3 how Reijnierse and Potters [44] extended Gal's analysis to *weakly cyclic* networks, which have the structure of a tree with some nodes replaced by cycles. We describe the solution of Gal's game on

these networks, and how Gal proved an analogous result for *weakly Eulerian* networks, which have the structure of trees with some nodes replaced by Eulerian cycles. Weakly cyclic and weakly Eulerian networks are defined more precisely in Section 1.3.

1.1 The birth of search games

Search games were first introduced by Rufus Isaacs in his 1965 book, Differential Games [28]. The book was originally motivated by combat problems, and indeed, many of the problems discussed in the book have a military focus to them. Earlier chapters in the book are concerned with so called *Pursuit Games*, in which a *Pursuer* (or *Pursuers*) aim to capture an *Evader* whose location is known to him at all times during the game. Search games are introduced later in the book in a chapter called 'Toward a Theory with Incomplete Information'. The model presented differs from Pursuit Games in that Pursuers now aim to capture an Evader about whose position the Pursuers do not have complete information. The terminology changes: the Evader becomes the Hider and the Pursuers become the Searchers. This terminology has stuck and is now widely used in the search games literature.

Isaacs begins by defining what he calls the *simple search game*. This could be regarded as the simplest and most general possible search game, and is described in informal terms. In an arbitrary region $\mathcal{R}$, which may be a subset of Euclidean space of any dimension, a Hider picks a hiding point (that is a point in $\mathcal{R}$). The Searcher then picks some sort of unit speed trajectory in the

region. The payoff, now widely referred to as the *search time*, is the time taken until the Searcher's trajectory meets the Hider. There is an assumption that the Searcher is able to find a tour of the whole region that is not wasteful, so that it does not 'double back' on itself. The solution of the game Isaacs gives is simple: the Searcher picks one such tour S, then follows it with probability $1/2$ and follows the reverse tour with probability $1/2$. Supposing $\mathcal{R}$ has measure μ, if S finds a point in $\mathcal{R}$ at time t, the reverse of S will find the same point at time $\mu - t$. Hence the expected time T to find any given point is given by

$$T = 1/2t + 1/2(\mu - t) = \mu/2.$$

The value of the game is therefore at most $\mu/2$. The Hider can ensure the payoff is no more than $\mu/2$ by hiding uniformly in $\mathcal{R}$, so that the probability he hides in any subset of $\mathcal{R}$ is proportional to its measure. By using this strategy, the Hider ensures that the probability the Searcher finds him before time t is no more than t/μ for $0 \leq t \leq \mu$, so the probability the search time is t or more is at least $1 - t/\mu$ Hence the expected time T satisfies

$$\begin{aligned} T &= \int_0^\infty Pr(\text{search time is } \geq t)dt \\ &\geq \int_0^\mu (1 - t/\mu)\, dt \\ &= \mu/2. \end{aligned}$$

The value of the game is therefore at least $\mu/2$, and combining the bounds we have

Theorem 1 (Isaacs) *The value of the simple search game is* $\mu/2$.

These strategies given by Isaacs are important and direct a lot of the later research on search games.

1.2 Search games on networks

A more precise formulation of Isaacs' game is given by Gal ([21] and [22]). Gal focuses on the game played on a network Q, which is any connected finite set of arcs of measure μ with a distinguished starting point O, called the root. We refer to the points at either end of an arc as *nodes*. The Hider picks a point H in Q, which is not limited to the nodes, and could be anywhere on the network; the Searcher picks a unit speed path S starting from O. The payoff (or search time) is the time taken for the path to reach H.

Gal shows that this game always has a value V, and uses Isaacs' Hider strategy to give a lower bound for V: by hiding uniformly in the network the Hider can ensure that the search time is always at least $\mu/2$. We call this strategy u. However, the assumption made by Isaacs that the Searcher can find a non-wasteful trajectory is not made, so the Searcher strategy given by Isaacs in [28] is not always available and the value of the game may be greater than $\mu/2$. The Searcher is also restricted to picking a path which starts from O, so it may not be possible for him to implement the "reverse" of a path. For instance, if Q is a single arc with the root O at one end and a point A at the other, the value of this game is clearly the length of the arc, $\mu > \mu/2$. The Hider simply uses the pure strategy of hiding at A and the Searcher picks the path from O

to A.

However, adapting Isaacs' strategy, Gal gives an upper bound for the value. The Searcher may not be able to find a non-wasteful, reversible path in Q, but he will always have some minimal time tour S of Q starting and ending at O of length $\bar{\mu} \geq \mu$. He can then use the mixed strategy where he picks S with probability $1/2$ and the reverse of S with probability $1/2$, ensuring that he finds every point in Q in expected time no more than $\bar{\mu}/2$. The Searcher's minimal tour S is later called a Chinese Postman Tour (CPT) in [23], and the randomised strategy given here is called the Random Chinese Postman Tour (RCPT). The RCPT gives an upper bound for the value V, and combining this with the lower bound we have

$$\mu/2 \leq V \leq \bar{\mu}/2 \tag{1}$$

Gal examines when these two bounds are tight. Suppose Q is Eulerian, so that it has a continuous closed path that visits each point of Q exactly once. Then the Searcher's CPT is one such Eulerian path starting at O. Since the length $\bar{\mu}$ of this tour is μ, the bounds in (1) are tight and we have $V = \mu/2 = \bar{\mu}/2$. The uniform strategy u is optimal for the Hider. It is easy to see that Eulerian networks are the only networks for which $\bar{\mu} = \mu$.

We can also consider the game played on a tree, that is a network without any cycles. In a sense, a tree is the opposite of an Eulerian network since the CPT of a tree has the maximum possible length, $\bar{\mu} = 2\mu$, as all arcs must be traversed in both directions. The inequalities (1) therefore become $\mu/2 \leq V \leq \mu$. Clearly

the uniform Hider strategy u is not optimal for the Hider, since every point H of Q is dominated in strategies by a leaf node (a node of degree 1). Hence an optimal Hider strategy must be some distribution on the leaf nodes. In [21] Gal defines a Hider distribution later called the Equal Branch Density (EBD) distribution in [23], and shows that it is optimal for the Hider, guaranteeing him an expected search time of no less than $\mu = \bar{\mu}/2$, which is the value of the game. The RCPT is optimal for the Searcher.

The EBD distribution can be defined in terms of a concept called *search density*, which extends to general search spaces Q that may not be networks.

Definition 2 *For a connected subset A of a search space Q and a Hider hidden on Q according to a fixed distribution, the search density $\rho(A)$ is defined as the probability the Hider is in A divided by the time taken for the Searcher to tour A.*

Consider a tree Q and a node x of Q that has degree at least 3. We call x a *branch node*. The arcs meeting at x consist of one arc on the path from x to O and some other arcs, which we call the *outward* arcs. For each outward arc a, we define a branch Q_a at x which consists of a together with all arcs *above* a (that is, those whose unique path to O intersects a). The EBD distribution is the unique Hider distribution on the leaf nodes of Q that ensures that at every branch node of Q, all branches have equal search density. Summing up:

Definition 3 *The Equal Branch Density (EBD) distribution is the unique distribution on a tree for which at every branch node, all branches have equal search*

density.

We illustrate the EBD distribution with an example. In Figure 1 nodes are labelled by letters and arc lengths indicated by numbers. To calculate the EBD distribution on this network, first note that there are two branches at O, which must have equal search density. This can be achieved by assigning Hider probability $3/9 = 1/3$ to the branch consisting of the arc OC, and probability $2/3$ to the other branch. The branch node D has two branches, and to ensure these have equal search density, the Hider probability assigned to the arcs AD and BD must be proportional to 2 and 3, respectively. Hence the probabilities the Hider is at nodes A and B are $2/5 \cdot 2/3 = 4/15$ and $3/5 \cdot 2/3 = 6/15$ respectively. The probability the Hider is at C is $1/3$.

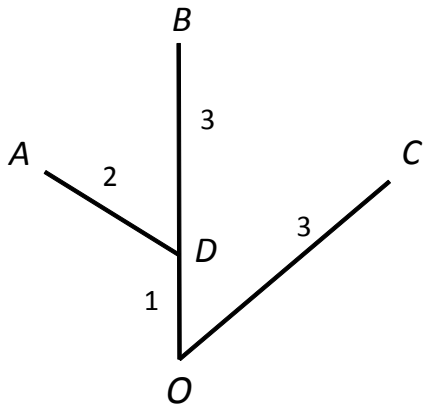

Figure 1: A tree network.

In [21], Gal shows that if the Hider uses the EBD distribution, this ensures that any depth-first search of Q, and in particular any CPT finds the Hider in

expected time exactly $\mu = \bar{\mu}/2$, which must therefore be the value of the game. In the case of the network in Figure 1, the value of the game is $\mu = 9$.

Hence we have

Theorem 4 (Gal) *If Q is an Eulerian network or a tree then the value of the search game with an immobile Hider played on Q is $\bar{\mu}/2$.*

However, Gal goes on to give a class of networks for which the value is not $\bar{\mu}/2$. The class he describes is that of networks consisting of two nodes O and A connected by a set of k disjoint unit length arcs, where k is an odd number. Networks of this form are not Eulerian, and they are certainly not trees. Gal gives a Searcher strategy that finds the Hider in expected time strictly less than $\bar{\mu}$.

The exact solution of the game played on a network of this form was not found for even $k = 3$ for another 14 years, when Pavlovic [40] gave the solution for general k. The optimal strategies for both players are not straightforward. For the 'three-arc network' depicted in Figure 2 (that is, when $k = 3$), the Hider chooses one arc equiprobably, and hides on this arc at a distance determined by some probability density function. The Searcher has an equally complicated optimal strategy involving doubling back.

1.3 Weakly cyclic and weakly Eulerian networks

Solutions of the game described in the previous section are not limited to trees and Eulerian networks. In [44] Reijnierse and Potters solve the game for *weakly cyclic* networks, showing that the RCPT is optimal for the Searcher, so that the

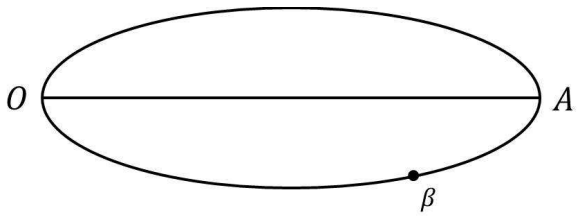

Figure 2: The three-arc network.

value is $\bar{\mu}/2$. A weakly cyclic network can be thought of as a tree network for which some of the nodes have been replaced with cycles. Alternatively, a weakly cyclic network can be defined more precisely as a network for which there are at most two disjoint paths between any two nodes. Weakly cyclic networks cannot contain any subnetwork that is topologically homeomorphic to the three-arc network depicted in Figure 2. A weakly cyclic network is depicted on the left hand side of Figure 3; the cycles are indicated by the dotted lines.

Reijnierse and Potters give an algorithm to calculate the optimal Hider distribution, in which the Hider hides with some probability on leaf nodes and with some probability hides uniformly on the cycles. Alpern and Gal [10] later give an alternative version of the algorithm, in which every cycle in the network is replaced with a leaf arc of half the length of the cycle, and the EBD distribution is calculated on the new network. The network depicted on the right hand side of Figure 3 is the modification of the weakly cycle network on the left. The Hider probability that should be assigned to a cycle in the original network is then the probability assigned to the end of the associated leaf arc in the new network.

Reijnierse [43] later showed that the equivalent result holds if we replace 'weakly

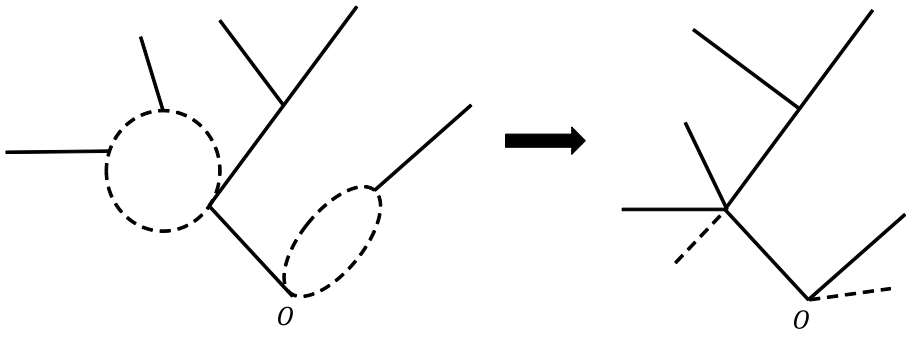

Figure 3: A weakly cyclic network and its modification.

cyclic' with 'weakly Eulerian'. A network is weakly Eulerian if it can be obtained from a tree by replacing some nodes with Eulerian networks. Gal [23] found a simple proof of this result, showing not only that the value V of the game is $\bar{\mu}/2$ for weakly Eulerian networks, but these are the only networks for which this is the value, and the RCPT is optimal. In summary:

Theorem 5 (Gal) *The value of the search game with an immobile Hider played on a network Q is $\bar{\mu}/2$ if and only if Q is weakly Eulerian.*

Notice that the class of weakly Eulerian networks includes both trees and Eulerian networks, so Theorem 5 generalises Theorem 4.

1.4 Layout of thesis

There have been other extensions of Gal's classic search games model, for example see Baston and Bostock [18], Kikuta and Ruckle [31] and Jotshi and Batta [29]. In the bulk of this thesis we consider further extensions of Gal's model.

We begin in Chapter 2 by studying a model of search on a *variable speed* network: that is, a network on which the speed of the Searcher depends on his location and direction of travel. This model was introduced by Alpern [4], who established that if the network is a tree it is optimal for the Hider to use a generalisation of the Equal Branch Density distribution, and found the optimal Searcher strategy in recursive form. Here we give the optimal Searcher strategy in closed form and also give a closed form expression for the value of the game. We then solve the game for some other networks that are equivalent to a circle, topologically. This chapter is based upon parts of Alpern and Lidbetter [13].

In Chapter 3 we consider a model of search called *expanding search*. An expanding search can be thought of as a sequence of unit speed paths on a network Q, starting at O, each of which starts from a point already reached by the Searcher. Another way to think of expanding search is as a family of connected subsets of Q starting with O and expanding at unit speed. To differentiate expanding search from the type of search used in Gal's model, we call the latter *pathwise search*. Expanding search provides a model of mining, in which the time taken to recommence mining from a location already reached is small compared to the time taken up by the mining itself. We give the optimal (minimal expected search time) expanding search in the case that a Hider is located on a tree according to a known distribution. In Chapter 4 we then consider the expanding search game (where the Hider chooses a hiding distribution) on a tree, and show that the solution of this game can be derived from the solution of the search game on a variable speed network, as found in

Chapter 2. This chapter is based upon parts of Alpern and Lidbetter [14].

The remainder of the thesis considers search games with multiple hidden objects. We begin in Chapter 4 by analysing a game in which the Searcher wishes to find k balls hidden amongst $n > k$ boxes. There is a known cost of searching each box, and the Searcher seeks to minimise the total expected cost of finding all the objects in the worst case. We show that it is optimal for the Searcher to begin by searching a k-subset H of boxes with probability $\nu(H)$, which is proportional to the product of the search costs of the boxes in H. The Searcher should then search the $n - k$ remaining boxes in a random order. A worst case Hider distribution is the distribution ν. We distinguish between the case of a *smart* Searcher who can change his search plan as he goes along and a *normal* Searcher who has to set out his plan from the beginning. We show that a smart Searcher has no advantage. We then show how the game can be formulated in terms of an expanding search game for multiple objects, and in Chapter 5 we go on to give upper and lower bounds for the value of the game on an arbitrary network. For 2-arc connected networks (networks that cannot be disconnected by the removal of fewer than 2 arcs), we solve the game for a smart Searcher, and give an upper bound on the value for a normal Searcher. This bound is tight if the network is a circle. This chapter is based upon Lidbetter [35].

Finally, in Chapter 6 we examine a *caching game* in which a Squirrel hides nuts from a Pilferer. The Squirrel wishes to bury m nuts amongst n discrete locations, and has enough energy to dig a total depth of D_S. A Pilferer, who has

total digging resources D_P then digs in these locations, attempting to locate the nuts. If the Squirrel is left with at least k nuts he wins, otherwise the Pilferer wins. We considering the cases of both a smart and normal Pilferer, and solve the games for $n = 2$ and $k = 1$. We then solve the game in some special cases (very large or very small D_P) for general n and $k = 1$. Finally we consider the case of arbitrary k and give a bound on the value of the game in terms of a game where $k = 1$. Parts of this chapter are based on Alpern *et al [9].*

2 Search on Variable Speed Networks

This chapter analyses the search game played between an immobile Hider and a mobile Searcher on a finite network Q with a distinguished root node O. The Hider simply picks a point H in Q (the hiding place). The Searcher chooses a path $S = S(t)$ in Q which starts at O, covers Q, and satisfies a time constraint

$$d(S(t_1), S(t_2)) \leq t_2 - t_1, \text{ for } t_1 < t_2,$$

where $d(x, y)$ is a given quasimetric (satisfying all axioms for a metric except possibly symmetry in x and y) denoting the minimum time required to go *from* x *to* y. We denote the travel time within the subset W by $d_W(x, y)$. If d is symmetric ($d(x, y) = d(y, x)$), and thus a metric, then we say the Q is *time-symmetric*. The payoff of this zero-sum game (to the minimising Searcher) is the *capture time* T given by

$$T(S, H) = \min\{t : S(t) = H\}.$$

Although both players have infinite pure strategy sets, the value V of this game $\Gamma = \Gamma(Q, O, d)$ exists by the usual application of the minimax theorem of Alpern and Gal [10], and we call this number $V(Q)$ the *search value* of Q (or of Q, O, d). In general, both players will require mixed strategies. In the case where Q is a tree, studied in Section 2.3, there are only finitely many undominated pure Hider strategies – namely the leaf nodes. So there the usual minimax theorem for finite games would suffice. As long as there is a cycle in the network, all points in the cycle are undominated, and so even the undominated

Hider strategies are infinite. In some cases the probability distribution function of the Hider's optimal mixed strategy has infinite support, as in the 'two-arc' network whose solution is given in Theorem 19. In keeping with the original notional convention of Gal, we use upper case $S = S(t)$ and H for pure strategies and lower case s and h for mixed strategies. A mixed strategy for the Hider is simply a probability distribution over Q. We consider such a distribution as a probability measure, as that notation is easier, with $h(W)$ denoting the probability the the Hider is in the subset W of Q. For mixed Searcher and Hider strategies s and h we define $T(s,h)$ as the expected search time

$$T(s,h) = \int T(S,H)d(s \times h),$$

where $s \times h$ is the product measure.

The time-symmetry assumption has recently been dropped in Alpern's article [4] on trees, which gave a recursive method of determining the search value of any tree. Here, we extend that work to general networks by determining an explicit formula for the search value of a tree using the recursions in [4]. Much of this chapter is taken from Alpern and Lidbetter [13], although whereas the main results are derived here from Alpern's recursions in [4], they are proved independently in [13]. Our *search value formula for trees* is $V = (1/2)(\tau + \Delta)$, where τ (called the *tour time*) is the minimum time required to tour the tree and Δ (called the *incline* of Q) is a measure of the asymmetry of the quasi-metric d. We define the *height* $\delta(x)$ of a point x in Q as the time difference in getting to and from x with respect to the root O, that is, $\delta(x) = d(O,x) - d(x,O)$. The incline Δ is a weighted average of the heights of the leaf nodes of the tree.

The terminology is based on the idea that going up takes more time than going down.

Our methods also yield a new explicit formula for the value of the Kikuta search game with node searching costs [30] and a simpler derivation of the solution to the foraging (find-and-fetch) search problem of Alpern [5].

We first obtain a complete solution (Theorem 18) for the search game on the circle with concave travel times in both directions. We then consider the more complicated problem where the circle consists of two arcs from the root to its antipode which have identical travel time functions. It turns out that when the antipode is 'downhill' from the root the solution is quite complicated, requiring both players to use distributions over a continuum of pure strategies. In particular, the Searcher sometimes goes all the way around the circle but sometimes reverses direction before completing the tour. This is in stark contrast to the way Gal found that Eulerian networks are searched for symmetric (not variable speed) networks.

The chapter is organised as follows. Section 2.1 gives our assumptions and notations for travel times. Section 2.2 gives a general result about search densities, Section 2.3 gives our derivation of the search value formula for a tree and applications to special cases, including the Kikuta game. Section 2.4 analyses the simplest non-tree, a network consisting of a single loop (a circle). This is the only section where variations of travel times within an arc are of importance. Section 2.5 looks at the circle as consisting of two identical arcs between two points and shows that in certain cases the solution is very complicated, involving

backtracking and a continuum of pure strategies.

2.1 Assumptions and notations for travel times

The most natural way to define travel times is by having a notion of arc length and to specify two speed functions (one for each direction) along the arcs, piecewise continuously. However it turns out to be easier to work directly from the quasimetric $d(x, y)$ giving the travel time from x to y.

In most cases considered here (in particular, for trees), we will only need to know the travel times from one end of an arc (a^-) to the other (a^+). To this end, we define forward and reverse travel times on a, denoted F_a and R_a, by

$$F_a = d_a\left(a^-, a^+\right),\ R_a = d_a\left(a^+, a^-\right),\ \text{and their difference by } D_a = F_a - R_a.$$

(We require the subscript in the form d_a because the shortest time between a^- and a^+ might not be via the arc which connects them.) The orientation of arcs involved in defining F_a and R_a is a matter of choice: for trees, we will always orient arcs away from the root O. For general networks we often choose the orientation so that $F_a \geq R_a$.

2.2 Searching higher density regions first

In both Chapters 2 and 3 we will need to use an elementary result about search densities. As already mentioned above in Definition 2, the search density, or just *density*, of a region of Q is defined as the probability the Hider is in the region divided by the time taken to search the region. We give a general analysis of the well known principle of *searching the higher density region* first, as established

in Proposition 3 of Alpern and Howard [12] and also used in this form in Alpern [4].

We begin by fixing a network Q and a Hider distribution (mixed strategy) h on Q. If S is a search strategy with cumulative capture distribution $G(t) = \Pr(T(S,H) \leq t) = h(S([0,t]))$, then the expected search time, $T(S,h)$ is given by $T(S,h) = \int_0^\infty t \; dG(t)$. Suppose that S searches disjoint regions A and B of Q in time intervals $[a,b]$ and $[b,c]$, that is, $S([a,b]) = A$ and $S([b,c]) = B$. The following lemma considers the question of when the Searcher will do better (reduce T) by searching in the opposite order. The answer is given by the *Search Density Lemma*.

Lemma 6 (Search Density) *Fix a network Q and a Hider distribution h. Suppose S (with cumulative capture distribution G) searches disjoint regions A and B for the first time during time intervals $[a,b]$ and $[b,c]$, while S' searches in the other order (B during $[a, a+(c-b)]$ and A during $[a+(c-b), c]$).*

$$\text{If } \frac{G(c) - G(b)}{c-b} \geq \frac{G(b) - G(a)}{b-a}, \text{ then } T(S,h) \geq T(S',h).$$

In other words the search with higher search density should be carried out first. If two searches have the same search density they can be carried out consecutively in either order.

Proof. The difference $T(S,h) - T(S',h)$ is given by

$$\begin{aligned}
& \left[\int_a^b t\ dG(t) + \int_b^c t\ dG(t)\right] - \left[\int_a^b (t+(c-b))\ dG(t) + \int_b^c (t-(b-a))\ dG(t)\right] \\
= & -\int_a^b (c-b)\, dG(t) + \int_b^c (b-a)\ dG(t) \\
= & -(c-b)(G(b)-G(a)) + (b-a)(G(c)-G(b)) \\
= & (b-a)(c-b)\left(\frac{G(c)-G(b)}{c-b} - \frac{G(b)-G(a)}{b-a}\right) \geq 0.
\end{aligned}$$

■

Notice that the Searcher Density Lemma is not particular to search on a variable speed network, but any type of search for a Hider with cumulative capture distribution G that searches disjoint regions A and B. We will also use the Search Density Lemma later in Chapter 3 in the context of expanding search.

2.3 The search value of a tree

In this section we take Q to be a rooted tree, and for simplicity, a *binary* tree (one with at most two outward arcs at any node, degree at most three). Any tree can be made into a binary tree by adding arbitrarily small additional arcs, so this assumption is not critical. We assume that all arcs are oriented away from O.

Alpern's earlier paper [4] gave a recursive method for computing the search value of a tree. Here we present an explicit formula for the search value, using the notion of the *height* of a point x in Q as the difference between the time to reach x from O and the time to return to O from x. That is, the height $\delta(x)$

of a point x (relative to O which has height 0) in Q is given by

$$\begin{aligned} \delta(x) &= d(O,x) - d(x,O), \text{ or more generally,} \\ \delta_y(x) &= d(y,x) - d(x,y) \text{ for the relative height of } x \text{ above } y. \end{aligned} \tag{2}$$

It is clear that the Hider should only hide at leaf nodes, as all other points of Q are dominated by these. We denote the set of leaf nodes by $\mathcal{L}$. It was already shown in [4] (and by Gal [21] for symmetric trees) that the optimal hiding distribution over the leaf nodes is the *Equal Branch Density* distribution e, as defined in Definition 3. We define the *incline* of Q, denoted Δ, as the mean height of the leaf nodes, with respect to the distribution (notated as a measure) e, that is

$$\Delta = \sum_{i \in \mathcal{L}} e(i) \cdot \delta(i).$$

The sign of Δ determines the relative height of the leaf nodes compared to the root: if $\Delta > 0$, the (weighted) mean height of the leaf nodes is above the root and if $\Delta < 0$, the mean height is below the root. For a tree, the tour time τ is simply the sum of all the forward and reverse times of its arcs, $\tau = \sum_{\text{arcs } a} (F_a + R_a)$. The main result of this section is the value formula for trees:

$$V = \frac{1}{2}(\tau + \Delta).$$

2.3.1 Subtrees and optimal strategies

This subsection shows how our notions of e and Δ can be adapted to subtrees and defines the optimal strategies for the players. By a subtree of a tree Q we

mean a subset of Q which is itself a tree.

Definition 7 (subtrees σ_z, EBD distribution e) *If z is a node or (closed) arc of a rooted tree Q,O, let Q_z denote the subtree consisting of all points of Q whose unique path to O intersects with z. Define the tour time τ_z to be the time taken to tour Q_z (the sum of all the forward and reverse arcs of Q_z). The EBD distribution (measure) e is the unique one concentrated on the leaf nodes that at every branch node x with out arcs a and b gives equal search density to the two branches Q_a and Q_b. That is,*

$$\frac{e\left(Q_a\right)}{\tau_a} = \frac{e\left(Q_b\right)}{\tau_b}, \text{ or simply } \frac{e(Q_a)}{e(Q_x)} = \frac{\tau_a}{\tau_x} = \frac{\tau_a}{\tau_a + \tau_b}. \tag{3}$$

Figure 4 illustrates the calculation of leaf node measure e and height δ on the given tree, which are indicated above each of the four leaf nodes. We recall the convention of putting the travel times F_a and R_a to the left and right of the arc a. The right side is one fourth the tour time of the tree, so its EBD measure e is 1/4, while the left side's is 3/4. Similar ideas give the secondary division of 1/4 into two weights of 1/8 and 3/4 into two weights of 1/2 and 1/4. The leftmost δ is calculated as $(4+7)-(5+2)=4$. The weighted average of the leaf node heights $\delta\left(i\right)$ is

$$\Delta = \frac{1}{2}\left(4\right) + \frac{1}{4}\left(0\right) + \frac{1}{8}\left(-1\right) + \frac{1}{8}\left(+1\right) = 2, \text{ so } V = \frac{1}{2}\left(\tau + \Delta\right) = 17. \tag{4}$$

For the moment, ignore the information about arcs a and b. Readers familiar with the earlier paper [4] will note that there the value (of the full tree) could not be calculated without first analysing the values of the subtrees, which is not what we do here. This approach is forward looking whereas the earlier was

backwards looking (using backwards recursion of the value). A similar difference in approach to the Searcher branching pattern will be seen later on.

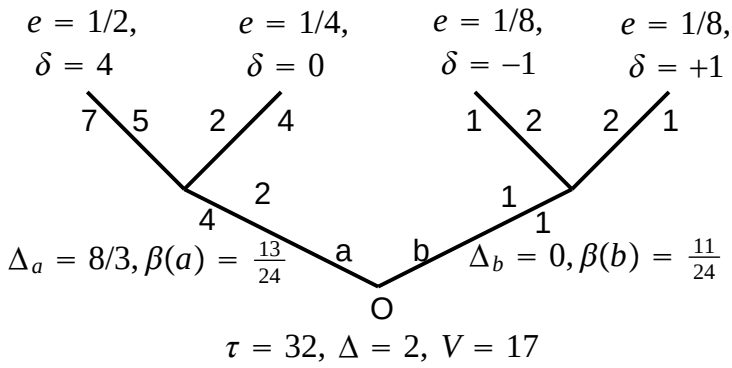

Figure 4: Tree with solution method indicated.

Definition 8 *For any arc or node z, let $\mathcal{L}_z = \mathcal{L} \cap Q_z$ denote the set of leaf nodes of Q_z and let $e_z = e/e(Q_z)$ denote the probability measure induced by e on Q_z. Define the incline Δ_z of Q_z by*

$$\Delta_z = \sum_{i \in \mathcal{L}_z} e_z(i)\delta_z(i).$$

Note that we may rewrite Δ_z in terms of the arcs a in Q_z as

$$\begin{aligned}
\Delta_z &= \sum_{i \in \mathcal{L}_z} e_z(i)\delta_z(i) = \sum_{i \in \mathcal{L}_z} e_z(i) \sum_{\{\text{arcs } a : z \le a < i\}} D_a \\
&= \sum_{\text{arcs } a \in Q_z} D_a \sum_{i \in \mathcal{L}_a} e_z(i) \text{ (since } i \in L_z \text{ and } z \le a < i \iff a \in Q_z \text{ and } i \in \mathcal{L}_a) \\
&= \sum_{\text{arcs } a \in Q_z} e_z(Q_a) D_a \qquad (5)
\end{aligned}$$

We now need a technical result relating to the inclines.

Proposition 9 *If a node x has outward arcs a and b, then*

(i) $\Delta_x = \frac{\tau_a}{\tau_x} \cdot \Delta_a + \frac{\tau_b}{\tau_x} \cdot \Delta_b$, *and*

(ii) $|\Delta_a| + |\Delta_b| \leq \tau_x$.

Proof. For (i) we calculate

$$\begin{aligned}
\Delta_x &= \sum_{i \in \mathcal{L}_x} e_x(i)\delta_x(i) = \sum_{i \in \mathcal{L}_a} e_x(i)\delta_x(i) + \sum_{i \in \mathcal{L}_b} e_x(i)\delta_x(i) \\
&= \sum_{i \in \mathcal{L}_a} e_x(Q_a)e_a(i)\delta_x(i) + \sum_{i \in \mathcal{L}_b} e_x(Q_b)e_b(i)\delta_x(i) \\
&= e_x(Q_a)\ \Delta_a + e_x(Q_b)\ \Delta_b = \frac{\tau_a}{\tau_x} \cdot \Delta_a + \frac{\tau_b}{\tau_x} \cdot \Delta_b \text{ by (3).}
\end{aligned}$$

For (ii), we calculate

$$\begin{aligned}
|\Delta_a| + |\Delta_b| &\leq \sum_{\text{arcs } c \in Q_a} |e_a(Q_c)| \cdot |D_c| + \sum_{\text{arcs } c \in Q_b} |e_b(Q_c)| \cdot |D_c| \ \text{(by (5))} \\
&\leq \sum_{\text{arcs } c \in Q_x} |D_c| = \sum_{\text{arcs } c \in Q_x} |F_c - R_c| \leq \sum_{\text{arcs } c \in Q_x} |F_c| + |R_c| = \tau_x.
\end{aligned}$$

■

We now introduce the optimal Searcher strategy β, which produces depth-first searches through a branching process.

Definition 10 (Depth-first (DF) path) *A depth-first (DF) path in a tree is one that, whenever arriving at a node, always takes an unsearched outward arc, if available; otherwise it takes the unique reverse arc.*

We give an explicit definition of branching strategies, and define the branching strategy β that turns out to be optimal for the Searcher. A recursive definition of an optimal branching strategy was given in [4], and we show that these definitions are equivalent, so that β is optimal.

Definition 11 (Biased depth-first (BDF) strategy β) *At every branch node x with out arcs a and b, let q be a probability distribution over these arcs. We interpret the branching strategy q as follows:*

1. *When arriving at a branch node for the first time, choose an outward arc a with probability $q(a)$.*
2. *When arriving at a branch node the second time, choose the unique untraversed outward arc.*
3. *When arriving at a branch node the third time, choose the unique inward arc.*

*We call the particular branching strategy β defined below the **biased depth-first (BDF)** strategy:*

$$\beta(a) = \frac{1}{2} + \frac{1}{2\tau_x}(\Delta_a - \Delta_b). \tag{6}$$

(Since Proposition 4 (ii) shows that $|\Delta_a|+|\Delta_b| \leq \tau_x$, this is indeed a probability.)

Note that branching strategies produce as sample paths only DF paths. For the tree illustrated in Figure 4, we have $\Delta_a = (2/3)(4) + (1/3)(0) = 8/3$, and $\Delta_b = (1/8)(-1) + (1/8)(1) = 0$, so that $\beta(a) = 1/2 + (8/3)/64 = 13/24$. Note in particular that we computed the optimal initial branching (at the root) without working backwards (unlike our recursive approach in [4]).

2.3.2 A simple formula for the search value of a tree

We now use the recursive form of the search value of a tree found in [4] to derive a simple formula for the value. In [4], the following recursive equations were

defined for branch nodes x with outward arcs a and b and arcs c with forward nodes y.

$$v_x = \frac{\tau_a v_a + \tau_b v_b + \tau_a \tau_b}{\tau_x},\ v_c = F_c + v_y. \tag{7}$$

$$r(a) = \frac{\tau_b + v_a - v_b}{\tau_x} \tag{8}$$

Theorem 12 (Theorem 6 of [4]) *For the search game Γ on a rooted tree Q, O,*

1. *The EBD distribution is uniquely optimal for the Hider.*
2. *The branching strategy r defined recursively by (7) and (8) is optimal for the Searcher. On a binary tree, this branching strategy is the only one that is optimal for the Searcher.*
3. *The value V can be calculated recursively by (7). For a branch node or arc z, the value of the search game played on Q_z is equal to v_z.*

Our main theorem follows easily from this.

Theorem 13 *The search value of a rooted tree Q, O is half the sum of its tour time and its incline,*

$$V = \frac{1}{2}(\tau + \Delta). \tag{9}$$

The Equal Branch Density strategy e is uniquely optimal for the Hider and the Biased Depth-first strategy β is optimal for the Searcher.

Proof. We prove (9) by induction on the number of arcs of Q. If Q has only one arc a, then since it is optimal for the Hider to hide at the leaf node, the

value is trivially $F_a = \frac{1}{2}((F_a + R_a) + (F_a - R_a)) = \frac{1}{2}(\tau + \Delta)$. Suppose (9) is true for all trees with fewer arcs than Q. There are two cases: either O is a branch node or it has only one outward arc.

In the first case, suppose O has outward arcs a and b. Then applying the induction hypothesis to Q_a and Q_b and using Theorem 12,

$$\begin{aligned} V &= v_O = \frac{\tau_a v_a + \tau_b v_b + \tau_a \tau_b}{\tau_x} \text{ (by (7))} \\ &= \frac{\tau_a(\tau_a + \Delta_a) + \tau_b(\tau_b + \Delta_b) + 2\tau_a\tau_b}{2\tau_x} \\ &= \frac{1}{2}\left(\frac{(\tau_a + \tau_b)^2}{\tau_x} + \frac{\tau_a}{\tau_x}\Delta_a + \frac{\tau_b}{\tau_x}\Delta_b\right) \\ &= \frac{1}{2}(\tau + \Delta) \text{ (by Proposition 9 (i))} \end{aligned}$$

In the other case, if O has only one outward arc c with forward node y, then applying the induction hypothesis to Q_y and using Theorem 12,

$$\begin{aligned} V &= v_O = F_c + v_y \\ &= F_c + \frac{1}{2}(\tau_y + \Delta_y) \\ &= \frac{1}{2}((\tau_y + F_c + R_c) + \Sigma_{\text{leaf nodes } i} e(i)(\delta_y(i) + F_c - R_c)) \\ &= \frac{1}{2}(\tau + \Delta) \end{aligned}$$

The fact that the EBD is uniquely optimal for the Hider was already established in Theorem 12. The fact that the Biased Depth-first strategy β is optimal for the Searcher follows from Theorem 12 and (8), since the branching strategy r

is optimal, and for a branch node x with outward arcs a and b,

$$\begin{aligned} r(a) &= \frac{\tau_b + v_a - v_b}{\tau_x} \\ &= \frac{1}{\tau_x}\left(\tau_b + \frac{1}{2}\left(\tau_a + \Delta_a\right) - \frac{1}{2}\left(\tau_b + \Delta_b\right)\right) \text{ (by (9))} \\ &= \frac{1}{\tau_x}\left(\frac{1}{2}(\tau_a + \tau_b) + \frac{1}{2}(\Delta_a - \Delta_b)\right) \\ &= \frac{1}{2} + \frac{1}{\tau_x}\left(\Delta_a - \Delta_b\right) \\ &= \beta(a). \end{aligned}$$

■

2.3.3 Applications of the value formula to special trees

The first special case of our value formula (9) is Gal's classic result [21] for time-symmetric trees.

Corollary 14 *If Q is a time-symmetric tree, its search value is half its tour time, $V = \tau/2$, which is simply the total length μ of the tree.*

Proof. For time-symmetric trees, $\Delta = 0$. ■

We say a tree has *constant leaf height* if $\delta(i) = \gamma$ for all $i \in \mathcal{L}$, that is, all the leaf nodes have the same height. Gal's [21] result on the optimality of the Random Chinese Postman Tour remains true (taking $\gamma = 0$) for constant leaf height trees. A Chinese Postman Tour (CPT) is closed path of minimal tour time, and a Random Chinese Postman Tour (RCPT) is an equiprobable mixture of a Chinese Postman tour and its reverse tour.

Corollary 15 *Suppose Q has constant leaf height γ. Then*

1. $V = \frac{1}{2}(\tau + \gamma)$,

2. *An optimal strategy for the Searcher is the RCPT, that is, an equiprobable mixture of a CPT S_1 of Q, and its reverse tour S_2.*

Proof. Part 1 follows immediately from (9) and the definition of constant leaf height. For part 2, fix any leaf node i, and note that in the sum $T(S_1, i) + T(S_2, i)$, the arcs in the unique path $\mathcal{P}$ from O to i appear twice in their forward direction, and all other arcs appear once in each direction. Hence twice the capture time is given by

$$\begin{aligned}
& 2 \cdot T(RCPT, i) = T(S_1, i) + T(S_2, i) \\
= \;& 2 \sum_{a \in \mathcal{P}} F_a + \sum_{a \in \mathcal{P}^c} (F_a + R_a) \\
= \;& \sum_{a \in \mathcal{P}} F_a + \sum_{a \in \mathcal{P}} R_a + \sum_{a \in \mathcal{P}^c} (F_a + R_a) + \sum_{a \in \mathcal{P}} D_a \\
= \;& \sum_{a \in \mathcal{A}} (F_a + R_a) + \sum_{a \in \mathcal{P}} D_a, \\
= \;& \tau + \gamma, \text{ since } \sum_{a \in \mathcal{P}} D_a = \delta(i) = \gamma \text{ by constant leaf height.}
\end{aligned}$$

■

2.3.4 Application to the Kikuta game with search costs

We now consider the application of the theorem to the search game $K = K(Q, O)$ formulated by Kikuta [30] on a time-symmetric rooted tree Q, O. Kikuta's game is similar to Γ except that each node i is assigned a search cost $c_i \geq 0$, with $\sum c_i = C$. When encountering a node, the Searcher can either search it at cost (time loss) c_i or bypass it (to search it later) without incurring a cost. It has already been observed in [4] that $K(Q, O)$ is equivalent

to our game Γ on a time-asymmetric tree Q'. We obtain Q' by replacing the search costs with *search arcs* a_i between each node i of Q and a new leaf node i' of Q', with $F_{a_i} = c_i$ and $R_{a_i} = 0$, so that $D_{a_i} = c_i$. We present here an explicit formula for the value of Kikuta's game, as a corollary of our formula for the search value of a tree. Let τ be the tour time of the original network, not including the search costs of the nodes.

Corollary 16 *The value of Kikuta's game K on a rooted time-symmetric tree Q, O of total length μ and search costs c_i totalling to C is given by*

$$V = \mu + \frac{1}{2}\left(C + \sum_{nodes\ i\ of\ Q} e\left(i'\right) \cdot c_i\right). \tag{10}$$

where e is the EBD distribution on the associated Q'. If the costs at all n nodes of Q are equal to c, then $V = (1/2)(\tau + (n+1)c)$ and the Random Chinese Postman Tour (and searching every node when you come to it) is optimal.

Proof. Since $V = V(K(Q, O)) = V(Q', O)$, and Q' is a (time-asymmetric) tree with no additional search costs, we have that the value V of Kikuta's game is equal to $(1/2)(\tau' + \Delta')$, where τ' and Δ' are the tour time and the incline of Q'. Clearly

$$\tau' = \tau + \sum_{\text{nodes } i \text{ of } Q} (F_{a_i} + R_{a_i}) = \tau + C = 2\mu + C$$

and $\Delta' = \sum_{\text{nodes } i \text{ of } Q} e\left(i'\right) \cdot c_i$, establishing (10). If all the $c_i = c$, then any weighted average is c, so $\Delta' = c$ and $C = nc$. In this case Q' has constant leaf node height c, so the optimality of the RCPT follows from the second part of Corollary 16. ■

The case of equal search costs (but not the general case) can also be tackled within the time-symmetric tree theory by adding time-symmetric rays with travel times equal to $c/2$ at each node, observing that while this is not equivalent to the Kikuta problem, it always finds the Hider at time $c/2$ earlier. See Alpern and Gal [11]. This problem can also be attacked in the more difficult context of an arbitrary Searcher starting point - see Baston and Kikuta [19].

2.3.5 Application to Alpern's find-and-fetch game

Our methods can be similarly applied to the find-and-fetch game recently introduced by Alpern [5]. This search game $F = F(Q, O)$ is played on a rooted time-symmetric network on which the Searcher not only wishes to find a Hider but also wishes to return to the root O. This models common problems such as search-and-rescue and foraging problems in which an animal must find food and then return to its lair. As in Gal's model, the Searcher follows a unit speed path from O, but then upon reaching the Hider takes the shortest path back to O at speed ρ. The payoff is the total time to find the Hider and return to O. In the case of a bird being weighed down by food he is taking back to his nest we might have $\rho < 1$, whilst $\rho > 1$ might be more appropriate for the case of someone searching for a contact lens, where the return speed would be quicker.

Alpern finds that if Q is a tree, the optimal strategy for the Hider is still the EBD distribution in this game. However, the RCPT is no longer optimal for the Searcher. Instead, he randomises between all possible depth-first searches using a type of strategy called a branching strategy. Upon reaching a node for

the first time the Searcher chooses which outward branch to take according to a certain probability. Alpern proves the following.

Theorem 17 (Alpern) *The value V of the find-and-fetch game on a tree is*

$$V = \mu + D/\rho, \tag{11}$$

where $D = D(Q)$ is the mean distance from O to the leaf nodes of Q, weighted according to the EBD distribution.

We show how this can be deduced from Theorem 13. It is clearly optimal for the Hider to choose a leaf node x, and for any such choice of x at shortest distance $d(x, O)$ from O, the Searcher must travel for additional time $d(x, O)/\rho$ after finding the Hider. We therefore form a new network Q' from Q by adding an asymmetric arc from x to a new leaf node x^+ with forward travel time (from x to x^+) of $d(x, O)/\rho$ and backward travel time $-d(x, O)/\rho$. The variable speed game played on Q' is then equivalent to the find and fetch game played on Q: travelling to x^+ in Q' is equivalent to travelling to x in the original network and then back to O at speed ρ, and if the Hider is not at x the extra arc from x to x^+ makes no contribution to the search time. Hence the two models are equivalent.

The total tour time τ of Q' is equal to twice the length 2μ of Q, and in the Q' the leaf node x^+ has height $2d(x, O)/\rho$, so $\Delta = \Delta(Q')$ is the mean value of $2d(x, O)/\rho$, weighted with respect to the EBD distribution, which is equal to $2D(Q)/\rho$. Hence by (9), the value is

$$\begin{aligned} V &= 1/2(2\mu + 2D/\rho) \\ &= \mu + D/\rho, \end{aligned}$$

as in (11).

It must be noted that strictly speaking this approach is invalid, since it is prohibited by the fact that our definition of variable speed networks required the forward and backward travel times to be derived from a quasimetric, which must necessarily take non-negative values. However, the fact that this method produces a solution to the game that is consistent with that found in Alpern [5] indicates that the results in this chapter may hold without the quasimetric assumption. On the other hand, it is clear that if negative travel times are used without restriction then the players' optimal strategies may not be well defined. For example, for a tree consisting of two branches, one of which has a positive tour time and the other a negative, the EBD distribution is meaningless. Further work is required to establish criteria under which the results of this chapter hold for negative travel times.

2.4 Circle with concave travel times

The simplest non-tree network is the circle, represented as a single loop arc a, with its initial and terminal ends a^- and a^+ identified with the start node O. For simplicity, we call the forward direction clockwise. Here we consider the variable speed search game played on a circle. We restrict ourselves to cases in

which we can parameterise the loop a (directed from end a^- to a^+) by $0 \leq x \leq 1$ so that $d(a^+, x)$ is a decreasing linear function of x and $f(x) = d(a^-, x)$ is an increasing continuous function. In this case we say that the arc a has *constant velocities* if f is linear and has *concave travel times* if f is concave. Note that the fact that an arc has concave travel times is independent of the orientation choice for the arc. We define the *in-midpoint* m of a as the unique point for which $d(a^-, m) = d(a^+, m) = \rho$, where ρ is called the *in-radius* of a.

In the time-symmetric case there is a simple solution: the value is half the travel time along the loop, two optimal strategies for the Hider are hiding uniformly along the arc or hiding at the midpoint m, the optimal Searcher strategy is to tour the loop equiprobably in either direction. The general solution for arbitrary travel times is unknown. Even for simple travel times (as shown in the next section when the circle is viewed as two identical arcs, with uniform velocities, from O to m) the solution can be very complicated, requiring backtracking paths and mixtures over a continuum of pure strategies.

In this section we see when the solution for the time-symmetric circle has a natural generalisation to hiding at the in-midpoint of the circle and searching *with unequal probabilities* in the clockwise or anticlockwise directions. The travel time assumption needed for this simplification is concave travel times, as defined above.

Theorem 18 *Let C be the network consisting of a single loop a at the start node O, with concave travel times. Let ρ and m denote the in-radius and in-midpoint of a. Then*

1. $V \equiv V(C,O) = \rho$

2. *Hiding at m is optimal.*

3. *The mixed Searcher strategy $\bar{s}_p$ of going around loop a clockwise (forwards) with probability $p = 1/(1+\lambda)$ and anticlockwise with probability $1-p$, is optimal for any $\lambda \in \left[\frac{f'_+(m)}{R_a}, \frac{f'_-(m)}{R_a}\right]$, where f'_+ and f'_- are respectively the right and left derivatives of f.*

4. *In particular, if loop a has constant velocities, then $m = R_a/(F_a + R_a)$, $V = \rho = F_a m$ and the unique optimal value of p is m.*

Proof. If the Hider chooses the in-midpoint m, then clearly the Searcher cannot find him in time greater than ρ, so $V \geq \rho$.

Since the forward time function $f(x) = d_a(a^-, x)$ is concave, it has left and right derivatives $f'_-(x_0)$ and $f'_+(x_0)$ at any x_0 in the interior of a such satisfying

$$f'_-(x_0) = \inf_{x<x_0} \frac{f(x_o) - f(x)}{x_o - x} \geq f'_+(x_0) = \sup_{x>x_0} \frac{f(x) - f(x_0)}{x - x_0}. \tag{12}$$

Suppose that the Searcher adopts the strategy $\bar{s}_p$, $p = 1/(1+\lambda)$, for some $\lambda \in \left[\frac{f_+(m)}{R_a}, \frac{f_-(m)}{R_a}\right]$. If the Hider is anticlockwise of m, that is, at some point

$H = x \leq m$, then

$$\begin{aligned}
T(\bar{s}_p, x) - \rho &= p\ f(x) + (1-p)\ g(x) - \rho \\
&= p\ f(x) + (1-p)\ R_a(1-x) - f(m) \\
&= \frac{f(x)}{1+\lambda} + \frac{\lambda R_a(1-x)}{1+\lambda} - \frac{f(m)(1+\lambda)}{1+\lambda} \\
&= \frac{f(x) + \lambda R_a(1-x) - f(m) - \lambda R_a(1-m)}{1+\lambda} \\
&= \frac{f(x) - f(m) + \lambda R_a(m-x)}{1+\lambda} \\
&\leq \frac{f(x) - f(m) + f'_-(m)(m-x)}{1+\lambda} \text{ (by definition of } \lambda) \\
&\leq \frac{m-x}{1+\lambda}\left(f'_-(x) - \frac{f(m)-f(x)}{m-x}\right) \text{ (by 12)} \\
&\leq 0.
\end{aligned}$$

By an analogous argument, if the Hider hides at some $x \geq m$ the Searcher will find him in expected time $\leq \rho$. Hence $V \leq \rho$, so that $V = \rho$. Points 2 and 3 follow easily.

If a has constant velocities, then the time taken for the Searcher to travel from either a^- or a^+ to $R_a/(F_a + R_a)$ is $F_a R_a/(F_a + R_a)$, so this must be the in-midpoint, and point 4 follows. ■

2.5 Solution of two-arc networks

In the previous section we showed that the circle network has a simple solution if it has concave travel times. In this section we show that the solution can become quite complicated if concavity is lost, even for a very simple class of circle networks $U(b)$ depicted in Figure 5, consisting of two identical constant velocity arcs from O to m (so labelled because it is the in-midpoint of $U(b)$ if

we view it as a single loop). That is, the two arcs have identical forward and reverse travel times F and R. Without loss of generality we can take the forward travel time F to be 1, and for notational simplicity denote $R = b$. Of course if $b \leq 1$ then the network can be viewed as a single arc (loop) with concave travel times, so in this case $V(U(b)) = \rho = 1$. Optimally, the Hider goes to m and the Searcher adopts strategy $\bar{s}_p$ with $p = 1/2$ (in fact any $p \in \left[\frac{b}{1+b}, \frac{1}{1+b}\right]$). As we shall see, the case $b > 1$ (where the network goes 'downhill' from the start O) has a rather complicated solution, reminiscent of the solution of Gal's search game on three (time-symmetric) arcs given by Pavlovic [40]. We view each arc from O to m as having unit length, parameterised by x going from 0 to 1, with forward velocity 1 and reverse velocity $1/b$.

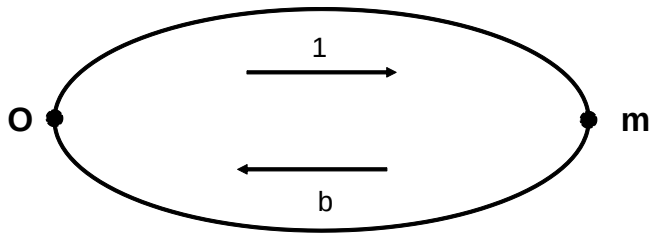

Figure 5: The network $U(b)$.

Theorem 19 *Consider the network $U(b)$ consisting of two identical arcs from O to m with forward travel time 1 and reverse time $b \geq 1$. Then*

1. *The search value is $V = V(U(b), O) = 1 + \frac{1}{2}(b-1)\ln 2$.*

2. *An optimal strategy for the Hider is to pick x according to the density function $4e^{-2x}$ on the interval $[0, \ln 2/2]$. Then he hides equiprobably on*

the two points at forward distance x from O.

3. *An optimal strategy for the Searcher begins by choosing a number y from $[0, \ln 2/2]$ according to the density function $2e^{2y}$. With probability $p = (b+3)/(2b+2)$ he tours the circle equiprobably in either direction. With probability $1-p$ he goes around in an equiprobable direction until he is at forward distance y from O; then reverses direction and goes around the circle until he has reached all points.*

Proof. Suppose the Hider follows the strategy described in the statement of the proof. Then the expected discovery time if the Searcher goes all the way around the circle is

$$\frac{1}{2}\int_0^{\frac{1}{2}\ln 2} 2x 4e^{-2x}dx + \frac{1}{2}\int_0^{\frac{1}{2}\ln 2}(1+b(1-2x))4e^{-2x}dx = 1+\frac{1}{2}(b-1)\ln 2.$$

If the Searcher backtracks at some point $y \le \frac{1}{2}\ln 2$ then the expected discovery time is

$$\begin{aligned} & \frac{1}{2}\left(\int_0^{\frac{1}{2}\ln 2}(2(1+b)y+2x).4e^{-2x}dx\right) + \\ & \frac{1}{2}\left(\int_0^{y} 2x\ 4e^{-2x}dx + \int_y^{\frac{1}{2}\ln 2}(2(1+b)y+1+b(1-2x))4e^{-2x}dx\right) \\ = \quad & 1+\frac{1}{2}(b-1)\ln 2. \end{aligned}$$

If the Searcher backtracks a some point $y > \frac{1}{2}\ln 2$, the expected search time will be greater than if he backtracks at $y = \frac{1}{2}\ln 2$.

Suppose the Searcher follows the strategy described in the statement of the proof. Then, if the Hider is at a distance $x > \frac{1}{2}\ln 2$ from O, the expected search

time is

$$\begin{aligned}
& \frac{b+3}{2b+2}\left(\frac{1}{2}2x+\frac{1}{2}(1+b(1-2x))\right)+ \\
& \frac{b-1}{2b+2}\left(\frac{1}{2}\int_0^{\frac{1}{2}\ln 2}(2(1+b)y+1+b(1-2x))2e^{2y}dy+\frac{1}{2}\int_0^{\frac{1}{2}\ln 2}(2(1+b)y+2x)2e^{2y}dy\right) \\
= \quad & 1+(b-1)\ln 2-(b-1)x \\
\leq \quad & 1+(b-1)\ln 2-(b-1)\frac{1}{2}\ln 2 \\
= \quad & 1+\frac{1}{2}(b-1)\ln 2
\end{aligned}$$

If the Hider is at a distance $x \leq \frac{1}{2}\ln 2$ then the expected search time is

$$\frac{b+3}{2b+2}\left(\frac{1}{2}2x+\frac{1}{2}(1+b(1-2x))\right)+\left(\frac{b-1}{2b+2}\right)\cdot$$
$$\left(\frac{1}{2}\left(\int_0^x(2(1+b)y+1+b(1-2x)2e^{2y}dy+\int_x^{\frac{1}{2}\ln 2}2x2e^{2y}dy+\frac{1}{2}\int_0^{\frac{1}{2}\ln 2}(2(1+b)y+2x)2e^{2y}dy\right)\right)$$
$$=1+\frac{1}{2}(b-1)\ln 2$$

Hence the value is $1+\frac{1}{2}(b-1)\ln 2$. ■

The method for discovering these strategies is as follows. Suppose $b=2$. For the Hider distribution, we would like to find a density function $h(x)$ where $x \in [0,z]$ such that the expected search time is the same whether the Searcher backtracks after travelling some distance $y \leq z$, or goes all the way around the circle (which is effectively backtracking after travelling distance 0). That is,

$$\frac{1}{2}6y+\int_0^z 2xh(x)dx+\int_0^y 2xh(x)dx+\int_y^z(6y+3-4x)h(x)dx=C.$$

where C is a constant, independent of y. Simplifying this and differentiating with respect to y we obtain

$$4-h(y)-2\int_0^y h(x)dx=0.$$

Putting $H(y) = \int_0^y h(x)dx$ gives the differential equation

$$\frac{dH}{dy} = 4 - 2H.$$

Solving this gives $h(x) = 4e^{-2x}$, and using $\int_0^z h(x)dy = 1$ we obtain $z = \frac{1}{2}\ln 2$.

A similar method can be used to find the Searcher strategy.

3 Expanding Search on a Tree

In Chapter 2 we generalised the usual model of search in which a Searcher moves on a network at unit speed, by considering a model in which the speed of the Searcher depends on his location and direction of travel. A Searcher strategy was a path in the network starting at the root, so we will refer to this model of search as *pathwise search*. This chapter discusses a new search paradigm, which we call *expanding search*, where the Searcher may restart the search at any time from any previously reached point. Such searches are routinely carried out in many contexts, sometimes by a team of agents. Under such searches, the portion $S(t)$ of the search region that has been covered by time t expands in a continuous way until the first time T (the search time) that it contains the target of the search. Expanding search was introduced in Alpern and Lidbetter [14], and this chapter is based upon parts of that paper. The work of Alpern and Howard [12] considered a related problem of a single Searcher who alternates between looking for a single Hider at two locations, which can now be seen as a special case (on a star network) of expanding search. There is also a connection to the two-Searcher *coordinated search* problem of Thomas [51] and Reyniers [45], [46] described in the next section. Megiddo et al [37] considered minimising the number of searchers, rather than the search time. Other interpretations of expanding search, such as the optimal mining of coal, will be described later.

An immobile hidden object, target, terrorist, or simply Hider, is located at an unknown point H of a known network Q, as usual. The network is endowed

with an arc length measure λ (linear Lebesgue measure) so that $\lambda(a)$ denotes the length of an arc a and $d(x,y)$ is the metric given by the length of the shortest path between points x and y in Q (as in Chapter 2, except here symmetry is satisfied so d is indeed a metric). We assume there are a finite number of arcs, each of finite length, so that Q is compact. The distribution (probability measure on Q) ν of H may be known or unknown. If it is known, we consider the *Bayesian Search Problem* of minimising the expected search, or capture, time. If it is unknown, we consider the zero-sum *expanding search game* where the Hider chooses H to maximise T. Starting at a given point of Q, called the root, a search team consisting of successively dividing groups spreads out over the network until the first (capture) time T that one of its members encounters the Hider. The agents are constrained to move with combined speeds of 1. This means that $\lambda(S(t)) = t$, where $\lambda(S(t))$ is the measure of the portion of Q covered by time t. When the Hider distribution ν is given on a tree, we solve the Bayesian Problem by an algorithm that determines the expanding search $S(t)$ that minimises the expected value of T. We also solve the search game when Q is a tree and later in Chapters 4 and 5 we will give solutions for an extension of the search game in which there are several hidden objects various different classes of networks.

In the important case where the Hider distribution ν is concentrated on the nodes of Q, an expanding search is simply a sequence of arcs $S = (a_1, a_2, \ldots, a_N)$, oriented so that the tail of a_1 is the root O of Q, the tail of every other arc a_i is the tip of a previous arc a_j, $j < i$, and the N non-root nodes of Q coincide with

the N tips of the arcs. If the Hider location H is the tip of arc a_k, the capture time $T = T(S, H)$ is given by

$$T(S, H) = \lambda(a_1) + \cdots + \lambda(a_k). \tag{13}$$

We show (Theorem 21) that when Q is a rooted tree and K is a subtree of maximum search density for some known Hider distribution ν, there is an optimal expanding search $\bar{S}(t)$ that begins by exhaustively searching K, that is, with $\bar{S}(t) = K$ at time $t = \lambda(K)$. The subtree K of maximum density can be found by considering the density of all the subtrees (of which there are a finite number) and picking the one with the largest density. While this optimality condition does not also hold for arbitrary networks, we can however use it to solve the Bayesian Search Problem on any network by considering its spanning trees.

Using the solution of the search game for variable speed networks, we are able to give a complete solution of the expanding search game for any rooted tree Q. We show (Theorem 24) that value of this game is given by

$$V = \frac{\mu + D(Q)}{2},$$

where $\mu = \lambda(Q)$ is the total length of the tree Q and $D(Q)$ is the mean distance from the root node O of Q to its leaf nodes, with respect to the Equal Branch Density (EBD) distribution e on the leaf nodes. We determine the optimal Searcher mixed strategy as a branching function which specifies the probability that each branch at a node should be searched first when reaching that node, regardless of how the search has proceeded up to that point.

To illustrate these ideas, consider the tree Q with root O depicted in Figure 6. We will use this network as an example at several points throughout this chapter, not always assigning the same lengths to the arcs.

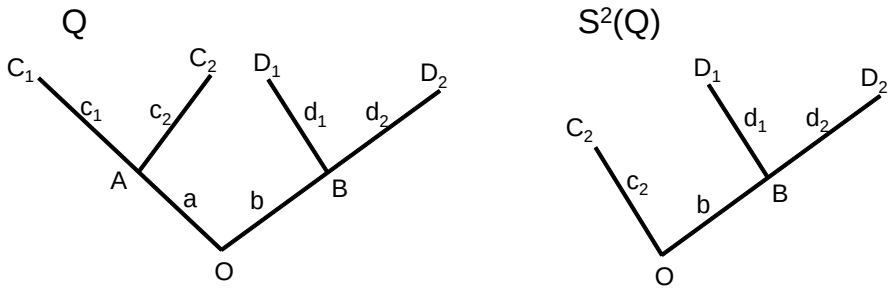

Figure 6: A rooted tree Q and its contraction $S^2(Q)$.

An example of an expanding search on Q is $S = (b, d_1, a, d_2, c_2, c_1)$. Consider the network Q_0 given by the following choice of arc lengths that is symmetric in the c_i and the d_i:

$$\lambda(a) = 1, \lambda(b) = 2, \lambda(c_1) = \lambda(c_2) = 1, \lambda(d_1) = \lambda(d_2) = 5 \tag{14}$$

For the search S, the time taken to reach, say C_2 is given by $T(S, C_2) = 2 + 5 + 1 + 5 + 1 = 14$, using the notation T for the search time introduced in (13). Note that the search S arrives at nodes A, B, C_1, C_2, D_1 and D_2 in respective times $8, 2, 15, 7, 13$ and 14. Hence if, for example, the Hider distribution ν takes the value $\frac{2}{7}$ on C_1 and $\frac{1}{7}$ on each of the other non-root nodes, then the expected search time is $T(S, \nu) = \frac{1}{7} \cdot 8 + \frac{1}{7} \cdot 2 + \frac{2}{7} \cdot 15 + \frac{1}{7} \cdot 7 + \frac{1}{7} \cdot 13 + \frac{1}{7} \cdot 14 = 7\frac{4}{7}$, using the notation $T(S, \nu)$ formally defined later in (18). In the case of this Hider distribution, there is a unique rooted subtree M of maximum density, $M = ac_1$,

which has density $(\frac{1}{7} + \frac{2}{7})/2 = \frac{3}{14}$. As we will show later in Theorem 21, this indicates that any optimal search must begin by searching M. After searching a and c_1 it is clear that an optimal search must continue by optimally searching the tree depicted on the right of Figure 6. This tree, which is obtained by contracting the first two arcs of Q traversed by this particular search S, shall be referred to as $S^2(Q)$ in Section 3.2. We can therefore apply Theorem 21 again by seeking the rooted subtree of maximum density in this new network. This is simply c_2, which has density $(\frac{1}{7})/1 = \frac{1}{7}$. The only arc available to search next is b, after which by symmetry d_1 and d_2 can be searched in either order. Hence an optimal search of Q_0 is $S_{opt} = (a, c_1, c_2, b, d_1, d_2)$, which searches the nodes a, b, c_1, c_2, d_1, and d_2 in respective times $1, 5, 2, 3, 10$, and 15, and hence has expected search time $T(S_{opt}, \nu) = \frac{1}{7} \cdot 1 + \frac{1}{7} \cdot 5 + \frac{2}{7} \cdot 2 + \frac{1}{7} \cdot 3 + \frac{1}{7} \cdot 10 + \frac{1}{7} \cdot 15 = 5\frac{1}{7}$.

Turning now to the *game* played on Q_0, we first note that any optimal Hider strategy must be restricted to the leaf nodes, C_i and D_i of Q_0, since all other points are dominated. Hence each player has a finite set of undominated pure strategies, and the game can be reduced to a matrix game. We note further that because of the symmetry in the network between the C_i and the D_i, it is clear that in the Hider's optimal mixed strategy he must choose equiprobably between the two nodes in each pair C_i and D_i, and in the Searcher's optimal strategy he must choose equiprobably which order to search the arcs c_1 and c_2 and the arcs d_1 and d_2. Hence we can simplify the matrix game by taking averages: for example in the matrix below, the entry for $(H, S) = (C, (b, d, a, c, c, d))$ corresponds to the average time taken to reach C_1 and C_2 by all four Searcher strategies

$(b, d_1, a, c_1, c_2, d_2)$, $(b, d_1, a, c_2, c_1, d_2)$, $(b, d_2, a, c_1, c_2, d_1)$, and $(b, d_2, a, c_2, c_1, d_1)$, that is $\frac{1}{2}((2+5+1+1)+(2+5+1+1+1)) = 9.5$.

	accbdd	*acbdcd*	*acbddc*	*bdaccd*	*bdacdc*	*bddacc*
C	2.5	6	8.5	9.5	12	14.5
D	12.5	12	11.5	11	10.5	9.5

(15)

Solving this matrix game numerically, we find that the Hider's optimal mixed strategy is to choose C with probability $\frac{1}{5}$ and D with probability $\frac{4}{5}$: that is C_1 and C_2 each with probability $\frac{1}{10}$ and D_1 and D_2 each with probability $\frac{2}{5}$. This is an example of the EBD distribution, as defined in Definition 3. Notice that the branch $\{a, c_1, c_2\}$ has density $(\frac{1}{10}+\frac{1}{10})/3 = \frac{1}{15}$, and the branch $\{b, d_1, d_2\}$ has equal density $(\frac{2}{5}+\frac{2}{5})/12 = \frac{1}{15}$. The Searcher's optimal strategy is to use the search (a, c, c, b, d, d) with probability $\frac{1}{3}$ and (b, d, d, a, c, c) with probability $\frac{2}{3}$. The game has value $V(Q_0) = 10.5$. As we shall see later in Theorem 21, this value is equal to half the sum of the total measure of the network, $\mu = 1+2+1+1+5+5 = 15$ and the quantity $D = D(Q_0) = \frac{1}{10}\cdot 2 + \frac{1}{10}\cdot 2 + \frac{2}{5}\cdot 7 + \frac{2}{5}\cdot 7 = 6$, the mean distance of the root node to the leaf nodes with respect to the EBD distribution. That is, $V(Q_0) = \frac{1}{2}(\mu + D) = \frac{1}{2}(15+6) = 10.5$.

3.1 Interpretation and applications of expanding search

The interpretation of an expanding search strategy in terms of a team of path-wise search agents is as follows. In the case where H is a node of Q, assume a group of m search agents starts at the root node. Then a large enough (for

later branching) subgroup takes initial arc a_1, while the rest remain at the root. Whenever a new arc a_k is chosen, some Searchers move along it, while some stay at its tail. An interesting problem which involves this group interpretation is to determine the number of agents that are required; either for a particular search strategy or for an optimal one. For trees, clearly the number of leaf nodes is a sufficient number of searchers. For example, the expanding search $(a, c_1, b, d_1, c_2, d_2)$ mentioned in the matrix (15) can be carried out by $m = 4$ agents, each adopting a pathwise search which includes waiting, as indicated by the left table in (16), where for example the fourth search agent follows the pathwise search a, w, w, w, c_2 (where w indicates waiting time). If only $m = 3$ agents are available, at least one of these must go backwards on an arc (indicated by the * in c_1^*) as in the table on the right.

a	c_1	b	d_1	c_2	d
$1, 4$	1	$2, 3$	2	4	3

a	c_1	b	d_1	c_1^*	c_2	d_2
1	1	$2, 3$	2	1	1	3

(16)

If a bound were put on the number of searchers, the expected time would be decreasing in the bound. When the bound is 1 (single agent), we get the usual pathwise search value, where a single Searcher moves at unit speed along the network. For sufficiently large bounds, we get the value obtained here for expanding search. For example, if the network is a line, with the root in the interior, then $m = 2$ searchers are enough for expanding search, whereas a single Searcher is faced with the well known linear search problem. A variation on the two-agent problem on the line, known as *coordinated search*, has been studied by Thomas [51] and Reyniers [45], [46], under the interpretation of 'how

to find your children when they are lost'. Here the quantity to be minimised is the time when the two agents (parents) are back at the root (their car) after one of them has found the Hider (child). If the Hider is mobile, the question of how many searchers are needed to guarantee finding him is considered in the classic paper of Megiddo et al [37].

Another interpretation, involving a single 'Searcher' (actually researcher) is that of tackling, say a mathematical problem for which one can map out in advance which facts need to be checked (calculated) before going onto the next step, perhaps seeking a proof or counterexample to a finite conjecture. Some steps of this process must be completed before other steps should be carried out, perhaps first checking the 'obvious' counterexample or proof. We assume here that after one branch of the research tree ends in failure, the researcher can go without time loss back to a previous method of attack. In some contexts the researcher here could be replaced by a computer program, so that we are in fact seeking a program which minimises the expected time to resolve some question.

The Bayesian problem for a tree, with a known distribution ν, has an interpretation in terms of optimal mining of coal. Suppose that by seismic analysis the density of coal along the tree network of its veins is known. We assume that time or effort involved in moving mining equipment (or miners) along the dug out regions (corresponding to what we called $S(t)$) of the mine is negligible with respect to the digging effort. So expanding search consists of digging in an area for a while, then moving the effort (miners or digging machines) to another

area, starting from the last bit dug. The probability density function of the capture time T when ν is the Hider distribution is the same as the extraction rate of coal at time T. So an optimal expanding search strategy corresponds to an extraction policy which minimises the mean time that the coal is ready for sale. This would make sense in an environment where the discount rate, used in calculating the net present value of coal mined over time, is linear. This is a reasonable approximation in certain economic conditions. Our optimality condition is to mine veins leading to the highest coal density, a semi-greedy policy. If there are enough excavators, it would be possible to always leave one at the last place mined along any vein. Our restriction on the rate of increase of the covered portion might relate to the number of miners or a constraint on the maximum electrical power available.

3.2 Known Hider distribution on nodes

We begin our analysis of expanding search in the simplest case where the Hider distribution ν is concentrated on the nodes of a network Q. In this case we will consider *expanding arc sequences*, sequences of arcs ordered and oriented so that the tail of each arc is the tip of a previous one. In this section an expanding search S, sometimes referred to as just a *search,* is simply an expanding arc sequence $(a_1, \ldots, a_N)$ where the tail of a_1 is the root node and tips of the arcs are the N non-root nodes of Q. We will refer to a search which is an expanding arc sequence as a *combinatorial search.* If S is an expanding search of Q and $k = 0, \ldots, N$, we denote by $S_k = S_k(Q)$ the rooted subnetwork (it is a tree)

of Q formed by the first k arcs searched by S. In particular, every search S of Q is associated with a spanning tree S_N. Note that in the tree S_k, the *ordering* of the arcs in S has been lost. We also let $S^k = S^k(Q)$ be the rooted network formed by shrinking S_k to O in Q. In Section 3.1 we saw $S^2(Q)$ depicted in Figure 6. After the Searcher has chosen the first k arcs to search, the problem that remains is how to search the remaining network S^k, so using dynamic programming we observe that an optimal expanding search S must be optimal for all of the subproblems (subnetworks) S^k. Given any expanding search $S = (a_1, \ldots, a_N)$ and any Hider location at the tip A_k of arc a_k, the search time $T = T(S, H)$ is given, as in (13), by

$$T(S, H) = \lambda(a_1) + \cdots + \lambda(a_k), \tag{17}$$

and the expected time for S to find a Hider hidden according to distribution (measure) ν is denoted by

$$T(S, \nu) = \sum_{k=1}^{N} \nu(A_k) \cdot T(S, A_k). \tag{18}$$

For a given Hider distribution ν, we say that $\bar{S}$ is optimal (against ν) if it minimises the expected search time $T(S, \nu)$ over all expanding searches S. (In the context of this section, there are only finitely many expanding searches, so the existence of optimal ones is not in question.)

The Search Density Lemma (Lemma 6) will be useful in our analysis, and we make an additional observation here about densities. For disjoint subsets A and B, the density of $A \cup B$ is a weighted average of the densities of A and B,

that is

$$\begin{aligned}\rho(A\cup B) &= \frac{\nu(A)+\nu(B)}{\lambda(A)+\lambda(B)} = \frac{\lambda(A)}{\lambda(A)+\lambda(B)}\frac{\nu(A)}{\lambda(A)} + \frac{\lambda(B)}{\lambda(A)+\lambda(B)}\frac{\nu(B)}{\lambda(B)} \\ &= \tfrac{\lambda(A)}{\lambda(A)+\lambda(B)}\ \rho(A) + \tfrac{\lambda(B)}{\lambda(A)+\lambda(B)}\ \rho(B)\,.\end{aligned}$$

Consequently if $\rho(B) \le \rho(A)$ then

$$\rho(B) \le \rho(A\cup B) \le \rho(A)\,, \text{with strict inequalities if and only if } \rho(B) < \rho(A)\,. \tag{19}$$

3.3 Known Hider distribution on nodes of a tree

We fix a Hider distribution ν and consider the densities of all subtrees of Q rooted at O. We will be particularly concerned with those subtrees which have maximum density $r = r(Q)$. Generically, there will be a unique subtree of maximum density r, and the main result of this section is that any optimal search must begin with the arcs of this subtree (in some order). The complicating factor is that it is possible there are multiple subtrees of maximum density, which is why Theorem 21 has a more complicated statement.

The set of subtrees $\mathcal{M}$ of maximum density r in Q may be ordered under set inclusion, in which case we call its minimal elements (those without any proper subtree of density r) *min-max subtrees.* If we include the empty subtree of no arcs, ϕ in $\mathcal{M}$ then it follows that $\mathcal{M}$ is closed under intersection and union. To see this, first note that if $A, B \in \mathcal{M}$ are disjoint (by which we mean they have no arcs in common) or if one contains the other, the result is trivially true. If $A - B$ and $B - A$ are both non-empty, then $A \cap B$ must have density r: it

certainly has density no greater than r by definition of r, and if it had density strictly less than r then by (19) we would have $\rho(A-B) > r$. But this cannot be possible, since by (19) we would then have $\rho(A \cup B) = \rho((A-B) \cup B) > r$, a contradiction. It follows that $\rho(A-B) = r$ and $\rho(B-A) = r$ so $\rho(A \cup B) = r$. We call the unique maximal element of $\mathcal{M}$ (the union of all its elements) the *max-max subtree,* denoted by $M = M(Q, \nu)$. Generically, there is a unique subtree of maximum density (namely M). But in general M may include more than one root arc, in which case all the corresponding branches of M have a unique min-max subtree, and these are all the min-max subtrees (one for each arc of M at the root of Q).

Lemma 20 *Let ν be a measure on the nodes of a tree Q.*

(i) *Distinct min-max subtrees are disjoint.*

(ii) *Every branch of a maximum density subtree A has density r.*

(iii) *Every branch of $M = M(Q)$ contains a unique min-max subtree.*

(iv) *Every min-max subtree A of Q has only one branch.*

Proof. To prove (i), suppose $A, B \in \mathcal{M}$ are distinct min-max subtrees with intersection I, which must belong to $\mathcal{M}$, since it is closed under intersection. By the minimality of A and B, I cannot be a proper subset of either subtree, and cannot be equal to either A or B. Hence I must be the empty subtree ϕ, so that A and B are disjoint. Now suppose A is a maximum density subtree, and by definition of r every branch of A must have density at most r. (19) implies

that the density of each branch must be equal to r, which proves (ii). Part (iii) follows immediately, since every branch of M has density r and must therefore contain a minimum subtree of density r. If A is a min-max subtree, then by (ii) every branch of A must have density r, and by the minimality of A must be equal to A, giving (iv). ■

We say that a search S *begins with* the subtree B if some initial block of arcs of S is some ordering of the arcs of B. Generically, there will be a unique subtree of maximum density, and every optimal expanding search must begin with its arcs. In the more general case, the result is as follows.

Theorem 21 *Let ν be any Hider distribution on the nodes of a rooted tree Q. Then*

(i) *Every optimal expanding search of Q begins with some min-max subtree of Q.*

(ii) *Every optimal expanding search of Q begins with the max-max subtree $M = M(Q)$;*

(iii) *For any subtree A of maximum density r, there is an optimal expanding search of Q that begins with the arcs of A.*

Proof. (i) We prove the first part of the theorem by induction on the number of arcs: it is trivially true for a network consisting of one arc. So assume that (i) is true for all trees with at most $N-1$ arcs, and let $S = (a_1, a_2, ..., a_N)$ be an optimal search of the tree Q having N arcs. Let k be the smallest integer for which tree S_k (consisting of the first k arcs of S) contains some min-max

subtree A. Let a_j be the first arc of S contained in A, so that it must be a root arc of A.

We first assume that $j > 1$, that S begins with arcs not in A. After searching $S_{j-1} = (a_1, \ldots, a_{j-1})$, the induction hypothesis says S must continue by searching some min-max subtree B of the remaining tree S^{j-1}. Since B contains the root arc a_j of Q, it must also be a min-max subtree of Q. But as both A and B are min-max subtrees of Q starting with the same root arc a_j, Lemma 20 (i) says that $B = A$. Hence S can be written as $S = S_{j-1}, A, X$. If we had $\rho(S_{j-1}) \geq r$, then S_{j-1} would be a tree of maximum density, contradicting the minimality of k, so we must have that $\rho(S_{j-1}) < r = \rho(A)$. But this would contradict the optimality of the expanding search S, as Lemma 6 shows that the alternative expanding search $S' = A, S_{j-1}, X$ has a strictly lower expected search time. It follows that our assumption $j > 1$ is false, and this case is impossible.

So the search S begins with a sequence of arcs $S_l = (a_1, \ldots, a_l)$ belonging to A. Suppose l is the largest integer for which $S_l \subset A$. If $S_l = A$, the proof is complete. If not, by the induction hypothesis applied to the tree S^l, S must continue by searching some min-max subtree B of S^l. Note that B must be disjoint from A, by Lemma 20 (iv). By the maximality of $r(S^l)$, we must have $\rho(B) = r(S^l) \geq \rho(A - S_l)$, and by the minimality of A, $\rho(A) > \rho(S_l)$ so that $\rho(A - S_l) > \rho(S_l)$, by 19. Hence $\rho(B) > \rho(A)$, and $\rho(B \cup A) > \rho(A) = r$, contradicting the maximality of r.

(ii) By part (i), the first arc of S is certainly in M since S begins by searching a min-max subtree, which must be contained in M. Suppose k is the largest

integer for which $S_k \subset M$. If $S_k = M$, the proof is complete, so suppose not. Note that by definition of r, we must have $r(S^k) \geq \rho(M - S_k)$, since $M - S_k$ is a rooted subtree of S^k. Also, since $\rho(M) = r$ and $\rho(S_k) \leq r$, by 19, $\rho(M - S_k) \geq r$. Putting these together gives $r(S^k) \geq r$. By part (i), after searching S_k, S must search a min-max subtree, A of S^k with $\rho(A) = r(S^k) \geq r$. The tree A must begin with an arc not in M, and hence be disjoint from M by Lemma 20 (iv), so that $\rho(A \cup M) \geq r$, by (19), contradicting the maximality of M.

(iii) Let ε_n be a positive sequence tending to 0. For each n, define the measure ν_n on the nodes of Q such that $v_n(x) = (1 + \varepsilon_n)\ \nu(x)$ for all nodes x in M and $v_n(y) = (1 - \omega_n)\ \nu(y)$ for all nodes y not in M, where ω_n is chosen to make ν_n a probability measure. For each n, let S_n be an optimal expanding search on Q with the Hider distribution ν_n. Since there are only a finite number of expanding searches, one of them, call it S', must appear infinitely often in the sequence $(S_n)_{n=1}^{\infty}$, and hence also be optimal for the limiting distribution ν. But for any of the measures ν_n, the tree A is the unique, and hence maximal, subtree of maximum density, so by part (ii), any optimal expanding search must start with A. Hence in particular the optimal expanding search S' starts with A. ■

We now have a simple algorithm for constructing an optimal expanding search (indeed all such searches) on a tree when the Hider has a known distribution on the nodes. It is sufficient, after the first k arcs of an optimal expanding search S have been chosen, to determine which initial arcs of the remaining

tree S^k lead to optimal expanding searches. The answer is that any arc of the max-max subtree of S^k can be chosen, or equivalently the unique root arc of any min-max subtree of S^k. Summarising, we have the following.

Corollary 22 *Let ν be a Hider distribution on the nodes of a tree Q and let M be the max-max subtree of Q. Then any arc a at the root of Q which belongs to M can be taken as the first arc a_1 of an optimal expanding search. Repeating this for the subtree Q_1 obtained by deleting a_1 from Q and identifying its tip with the root, gives all possibilities for a_2, and so on.*

We saw in Section 1 how this algorithm can be used to find the optimal search of the network Q_0 of Figure 6 with arc lengths given by (14). We now illustrate how the algorithm can be applied to a network whose maximum density subtree is not unique. Consider the network Q depicted in Figure 6, but this time suppose all the arcs have length 1 and the nodes A, B, C_1, C_2, D_1, and D_2 have respective measures $0.1, 0.2, 0.3, 0.1, 0.2$, and 0.1. The first step of the algorithm is to identify the max-max subtree, $M(Q)$, which can easily be found to be abc_1d_1. The next step is to choose one of the root arcs of $M(Q)$ as the first arc of the search. Depending on whether we choose a or b, we obtain a different new network, Q_1 in which we have shrunk a to the root, or Q_1', in which we have shrunk b to the root, as depicted in Figure 7.

Suppose we choose to begin with b. Then we find the max-max subtree of Q_1' is ac_1d_1, so that we can choose either a or d_1. Suppose we choose d_1 and shrink this arc to the root. The max-max subtree of the resulting network is ac_1, and we continue by searching these two arcs next. Finally, shrinking these

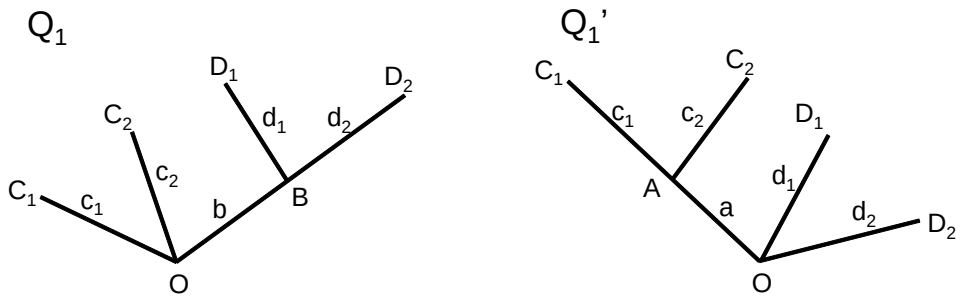

Figure 7: The contracted networks Q_1 and Q_1'.

arcs to the root we are left with the network consisting of only the arcs c_2 and d_2. These arcs have equal density so the max-max subtree is the whole network, and c_2 and d_2 can be searched in either order. The result is an optimal search $(b, d_1, a, c_1, c_2, d_2)$ with expected search time equal to $0.2(1) + 0.2(2) + 0.1(3) + 0.3(4) + 0.1(5) + 0.1(6) = 3.2$. Had we made different choices when applying the algorithm we may have produced any one of the alternative optimal strategies: $(b, d_1, a, c_1, d_2, c_2), (b, a, c_1, d_1, c_2, d_2), (b, a, c_1, d_1, d_2, c_2), (a, c_1, b, d_1, c_2, d_2)$, or $(a, c_1, b, d_1, d_2, c_2)$.

It is worth noting that the principle of searching the subtree of highest density first does not hold in the analogous pathwise search problem, where a Searcher must take a continuous path in a rooted network to minimise the time taken to find a Hider located on the network according to some distribution. For example, consider the network consisting of the closed interval $[-2, 2]$ rooted at 0, where the Hider is located at $-2, +1, +2$ with respective probabilities $\frac{7}{13}, \frac{4}{13}$ and $\frac{2}{13}$. The unique maximum density subtree is $[0, 1]$, but if a search begins with a path from 0 to 1, the minimum possible expected search time is given by the path which continues to -2 before going to $+2$. This has expected search

time $\frac{48}{13}$, whereas a search which begins by going to -2 before going to 1 and then 2 has a smaller expected search time of $\frac{46}{13}$.

3.4 Expanding search game on trees

We now assume that the Hider distribution ν is not known to the Searcher. In this case we consider the problem of finding the mixed Searcher strategy (probability measure over expanding searches) which minimises the expected search time in the worst case. An equivalent problem, which we prefer to adopt, is the zero-sum expanding search game $\Lambda = \Lambda(Q, O)$. Here the maximising Hider picks a location H in Q, the minimising Searcher picks an expanding search S in $\mathcal{S}$, and the payoff is the search (capture) time $T(S, H)$. For trees Q, it is clear that the Hider must pick a leaf node, as all other points in Q are dominated. Hence $\Lambda(Q, O)$ is a finite game: the Hider picks a leaf node and the Searcher picks an ordering of the arcs of Q.

We can derive the solution of $\Lambda(Q, O)$ for trees Q from the solution of the variable speed game $\Gamma(Q, O, d)$ for a particular choice of d. We will first show that if the Hider follows the EBD distribution in Λ, then it is optimal or the Searcher to perform a depth-first search.

Lemma 23 *Suppose Q is a tree. Then in the game $\Lambda(Q, O)$ if the Hider hides according to the EBD distribution, then a best response for the Searcher must be a depth-first search.*

Proof. Suppose the Hider is hidden according to the EBD distribution and the Searcher does not use a depth-first search. Then there must exist some node x

such that after reaching x for the first time, the Searcher proceeds to search a branch A of Q_x before searching a non-empty region X which is disjoint from Q_x, and then the other branch B of Q_x. Let a be the arc whose forward node is x. Then this portion of the search may be notated as $aAXB$. Since the Hider is using the EBD distribution, we must have $\rho(A) = \rho(B)$. But by the Search Density Lemma, $\rho(A) \geq \rho(X) \geq \rho(B)$, so we must have $\rho(X) = \rho(A) = \rho(B)$. But in that case $\rho(a \cup A) < \rho(X)$, so using the Search Density Lemma again, the Searcher could improve his search time by swapping the order of search of $a \cup A$ and X, contradicting the optimality of the search. Hence the Searcher must use a depth-first search. ■

So if the Hider uses the EBD distribution then the Searcher uses a depth-first search, and in this case it is easy to see that an expanding search of Q is equivalent to a pathwise search of Q endowed with the quasimetric d defined such that all the reverse travel times R_a are equal to 0. (Strictly speaking, $d(x, y)$ should only take the value 0 when $x = y$, as d is a quasimetric, but this problem can be resolved by setting the reverse travel times to be some small $\varepsilon > 0$ and using a limiting argument.) Hence we can read off the solution of this game from the solution to the pathwise search game on a variable speed network. The incline, Δ is now D, the mean distance from the root to the leaf nodes, with respect to the EBD. The branching probability (6) for an arc a at branch node x is now

$$\beta(a) = \frac{1}{2} + \frac{1}{2\mu_x}(D_a - D_b), \tag{20}$$

where μ_x is the length of the subtree Q_x and D_a and D_b are the mean distances

from x to the leaf nodes of Q_a and Q_b respectively. Hence we have the following:

Theorem 24 *In the expanding search game $\Lambda(Q, O)$ on a tree Q, the branching strategy defined by the branching probabilities (20) is optimal for the Searcher; the EBD is optimal for the Hider; the value V of the game is given by*

$$V = \frac{1}{2}(\mu + D).$$

4 The expanding search game with multiple objects on a tree

In this chapter we extend the idea of the expanding search game on a network introduced in Chapter 3, by allowing the Hider to hide k objects on the network (where k is some fixed number), all of which the Searcher wishes to find. We begin by considering this game played on a *star network*: that is a network consisting of a set of arcs that meet at the root. As we will explain in more detail in Section 4.4, this is equivalent to the following game where balls are hidden in boxes. The Searcher who wishes to find a number of objects (or balls) hidden by a Hider amongst a set of discrete locations (or boxes) each of which has a designated *search cost*. The Searcher looks in the boxes one by one, paying the search costs associated with the boxes he looks in, until he has found all the balls. He wishes to minimise the total search cost of finding the balls, and the Hider wishes to maximise it, so we view the problem as a zero-sum game between the Searcher and the Hider. This is a natural problem to consider, and one which we face on an everyday basis. For example, before leaving the house in the morning we may wish to locate a certain essential items such as wallet, phone and keys. There is a set of discrete locations around the house where these objects may be hidden, each of which takes a particular amount of time to search, and we wish to minimise the total time it takes to find all the items. The problem provides a simple model for other practical search scenarios, such as a search for a number of corrupted files which may be distributed amongst several

folders, or a search for bombs hidden in several locations. The problem is also relevant to studies such as Pravosudov and Clayton [41] which have examined how *scatter hoarders* like squirrels search for food they have previously cached over a number of sites.

After we have formally defined the game in Section 4.1, we will see in Section 4.2 that if both parties are allowed to use mixed strategies it is optimal for the Searcher to begin his search with a subset of k boxes chosen with probability proportional to the product of their search costs, and then search the remaining boxes in a random order. It is optimal for the Hider to choose a subset of k boxes with probability proportional to the product of their search costs. In Section 4.3 we then distinguish between two versions of the game, the first of which restricts the Searcher to setting out his search plan from the start, and the second of which allows him to change his plan during the search, based on information gathered. We call the second type of Searcher a *smart* Searcher to distinguish from the first type which we call a *normal* Searcher. The idea of a smart Searcher was introduced by Alpern *et al* [9], in the context of scatter hoarders. A smart Searcher is clearly at an advantage over a normal Searcher, and this formulation perhaps provides a more realistic model of an intelligent Searcher. However, we will show that in this game a smart Searcher cannot do any better than a normal Searcher against a Hider who is playing optimally.

In Section 4.4 we show how the game can be formulated in terms of an expanding search game with several hidden objects. We go on to discuss the game on the more general tree network, and show that in general, a smart

Searcher has an advantage over a normal Searcher.

The box-searching game solved in this chapter is an original problem, introduced by the author in [35], upon which this chapter and the next is based. The problem of searching for a *single* object hidden according to a known distribution in boxes with search costs where there is an 'overlook probability' associated with each box is a known problem solved by Blackwell (reported in [36]). An alternative solution to the problem is presented by Gittins *et al* [26] using Gittins' well known index for multi-armed bandit processes. A study of the zero-sum game version of the problem can be found in Bram [20] and Ruckle [49]. In Neuts [39], the game is extended so that the boxes each have allocated search costs. Interest in searching in boxes also extends to the field of economics, for example in Weitzman [53] where a Searcher faces a problem of when to stop searching a set of boxes with fixed search costs, and rewards assigned according to a known probability distribution.

To date there has been little study in Search Theory of problems involving multiple hidden objects. Alpern et al. [7] introduced a game in which a Searcher with limited 'digging' resources seeks several objects buried in a choice of locations. An extension of Blackwell's problem in which a Searcher looks for *one* of many hidden objects is examined in Assaf and Zamir [16] and Sharlin [50]. In [38], two Searchers compete to find different objects before each other. Press [42] considers a search problem in which a Searcher samples with replacement to find 'rare malfeasors' hidden according to a known distribution amongst a population. This could be viewed as a search for balls distributed amongst sev-

eral boxes according to a known distribution. In [8], a search game is considered in which a Hider distributes a *continuous* amount of wealth amongst discrete locations.

The related problem of multiple *Searchers* trying to locate an object has been studied by Reyniers [45], [46] and Ruckle [49]. In these studies, two Searchers coordinate to find a single Hider. In Weber [52] two agents aim to minimise the time to find *each other* in a set of discrete locations. This problem lies in the field of *Rendezvous Search*, a category of problems first posed informally in Alpern [1] and later developed in work such as Alpern [2] and Anderson and Weber [15].

4.1 Search for k balls in n boxes

A Searcher wishes to find k balls hidden in a set $B = \{1, ..., n\}$ of n boxes. He can search them in any order, but incurs a known cost c_i when he searches box i. If the balls are hidden (uniformly) randomly, then clearly the best search strategy is to search the boxes in order of increasing cost. We wish to find the randomised search strategy which minimises the expected total cost, in the worst case. For any pure Searcher strategy - that is, any ordering (or permutation) of the boxes B - the worst case occurs when the last (nth) box searched contains a ball. So any pure search strategy is a pure minimax strategy, with total cost equal to the sum of the search costs $\Sigma_{i=1}^{n} c_i$, which we denote by C_0.

So we are naturally led to consider mixed search strategies, and to seek a distribution over orderings of B which minimises the expected total cost, in

the worst case. This problem is equivalent to finding optimal strategies in the zero-sum game $\Theta = \Theta(n, k; c_1, ..., c_n)$ against Nature, where a malevolent Hider (Nature) chooses a k-subset H of B; the Searcher chooses an ordering $(i_1, ..., i_n)$ of B; and the payoff (to the maximising Hider) is the total cost

$$C = C(i_1, ..., i_n; H) = c_{i_1} + c_{i_2} + ... + c_{i_j}$$

if the last ball is found in the jth box to be searched. This is a finite game and has a value which we denote by $V = V(n, k; c_1, ..., c_n)$.

4.2 Optimal strategies

A Hider mixed strategy is a probability distribution over the set $\mathcal{H} = B^{(k)}$ of k-subsets (that is, subsets of size k) $H \subset B$. The optimal Hider distribution $\nu = \nu_{B,k}$ turns out to be the distribution which assigns to each $H \in \mathcal{H}$ a probability proportional to $\pi(H) := \Pi_{i \in H} c_i$, the product of the costs of searching the k boxes in H. That is,

$$\nu(H) = \frac{\pi(H)}{\sum_{A \in \mathcal{H}} \pi(A)}.$$

The distribution is related to a Hider distribution on trees determined recursively by Gal [21]. The main result of this section is the following.

Theorem 25 *The Searcher strategy which minimises the expected total search cost to find k balls amongst n boxes in the worst case is to first search a k-subset of boxes in any order, chosen according to the distribution ν, and then to search*

the remaining $n-k$ boxes in a (uniformly) random order. The order of searching the first k boxes does not affect the payoff. A worst case Hider distribution is the distribution ν.

To prove this theorem, we will first show that the Hider strategy ν ensures the same expected search cost C against any strategy of the Searcher. This puts a lower bound on the value of the game. We then restrict the Searcher to a subset of his strategy set and show that even with this restriction he has a mixed strategy that ensures the same expected search cost C against any Hider strategy. This puts an upper bound on the value which when combined with the lower bound provides the desired result.

We begin by defining a quantity $S_j(H)$ for all subsets $H \subset B$ and for all $j = 1, ..., |H|$. $S_j(H)$ is calculated by taking each j-subset A of H, multiplying together the search costs of the boxes in A, and summing these products.

Definition 26 *For $H \subset B$ and $j = 1, ..., |H|$, let $S_j(H) = \sum_{A \in H^{(j)}} \pi(A)$. We write simply S_j for $S_j(B)$.*

Note that this allows us to notate the Hider strategy ν more concisely: for k-subsets H, we can write $\nu(H) = \pi(H)/S_k$. We will show that ν makes the Searcher indifferent between all his pure strategies, ensuring the same expected cost for any ordering of the boxes. Following standard notation, we write $[j]$ for the set $\{1, ..., j\}$, so that $B = [n]$.

Lemma 27 *For any ordering $(i_1, ..., i_n)$ of the boxes, the expected total search*

cost if the balls are hidden according to ν is

$$C = C(i_1, ..., i_n; \nu) = C_0 - \frac{S_{k+1}}{S_k}. \tag{21}$$

Proof. By a relabelling argument we can assume that the Searcher chooses the ordering $1, 2, ..., n$. We calculate the expected cost of boxes not searched and subtract this from the total cost of the boxes, C_0. All the boxes $1, ..., k$ will certainly be searched, and for $i \geq k+1$, the probability box i is not searched is the probability all k balls are in $[i-1]$, which is

$$\frac{S_k([i-1])}{S_k}.$$

Thus the expected cost of boxes not searched is

$$\frac{1}{S_k} \sum_{i=k+1}^{n} c_i S_k([i-1]).$$

This sum is clearly equal to S_{k+1}, so the expression above is S_{k+1}/S_k, and subtracting this from C_0 gives (21). ■

We will see that the Hider strategy ν set out at the beginning of the section is optimal, and that the value of the game is given by (21). To this end, we now define a restricted game $\Theta' = \Theta'(n, k; c_1, ..., c_n)$ in which the Searcher's strategy set is reduced, and we will show that in Θ' the Searcher can attain expected search time $C_0 - S_{k+1}/S_k$ against any Hider strategy.

Definition 28 *Let Θ' be the same as Θ except that after searching the first k boxes the Searcher must search the remaining boxes in a (uniformly) random order. Let $V' = V'(n, k; c_1, ..., c_n)$ be the value of this game. We must have $V' \geq V$.*

This reduces the number of pure strategies for the normal Searcher to $\binom{n}{k}$, since a pure strategy can now be specified by some $A \in B^{(k)}$ which corresponds to the first k boxes of his search. Hence the Hider and the Searcher have the same strategy set, $\mathcal{H} = B^{(k)}$. We now show that Θ' is a symmetric game, in the sense that its payoff matrix is symmetric.

Lemma 29 *Let $H, A \in \mathcal{H}$ be strategies in Θ' for the Hider and Searcher, respectively. Then $C(A, H) = C(H, A)$.*

Proof. $C(A, H)$ is the expected cost of a search that begins with A and searches the rest of the boxes in a random order until all the balls in H are found. In such a search all the boxes in $A \cup H$ must be searched before the balls have all been found, since the Searcher begins by searching all the boxes in A, and cannot stop until he has searched all the boxes in H. As for the remaining boxes not in $A \cup H$, these are each searched with the same probability q after all the boxes in A have been searched. Note that the value of q depends only two things. The first is the number $j = |A - H|$ of remaining balls to be found after the boxes in A have been searched, and the second is the number $m = n - k$ of remaining unsearched boxes at this point. Hence

$$C(A, H) = \sum_{i \in A \cup H} c_i + q \sum_{i \notin A \cup H} c_i.$$

Notice that $|A - H| = |H - A|$, and so j, m and q are unchanged if the set A and H are interchanged. Thus $C(A, H) = C(H, A)$. ■

The solution of the game Θ' follows immediately from this lemma, using the well-known result below about finite zero-sum games with symmetric payoff

matrices.

Lemma 30 *For a 2-player, zero-sum game between Players I and II with $n \times n$ symmetric payoff matrix M, if $x^* = (x_1^*, ..., x_n^*)^T$ is a mixed strategy for Player I that makes Player II indifferent between all his strategies, then the strategy pair (x^*, x^*) forms an equilibrium.*

Proof. Since x^* makes Player II indifferent, there is some number U such that $U = (x^*)^T My$, for all strategies y of Player II. Equivalently,

$$\begin{aligned} U^T &= U = \left((x^*)^T My\right)^T \\ &= y^T M^T \left((x^*)^T\right)^T \\ &= y^T M x^*, \text{ since } M \text{ is symmetric.} \end{aligned}$$

Since this holds for all y, Player II can make Player I indifferent between all his strategies by playing x^*. Hence if both players play x^* each player is playing a best response to the other, and this is an equilibrium. ■

Corollary 31 *The value of Θ' is $V' = C_0 - \frac{S_{k+1}}{S_k}$. The strategy ν is optimal for both the Hider and the Searcher.*

Proof. If the Hider uses the strategy ν, by Lemma 27, the Searcher will be indifferent and the expected cost will be $C_0 - \frac{S_k}{S_{k+1}}$. By Lemmas 29, Γ' is symmetric, so by Lemma 30, if the Searcher also uses the mixed strategy ν, this forms an equilibrium and the value of the game is $V' = C(\nu, \nu) = C_0 - \frac{S_{k+1}}{S_k}$. ■

Theorem 25 follows by combining our results.

Proof of Theorem 25. By Lemma 27, the Hider can ensure expected search cost no less than $C_0 - \frac{S_{k+1}}{S_k}$ by using ν, so $V \geq C_0 - \frac{S_{k+1}}{S_k}$. By Corollary 31, the Searcher can ensure expected search cost no more than $C_0 - \frac{S_{k+1}}{S_k}$ by using the strategy ν, so $V \leq V' = C_0 - \frac{S_{k+1}}{S_k}$. Hence we must have equality, and ν is optimal for both players. ■

Note that the value of the game must be increasing in k since the Searcher's optimal strategy for the game $\Theta(n, k+1; c_1, ..., c_n)$ will find k hidden balls with total search cost no greater than $V(n, k+1; c_1, ..., c_n)$. Therefore we must have

$$\begin{aligned} C_0 - \frac{S_{k+1}}{S_k} &\geq C_0 - \frac{S_k}{S_{k-1}}, \text{ so that} \\ S_k^2 &\geq S_{k-1}S_{k+1}, \text{ and} \\ \log S_k &\geq \frac{\log S_{k-1} + \log S_{k+1}}{2}. \end{aligned}$$

It follows that $S_1, S_2, ...$ is a logarithmically concave sequence. We can also see this in another way using the concept of logarithmically concave polynomials (that is, polynomials whose sequence of coefficients is logarithmically concave). It is well known that the product of logarithmically concave polynomials is logarithmically concave [27]. That is, if $A(x) = a_0 + a_1x + a_2x^2 + ...$ and $B(x) = b_0 + b_1x + b_2x^2 + ...$ are logarithmically concave, so that they satisfy $a_k \geq a_{k+1}a_{k-1}$ and $b_k \geq b_{k+1}b_{k-1}$ for all $k \geq 1$, then $C(x) = A(x)B(x)$ is logarithmically concave. Let $S(x)$ be the polynomial whose coefficients are the S_k, so

$$S(x) = 1 + S_1x + S_2x^2 + ... + S_nx^n.$$

Then we can factorise $S(x)$ as

$$S(x) = \prod_{j=1}^{n} (1 + c_j x)\,,$$

and since the polynomial $\left(1 + c_j x + 0x^2 + 0x^3 + ...\right)$ is logarithmically concave for each j, $S(x)$ is the product of logarithmically concave polynomials and must be logarithmically concave itself.

4.3 Smart and normal strategies

So far we have assumed that the Searcher must set out his search plan at the start of the search, but we now consider a variation of the game where the Searcher is permitted to adapt his plan during the search using information gathered. We call such a Searcher *smart*, as opposed to a *normal* Searcher who is required to set out his search plan from the start. In the case of a single hidden ball it is clear that a smart Searcher cannot do any better than a normal Searcher. In Alpern et al. [9] the authors showed that for their model of scatter hoarding behaviour it was advantageous for a Searcher to be smart. However, we find the contrary here, that a smart Searcher has no advantage over a normal Searcher.

We denote the box-searching game with a smart Searcher by $\tilde{\Theta} = \tilde{\Theta}(n, k; c_1, ..., c_n)$, and let the value of the game be $\tilde{V} = \tilde{V}(n, k; c_1, ..., c_n)$. Clearly the value of this game is no greater than the value of the equivalent game with a normal Searcher, since a smart Searcher can always use a normal strategy. We combine this observation with Theorem 25 to give the lemma below.

Lemma 32 *$\tilde{V} = \tilde{V}(n, k; c_1, ..., c_n)$ satisfies $\tilde{V} \leq C_0 - \frac{S_{k+1}}{S_k}$.*

To show that a smart Searcher cannot do better, we will need a technical lemma about the function S_k, which is a straightforward calculation. For any set $H \subset B$ write $S_j^i = S_j(B - i)$.

Lemma 33 *For any $i = 1, ..., n$,*

$$S_k = c_i S_{k-1}^i + S_k^i.$$

Proof. We split the sum $S_k = \sum_{A \in B^{(k)}} \pi(A)$ into the sum over subsets A which include i and those that do not. Accordingly,

$$\begin{aligned} S_k &= \sum_{A \in (B-i)^{(k)}} \pi(A) + \sum_{A \in (B-i)^{(k-1)}} c_j \pi(A) \\ &= c_i S_{k-1}^i + S_k^i. \end{aligned}$$

■

We use this lemma to give an inductive proof that a smart Searcher strategy cannot do any better than a normal Searcher strategy against ν. The idea of the proof is that the information gathered during a search is of no use to the Searcher because ν has the property that at all points of a search, the remaining balls to be found will be hidden according to the optimal Hider distribution on the unsearched boxes. We formalise this in the next lemma.

Lemma 34 *Suppose the k balls are hidden according to the optimal Hider distribution ν for the game $\Theta(n, k; c_1, ..., c_n)$, and let $i \in B$. Let $\nu' = \nu_{B-i,k-1}$ be the optimal Hider distribution for $k - 1$ balls hidden in the boxes $B - i$ and*

let $\nu'' = \nu_{B-i,k}$ be the optimal Hider distribution for k balls in the boxes $B-i$. Then given that there is a ball in box i, the remaining $k-1$ balls are hidden in $B-i$ according to ν'; given that there is no ball in box i, the k balls are hidden in $B-i$ according to ν''.

Proof. The probability α that there is a ball in box i is given by $\alpha = \frac{c_i S^i_{k-1}}{S_k}$. Hence, conditional on there being a ball in box i, the probability the remaining $k-1$ balls are hidden in some $(k-1)$-set $H \in (B-i)^{(k)}$ is

$$\frac{\nu(H \cup i)}{\frac{c_i S^i_{k-1}}{S_k}} = \frac{\frac{\pi(H) c_i}{S_k}}{\frac{c_i S^i_{k-1}}{S_k}} = \frac{\pi(H)}{S^i_{k-1}} = \nu'(H).$$

Similarly, the probability there is no ball in box i is $1-\alpha = 1 - \frac{c_i S^i_{k-1}}{S_k} = \frac{S_k - c_i S^i_{k-1}}{S_k} = \frac{S^i_k}{S_k}$ (by Lemma 33). Hence, conditional on there not being a ball in box i, the probability the remaining k balls are hidden in some k-set $H \subset B-i$ is

$$\frac{\nu(H)}{\frac{S^i_k}{S_k}} = \frac{\frac{\pi(H)}{S_k}}{\frac{S^i_k}{S_k}} = \frac{\pi(H)}{S^i_k} = \nu''(H)$$

■

From this property of ν it follows that a smart Searcher cannot guarantee a total search cost any smaller than a normal Searcher.

Theorem 35 *The value of $\tilde{\Theta}$ is $\tilde{V} = C = C_0 - \frac{S_{k+1}}{S_k}$. The strategy ν is optimal for both the Hider and the Searcher.*

Proof. We prove by induction on n that if the balls are hidden according to ν then the expected search cost of any smart search is C. This is clearly true for $n = 2$, since every smart search is a normal search. Assume it is true for

$n-1$, and suppose without loss of generality that a smart Searcher begins by searching box 1. Subsequent to searching this box, regardless of whether he finds an object, Lemma 34 implies that the remaining objects will be hidden optimally. Hence, by the induction hypothesis and Lemma 27 the expected search cost of all smart searches of the remaining boxes is the same, including the search which opens the boxes in the order $2, 3, ..., n$. So the total expected search cost is the same as that of the normal search $1, 2, ..., n$, which by Lemma 27 is C, completing the induction. It follows that $\tilde{V} \geq C$, and combining this with Lemma 32 completes the proof. ■

We can also consider variations of the game in which the Hider is smart, so that every time the Searcher opens a box, he can rearrange the remaining balls that have not been found. This gives rise to two more games, one in which the Searcher is also smart and one in which he is normal. The value of these games is clearly greater than the values of the corresponding games in which the Hider is normal, which we have seen are both equal to $C = C_0 - \frac{S_{k+1}}{S_k}$.

In the case where only the Hider is smart, the value is strictly greater than C in general, though the game is non-trivial to analyse and there are no examples for which the value is greater than C when $n \leq 3$. We present here an example for $n = 4$. Suppose the boxes $(1, 2, 3, 4)$ have search costs $(1, 10, 50, 50)$ and there are 2 balls. We give a Hider strategy that ensures expected search cost greater than C for any normal Searcher strategy. The Hider starts by hiding the balls according to the optimal normal strategy ν. If the Searcher opens box 1 first and finds a ball, the Hider puts the remaining ball in box 2. If not, he

hides the two remaining balls in boxes 3 and 4 with probability 197/245, and otherwise hides them equiprobably in boxes 2 and 3 or boxes 2 and 4. If the Searcher starts with box 2 and finds a ball, the Hider puts the remaining ball in box 1 with probability 361/10201, and otherwise hides them equiprobably in box 3 or 4. If not he puts the other two balls in boxes 3 and 4. If the Searcher starts with box 3 or 4 and finds a ball, the Hider puts the remaining ball in boxes 1 and 2 with respective probabilities 361/18605 and 9122/55815 and otherwise puts it in the other box of cost 50. If not he puts the two balls in box 2 and the remaining box of cost 50. We leave it as an exercise to the reader to check that against any normal Searcher strategy the expected cost is greater than C.

If both the Hider and the Searcher are smart, then it turns out the value is C. To prove this, it is sufficient to give a strategy for a smart Searcher that makes the Hider indifferent between all his strategies, since we already know that the Hider strategy ν ensures an expected search cost of C against any smart search, by Theorem 35. To define a smart search we simply need to specify the probability p_i that the Searcher begins with some box $i = 1, ..., n$, for given n and k. Let

$$p_i = \left(\frac{S_k^i}{S_{k-1}^i} - \frac{S_{k+1}^i}{S_k^i} \right) \left(\sum_{j=1}^{n} \frac{S_k^j}{S_{k-1}^j} - \frac{S_{k+1}^j}{S_k^j} \right)^{-1}.$$

Since S_k^i is a logarithmically concave sequence, $\left(S_k^i / S_{k-1}^i - S_{k+1}^i / S_k^i \right) \geq 0$, so p_i is certainly a probability. We then prove by induction on n that when the Searcher uses p, the expected search cost of boxes not searched is the same whatever strategy the Hider uses, so that it must be equal to S_{k+1}/S_k. The

base case $n = 2$ is easily verified. Assuming the induction hypothesis is true for $n-1$, the expected search cost against the Hider strategy H is

$$\sum_{i \in H} p_i \frac{S_k^i}{S_{k-1}^i} - \sum_{i \notin H} p_i \frac{S_{k+1}^i}{S_k^i}. \tag{22}$$

It is sufficient to show that expected search cost is the same against any two Hider strategies H and H' that differ only in that H contains i and H' contains j. The difference in the quantity (22) for two such Hider strategies is

$$p_i \left(\frac{S_k^i}{S_{k-1}^i} - \frac{S_{k+1}^i}{S_k^i} \right) - p_j \left(\frac{S_k^j}{S_{k-1}^j} - \frac{S_{k+1}^j}{S_k^j} \right),$$

which is 0, by our choice of p. Summing this up, we have the following.

Theorem 36 *The value of the game with a smart Searcher and smart Hider is C. The strategy ν is optimal for the Hider and the strategy p given above is optimal for the Searcher.*

4.4 Box search as an expanding search on trees

We can reformulate the games analysed in this chapter so far in terms of an expanding search game. In this section we begin by describing how searching in boxes can be thought of as a special case of expanding search on a network, and we define precisely the expanding search game for multiple objects on a network.

Consider the star network Q_n consisting of n arcs $a_1, ..., a_n$ which meet at the root O, as depicted in Figure 8.

For given costs $c_1, ..., c_n$, let arc a_i have length c_i. We can now define a search game, $\Lambda_k(Q_n, O)$ in which a normal Searcher chooses an expanding search

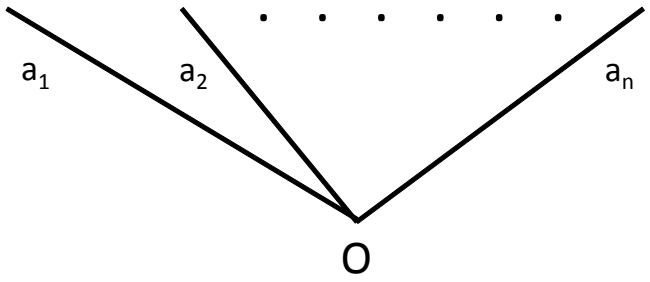

Figure 8: The network Q_n.

starting at O and the Hider chooses k points on Q_n. An expanding search on Q_n is essentially an ordering of the arcs $a_1, ..., a_n$. We define the payoff of the game as the first time all the points chosen by the Hider have been reached by the search. It is optimal for the Hider to choose k distinct leaf nodes (that is, the tips of the arcs), and so it is easy to see that $\Lambda(Q_n, k)$ is equivalent to the box-searching game $\Theta(n, k; c_1, ..., c_n)$, which we have solved in Section 4.2. Of course, for any network Q with root O we can define in the natural way an associated expanding search game, $\Lambda_k(Q, O)$ with k hidden objects, where a Hider chooses k points on Q and a normal Searcher chooses an expanding search of Q starting at O. Of course, when $k = 1$, $\Lambda_k = \Lambda$. We denote the value of the game (if it exists) by $V_k = V_k(Q, O)$.

For $k = 1$ the solution of the game $\Lambda_k(Q, O)$ is given in Chapter 3 (Theorem 24) for general tree networks Q, and a the value is $V = (\mu + D)/2$, where μ is the total length of the network and D is an average of the distances of the root to the leaf notes, weighted with respect to the EBD distribution. For a star,

$\mu = C_0$ and $D = \sum \frac{c_i}{C_0} c_i = \frac{1}{C_0} \sum c_i^2$. Our formula from Theorem 25 gives:

$$\begin{aligned} V &= C_0 - \frac{S_2}{S_1} \\ &= C_0 - \frac{\sum_{i<j} c_i c_j}{C_0} \\ &= \frac{C_0}{2} + \frac{C_0^2 - 2\sum_{i<j} c_i c_j}{2C_0} \\ &= \frac{C_0}{2} + \frac{\sum c_i^2}{2C_0} \\ &= \frac{1}{2}(C_0 + D). \end{aligned}$$

This shows that the two formulas are equivalent for $k = 1$. The optimal Hider strategy given in Theorem 24 is also the same, but the optimal Searcher strategy is different. It mixes between 2^{n-1} pure strategies, whereas the optimal strategy presented in Section 4.2 mixes between all $n!$ pure strategies.

For the game $\Lambda_k(Q, O)$ played on general trees Q, it is clear that it is optimal for the Hider to place the objects at distinct (if possible) leaf nodes of the tree. A Searcher pure strategy is a sequence of arcs of the tree, the base of each of which touches one of the arcs already searched. Λ_k can therefore be reduced to a finite game, and must have a value. In a small number of special cases, the solution of the game on a tree can be deduced from the solution of the game on a star. For example, consider the tree depicted on the left of Figure 9. For the game with 2 hidden objects, it is clear that the arc a must be traversed by the Searcher before he finds all the hidden objects, since there must be an object at the leaf node of arc b or c. Hence this network has the same value as the network depicted on the right of Figure 9, since the Searcher may as well begin by traversing the arc a. The solution of the game on the network on the right

follows easily from the solution of the game on a star network.

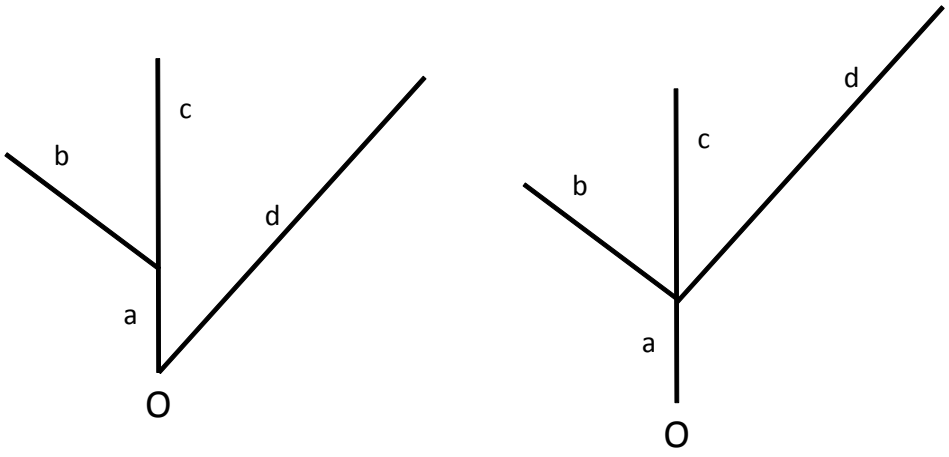

Figure 9: Two tree networks.

This principle can be extended to more general trees, but does not produce strong results. If a tree is binary (each branch node has two outward arcs), then the solution of the search game for k Hiders can be deduced from the solution on a star only if there are no more than $k+1$ leaf nodes.

We investigate the solution of the game for $k = 2$ on a particular tree Q, depicted in Figure 10, with leaf nodes A, B, C, D. All arcs have unit length.

Up to symmetry, the Hider has only 2 pure strategies: hide on the same side (choose nodes AB or CD) or hide on different sides (choose nodes AC, AD, BC or BD). Similarly, the Searcher only has two pure strategies up to symmetry: start with two nodes on the same side or start with two nodes on different sides. Hence the game can be reduced to a 2×2 matrix game in which the Hider chooses between the strategies '*same*' and '*diff*' and the Searcher chooses

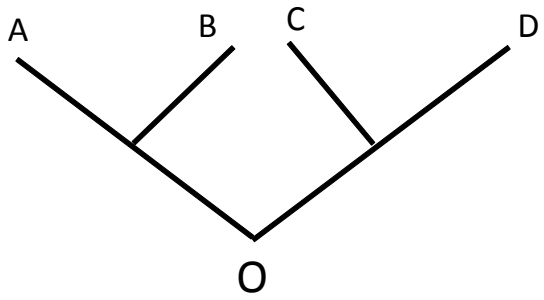

Figure 10: The network Q.

between '*SAME*' and '*DIFF*', the interpretation being that if, say the Hider chooses *same*, he randomises between AB and AC. This gives the following representation of the game in strategic form.

Hider/Searcher	*SAME*	*DIFF*
same	4.5	5.5
diff	5.5	5.25

Solving this gives a value of $V = 5.3$, with both players' unique optimal strategies given by the probability vector $(1/5, 4/5)$. Note that in general the game played on a tree does not have a symmetric payoff matrix.

As before, we can also define the related game $\tilde{\Lambda}_k(Q, O)$ on a network Q with root O, in which the Hider chooses k points on the network, and a *smart* Searcher chooses an expanding search of Q starting at O, so that at any point during the search the Searcher is permitted to change his search plan. We denote the value of this game (if it exists) by $\tilde{V}_k = \tilde{V}_k(Q, O)$, which can be no greater than V_k. If Q is a tree, both a smart and normal Searcher have a finite strategy set, so V_k and $\tilde{V}_k$ both certainly exist.

Theorems 25 and 35 show that if Q is a star network then $V_k(Q,O) = \tilde{V}_k(Q,O)$. We now show that this may not be true for general trees Q. Consider the smart search game $\tilde{\Lambda}_2(Q,O)$ played on the tree Q in Figure 10 with $k=2$, and suppose a smart Searcher uses the following strategy. He searches the nodes in the order $ACDB$ if there is an object at A, and otherwise he searches them in the order $ABCD$. If the Hider uses the strategy $(1/5, 4/5)$ which is uniquely optimal in the game $\Lambda(Q,k)$, then the Searcher will find an object at A with probability $1/2$. If he does find an object here, then with probability $4/5$ he finds the remaining object at one of the nodes on the other side after total expected time 4.5; with probability $1/5$ he finds the remaining object only after searching the whole network, at time 6. So if he finds an object at A the expected search time is $4/5(4.5) + 1/5(6) = 4.8$. Similarly, if he doesn't find an object at A, then with probability $4/5$ he finds an object at B, and the remaining object after expected time 5.5; with probability $1/5$ he finds both objects on the other side after time 6. So if he doesn't find an object at A the expected search time is $4/5(5.5) + 1/5(6) = 5.6$. So on average the expected search time is $1/2(4.8) + 1/2(5.6) = 5.2 < V(Q,2)$.

We have showed that against the Hider strategy $(1/5, 4/5)$ there is a smart search that guarantees expected search time strictly less than $V_2(Q,O)$. If the Hider follows any other strategy then this must be sub-optimal in the normal search game $\Lambda_2(Q,O)$, so there is a normal search (and hence a smart search) which guarantees expected time strictly less than $V_2(Q,O)$. Hence by the minimax theorem for zero-sum games, the value $\tilde{V}_2(Q,O)$ of the smart search game

$\tilde{\Lambda}_2(Q, O)$ satisfies $\tilde{V}_2(Q, O) < V_2(Q, O)$.

It can be shown that up to symmetry a smart Searcher has 6 pure strategies in the game $\tilde{\Lambda}_2(Q, O)$, and the value is $\tilde{V}_2 = 5.25$. The optimal Hider strategy is $(1/4, 3/4)$ and the optimal Searcher strategy can be described as follows. Pick a node at random to start with; if there is no object here, then search the other node on the same side and the remaining two in a random order; if there is an object here, with probability 1/2 search the other node on the same side and the remaining two in a random order, and with probability 1/2 search the two nodes on the other side first then the remaining node.

5 Expanding Search for Multiple Objects on General Networks

In this chapter we begin in Section 5.1 by giving a general result about searching for multiple objects before going on in Section 5.2 to define more precisely the expanding search game with multiple hidden objects on a network and to prove that the game always has a value. In Section 5.3 we define strategies for both a normal and a smart Searcher which can be viewed as analogues of the Random Chinese Postman Tour introduced by Gal [21]. These strategies give upper bounds on the value of the game. Finally we examine the game played on 2-arc-connected networks. A 2-arc connected network is a network that cannot be disconnected by the removal of fewer than 2 arcs. We give the solution of the game for a smart Searcher, and an upper bound for the value of the game for a normal Searcher, showing that although this bound is not tight in general, it is if the network is a circle.

5.1 A generalised uniform strategy for the Hider

In this section we define a particular mixed Hider strategy for the expanding search game with multiple hidden objects which will give us a lower bound on the expected search time against any Searcher strategy (including smart Searcher strategies). The Hider strategy is a generalisation of the uniform strategy u first introduced by Isaacs [28], and discussed in Chapter 1. The lower bound on the expected search time generalises Isaacs' 1969 result given in Theorem 1.

In particular, we will generalise the version of this result given in [11], which proves that the lower bound holds against any *generalised search strategy* (as defined below) in an arbitrary search space Q (that is, some finite, measurable subset of Euclidean space).

Definition 37 *For a search space Q, a **generalised search strategy** is defined by the sets $X(t) \subset Q$ that have been 'discovered' by time t ($t \geq 0$). The sets $X(t)$ must satisfy the conditions*

$$X(t) \subset X(t') \text{ for all } t < t', \text{ and } \lambda(X(t)) \leq t \text{ for all } t > 0,$$

where λ is the Lebesgue measure of Q. The set of generalised search strategies is denoted by $\mathcal{S} = \mathcal{S}(Q)$.

In practice, from now on we will assume that $\lambda(X(t)) = t$ for all t, so the Searcher is searching as quickly as possible, and hence $X(\mu) = Q$. Notice that expanding search satisfies the conditions to be a generalised search strategy (see Section 5.2).

The notion of a *smart* generalised search strategy is defined analogously to that of a smart expanding search strategy as given in Chapter 4. In other words, a smart generalised search strategy is a generalised search strategy that can be changed by the Searcher whenever he finds one of the hidden objects. The set of all smart search strategies is complicated to describe, so we limit ourselves to this informal verbal description. As before we distinguish generalised search strategies from smart generalised search strategies by calling the formal *normal* generalised search strategies.

We also describe formally a strategy for a Hider with k objects in a search space Q.

Definition 38 *A Hider strategy in a search space Q is an ordered set of k points $H = (H_1, ..., H_k)$ in Q. The set of Hider strategies is denoted by $\mathcal{H}_k$.*

The order is irrelevant for the models presented here, but a Hider strategy cannot be defined simply as a set because this would not allow the Hider to place two objects in the same place. For a given Hider strategy $H \in \mathcal{H}_k$ and generalised search strategy X, we define the search time $T(X, H)$ as

$$T(X, H) = \inf\{t : H_i \in X(t) \text{ for all } i = 1, ..., k\}.$$

We think of $T(X, H)$ as the first time t that $X(t)$ contains all the points $H_1, ..., H_k$.

A mixed strategy always available to the Hider is the uniform strategy where each of the k objects is independently hidden uniformly on Q, so that the probability any given object is contained in some measurable subset of Q is proportional to the measure of that subset. We denote this Hider strategy by $u_k = u_k(Q)$. We will show that against this Hider strategy any smart generalised search strategy has the same expected search time.

Theorem 39 *Suppose X is a smart generalised search strategy, and suppose a Hider hides k objects on Q according to the uniform strategy u_k. Then the expected search time $T(X, u_k)$ is given by*

$$T(X, u_k) = \left(1 - \frac{1}{k+1}\right) \mu$$

Proof. We prove the theorem by induction of k. For $k = 1$, smart search strategies are the same as normal search strategies, and the result follows from Theorem 3.3 of Alpern and Gal [11]. For completeness we give the proof here as well. (It is also more or less identical to the proof of 1 given in Chapter 1.) The probability the single hidden object is found after some time t is $1 - t/\mu$, so the expected time to find the object is given by

$$\begin{aligned} T(X, u_1) &= \int_0^\mu (1 - t/\mu) dt \\ &= \left[t - \frac{t^2}{2\mu} \right]_{t=0}^{\mu} \\ &= \mu/2. \end{aligned}$$

Now suppose the result is true for all $j < k$ with $k \geq 2$, and suppose that k hidden objects are hidden according to u_k. Let T_1 be the time taken for a smart generalised search strategy X to find the first object, so that

$$T_1 = \inf\{t : H_i \in X(t) \text{ for some } i = 1, ..., k\}.$$

The probability T_1 is greater than some t is the probability that all the objects are in a set of measure $\mu - t$, which is $(1 - t/\mu)^k$. Hence the probability that T_1 is at most some $t \geq 0$ is

$$\Pr(T_1 \leq t) = 1 - \Pr(T_1 > t) = 1 - (1 - t/\mu)^k.$$

So, differentiating, the probability density function f of T_1 is

$$f(t) = \frac{k}{\mu} (1 - t/\mu)^{k-1}.$$

Given that the first object is found at time t, the remaining objects are clearly hidden uniformly in $Q' = Q - X(t)$ according to $u_{k-1}(Q')$, and from time t, the

Searcher performs a smart generalised search for $k-1$ objects on Q', which has measure $\mu - t$. Hence, by induction the remaining expected search time T' is

$$T'_t = (1 - 1/k)(\mu - t).$$

Putting this together, the expected search time of all k objects is

$$\begin{aligned} T(X, u_k) &= \int_{t=0}^{\mu} (t + T'_t) f(t) dt \\ &= \int_{t=0}^{\mu} (t + (1 - 1/k)(\mu - t)) \frac{k}{\mu} (1 - t/\mu)^{k-1} dt \\ &= \left(1 - \frac{1}{k+1}\right) \mu. \end{aligned}$$

So by induction, the theorem is true for all k. ■

This theorem is a generalisation of Isaacs' 1969 result from [28], noted in the Introduction (Theorem 1).

5.2 Existence of value for expanding search game with multiple hidden objects

We now turn back to the expanding search game with multiple hidden objects, beginning with a more general definition of an expanding search S. The idea behind the definition concerns the closed region $S(t)$ of Q that it has searched by time t. As t increases, its size (total length) cannot grow too fast and it cannot jump to new points detached from those it has already searched. A Hider at H is captured when H first belongs to the searched set $S(t)$. These ideas are captured in the following definition.

Definition 40 *An expanding search S on a network Q is a nested family of connected closed sets $S(t)$, $0 \le t \le \mu$, which satisfy*

(i) $S(0) = O$ *(starts at the root of* Q*),* $S(\mu) = Q$ *(exhaustive search),*

(ii) $S(t') \subset S(t)$ *for* $t' < t$*, and*

(iii) $\lambda(S(t)) \leq t$.

The set of all expanding searches is denoted by $\mathcal{S}$.

Notice that apart from the restrictions that $S(0) = O$ and that the sets $S(t)$ must be connected, the definition is identical to that of generalised search. In particular, an expanding search *is* a generalised search strategy, so Theorem 39 holds for expanding search.

It follows from (ii) and (iii) that $\lambda(S(t) - S(t')) = t - t'$ and consequently by connectedness that $d_{Haus}(S(t), S(t')) \leq t - t'$, where d_{Haus} is the Hausdorff metric on Q. The set $\mathcal{S}$ is compact in the uniform Hausdorff metric

$$d^*(S, S') = \max_{0 \leq t \leq \mu} d_{Haus}(S(t), S'(t)). \tag{23}$$

For $S \in \mathcal{S}$ and a Hider strategy $H = (H_1, ..., H_k) \in \mathcal{H}_k$, let $T^* = \inf\{t : H_i \in S(t) \text{ for all } i = 1, ..., k\}$. Suppose some $H_i \notin S(T^*)$. Then for $T^* < t < T^* + d_{Haus}(S(t), S(T^*))$ we also have that $H \notin S(t)$, contradicting the definition of T^*. Consequently $H_i \in S(T^*)$ for all i. This shows that the infimum is a minimum, and leads to the definition

$$T(S, H) = \min\{t : H_i \in S(t) \text{ for all } i = 1, ..., k\}.$$

This shows that the expanding search game $\Lambda_k(Q, O)$ with k hidden objects is well defined. This level of rigour was unnecessary in Chapter 4 as all the games considered were finite.

Next we show that for any fixed $H \in \mathcal{H}_k$, $T(S,H)$ is lower semicontinuous for $S \in \mathcal{S}$ with respect to the uniform Hausdorff metric d^*. Suppose $T(S,H) > t_0$. Then $H_i \notin S(t_0)$ for some i and furthermore if $d^*(S,S') < d(H_i, S(t_0))$ it follows that $H_i \notin S'(t_0)$ and so $T(S',H) > t_0$. Thus $T(S,H)$ is lower semicontinuous in S for fixed H and hence also $T(S,h) = \int_Q T(S,H)\ d\nu(H)$ is lower semicontinuous for any fixed Hider distribution ν on Q.

To summarise, we have shown that $\mathcal{S}$ is compact in the uniform Hausdorff metric (23) and that T is lower semicontinuous in S for fixed H. So the Minimax Theorem of Alpern and Gal [10] gives the following.

Theorem 41 *For any network the expanding search game $\Lambda_k(Q,O)$ has a value $V = V_k(Q,O)$, the Searcher has an optimal mixed strategy and the Hider has ε-optimal mixed strategies.*

We do not give an existence proof for the value of the expanding search game $\tilde{\Lambda}_k(Q,O)$ with a smart Searcher, but we will demonstrate that it exists for certain networks by calculating the value and giving optimal strategies explicitly.

5.3 Pointwise search and upper bounds for value

For an expanding search it is not in general possible to say where the Searcher is located at a particular time t, unlike for example in the pathwise paradigm in Gal's analysis of search games, where the Searcher simply follows a continuous path in Q. Of course if $S([0,t]) - S([0,t))$ is a single point $P(t)$, this is where the Searcher would be located at time t. In fact we can characterise functions $P(t)$ which arise in this manner as *pointwise searches*. Unlike pathwise searches,

pointwise searches are not in general continuous. There are certain *restart times* $\mathcal{T}_S = \{t : P(t) = P(t'),\ \text{some } t' < t\}$ when the search returns to *restart points* $P(\mathcal{T}_P)$ which it has visited earlier. In the special case of combinatorial searches (expanding arc sequences) considered in Chapter 3, the restart points are those nodes which are tails of two or more arcs involved in the search. The fact that intervals (arc interiors) of zero density are searched without interruption (that is, full arcs are searched) is obvious. This result also applies to arcs where the distribution is uniform. This is a version of Lemma 6 - see Corollary 8 of [12].

Given a pointwise search $P \in \mathcal{S}^p$ and a hiding point $H \in Q$, the capture time $T = T(P, H)$ is given by

$$T(P, H) = \min\{t : P(t) = H\}.$$

Furthermore the set $\mathcal{S}^p$ of pointwise searches is dense in $\mathcal{S}$ in the uniform Hausdorff topology given by (23). To see this, fix any search $S \in \mathcal{S}$ and for a positive integer n let $t_i = i\mu/n$ for $i = 0, \ldots, n$. Let $P \in \mathcal{S}^p$ be a search such that $P([0, t_i]) = S(t_i)$, $i = 0, \ldots, n$. Such a P is easily constructed, as the subarcs in $S(t_i) - S(t_{i-1})$ can be traversed one at a time by P in the interval $[t_{i-1}, t_i]$. A similar argument in a slightly simpler setting is worked out in detail in Lemma 2 of [12]. The significance of the fact that pointwise searches are dense is that we can obtain $\varepsilon-$optimal expected search strategies using them against any Hider distribution.

An example of a Q, ν where the minimum search time requires a general expanding search rather than any pointwise search is the following.

Example 42 *Let Q be the interval $[-1,1]$ with its root O at the center 0. Let ν be given by the density function $h(x) = 1 - |x|$. The unique optimal expanding search $S(t)$ is given by $S(t) = [-t/2, t/2]$ for $0 \leq t \leq 2 = \mu$. (A derivation of this fact is given in [14]) Clearly the expanding search S is not induced by any pointwise search P, as $S([0,t]) - S([0,t)) = \{-t/2, t/2\}$ is not a singleton set.*

Every pointwise expanding search P has a length $\lambda = \min\{t : P([0,t]) = Q\}$, which is the first time the whole network has been searched. If P satisfies $P([0,t]) = t$ for all t, then clearly it has length μ (the total length of the network). For a pointwise expanding search of length λ we can define P^{-1}, the *reverse* of P by $P^{-1}(t) = P(\lambda - t)$ for $0 \leq t \leq \lambda$ and $P^{-1}(t) = O$ for $t > \lambda$. The function P^{-1} may not be a pointwise expanding search at all, but if it is then we say P is *reversible*. A reversible pointwise expanding search P of minimal length is called a *minimal reversible expanding search* (or *MRES*), and for a given network Q we denote the length of an MRES of Q by $\bar{\mu} = \bar{\mu}(Q)$.

It may be the case that a network's MRES 'doubles back' on itself, as in the network in Figure 11 in which all the arcs a, b, c and d have unit length. An example of an MRES on this network can be described as: take the path along arc a from O to A, then the path along b from O to A, then the path from A to B and back again, then the path from A to O along c. Here the arc d is necessarily traversed twice, so that the length $\bar{\mu}$ of the MRES is 5.

We use the concept of the MRES to define a mixed strategy for a smart Searcher, called the smart k-uniform MRES.

Definition 43 *Let P be an MRES on a network Q, and for each $j = 0, 1, ..., k$*

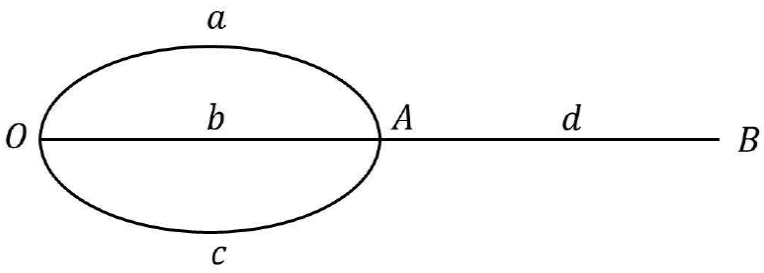

Figure 11: A network with an MRES that doubles back on itself.

let P_j be the smart search that follows P until j objects have been found, than follows P^{-1} until the remaining $k-j$ objects have been found. The smart k-uniform MRES P_* is defined as an equiprobable choice of the P_j.

Following P_* gives an upper bound on the expected search time, as we now show.

Theorem 44 *If a smart Searcher follows a smart k-uniform MRES P_*, he ensures an expected search time of no more than $\left(1-\frac{1}{k+1}\right)\bar{\mu}$ against any Hider strategy H in the game $\tilde{\Lambda}_k(Q,O)$.*

Proof. Suppose the MRES P finds the k objects at times $t_1, t_2, ..., t_k$, where $t_1 \leq t_2 \leq \cdots \leq t_k$, and let $t_0 = 0$ and $t_{k+1} = \bar{\mu}$. Then for $j = 0, 1, ..., k$, the expanding search P_j finds j objects after time t_j, and after further time $\bar{\mu} - t_{j+1}$ will have found all the remaining objects (in fact the remaining objects may be found by an earlier point in time). So the total time taken for P_j to find all the objects is no greater than $t_j + (\bar{\mu} - t_{j+1})$. Since P_* is an equiprobable choice of

the P_j, the total expected search time $T(P_*, H)$ satisfies

$$\begin{aligned} T(P_*, H) &\leq \frac{1}{k+1} \sum_{j=0}^{k} t_j + (\bar{\mu} - t_{j+1}) \\ &= \frac{1}{k+1} \left((k+1)\bar{\mu} - \sum_{j=0}^{k} (t_{j+1} - t_j) \right) \\ &= \frac{1}{k+1} \left((k+1)\bar{\mu} - \bar{\mu} \right) \\ &= \left(1 - \frac{1}{k+1} \right) \bar{\mu}. \end{aligned}$$

■

For the network in Figure 11, for $k = 3$, we have $(1-1/(k+1))\bar{\mu} = 5(3/4) = 3.75$ and $(1-1/(k+1))\mu = 4(3/4) = 3$, so Theorems 39 and 44 imply that the value of the smart search game for 3 objects on this network is between 3 and 3.75.

We can also define a mixed strategy for a normal Searcher on any network, giving us an upper bound for the expected search time in the normal game.

Definition 45 *Let P be an MRES on a network Q, and let I_j be the portion of this MRES that is traversed during the time interval $\left[\frac{j-1}{k}\bar{\mu}, \frac{j}{k}\bar{\mu}\right)$, $j = 1, ..., k$. There is an expanding search for which I_j is the last part of Q to be searched, and it is searched in the backward direction: namely the search which starts by following P until time $\frac{j-1}{k}\bar{\mu}$ and then follows P^{-1} for the remaining time. Similarly there is a search for which I_j is the last part of Q to be searched, and in the forward direction, which starts by following P^{-1} until time $\left(1 - \frac{j}{k}\right)\bar{\mu}$ and then follows P for the remaining time. The normal k-uniform MRES P^* is defined as the search strategy which makes an equiprobable choice between all $2k$*

expanding searches so described.

Theorem 46 *If a normal Searcher follows a normal k-uniform MRES P^*, he ensures an expected search time of no more than $\left(1 - \frac{1}{2k}\right)\bar{\mu}$.*

Proof. First note that if the MRES P doubles back on itself for any part of Q then we can simply add some more nodes and arcs to Q to create a new network which has the same $\bar{\mu}$ but in which the MRES never doubles back. To do this, we simply add a new arc whenever P starts to retrace some part of Q that has already been searched. It is clear that this is a disadvantage to the Searcher, so it will only increase the expected search time. So without loss of generality we can restrict the proof to Q for which the MRES traverses each arc of Q only once, so that $\bar{\mu} = \mu$.

By rescaling we can assume $\bar{\mu} = 1$ so each I_j has length $1/k$. Supposing that I_j is the last part of Q to be searched by P^*, it is equally likely to be searched in the forward direction (by P) as it is in the reverse direction (by P^{-1}). It follows that if I_j contains just one object then the expected search time will be $1 - (1/2)(1/k)$. If I_j contains no objects then by the time I_j is searched all the objects will have been found and the search time is no more than $1 - 1/k$. If I_j contains more than one object, then the search time may be as great as 1. However, as the number of I_j containing two or more objects must be no greater than the number of I_j containing no objects, the expected search time is bounded by the average of $1 - 1/k$ and 1, which is $1 - (1/2)(1/k)$. ■

For the network in Figure 11, for $k = 3$, we have $\bar{\mu}(1 - 1/(2k)) = 5(5/6) \approx 4.167$ so Theorems 39 and 46 imply that the value of the normal search game

for 3 objects on this network (if it exists) is between 3 and 4.167.

When $k = 1$, the two lower bounds given in Theorems 44 and 46 are both equal to $\bar{\mu}/2$. This bound for the expanding search game with one hidden object can be found in Alpern and Lidbetter [14], and our results therefore both generalise this bound.

5.4 Search for k objects on a 2-arc-connected network

In this section, we start by examining when the two bounds given in Theorems 39 and 44 are equal. That is, when $\mu(Q) = \bar{\mu}(Q)$. For this equality to hold, we must be able to find an MRES on Q which doesn't 'double back' on itself. We will see that this is possible precisely for 2-arc connected networks, so that the value $\tilde{V}_k(Q, O)$ of the smart search game played on such networks is $\mu(1 - 1/(k+1))$. Recall that a network is 2-arc-connected if and only if it cannot be disconnected by the removal of fewer than 2 arcs. We also show that the bounds given by Theorem 46 is tight if Q is a circle.

We will need the following characterisation of 2-arc-connected networks due to Robbins [47] in terms of *orientable networks*, those for which the arcs can be oriented in such a way so that there is a directed path from any point x to any other point y in Q.

Theorem 47 *The following are equivalent for a network Q :*

(i) *Q is 2-arc connected.*

(ii) *Q is orientable.*

(iii) *There is an increasing sequence of subnetworks* $Q_0, Q_1, \ldots, Q_k = Q$, *such that* Q_0 *is a cycle and each* Q_i, $i > 0$, *is obtained from* Q_{i-1} *by adding a path between some two points* x_i *and* y_i *of* Q_{i-1}. *(This sequence is called an ear decomposition.)*

In his elegant paper, Robbins explained his result in terms of one-way streets and robustness in terms of repairs on a given street. It follows from his proof that (iii) can be extended to say that Q_0 can be chosen to contain any given point, in our case the root of Q.

Our motivation for what follows is the proof of Gal [21] that $V^p = \mu/2$ for Eulerian networks. He simply takes any Eulerian tour of Q, equiprobably with its reverse tour, as the Searcher mixed strategy. For expanding pointwise searches $P(t)$, unlike pathwise searches, the reverse function $P(\mu - t)$ may not be a pointwise (expanding) search. In order to adapt Gal's idea to the expanding search context we need to assume that the pointwise search $P : [0, \mu] \to Q$ is *reversible*, by which we mean that $P^{-1}(t) = P(\mu - t)$ is a pointwise expanding search (see Section 5.2). Note in particular that a reversible pointwise search must end at $P(\mu) = P^{-1}(0) = O$, the root.

Theorem 48 *A network* Q *is* 2*-arc connected if and only if it has a reversible combinatorial search.*

Proof. First suppose Q has a reversible combinatorial search. Suppose an arc a is traversed between times t_1 and t_2, that is, $a = P(\{t : t_1 < t < t_2\})$. Then $Q - a = P([0, t_1)) \cup P^{-1}([0, \mu - t_2))$ is the union of connected sets with a

common point $O = P(0) = P^{-1}(0)$ so it must be connected. Hence Q is 2-arc connected.

Now suppose Q is 2-arc connected. Then by Theorem 21 it has an ear decomposition starting with a cycle, $Q_0, Q_1, ..., Q_k$. We can assume that the cycle, Q_0 includes O. We construct reversible combinatorial searches P_i on Q_i inductively, where the P_i are sequences of arcs, with no arcs repeated. Let S_0 be the cycle on Q_0 starting at O. This is clearly a reversible combinatorial search on Q_0. Assume we have constructed P_i, a reversible pointwise search on Q_i, for $1 \leq i < k$. We have $Q_{i+1} = Q_i \cup A_i$, where A_i is a path from a node $x \in Q_i$ to a node $y \in Q_i$. We can assume that x occurs before y in P_i, otherwise we can relabel the nodes. Let P_{i+1} be the combinatorial search on Q_{i+1} which follows P_i until reaching x, then follows the path A_i from x to y, and finally follows the remainder of P_i from x to O. Then P_{i+1}^{-1} consists of the following three expanding arc sequences: firstly the path along P_i^{-1} from O to x, next the path along A_i^{-1} from y to x, and finally the path along P^{-1} from x to O. Each of these expanding arc sequences starts from a point that has already been reached by P_{i+1}^{-1}, so P_{i+1}^{-1} is a combinatorial expanding search, and P_{i+1} is reversible, as required. ■

We can now give the solution of the smart search game $\tilde{\Lambda}_k(Q, O)$ for 2-arc connected networks, which generalises the analogous theorem for a single hidden object given in [14].

Theorem 49 *If Q is 2-arc-connected then the value of the smart search game $\tilde{\Lambda}_k(Q, O)$ is $\tilde{V}_k(Q, O) = \mu\left(1 - \frac{1}{k+1}\right)$. The k-uniform strategy u_k is optimal for*

the Hider, and the smart k-uniform MRES, P_* is optimal for the Searcher.

Proof. If the Hider follows u_k, then by Theorem 39 he ensures an expected search time of at least $\mu\left(1-\frac{1}{k+1}\right)$, so $\tilde{V} \geq \mu\left(1-\frac{1}{k+1}\right)$. If the Searcher follows P_*, then by Theorem 44 he ensures an expected search time of at most $\bar{\mu}\left(1-\frac{1}{k+1}\right)$, so $\tilde{V} \leq \bar{\mu}\left(1-\frac{1}{k+1}\right)$. Since Q is 2-arc-connected, $\mu = \bar{\mu}$, by Theorem 48, so $\tilde{V} \leq \mu\left(1-\frac{1}{k+1}\right)$, and we must have equality. ■

For a normal Searcher, Theorem 46 and Theorem 48 imply that if Q is 2-arc-connected, the value of the game is no more than $\mu(1-1/(2k))$. We sum this up in the Theorem below.

Theorem 50 *If Q is 2-arc connected, the value of the normal search game $\Lambda_k(Q,O)$ satisfies $V_k(Q,O) \leq \mu\left(1-\frac{1}{2k}\right)$.*

This bound is tight in the case that Q is a circle, as we now show. We denote the circle C by the interval $[0,\mu]$, identifying the points 0 and μ, which is the root.

Theorem 51 *The value normal search game $\Lambda_k(C,O)$ is $V_k(C,O) = \mu\left(1-\frac{1}{2k}\right)$. The normal k-uniform MRES P^* is optimal for the Searcher. It is optimal for the Hider to follow the strategy h in which he picks some x uniformly from the interval $[0,\mu/k)$ and hides the objects at the points $\{x, x+\mu/k, x+2\mu/k, ..., x+(k-1)\mu/k\}$.*

Proof. By Theorem 50, $V(C,k) \leq \mu\left(1-\frac{1}{2k}\right)$ since C is 2-arc-connected, so it remains to be shown that the Hider's strategy h ensures an expected search

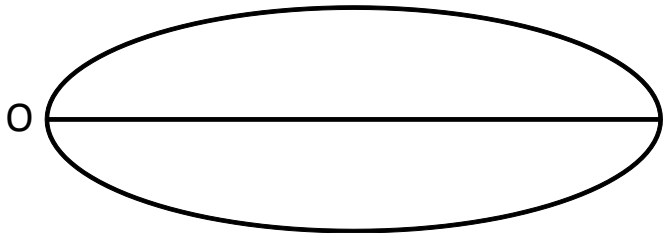

Figure 12: The three-arc network.

time no greater than $\mu\left(1-\frac{1}{2k}\right)$. For any normal Searcher strategy S, $k-1$ of the objects will be found be S by time $\mu\left(1-1/k\right)$, and the whole of C will have been searched except for an interval $I=[a,a+\mu/k)$, for some $a\leq\mu\left(1-1/k\right)$. The remaining object that has not been found is located uniformly on I, so any search will find it in additional time $\frac{\mu}{2k}$. The total search time is therefore $\mu\left(1-\frac{1}{k}\right)+\frac{\mu}{2k}=\mu\left(1-\frac{1}{2k}\right)$. This completes the proof. ■

The bound in Theorem 50 is not tight for general 2-arc-connected networks, as demonstrated by the three-arc network Q depicted in Figure 12 which consists of 3 arcs of length 1 (also discussed in the Introduction).

Consider the expanding search game $\Lambda_2(Q,O)$ with 2 hiders played on Q with a normal Searcher. Theorem 50 gives the bound $V(Q,2)\leq 3(1-1/4)=9/4$. However, the Searcher can improve on this bound. Consider the Searcher strategy where he picks two arcs at random and uses his optimal strategy for the circle with $k=2$ on this subnetwork, then uses his strategy for $k=1$ on the remaining arc, which can now be regarded as a circle.

Suppose the Hider places his two objects on different arcs. If the Searcher chooses these arcs first (probability 1/3), then by Theorem 51, the expected

capture time will be $2(1-1/4) = 3/2$. If he chooses a different pair of arcs first (probability 2/3), then again, by Theorem 51, the expected capture time will be $2+1(1-1/2) = 5/2$. So the overall expected capture time is $1/3 \cdot 3/2 + 2/3 \cdot 5/2 = 13/6$.

Now suppose the Hider places both objects on the same arc. If one of the first two arcs the Searcher chooses is this one (probability 2/3), the expected capture time is $2(1-1/4) = 3/2$. If the Searcher chooses the two other arcs first (probability 1/3), then the expected capture time is no greater than 3. So the overall expected capture time is no greater than $2/3 \cdot 3/2 + 1/3 \cdot 3 = 2 < 13/6$. So $V(Q, 2) \leq 13/6 < 9/4$.

It can be shown that $V(Q, 2) = 13/6$, as the Hider can ensure expected capture time at least 13/6 by picking two arcs at random and using his optimal strategy for the circle with $k = 2$ on this subnetwork. A detailed proof is omitted, but it is sufficient to show that the Searcher has a best response to this Hider strategy which begins by searching only two of the arcs. If the Searcher uses a strategy of this type, then given he finds the final object before time 2, the expected capture time is 3/2; given he finds the final object after time 2, the expected capture time is 5/2. Hence the overall expected capture time is precisely $1/3 \cdot 3/2 + 2/3 \cdot 5/2 = 13/6$.

6 The nut caching game

In this final chapter we consider a *caching game* between a Squirrel and a Pilferer that shares some similarities with the expanding search game for multiple hidden objects. Suppose a squirrel has m nuts which he can bury at n possible caching sites. He has enough energy to dig a given total depth, which we normalise to 1, so the sum of the depths reached at all the sites must be equal to 1. Thus a Squirrel strategy specifies how he places his m nuts at various depths among the n caching sites, subject to his energy (digging) constraint. He has the option of placing nuts at different depths at the same site, although this may not at first glance appear to be a good idea. The squirrel needs $k < m$ of them to survive the winter ('win'), so he wishes to hide them in such a way as to maximise the probability that k nuts remain after pilfering. We can think of a Squirrel strategy as a set of m points on a star network with n arcs of length 1, subject to the constraint that the sum of the distances of the furthest points on each arc from the root is 1.

After the Squirrel places his nuts, the Pilferer arrives. He knows the location of the n sites but not which ones have nuts or at what depths they are hidden. He can dig to different depths at the sites, subject to his own energy constraint that the total depth (summed over all sites) does not exceed some constant D. He wishes to minimise the probability that k nuts remain after pilfering. We can think of a Pilferer strategy as an expanding search terminating after time D on a star network with n arcs of length 1. As in previous chapters, A *smart* Pilferer can adapt his digging strategy as he goes along; a *normal* Pilferer can

just choose the depths at each site. In this chapter we will analyse both cases. An empirical question raised by this distinction is whether Pilferers switch sites when they find a cache, or continue to look for a deeper cache at the same site. We assume throughout that the Pilferer can only detect a nut visually, when the earth above it has been removed.

The payoff of the game is 1 if the Squirrel survives and therefore wins (that is, there are at least k undiscovered nuts) and 0 otherwise. We denote the game with a normal Searcher by $\Phi_k(n, m, D)$ and with a smart Searcher by $\tilde{\Phi}_k(n, m, D)$, with the corresponding values denoted by $V = V_k(n, m, D)$ and $\tilde{V} = \tilde{V}_k(n, m, D)$. Note that the values always satisfy $V \geq \tilde{V}$. In [7] the existence of the value is proved for the case of a normal Pilferer, but not for a smart Pilferer. However, for the examples considered in this chapter, we will prove the value exists by giving optimal strategies for the players.

This caching game is a modification of so called *accumulation games*, studied in [32], [33], [34] and [17], and more generally of *geometric games* [48]. In general, in accumulation games the Hider repeatedly adds material (corresponding to our nuts) to the hiding sites as they are pilfered over time. However, unlike in our model, *depth* of caching is not usually considered. This chapter uses some material from [9]. Further results can be found in [7].

6.1 Optimal strategies with smart Pilferers, $m = n = 2$, $k = 1$

We begin by analysing the simple case where the squirrel has to hide two nuts at two sites and survives (wins) if at least one nut remains after pilfering (so the parameter values are $m = n = 2$ and $k = 1$). Here we assume the Pilferer is smart, as defined above, in that he can alter his digging depending on what he does or does not find up to a given time.

Note that if $D < 1$, the squirrel can always win (so $V = 1$) by hiding both nuts together in one site at depth 1. If $D \geq 2$, and the pilferer digs to depth 1 at both sites, any nuts placed by the squirrel will be lost. Thus there cannot be any strategy for the squirrel giving him a positive probability of surviving, and hence in this case V is 0. So to exclude these trivial cases we assume that $1 \leq D < 2$. The solution of this problem splits into two cases:

Proposition 52 *If the pilferer's digging depth constraint D satisfies $1\frac{1}{2} \leq D < 2$, then it is optimal for the squirrel to place both nuts at maximum depth 1 at a random site. The value $\tilde{V}$ of the game is $1/2$.*

Proof. Suppose the squirrel places both nuts at depth 1 at a site chosen randomly. Then since $D < 2$ the pilferer cannot dig to depth 1 at both sites, and so if he guesses wrongly (which has probability $1/2$) he will dig at the wrong site and the squirrel will survive. This squirrel strategy thus guarantees a survival probability of $1/2$, so that $\tilde{V} \geq 1/2$.

The pilferer can guarantee the squirrel will win with probability no more than

1/2 by digging to depth 1 at one site and to depth 1/2 at the other (choosing equiprobably between the sites). It is then easy to verify that however the nuts are buried, the pilferer will find them both at least 1/2 the time. To see this, note that the squirrel can plant a nut at depth greater than 1/2 at only one location, and if this is the location where the pilferer digs to depth 1 the squirrel loses. Since this occurs with probability 1/2, we have established the claim that $\tilde{V} \leq 1/2$.

Combining the bounds on $\tilde{V}$ demonstrated in the previous two paragraphs shows that $\tilde{V} = \frac{1}{2}$, as claimed in the proposition. ■

The second case is as follows.

Proposition 53 *If the pilferer's digging depth constraint D satisfies $1 \leq D < 1\frac{1}{2}$, then it is optimal for the squirrel to hide his two nuts at depths 1/2 and 1 at a random location, with probability 2/3; and at depth 1/2 at both locations, with probability 1/3. (The three equiprobable configurations are shown below in Figure 13, where S_1 and S_2 denote the two sites.) In this case the value $\tilde{V}$ of the game is 2/3.*

Proof. Suppose that the squirrel hides his two nuts at depths 1/2 or 1 in one of the configurations shown in Figure 13, with equal probabilities of 1/3 for each. It is easy to see that the pilferer's digging constraint $D < 1\frac{1}{2}$ prevents him from finding both nuts in more than one of these three hiding configurations: if the pilferer digs to depth 1 at one site, he will win in one of the configurations a) or c) but in neither of the others; otherwise, he will only be able to win in

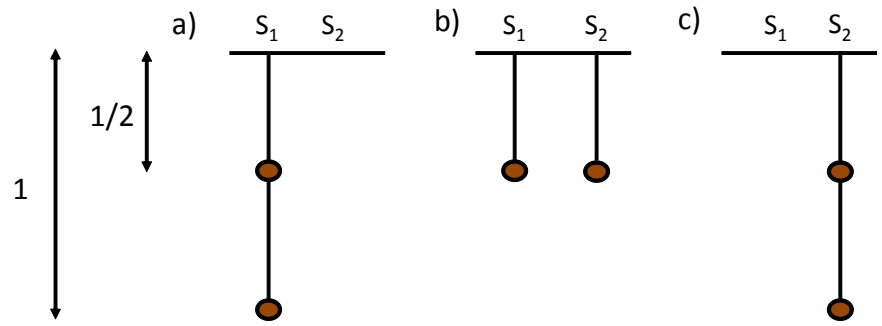

Figure 13: Three caching strategies of the squirrel.

configuration b). Consequently $\tilde{V} \geq 2/3$.

Suppose that the pilferer adopts the following digging strategy: he guesses equiprobably how the nuts are distributed between the three sites. In other words, with equal probability he chooses $(i, j) = (2, 0)$, $(1, 1)$ or $(0, 2)$. He then digs in site S_1 till he finds i nuts and digs in site S_2 till he finds j nuts. If the Pilferer guesses the correct way the nuts are distributed, he will be sure to find both nuts, and since he guesses correctly with probability $1/3$, this guarantees $\tilde{V} \leq 2/3$.

Combining the two bounds on $\tilde{V}$, we have $\tilde{V} = 2/3$. ■

The strategy drawn in Figure 13 for the squirrel has the unusual property of nuts at different levels at the same site. Why can't the squirrel improve (or at least do as well) by changing the placement in cases a) and c) to putting both nuts at the bottom? To see why this does not work, suppose that the pilferer always uses the switch strategy. Then the squirrel loses *always* when he adopts b) and loses half the time when he adopts a) or c). So if, as before, he

adopts all three with probability 1/3, he wins (survives) with probability only $(1/2) \cdot (2/3) = 1/3$ which is worse than 2/3.

6.2 Optimal strategies with normal Pilferers, $m = n = 2$, $k = 1$

Note that in the last proof, the pilferer's strategy requires that he is *smart*, in the sense we defined in the introduction. Suppose, on the other hand, that the pilferer is *normal*, and adopts a *simple strategy* that picks two depths d_1 and $d_2 = D - d_1$ for the sites, which will be dug without reference to what is found. It is clear that in this version every squirrel strategy is dominated by one which places the nuts at the two sites with respective depths s_1 and $s_2 = 1 - s_1$, and there is no point in placing nuts at different depths at the same site. For this reason, it is also clear that the solution of the game played with $m > 2$ nuts is the same as the solution for $m = 2$.

We now show that restricting the pilferer to simple strategies does not help the squirrel when $D \geq 1\frac{1}{3}$, but it does indeed help him when $D < 1\frac{1}{3}$. For example, when $D = 5/4$, $\tilde{V} = 2/3$ but $V = 3/4$, as given by the result below.

Proposition 54 *Let q be a positive integer and let the pilferer's total digging depth D satisfy $1 + \frac{1}{q+1} \leq D < 1 + \frac{1}{q}$. Assume the pilferer is normal, as defined above. Then the optimal squirrel strategy is to pick i randomly from the $q + 1$ values $0, 1, \ldots, q$ and to bury his two nuts at respective depths i/q and $(q - i)/q$ at the two sites, in random order. The optimal strategy for the pilferer is to pick $d_1 = \frac{j}{q+1}$ with $j = 1, 2, \ldots, q + 1$ chosen equiprobably among the $q + 1$*

possibilities. The value of the game is $V = 1 - \frac{1}{q+1}$*, as graphed in Figure 14 against* D*.*

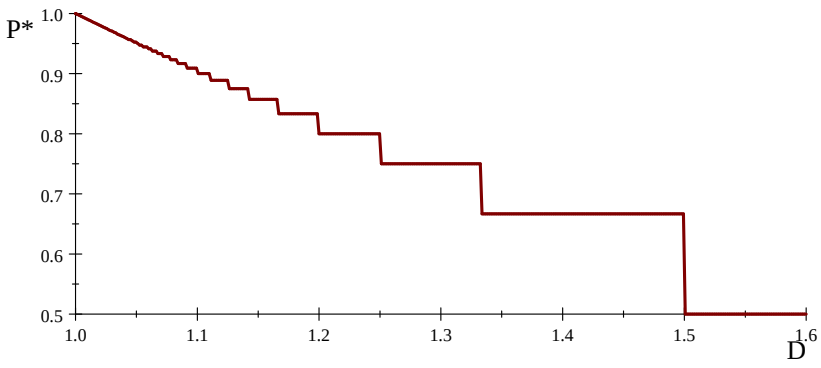

Figure 14: The value V of the smart game for $1 \le D \le 2$.

Proof. Consider the squirrel mixed strategy S^q which picks $s_1 = \frac{i}{q}$ with the $q+1$ values $i = 0, \ldots, q$ taken equiprobably. Suppose a pilferer strategy $d = (d_1, d_2)$ wins against more than one of the squirrel strategies. Then it is clear it wins against two consecutive ones, say i and $i+1$. Then we must have

$$d_1 \ge \frac{i+1}{q} \text{ and } d_2 \ge \frac{q-i}{q}, \text{ so } D = d_1 + d_2 \ge 1 + \frac{1}{q},$$

which is larger than we are allowing. So all but one of the strategies i must win, and hence $V \ge \frac{q}{q+1}$.

Next consider the pilferer simple mixed strategy of $d_1 = \frac{j}{q+1}$ with $j = 1, 2, \ldots, q+1$ chosen equiprobably among the $q+1$ possibilities. If a squirrel strategy s_1 wins against all of these, then for every j we have either

$$s_1 > \frac{j}{q+1} \text{ or } 1 - s_1 > D - \frac{j}{q+1}.$$

This means that for all j,

$$s_1 \text{ is not in the interval } I_j = \left[\frac{j}{q+1} + (1-D), \frac{j}{q+1}\right].$$

This is equivalent to saying that the intervals I_j cannot overlap, or that

$$D - 1 < \frac{1}{q+1}, \text{ which is the same as } D < 1 + \frac{1}{q+1},$$

contrary to our assumption. So at least one of the $q+1$ strategies $d_1 = \frac{j}{q+1}$ gets both nuts, and hence $V \leq \frac{q}{q+1} = 1 - \frac{1}{q+1}$. The result follows by combining the two estimates. ■

6.3 Some solutions for arbitrary n and $k = 1$

For $n > 2$ the game is hard to solve. Even for $n = 3$ it is non-trivial, and some results in this case are presented in [7]. Here we give some results for arbitrary n. The first two propositions cover the cases that D is very large or very small.

Proposition 55 *Suppose $D \geq n - 1/2$. Then for $k = 1$ and any m, it is optimal for the Squirrel to hide all the nuts at depth 1 at a randomly chosen site; it is optimal for the Pilferer to dig a hole of depth $1/2$ in a randomly chosen site and holes of depth 1 in all other sites. The value of the game is $1/n$. This holds whether the Pilferer is smart or normal.*

Proof. The Squirrel's strategy clearly ensures that the value is at least $1/n$ since the Pilferer can check $n-1$ out of the n sites (whether he is smart or normal), so will fail to find the nuts with probability $1/n$. Suppose the Pilferer uses the strategy described, and chooses some site S_i at which to dig a hole of

depth $1/2$. The Pilferer fails to find all the nuts only if S_i contains a nut at depth greater than $1/2$. But at most one site can have a nut buried at greater depth than $1/2$, so this happens with probability less than $1/n$, so the value is at most $1/n$.

Hence the value is $1/n$. Notice that the Pilferer strategy is normal, so this is the value of the game whether the Pilferer is smart or normal. ■

We now consider the case that D is very small. In order to describe the optimal strategies, we first define an *allocation* of the m nuts to the n sites as a way of distributing the nuts amongst the sites. More precisely:

Definition 56 *An **allocation** of m nuts to n sites is a vector of non-negative integers $(m_1, ..., m_n)$ with $\sum_{i=1}^{n} m_i = m$. The parameter m_i is the number of nuts in site i.*

Every Squirrel strategy corresponds to an allocation, and for every allocation we define a unique strategy for the Squirrel which we call his *allocation strategy*: at site S_i, hide m_i nuts at depths $1/m, 2/m, ..., m_i/m$. Clearly the total depth dug by the Squirrel is $\sum_{i=1}^{n} m_i/m = 1$. For every allocation we also define a unique *allocation strategy* for a smart Pilferer: for each i, dig in site S_i till finding m_i nuts. It is non-trivial to calculate the total number of allocations, as we now demonstrate.

Lemma 57 *The number of allocations is $\binom{n+m-1}{m}$.*

Proof. For a given allocation $(m_1, ..., m_n)$, we define a path on a $(n-1) \times m$ lattice from the bottom left corner $(0, 0)$ to the top right corner $(n-1, m)$. The

path begins by taking m_1 steps up then one step right. This is followed by m_2 steps up then another step right. The path continues in this fashion. We can write this as the following algorithm:

1. Take m_1 steps up. Set $j = 2$.
2. Take one step right then m_j steps up.
3. Increase j by 1.
4. If $j < n$, go to step 2, otherwise stop.

Clearly every allocation corresponds to a unique path from $(0, 0)$ to $(n-1, m)$ and every such path corresponds to a unique allocation. Hence the number of allocations is equal to the number of paths, and the number of paths is easily seen to be $\binom{n+m-1}{m}$ since there are $n+m-1$ steps in total and m of them must be in the upwards direction. ■

The following Theorem is a consequence.

Theorem 58 *If $D < 1 + \frac{1}{m}$ and $k = 1$ then it is optimal for the both the Squirrel and the (smart) Pilferer to choose uniformly randomly between all their allocation strategies. The value of the game is*

$$1 - \frac{1}{\binom{n+m-1}{m}}$$

Proof. If the Squirrel uses his allocation strategy then the Pilferer clearly does not have enough digging resources to check more than one of the Squirrel's configurations of nuts. By Lemma 57 there are $\binom{n+m-1}{m}$ such configurations, so

the value is at least $1 - 1/\binom{n+m-1}{m}$. If the Pilferer uses his allocation strategy, then with probability $1/\binom{n+m-1}{m}$ he guesses the correct allocation and thereby finds all the nuts so the value is at most $1 - 1/\binom{n+m-1}{m}$, and hence we have equality. ■

6.4 Some results for arbitrary k

We now show that the value of the game $\Phi_k(n, m, D)$ for arbitrary k can be bounded below by the value of $\Phi_1(n, m', D)$, for some value of m', and analogously for the smart search game. We then show that this bound is tight for one particular example.

We first define a Squirrel strategy where he hides the nuts in *batches*. More precisely, for a given Squirrel pure strategy for the game $\Phi_1(n, m', D)$, where $m = m'k+r$, and $r < k$, we can define a corresponding Squirrel pure strategy for the game $\Phi_k(n, m, D)$ where a nut hidden in site S_i at depth x in $\Phi_1(n, m', D)$ corresponds to k nuts hidden at in site S_i at depth x in $\Phi_k(n, m, D)$. The remaining r nuts are all added to one of the existing batches. The resulting configuration is of $m' - 1$ batches of k nuts all in the same place, and one batch of $k+r$ nuts. For example, suppose $m = 20$ and $k = 3$, so that $m' = 6$ and $r = 2$. Then there will be 5 batches of size 3 and 1 batches of size 5. Corresponding mixed strategies are defined in the natural way. In order to win, the Pilfer needs to find all the batches. The following lemma is a consequence.

Lemma 59 *The value $V_k(n, m, D)$ of the game $\Phi_k(n, m, D)$ and the value $\tilde{V}_k(n, m, D)$*

of the game $\tilde{\Phi}_k(n, m, D)$ *satisfy*

$$\begin{aligned} V_k(n, m, D) &\geq V_1(n, m', D) \text{ and} \\ \tilde{V}_k(n, m, D) &\geq \tilde{V}_1(n, m', D), \end{aligned}$$

where $m = m'k + r$ *and* $r < k$.

We show that this bound is tight for $n = 2$ for at least one choice of parameters. As already noted, if the Pilferer is normal, then for $n = 2$ and $m > 2$ the game simply reduces to the analogous game where $m = 2$, so we consider only the case of a smart Pilferer.

Theorem 60 *If* $n = 2$, $m = m'k + r$, *where* $r < k$, *and* $D < 1 + 1/m'$, *then the value* $\tilde{V}_k(n, m, D)$ *of the smart search game* $\tilde{\Phi}_k(n, m, D)$ *is*

$$\tilde{V}_k(n, m) = 1 - \frac{1}{\binom{n+m'-1}{m'}} = 1 - \frac{1}{1 + m'}.$$

It is optimal for the Squirrel to hide the nuts in batches of size k *and* $k + r$ *according to his optimal strategy in* $\tilde{V}_k(n, m')$. *The optimal strategy for the Pilferer is to choose a number* j *uniformly from the set* $\{0, 1, ..., m'\}$, *and search for* jk *nuts in the first site before spending his remaining energy searching in the second site.*

Proof. By Lemma 59 and the paragraph preceding it, if the Squirrel uses his optimal strategy for $\tilde{\Phi}_k(n, m')$ hiding the nuts in batches, then he ensures that the value $\tilde{V}_k(n, m)$ is at least $\tilde{V}_k(n, m') = 1 - 1/(m' + 1)$.

Suppose the Pilferer uses the strategy described and there are t and $m - t$ nuts in the two sites S_1 and S_2, respectively, with $t = qk + s$, where $s < k$ and

$0 \leq q \leq m$. We will show that one of the Pilferer's $m' + 1$ equiprobable choices of pure strategy wins the game, so that he wins with probability $1/(m' + 1)$. The strategy he needs corresponds to picking $j = q$. In this case, the Pilferer finds qk nuts in the first site all $m - t$ nuts in the second site, making a total of $qk + (m - t) = qk + m - qk - s = m - s > m - k$ nuts. Thus the Squirrel is left with fewer than k nuts and loses the game. This shows that the value $\tilde{V}_k(n, m)$ is at most $1 - 1/(m' + 1)$, and we must therefore have equality. ■

7 Conclusion

In this thesis we began in Chapter 2 by analysing a search game played between a Hider and a Searcher on a variable speed network. We gave an explicit solution for the game played on trees, improving over the recursive approach given by Alpern [4] and generalising a classic result of Gal [21]. We then showed how the solution could be applied to related games including Kikuta's game with search costs [30] and Alpern's find-and-fetch game [5]. We also solved the game for some simple networks that are not trees, but we note that even in Gal's classic time-symmetric model of network search, the general solution in unknown. Future work could study how the analogous problem of rendezvous search (see [3],[6]) could be also be investigated on time-asymmetric networks.

In Chapter 3 we defined the notion of expanding search on a network, and solved the problem of how to find the optimal expanding search for a Hider located on the nodes of a tree according to a known probability distribution. We then saw how the theory of search games on variable speed networks developed in Chapter 2 could be applied to the expanding search game on a tree. The expanding search game could also be reformulated and studied from the point of view of rendezvous search.

We then extended the notion of the expanding search game in Chapter 4 to allow the Hider to hide several objects on a network, first analysing the case of a star network, and viewing this as a search game for k objects in n boxes. We distinguished between the cases where the Searcher is smart and normal, and solved both variations of the game. There are several simple extensions to

the game which merit further study. For example, the Searcher may wish to minimise the search cost incurred in finding some $j < k$ of the objects, or he may have a fixed budget, under which he wishes to maximise the number of objects he can find.

We turned in Chapter 5 to the more general expanding search game with multiple objects on a network, first giving a Hider strategy that generalises the Hider strategy given by Isaacs' [28] in his original formulation of search games. We then gave a solution (and upper bound) for the game played on 2-arc-connected networks for a smart (and respectively, normal) Searcher, generalising a result in [14]. There is scope for this game to be investigated on wider classes of networks.

Finally, in Chapter 6 we examined a caching game which extends the Kikuta-Ruckle accumulation games by adding a 'depth' element to the caching strategy. We solved the game when there are two caching sites, and gave some results for a general number of sites, again separating the cases of a smart and normal Searcher (or in this context, Pilferer). There are several ways in which this game could be extended to make it more realistic as a model of caching in nature. For example, imposing a spatial structure on the location of the caching sites or allowing the Squirrel to add material to the caches whilst the pilfering is taking place, as in the original formulation of accumulation games.

References

[1] S. Alpern. *Hide and Seek Games*, Seminar, Institut fur Hohere Studien, Wien, 26 July, 1976.

[2] S. Alpern. The Rendezvous Search Problem, *SIAM J. Control Optim.*, 33 (1995), 673-683.

[3] S. Alpern. Rendezvous search: a personal perspective. *Operations Research* 50 (2002):772-795.

[4] S. Alpern. Search games on trees with asymmetric travel times. SIAM J. Control Optim. 48 (2010), no. 8, 5547-5563.

[5] S. Alpern. Find-and-fetch search on a tree. Operations Research 59 (2011), 1258-1268.

[6] S. Alpern and A. Beck. Asymmetric rendezvous on the line is a double linear search problem. *Mathematics of Operations Research* 24 (1999), 604-618.

[7] S. Alpern, R. Fokkink, J. Op Den Kelder, and T. Lidbetter. *Disperse or unite? A mathematical model of coordinated attack*, Lecture notes in computer science, 6442 (2010), 220-233, ISSN 0302-9743.

[8] S. Alpern, R. Fokkink, and K. Kikuta. *On Ruckle's conjecture on accumulation games*, SIAM J. Control Optim., 48 (2010), 5073–5083.

[9] S. Alpern, R. Fokkink, T. Lidbetter, and N. S. Clayton. *A Search Game model of the Scatter Hoarder's problem*, J. R. Soc. Interface, 9 (2012), 869-879.

[10] S. Alpern and S. Gal. A Mixed Strategy Minimax Theorem without Compactness, *SIAM J. Control and Optim.* 26 (1988), 1357-1361.

[11] S. Alpern and S. Gal. *The Theory of Search Games and Rendezvous.* Kluwer International Series in Operations Research and Management Sciences, 319 pp, Kluwer, Boston (2003).

[12] S. Alpern and J. V. Howard. Alternating search at two locations, *Dynamics and Control* 10 (2000), no. 4, 319-339.

[13] S. Alpern and T. Lidbetter. Searching a Variable Speed Network, Mathematics of Operations Research (in press). [Draft version in LSE Department of Management Working Papers, LSEOR 10.129. ISSN 2041-4668 (Online)]

[14] S. Alpern and T. Lidbetter. Mining Coal or Finding Terrorists: The Expanding Search Paradigm, *Operations Research* 61 (2013), no. 2, 265-279.

[15] E. J. Anderson and R. R. Weber. The rendezvous problem on discrete locations, *J. Appl. Probab.*, 28 (1990), 839–851.

[16] D. Assaf and S. Zamir. Continuous and discrete search for one of many objects, *Oper. Res. Lett.*, 6 (1987), 205-209.

[17] V. J. Baston, F. A. Bostock, T. S. Ferguson. The number hides game. Proc. Am. Math. Soc. 107 (1989) 437–447.

[18] V. J. Baston and F. A. Bostock. An evasion game on a finite tree. *SIAM J. Control Optim.* 28 (1990), no. 3, 671-677.

[19] V. J. Baston and K. Kikuta. Search games on networks with travelling and search costs and with arbitrary searcher starting points. *Networks* (in press).

[20] J. Bram. A 2-Player N-Region Search Game, Operations Evaluation Group, Office of Chief Naval Operations, Washington, D.C., OEG IRM-31 (AD 402 914) (1962).

[21] S. Gal. Search games with mobile and immobile hider. *SIAM J. Control Optim.* 17 (1979), 99-122.

[22] S. Gal. *Search Games*, Academic Press, New York (1980).

[23] S. Gal. On the optimality of a simple strategy for searching graphs. *Int. J. Game Theory* 29 (2000), 533-542.

[24] S. Gal. Search games. Wiley Encyclopedia of Operations Research and Management Sci. (James J. Cochran, ed.), Wiley (2011)

[25] A. Garnaev. *Search Games and Other Applications of Game Theory* (Lecture Notes in Economics and Mathematical Systems Vol. 485), Springer (2000).

[26] J. C. Gittins, K. Glazebrook, and R. R. Weber. *Multi-armed Bandit Allocation Indices*, 2nd ed., Wiley (2011).

[27] S. G. Hoggar. Chromatic polynomials and logarithmic concavity, *J. Comb. Theory Ser. B*, 16 (1974), 248-254.

[28] R. Isaacs. Differential Games, Wiley, New York (1965).

[29] A. Jotshi and R. Batta. Search for an immobile entity on a network. *European Journal of Operational Research* 191 (2008), no. 2, 347-359.

[30] K. Kikuta. A search game with travelling cost on a tree, *J. Oper. Res. Soc. Japan*, 38 (1995), no. 1, 70-88.

[31] K. Kikuta and W. Ruckle. Initial point search on weighted trees. *Naval Res. Logistics* 41 (1994), no. 6, 821-831.

[32] Kikuta, K. & Ruckle, W. Accumulation games. Part 1: Noisy search. J. Optim. Theory Appl. 94 (1997), 395–408.

[33] K. Kikuta and W. Ruckle. Continuous accumulation games in continuous regions, *J. Optim. Theory Appl.,*106 (2000) 581–601.

[34] K. Kikuta and W. Ruckle. Continuous accumulation games on discrete locations, *Naval Res. Logist.* 41 (2001), 821–831.

[35] T. Lidbetter. Search games with multiple hidden objects, *SIAM J. Control and Optim.* (in press)

[36] D. Matula. A periodic optimal search, *Am. Math. Man.* 71 (1964), 15–21.

[37] N. Megiddo, S. L. Hakimi, M. R. Garey, D. S. Johnson, and C. H. Papadimitriou. The complexity of searching a graph. *J. ACM* 35 (1988), 18-44.

[38] T. Nakai. A preemptive detection game, *J. Inf. Optim. Sci.*, 11 (1990), 1-15.

[39] M. F. Neuts. A Multistage Search Game, *J. SIAM,* 11 (1963), 502-507.

[40] L. Pavlovic. A search game on the union of graphs with immobile hider. *Naval Res. Logistics* 42 (1995), no. 8, 1177-1199.

[41] V. V. Pravosudov and N. S. Clayton. A test of the adaptive specialization hypothesis: Population differences in caching, memory, and the hippocampus in black-capped chickadees (Poecile atricapilla), *Behav. Neurosci.*, 116 (2002), 515-522.

[42] W. Press. Strong profiling is not mathematically optimal for discovering rare malfeasors, *Proc. Natl. Acad. Sci. USA*, 106 (2008), 1716-1719.

[43] J. H. Reijnierse. Games, graphs and algorithms. Ph.D Thesis, University of Nijmegen, The Netherlands (1995).

[44] J. H. Reijnierse and J. A. M. Potters. Search games with immobile hider. *Int. J. Game Theory* 21 (1993), 385-394.

[45] D. J. Reyniers. Coordinating two searchers for an object hidden on an interval. *Journal of the Operational Research Society* 46 (1995), 1386-1392.

[46] D. J. Reyniers. Coordinated search for an object hidden on the line. *European Journal of Operational Research* 95 (1996), 663-670.

[47] H. E. Robbins. A theorem on graphs, with an application to a problem on traffic control. *Amer. Math. Monthly* 46 (1939), 281–283.

[48] W. Ruckle. *Geometric games and their applications.* Boston, MA: Pitman Press (1983).

[49] W. H. Ruckle. *A Discrete Search Game*, Stochastic Games and Related Topics, eds. T. E. S. Raghaven, T. S. Ferguson, T. Parthasarathy, and O. J. Vrieze (Kluwer Academic Publishers, Boston, MA) (1991), 29-43.

[50] A. Sharlin. Optimal search for one of many objects hidden in two boxes. *Eur. J. Oper. Res.*, 32 (1987), 251-259.

[51] L. C. Thomas. Finding your kids when they are lost. *Journal of the Operational Research Society* 43 (1992), 637-639.

[52] R. R. Weber. Optimal symmetric rendezvous search on three locations, *Math. Oper. Res.,* 37 (2012), 111-122.

[53] M. L. Weitzman. Optimal search for the best alternative, *Econometrica*, 47 (1979), 641–654.