The expanding search ratio of a graph

Spyros Angelopoulos*, Christoph Dürr*, and Thomas Lidbetter**

*Sorbonne Universités, UPMC Univ Paris 06, CNRS, LIP6, Paris, France, [Spyros.Angelopoulos@lip6.fr, Christoph.Durr@lip6.fr]
**Department of Mathematics, London School of Economics, Houghton Street, London WC2A 2AE, t.r.lidbetter@lse.ac.uk

Abstract

We study the problem of searching for a hidden target in an environment that is modeled by an edge-weighted graph. A sequence of edges is chosen starting from a given *root* vertex such that each edge is adjacent to a previously chosen edge. This search paradigm, known as *expanding search* was recently introduced for modeling problems such as coal mining in which the cost of re-exploration is negligible. We define the *search ratio* of an expanding search as the maximum over all vertices of the ratio of the time taken to reach the vertex and the shortest-path cost to it from the root. Similar objectives have previously been studied in the context of conventional (pathwise) search.

In this paper we address algorithmic and computational issues of minimizing the search ratio over all expanding searches, for a variety of search environments, including general graphs, trees and star-like graphs. Our main results focus on the problem of finding the *randomized expanding search* with minimum *expected search ratio*, which is equivalent to solving a zero-sum game between a *Searcher* and a *Hider*. We solve these problems for certain classes of graphs, and obtain constant-factor approximations for others.

Keywords: Search games, randomized algorithms, competitive analysis, game theory

1 Introduction.

We consider the problem faced by a *Searcher* of locating a stationary target or *Hider* located at the vertex of a connected edge-weighted graph G. We interpret the weight of an edge as the time taken to search that edge. The search must start at a given vertex O called the *root* and consists of a sequence of edges chosen in such a way that every edge must be adjacent to some previous edge, so that the set of edges that have been searched at any point forms a connected subgraph of G. For a given search and a given vertex v at which the Hider is located, the *search time* of v is the first time an edge in the search is incident to v.

This paradigm of search, recently introduced in Alpern and Lidbetter [2] is known as *expanding search*, in contrast to the more usual search paradigm, referred to here as *pathwise search* in which a search corresponds to a walk in a graph. The expanding search paradigm is an appropriate model for situations in which the cost of "re-exploration" is negligible compared to the cost of searching, for example when mining coal: here digging into a new site is far more costly than moving the drill through an area that has already been dug. Another interpretation, described in detail in Alpern and Lidbetter [2], is that of a team of searchers splitting up in the search for a target, for example a bomb or terrorist unit.

We illustrate the concept of expanding search on a graph with an example. Consider the graph Q depicted in Figure 1 with root O; vertices A, B, C and D; and edges OA, OB, BC and BD of lengths 3, 2, 2 and 1, respectively. An example of an expanding search on Q, which we will denote by S, is the sequence of edges, OA, OB, BD, BC. Under S, the search time of vertex D is $3 + 2 + 1 = 6$.

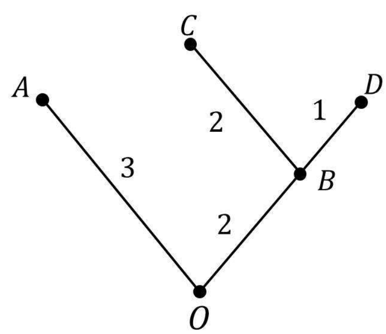

Figure 1: The graph Q.

Alpern and Lidbetter [2] take the approach of seeking randomized search strategies that minimize the expected search time in the worst case: that is, the maximum expected search time over all vertices. They also consider the problem of determining the search that minimizes some weighted average of the search times of the vertices. In this paper, we take an alternative approach by considering a normalized version of the search time obtained by dividing the search time of a vertex v by the length of the shortest path from O to v. For example, in the graph Q depicted in Figure 1, the normalized search time under S of vertex D is $6/3 = 2$ since D is at a distance of $2 + 1 = 3$ from the root. The maximum the normalized search time takes over all vertices of the graph is called the *search ratio*. In Q, the normalized search time of S is maximized at B, where it is equal to $(3 + 2)/2 = 2.5$, so this is the search ratio of S. This paper studies the problem of finding an expanding search with minimum search ratio.

Our choice of the search ratio as the objective for expanding search is motivated by earlier work by Koutsoupias et al. [30], who introduced this objective in the context of pathwise search. Their approach is analogous to the competitive analysis of online algorithms, in which the performance of an online algorithm is measured against the performance of an optimal offline algorithm; more precisely, the optimal offline algorithm corresponds to simply taking the shortest path to the target. As in Koutsoupias et al. [30], we consider not only deterministic searches, but also randomized searches, with the aim of finding the randomized expanding search that minimizes the expected value of the normalized search time in the worst case. Equivalently, we view this problem as a zero-sum game between a Hider who chooses a vertex of the graph and a Searcher who chooses an expanding search. The payoff, which the Hider seeks to maximize and the Searcher to minimize, is the normalized search time. This puts our work in the broader category of *search games*, a more general framework for games played between a Hider who chooses a point in some search space and a Searcher who makes some choice of how to navigate through the space with the aim of minimizing a given cost function.

It is worth mentioning that normalized cost formulations very similar to the search ratio have also been previously studied in the context of searching in unbounded domains (see, e.g., the work of Gal [20]). In such domains, the Hider can ensure the search time is arbitrarily large by choosing positions arbitrarily far from the root. This observation motivates the need for normalizing the search cost, which is accomplished by dividing this cost by the shortest-path cost from the root to the Hider.

In the spirit of the work of Koutsoupias et al. [30], in this paper we focus on computational and algorithmic issues of expanding search. We note that Alpern and Lidbetter [2] follow a purely mathematical approach to analyzing expanding search, with an emphasis on evaluating the value of the corresponding zero-sum games; computational and algorithmic issues are not considered. Table 1 illustrates the context of our work with respect to previous work. We note that the problem of minimizing the average search time of the vertices of a graph assuming the pathwise search formulation is precisely the well-known problem of minimizing the *latency* of a graph, also known as the *Traveling Repairman* problem (see [14, 24, 6, 38] for some representative results on this problem). The problem of choosing the randomized (pathwise) search that minimizes the *maximum expected search time* of points of a network was formalized by Gal [22], and has been extensively studied, as discussed in Subsection 1.1.

Table 1: Previous work and relations between search paradigms and objectives.

		Objective		
		Average search time	*Maximum expected search time*	*Search ratio*
Search paradigm	*Pathwise*	Min. Latency problem [14]	Gal's search game [22]	Searching a fixed graph [30]
	Expanding	Expanding search [2]	Expanding search [2]	This work

1.1 Related work.

Search theory (and its computational counterpart) has a very rich history of research. We give a summary of some results that are pertinent to this work.

Following the formalization of network search games by Gal [22] in the framework of pathwise search with un-normalized search time, the problem has had considerable attention, for example in Reijnierse and Potters [36], Pavlovic [35] and Gal [23]. In the latter work the solution of the game was found for all *weakly Eulerian* networks. Recent variations on Gal's original game include a setting in which the Searcher chooses his own starting point in Dagan and Gal [17] and Alpern et al. [1], and the setting in which the Hider is restricted to choosing vertices that have search costs in Baston and Kikuta, 2013 [9] and Baston and Kikuta, 2015 [10].

Expanding search was introduced by Alpern and Lidbetter [2] in the setting in which the payoff is the total (un-normalized) cost of finding the Hider. Among other results, Alpern and Lidbetter solved the game in the case that the network is either a tree or 2-edge-connected. This model was extended by Lidbetter [31] to a setting in which the Searcher must locate multiple hidden objects.

Much of the search games literature been purely mathematical, with less emphasis on issues of complexity, a notable exception being the work of von Stengel and Werchner [39]. The search ratio of pathwise search was studied in Koutsoupias et al. [30], who showed that the problem of computing the optimal search ratio in a given undirected graph is NP-complete (and MAX-SNP hard to approximate). They also gave a search strategy based on repeated executions of depth-first searches with geometrically increasing depths that achieves a constant approximation of the (deterministic) competitive ratio. Similar results can be obtained concerning the *randomized competitive ratio* (assuming that the Searcher randomizes over its strategy space). Connections between graph searching and other classic optimization problems such as the Traveling Salesman problem and the Minimum Latency problem were shown by Ausiello et al. [7]. The setting in which the search graph is revealed as the search progresses was studied by Fleischer et al. [19]. The latter also addressed connections between searching and exploring an environment, where the latter operation is defined as moving around the environment until all possible hiding positions are "visible" (the formal definition of visibility depends on the particular environment).

A specific search environment that has attracted considerable attention in the search literature is the *star-like* environment. More specifically, in the unbounded variant, the search domain consists of a set of infinite lines which have a common intersection point (the root of the Searcher); this problem is also known as *ray searching*. Ray searching is a natural generalization of the well-known *linear search* problem introduced independently by Beck [11] and Bellman [12] (informally called the "cow-path problem"). Optimal strategies were initially given by Gal [21] as well as by Baeza-Yates et al. [8] and Jaillet and Stafford [25]. Other related work includes the study of randomization by Schuierer [37] and Kao et al. [28]; multi-Searcher strategies by López-Ortiz and Schuierer [33]; searching with turn cost by Demaine et al. [18]; the variant in which some probabilistic information on target placement is known by Jaillet and Stafford [25] and Kao and Littman [26]; and the related problem of designing *hybrid algorithms* by Kao et al. [27].

Bounded star search, namely the case in which an upper bound is known on the distance of the target from the root was studied in López-Ortiz and Schuierer [32] and Bose et al. [15]. New performance measures that are applicable in the context of multi-target searching were introduced by Kirkpatrick [29] and McGregor, Onak and Panigrahy [34] (i.e., the setting in which there are more than one Hider and the Searcher must locate one of them). The problem of locating a certain number among the many Hiders was studied by Angelopoulos et al. [5].

It must be emphasized that star search has applications that are not necessarily confined to the concept of locating a target (which explains its significance and popularity). Indeed star search offers an abstraction that applies naturally in settings in which we seek an intelligent allocation of resources to tasks. More precisely, it captures decision-making aspects when the objective is to successfully complete at least one task, without knowing in advance the completion time of each task. Some concrete applications include drilling for oil in a number of different locations in McGregor et al. [34], as well as the design of algorithms that return acceptable solutions even if interrupted during their execution in Bernstein et al. [13] and Angelopoulos [3].

1.2 Contribution.

In this work we study expanding search under the search-ratio measure, assuming a variety of search spaces such as stars, trees, and general edge-weighted, undirected graphs. Our main motivation is to explore how the transition from pathwise to expanding search affects the deterministic and the randomized search ratios.

We begin in Section 2 with the definitions of the (expanding) search ratio and randomized search ratio. In Section 3 we show that the problem of finding the optimal (deterministic) search ratio is NP-hard (using a substantially more complicated reduction than for pathwise search in Koutsoupias et al. [30]). Applying well-known iterative deepening techniques, we obtain a $4\ln(4) \approx 5.55$ approximation.

Our first main technical results, presented in Section 4, apply to the setting where the graph is an unweighted graph or a tree. Here, it is easy to show that an optimal deterministic search strategy searches the vertices in non-decreasing order of distance from the root (and chooses the corresponding edges accordingly). This strategy is also a 2-approximation of the randomized search ratio. To see why the randomized search ratio might as little as half of the deterministic search ratio, suppose two vertices are at approximately the same distance from the root. Then it is possible that by using randomization, the expected search times of the vertices can be "smoothed out", which may decrease the randomized search ratio. Therefore, we define a randomized search strategy that approximates the randomized search ratio within a factor of 5/4, representing a significant improvement over the afore-mentioned 2-approximation. The idea of the strategy is to choose a subtree containing nodes within some randomly chosen radius of the root, search it, contract this subtree to the root, and repeat. The method of searching each of these subtrees is by what we call a *Random Depth-First Search*, which is an equiprobable choice of a depth-first search S and the depth-first search that arrives at the leaves of the trees in the reverse order to S. Thus vertices of the graph at a similar distance to each other are reached at roughly the same time, on average. Improved approximations via randomization are usually not easy to achieve (see, e.g. Koutsoupias et al. [30]). Our result confirms the intuitive expectation that randomization has significant benefits.

In Section 5 we consider the problem of finding the randomized search ratio of a star graph with n edges. In particular, we apply game-theoretic techniques to show that the ratio cannot exceed $(n+1)/2$, with equality if and only if the star is uniform (all the edges have the same length). This bound is deceptively difficult to prove, and we do so by exhibiting a randomized search strategy s_n whose (randomized) search ratio is no greater than $(n+1)/2$. Necessarily, s_n must be optimal on the uniform star, for which it is optimal to search the vertices in a uniformly randomly order; but it must also be optimal on stars whose edge lengths increase very rapidly, for which it is roughly optimal to simply search the edges in increasing order of length. Therefore, s_n is defined recursively in such a way that if the nth edge has roughly the same length as those preceding it, it is searched at a random point during the search s_{n-1} and if it is a lot longer than those preceding it, it is searched after the other edges. As a bonus, we find that s_n is in fact optimal for the special case that the lengths of the edges do not "increase too quickly". All the results we give for star graphs have analogous counterparts in the setting of pathwise search.

As argued earlier, star-search problems have applications that transcend searching. This is indeed the case in expanding search. Consider the following problem: we are given a collection of n boxes, among which only one contains a prize. We can open a box i at cost d_i. We seek a (randomized) strategy for locating the prize, and the randomized search ratio of the strategy is the total expected cost of all opened boxes, divided by the cost of the box that holds the prize. This problem is equivalent to the problem of finding the (randomized) search ratio of a star graph. We thus obtain a 5/4-approximation (as a corollary to our result on weighted trees) and we can further argue that a tight upper bound for the randomized search ratio for this problem is $(n+1)/2$.

We may also interpret expanding search on a tree as the scheduling of jobs with precedence constraints, where each vertex v of the tree corresponds to a job whose processing time is the length of the edge immediately preceding v, and all jobs on the path from v to the root must be executed before v can be executed. An expanding search corresponds to a feasible schedule, and the distance of v from the root corresponds to the minimum possible completion time of v over all choices of schedule, which we can interpret as the offline cost of completing v. We can then consider the problem of choosing a schedule to minimize the maximum ratio of the completion time of a job to its "offline cost". This is exactly our expanding search problem.

Since our main objective is to study the algorithmic and computational impact of *re-exploration* due to the transition from pathwise search to expanding search, it is important to compare our results to the best-known bounds in the context of pathwise search. More precisely, for unweighted graphs, [30] gives asymptotic approximations of the deterministic and randomized search ratios equal to 6 and 8.98, respectively, but its techniques appear to be applicable also to general graphs, at the expense of somewhat larger, but nevertheless constant approximations. Furthermore, Koutsoupias et al. [30] note that the problems of computing the search ratios of trees are "surprisingly hard". In contrast, for expanding search of unweighted graphs and (weighted) trees we obtain optimal algorithms and a 5/4 (asymptotic) approximation of the deterministic and randomized search ratios, respectively. We thus demonstrate that the transition from pathwise to expanding search can yield dramatic improvements in terms of the approximability of the search ratios. For general graphs, we note that our 5.55 approximation is strict, and not asymptotic. As a last observation, we note that the pathwise and expanding search algorithms appear to depend crucially on the approximability of the Traveling Salesman problem and the Steiner Tree problem, respectively.

2 Preliminaries.

Let $G = (\mathcal{E}, \mathcal{V})$ be an undirected, connected, edge-weighted graph with $|\mathcal{V}| = n + 1$, and a distinguished root vertex $O \in \mathcal{V}$. The weight or *length* of edge $e \in \mathcal{E}$, denoted by $\lambda(e)$, represents the time required to search that edge (we assume, via normalization, that $\lambda(e) \geq 1$ for all edges e). For subgraphs or subsets of edges X, we write $\lambda(X)$ for the sum of the lengths of all the edges in X. We will call a graph of unit edge weights *unweighted*, otherwise it is weighted.

An expanding search, or simply *search strategy* on G is a sequence of edges, starting from the root, chosen so that the set of edges that have been searched at any given point in the sequence is a connected, increasing set. More precisely:

Definition 1. *An expanding search S on a graph G with root vertex O is a sequence of edges $(e_1, \ldots, e_n)$ such that every* prefix *$\{e_1, \ldots, e_k\}$, $k = 1, \ldots, n$ is a subtree of G rooted at O. We denote the set of all expanding searches on (G, O) by $\mathcal{S} = \mathcal{S}(G, O)$.*

We note that if we wished, we could define search strategies less restrictively so that every prefix is simply a connected subgraph rather than a tree, but it will soon be clear that strategies fulfilling Definition 1 are dominant.

For a given vertex $v \in \mathcal{V}$ and a given search strategy $S = (e_1, \ldots, e_n)$, denote by S_v the first prefix $\{e_1, \ldots, e_k\}$ that covers v. The *search time*, $T(S, v)$ of v is the total time $\lambda(S_v)$ taken to search all the edges before v is discovered. Let $d(v)$ denote the length of the shortest path from O to v, which is the minimum time for the Searcher to discover v. For $v \neq O$ the *normalized search time* is denoted by $\hat{T}(S,v) = T(S,v)/d(v)$.

Definition 2. *The (deterministic) search ratio $\sigma_S = \sigma_S(G)$ of a search strategy S for the graph G is defined as*

$$\sigma_S(G) = \max_{v \in \mathcal{V} - \{O\}} \hat{T}(S,v).$$

The (deterministic) search ratio, $\sigma = \sigma(G)$ of G is defined as

$$\sigma(G) = \min_{S \in \mathcal{S}} \sigma_S(G).$$

If $\sigma_S = \sigma$ we say S is optimal.

We will also consider *randomized search strategies*, that is some probabilistic choice of search strategies. Following the standard notation, we denote randomized strategies by lower case letters, and for a randomized search strategy s and a vertex v, we extend the notation $T(s,v)$ to denote the expected search time of v. Similarly we write $\hat{T}(s,v)$ for the *expected normalized search time* $T(s,v)/d(v)$.

Definition 3. *The randomized search ratio $\rho_s = \rho_s(G)$ of a randomized search strategy s for the graph G is given by*

$$\rho_s(G) = \max_{v \in \mathcal{V} - \{O\}} \hat{T}(s,v).$$

The randomized search ratio, $\rho = \rho(G)$ of G is given by

$$\rho(G) = \inf_{s} \rho_s(G),$$

where the infinum is taken over all possible randomized search strategies s. If $\rho_s = \rho$ we say s is optimal.

We will view the randomized search ratio ρ through the lens of a finite zero-sum game between a Searcher and a malevolent Hider. The Searcher's pure strategy set is the set $\mathcal{S}$ of expanding searches and the Hider's pure strategy set is the set $\mathcal{V} - \{O\}$ of non-root vertices of G. For a Hider strategy $v \in \mathcal{V} - \{O\}$ and a Searcher strategy $S \in \mathcal{S}$, the payoff of the game is $\hat{T}(S,v)$, which the Hider wishes to maximize and the Searcher wishes to minimize. Since the strategy sets are finite, the game has a value and optimal mixed strategies for both players. By the standard minimax theorem for zero-sum games, the value of the game is equal to the randomized search ratio and an optimal randomized search strategy is an optimal mixed strategy for the Searcher in the game. A mixed strategy for the Hider is a probability distribution h over the vertices $\mathcal{V} - \{O\}$, and for mixed strategies h and s of the Hider and Searcher respectively, we write $T(s,h)$ and $\hat{T}(s,h)$ for the corresponding expected search time and expected normalized search time.

We will obtain lower bounds for $\rho(G)$ by giving explicit Hider strategies. More precisely, if h is a given mixed Hider strategy, the minimax theorem implies that $\rho(G) \geq \min_{S \in \mathcal{S}} \hat{T}(S,h)$.

3 General graphs.

In this section we show that the problem of computing the (deterministic) search ratio is NP-hard. We also give a search strategy that achieves a $4\ln(4) \approx 5.55$ approximation ratio.

Theorem 4. *Given a graph G with root O and a constant $R \geq 0$, it is NP-Complete to decide whether $\sigma(G) \leq R$.*

Proof. The proof is based on a reduction from 3-SAT. Given a 3-SAT instance consisting of n variables and m clauses with $m \geq n$, we construct an instance of our problem.

We construct the graph G consisting of vertices O, P, a vertex C_j for every clause (the *clause vertices*), vertices X_i (the *variable vertices*) and vertices X_i^0, X_i^1 (the *literal vertices*) for every variable. For every $i = 1, \ldots, n$ there are unit length edges of the form $(X_i, X_i^0), (X_i, X_i^1), (P, X_i^0), (P, X_i^1)$. For every variable x_i appearing positively in the j-th clause there is an edge (C_j, X_i^1) of length 2 and for every variable x_i appearing negatively in the j-th clause there is an edge (C_j, X_i^0) of length 2. For every $j = 1, \ldots, m$ there is an edge (O, C_j) of length 3 and for every $i = 1, \ldots, n$ there is an edge (O, X_i) of length 3. Finally, there is an edge (O, P) of length 3. We fix $R = 1 + \frac{2}{3}(n + m)$. The construction is shown in Figure 2.

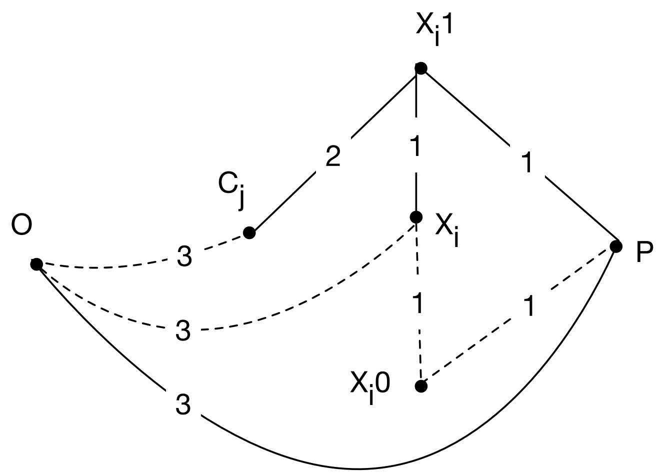

Figure 2: A schematic view of the graph G used in the reduction of Theorem 4

.

Note that the vertices can be partitioned according to their distance from O. In particular, vertex P, as well as variable and clause vertices have distances 3, whereas literal vertices have distance 4.

We must show that there exists a boolean assignment to the variables satisfying all clauses if and only if the search ratio of G is at most R.

For the easy direction of the proof, consider a boolean assignment $b \in \{0, 1\}^n$ to the variables satisfying all clauses. We will show that there is a search strategy with search ratio at most R. First we construct a tree H covering all distance 3 vertices with total length $3R$. The tree consists of the edge (O, P), the edges $(P, X_i^{b_i}), (X_i, X_i^{b_i})$ for every $i = 1, \ldots, n$, and for every clause C_j an edge from C_j to the literal vertex corresponding to a literal satisfying the clause. We denote the tree constructed from b by H^b. The total length of H^b is $3 + 2n + 2m$ which is exactly $3R$ by the choice of R. To turn the tree into a search strategy S we order the edges from H by increasing distance from 0. This sequence S is completed in arbitrary order with the remaining edges of the form (X_i, X_i^0) and (X_i, X_i^1). We have $\rho_S(P) = 1, \rho_S(C_j) \leq 3R/3, \rho_S(X_i) \leq 3R/3$ and $\rho(X_i^x) \leq (3R + n)/4 \leq R$ for every i, j, which shows that the search ratio of G is at most R.

For the hard direction, assume that there is a search strategy with search ratio at most R. Let H be its shortest prefix covering all distance 3 vertices. By the definition of the search ratio we know that $\lambda(H) \leq 3R$. Through a sequence of transformations we turn H into a tree of the form H^b with

$\lambda(H^b) \le \lambda(H)$. This will show that b is a satisfying assignment for the formula and complete the proof of the theorem.

- If (O, P) does not belong to H we add it. This must create a cycle, containing an edge of the form (O, v) with $v \ne P$. Now we remove this edge, and obtain a tree of the same length.
- If there is an edge (O, C_j) in H for some j, then we replace this edge by the edges $(C_j, v), (v, P)$, where v is a vertex corresponding to a literal from the j-th clause. Some of the added edges might already have been present. The result is a tree of no greater length.
- If there is an edge of the form (O, X_i) in H for some i, then we replace this edge by the edges $(X_i, X_i^0), (X_i^0, P)$. Again, the result is a tree with of no greater length.
- At this stage we know that O is only connected to P in the tree.
- If there is a vertex C_j connected to several vertices $v_1, \ldots, v_k$ for $k \ge 2$, then we remove the edges $(C_j, v_1), \ldots, (C_j, v_k)$. Hence, the tree now contains k components, each containing some distinct vertex v_i, and only one of them also containing P. Without loss of generality suppose that v_1 and P are in the same component. Then we add (C_j, v_1) back to H and add for each vertex v_i ($i = 2, \ldots, k$), a length 2 path to P, going through any literal vertex to which v_i is connected. This way we maintain a tree, and do not increase its length (it might even decrease if some of the added edges were already present).
- At this stage we know that every C_j vertex is adjacent to exact one length 2 edge. Also for every $i = 1, \ldots, n$, among the vertices $\{X_i, X_i^0, X_i^1, P\}$ there are at least two edges, one adjacent to X_i and one adjacent to P. The last edge is necessary since otherwise there would be no connection from the vertices $\{X_i, X_i^0, X_i^1\}$ to P, since by the previous point we know that such a path could not go through a clause vertex. Let k be the total number of additional edges that could exist among the vertex sets $\{X_i, X_i^0, X_i^1, P\}$ over all $i = 1, \ldots, n$. Then the total length of H is $3 + 2m + 2n + k$, which by assumption is at most $3R$. By the choice of R we have equality and thus $k = 0$. This shows that H is a tree of the form H^b for some $b \in \{0, 1\}^n$, which a satisfying assignment.

□

Using an approach similar to the doubling heuristic of Koutsoupias et al. [30], we obtain a constant-approximation algorithm for computing the search ratio. It is worth pointing out that the algorithm doubles the radius, and explores the resulting graph by computing a Steiner tree of the corresponding vertex set (in contrast to pathwise search, in which the resulting graph is simply explored depth-first).

Theorem 5. *There is a polynomial-time search algorithm that approximates $\sigma(G)$ within a factor of $4\ln(4) + \epsilon < 5.55$.*

Proof. For any $d \ge 1$, let $\mathcal{V}_d$ be the set of vertices of G at distance no more than d from the root O. Let G_d be the subtree of minimal length in G that contains all the vertices in $\mathcal{V}_d$. It is easy to see that σ is at least $\lambda(G_d)/d$.

We can view the problem of computing G_d as the problem of finding a minimum-cost Steiner tree for the set of vertices $\mathcal{V}_d$. The best known polynomial time algorithm that approximates the problem within a constant factor is that of Byrka et al. [16], which has approximation ratio $\ln(4)+\epsilon$. Let $\hat{G}_d$ be a subtree of G with total length no greater than $\ln(4)\lambda(G_d)$ that contains all the vertices in $\mathcal{V}_d$.

Consider the following family of search strategies S, for fixed $1 = d_0, d_1, \dots, d_k$, where d_k is the radius of G. In increasing order of $j \geq 1$, search all the edges of $\hat{G}_{d_j}$ in an arbitrary order (omitting those edges that have already been searched). Suppose the Hider is at some vertex v reached in the jth phase of the algorithm. Then $d(v)$ must be at least d_{j-1}, and

$$\hat{T}(S,v) \leq \frac{\sum_{i=1}^{j} \lambda(\hat{G}_{d_i})}{d_{j-1}} \leq \frac{\ln(4)\sum_{i=1}^{j} \lambda(G_{d_i})}{d_{j-1}} \leq \frac{\ln(4)\sigma\sum_{i=1}^{j} d_i}{d_{j-1}}.$$

It is optimal to choose $d_i = 2^i$ (for a proof of this, see Koutsoupias et al. [30]). So we obtain $\hat{T}(S,v) \leq 4\ln(4)\sigma < 5.55\sigma$. □

4 Trees and unweighted graphs.

In this section we present our main technical results that apply to unweighted graphs and (weighted) trees. For both classes of graph it is easy to show that it is optimal, in the deterministic setting, to search the vertices in non-decreasing order of their distance from the root.

If G is a graph with root O, for any $r > 0$ let $\mathcal{V}_r$ be the set of vertices in G at distance no more than r from the root O and let G_r be the induced subgraph of G with vertex set $\mathcal{V}_r$.

Proposition 6. *Let G be a rooted graph and suppose that G is a tree or an unweighted graph. Then an optimal search strategy is to search the vertices in non-decreasing order of their distance from the root. The search ratio σ is given by*

(i) $\sigma = \sup_{r>0} \frac{\lambda(G_r)}{r}$ *if G is a tree and*

(ii) $\sigma = \sup_{r>0} \frac{|\mathcal{V}_r|-1}{r}$ *if G is an unweighted graph.*

Proof. We label the non-root vertices $v_1, \dots, v_n$ in non-decreasing order of distance from O. Suppose a search S is optimal so that $\sigma_S = \sigma$, and visits the vertices in an order $v_{i_1}, \dots, v_{i_n}$ different to $v_1, \dots, v_n$. Let j be minimal with $i_j \neq j$, so that $d(v_{i_j}) \geq d(v_{i_{j+1}})$. Let S' be a search that visits the vertices in the same order as S except that v_{i_j} and $v_{i_{j+1}}$ are visited the other way around. It is easy to see that such an S' exists, and in both the case that G is a tree or it is unweighted, the new search times of these vertices satisfy $T(S', v_{i_j}) = T(S, v_{i_{j+1}})$ and $T(S', v_{i_{j+1}}) \leq T(S, v_{i_{j+1}})$.

Hence $\hat{T}(S', v_{i_{j+1}}) \leq \hat{T}(S, v_{i_{j+1}})$, and also

$$\hat{T}(S', v_{i_j}) = T(S, v_{i_{j+1}})/d(v_{i_j}) \leq T(S, v_{i_{j+1}})/d(v_{i_{j+1}}) = \hat{T}(S, v_{i_{j+1}}.$$

For every other vertex v, we have $\hat{T}(S', v) = \hat{T}(S, v)$. Thus $\sigma_{S'} \leq \sigma_S$.

Repeating this process a finite number of times results in a search that visits the vertices in the order $v_1, \dots, v_n$ and has a search ratio no greater than S, so S is optimal.

The two expressions for the search ratio of G in the statement of the proposition follow immediately. □

Note that the above argument does not work for unweighted graphs (as we should expect) since in general, swapping the order in which adjacent vertices are visited changes the search times of the other vertices.

In the randomized setting for trees and unweighted graphs we first show that the optimal deterministic search approximates the optimal randomized search by a factor of 2. To prove this, we use the following collection of lower bounds for ρ. For each non-root vertex v of G, let λ_v denote the length of the unique edge incident to v on the shortest path between O and v (so if G is an unweighted graph then $\lambda_v = 1$) and let $\lambda_O = 0$. For a set $\mathcal{A}$ of vertices, let $\lambda(\mathcal{A}) = \sum_{v\in\mathcal{A}} \lambda_v$ and let $\Delta(\mathcal{A}) = \sum_{v\in\mathcal{A}} \lambda_v d(v)$.

Lemma 7. *Suppose $\mathcal{A}$ is a set of non-root vertices of G, and suppose the Hider chooses each $v \in \mathcal{A}$ with probability $p_i = \lambda_v d(v)/\Delta(\mathcal{A})$. Then the normalized search time $\hat{T}(S,p)$ of any search S against the Hider strategy p satisfies*

$$\rho \geq \hat{T}(S,p) \geq \frac{\sum_{u,v\in\mathcal{V}(H)} \lambda_u\lambda_v}{\Delta(\mathcal{A})} \geq \frac{\lambda(\mathcal{A})^2}{2\Delta(\mathcal{A})}.$$

Proof. Suppose S is the search strategy that visits the vertices of $\mathcal{A}$ in the order $v_1,\ldots,v_m$. Then clearly $T(S,v_i) \geq \sum_{j\leq i} \lambda_{v_j}$ for all v_i so the expected value of the search ratio of S, under p is

$$\sum_{i=1}^{m} \frac{p_i T(S,v_i)}{d(v_i)} \geq \sum_{i=1}^{m} \frac{\lambda_{v_i} d(v_i)}{d(v_i)\Delta(\mathcal{A})} \sum_{j\leq i} \lambda_{v_j} = \frac{\sum_{u,v\in\mathcal{A}} \lambda_u\lambda_v}{\Delta(\mathcal{A})}.$$

The second inequality is trivial.

□

The next proposition follows directly from Lemma 7.

Proposition 8. *The randomized search ratio ρ of a tree or unweighted graph satisfies $\sigma/2 \leq \rho \leq \sigma$. Hence the optimal deterministic search is a 2-approximation of the optimal randomized search.*

Proof. Let S^* be the optimal deterministic search strategy that searches the vertices in non-decreasing order of their distance from the root, and suppose v is a vertex such that $\sigma = \sigma_{S^*} = \hat{T}(S^*,v)$. If $\mathcal{A}$ is the set of non-root vertices at distance no more than $d(v)$, then we must have $T(S^*,v) = \lambda(\mathcal{A})$, so by Lemma 7,

$$\frac{\sigma}{\rho} \leq \frac{\lambda(\mathcal{A})/d(v)}{\lambda(\mathcal{A})^2/(2\Delta(\mathcal{A}))} = \frac{2\Delta(\mathcal{A})}{d(v)\lambda(\mathcal{A})} \leq 2,$$

since $\Delta(\mathcal{A}) \leq \lambda(\mathcal{A})d(v)$.

□

We will show next that we can obtain improved approximations; more precisely we will present and analyze a randomized search for trees or unweighted graphs with approximation ratio asymptotically equal to 5/4. In the case that G is an unweighted graph, we will define the search on some shortest path tree (that is, a spanning tree of G comprising shortest paths from O to each vertex); if G is a weighted tree then we define the search on the whole of G. The idea of the strategy is to partition the vertices of the tree into subsets $\mathcal{V}_0,\mathcal{V}_1,\ldots$, each of which contains vertices whose distances from O are within some interval $[x_i,x_{i+1}]$, where the x_i are chosen randomly according to the method described later in Definition 10. The subsets are then searched one at a time, in increasing order of distance from O. Note that after visiting all the vertices in $\mathcal{V}_1\cup\ldots\cup\mathcal{V}_j$ we can contract all the edges searched so far to the root vertex O and consider the problem of how to search the induced subtree G_{i+1} with vertex set $\mathcal{V}_{j+1}\cup O$.

The method of searching each of these subtrees is according to a *random depth-first search* (or *RDFS*), which we define as follows. Given a set of vertices H and a depth-first search S of H starting at the root, consider the search S^{-1} which is the depth-first search of H that arrives at the leaf vertices of H in the reverse order to S. An equiprobable choice of S and S^{-1} is a RDFS of H. It is straightforward to calculate the maximum expected search time of a RDFS.

Lemma 9. *Under any RDFS of a rooted tree H, every vertex is found in expected time no more than $(\lambda(H) + d_{\max})/2$, where $d_{\max}$ is the distance of the furthest vertex from the root of H.*

Proof. Suppose S is some depth-first search of H and v is a vertex of H. Let A be the subset of edges searched by S up to and including when v is reached and let B be the subset of edges searched by S^{-1} up to and including when v is reached. It is easy to see that $A \cap B$ is the unique path from v to the root. If s is the RDFS that chooses S and S^{-1} equiprobably, then the expected time $T(s,v)$ of s to reach v is

$$\begin{aligned} T(s,v) &= (\lambda(A) + \lambda(B))/2 \\ &= (\lambda(A \cup B) + \lambda(X \cap Y))/2 \\ &\leq (\lambda(H) + d_{\max})/2. \end{aligned}$$

□

We can now define the randomized deepening strategy. Let t be the smallest integer such that every vertex of G is at distance less than 2^t from O.

Definition 10 (Randomized deepening strategy). *Suppose G is a tree with root O. For $i = 1, \ldots, t$, choose some x_i uniformly at random from the interval $[2^{i-1}, 2^i]$ and let $x_0 = 1$ and $x_{t+1} = 2^t$. For $i = 0, \ldots, t$, let $\mathcal{V}_i$ be the set of vertices of G whose distance from O lies in the interval $[x_i, x_{i+1})$. We call $\mathcal{V}_0, \ldots, \mathcal{V}_t$ the* levels *of the search, so that $\mathcal{V}_i$ is level i of the search. Let G_0 be the induced subtree of G with vertex set $\mathcal{V}_0 \cup O$ and we define $G_i, i > 0$ recursively as the induced subtree with vertex set $\mathcal{V}_i \cup O$ of the graph obtained by contracting $G_0 \cup \ldots \cup G_{i-1}$ to the root O. The* ***randomized deepening strategy*** *performs a RDFS of each of the trees G_i in the order $G_0, \ldots, G_t$.*

We need two straight-forward results before stating and proving the main theorem in this section, that the random deepening strategy has approximation ratio 5/4. Let $\mathcal{A}_i$ be set of vertices of G whose distance from O is in the interval $[2^{i-1}, 2^i)$, and let $\mathcal{A}^i = \cup_{j \leq i} \mathcal{A}_j$. (For the purposes of writing the proof of Theorem 13 we allow i to take any integer value, but note that $\mathcal{A}_i$ is only non-empty for $i = 1, \ldots, t$.)

Lemma 11. *The expected sum over all vertices $v \in \mathcal{A}_i \cap \mathcal{V}_{i-1}$ of edge lengths λ_v is $2\lambda(\mathcal{A}_i) - \Delta(\mathcal{A}_i)/2^{i-1}$.*

Proof. The probability that the distance $d(v)$ from O of a vertex v in $\mathcal{A}_i$ is less than x_i is $(2^i - d(v))/2^{i-1}$. So the expected sum over all vertices $v \in \mathcal{A}_i \cap \mathcal{V}_{i-1}$ of lengths λ_v is

$$\sum_{v \in \mathcal{A}_i} \left(\frac{2^i - d(v)}{2^{i-1}} \right) \lambda(v) = 2\lambda(\mathcal{A}_i) - \Delta(\mathcal{A}_i)/2^{i-1}.$$

□

We also make two simple observations about the parameters $\Delta(\mathcal{A}^i$.

Lemma 12. *For any $i = 1, \ldots, t$ we have*

(i) $\Delta(\mathcal{A}^i) \leq 2^i \lambda(\mathcal{A}^i)$ *and*

(ii) $\frac{\lambda(\mathcal{A}^i)-\Delta(\mathcal{A}^i)/2^i}{\rho} \leq 2^{i-1}$.

Proof. Item (i) follows from the fact that every vertex in $\mathcal{A}^i$ is at distance no further than 2^i from O.

For item (ii), we use Lemma 7 to bound ρ, giving

$$\frac{\lambda(\mathcal{A}^i) - \Delta(\mathcal{A}^i)/2^i}{\rho} \leq \frac{\lambda(\mathcal{A}^i) - \Delta(\mathcal{A}^i)/2^i}{\lambda(\mathcal{A}^i)^2/(2\Delta(\mathcal{A}^i)} = 2\left(\frac{\Delta(\mathcal{A}^i}{\lambda(\mathcal{A}^i}\right) - 2^{1-i}\left(\frac{\Delta(\mathcal{A}^i)}{\lambda(\mathcal{A}^i)}\right)^2.$$

The right-hand side of the expression above is a quadratic in $\Delta(\mathcal{A}^i)/\lambda(\mathcal{A}^i)$, which is easily shown to be maximized at $\Delta(\mathcal{A}^i)/\lambda(\mathcal{A}^i) = 2^{i-1}$, where it takes a value of 2^{i-1}.

□

Theorem 13. *Let G be a weighted tree or an unweighted graph. Let s be the randomized deepening strategy on G if G is a tree, or on some shortest path tree of G if G is an unweighted graph. Then the approximation ratio of s is asymptotically $5/4$. In particular, $\rho_s \leq (5/4)\rho + 1$.*

Proof. First suppose G is a tree. Let v be a vertex that maximizes the randomized search ratio of s and suppose v is at distance d from O. Let L be the expected sum over all vertices u in previous levels from v of the lengths λ_u plus half the expected sum over all vertices u in the same level as v of the lengths λ_u. It is easy to see that the expected difference between the closest and furthest vertices from O in the same level as v is no more than $2d$, so by Lemma 9, the expected search time of v is at most $L+(2d)/2 = L+d$. Hence

$$\frac{\rho_s}{\rho} \leq \frac{(L+d)/d}{\rho} \leq \frac{L/d}{\rho} + 1/\rho.$$

We just have to show that $L/(d\rho) \leq 5/4$. Suppose $v \in \mathcal{A}_k$ for some k and let L_1, L_2, L_3 be the contributions to L from vertices in $\mathcal{A}_{k-1}, \mathcal{A}_k, \mathcal{A}_{k+1}$, respectively, so that $L = \lambda(\mathcal{A}^{k-2}) + L_1 + L_2 + L_3$. We calculate L_1, L_2 and L_3 separately.

For L_1, observe that with probability $(d-2^{k-1})/2^{k-1}$ the vertex v is in level k, which does not intersect with any vertices in $\mathcal{A}_{k-1}$. Otherwise, v is in level $k-1$ which contains some vertices of $\mathcal{A}_{k-1}$ and by Lemma 11,

$$\begin{aligned} L_1 &= \left(\frac{d-2^{k-1}}{2^{k-1}}\right)\lambda(\mathcal{A}_{k-1}) + \left(\frac{2^k-d}{2^{k-1}}\right)(\lambda(\mathcal{A}_{k-1})/2 + (2\lambda(\mathcal{A}_{k-1}) - \Delta(\mathcal{A}_{k-1})/2^{k-2})/2 \\ &= \left(2-\frac{d}{2^k}\right)\lambda(\mathcal{A}_{k-1}) - \left(2-\frac{d}{2^{k-1}}\right)\frac{\Delta(\mathcal{A}_{k-1})}{2^{k-1}}. \end{aligned}$$

Similarly, for L_3, if v is in level $k-1$ then all vertices of A_{k+1} are in levels after the level of v. Otherwise v is in level k which contains some vertices of $\mathcal{A}_{k+1}$. Again applying Lemma 11,

$$L_3 = \left(\frac{d}{2^{k-1}} - 1\right)\left(\lambda(\mathcal{A}_{k+1}) - \frac{\Delta(\mathcal{A}_{k+1})}{2^{k+1}}\right).$$

Lastly, for L_2, observe that a vertex u in $\mathcal{A}_k$ at distance $d(u) \leq d$ is in the level before v if $d(u) \leq x_k < d$, otherwise it is in the same level as v. So the contribution u makes to L_2 is

$$\left(\frac{d-d(u)}{2^k}\right)\cdot 1 + \left(\frac{2^{k-1}-(d-d(u))}{2^{k-1}}\right)\cdot\frac{1}{2} = \frac{d-d(u)+2^{k-1}}{2^k}.$$

Similarly, if $d(u) > d$, then u is in the level after v if $d \le x_k < d(u)$, otherwise u is in the same level as v, so the contribution u makes to L_2 is

$$\left(\frac{d(u)-d}{2^k}\right)\cdot 0 + \left(\frac{2^{k-1}-(d(u)-d)}{2^{k-1}}\right)\cdot\frac{1}{2} = \frac{d-d(u)+2^{k-1}}{2^k},$$

which is the same. Hence L_2 is given by

$$L_2 = \sum_{u\in\mathcal{V}_k} \frac{d-d(u)+2^{k-1}}{2^k} = \left(\frac{d}{2^k}+\frac{1}{2}\right)\lambda(\mathcal{A}_k) - \frac{\Delta(\mathcal{A}_k)}{2^k}.$$

Using $L = \lambda(\mathcal{A}_{[k-2]}) + L_1 + L_2 + L_3$, combining our expressions for L_1, L_2 and L_3 and rearranging, we obtain

$$\begin{aligned} L = {} & \left(1-\frac{d}{2^k}\right)\left(\frac{\Delta(\mathcal{A}^{k-2})}{2^{k-2}} - \lambda(\mathcal{A}^{k-2})\right) + \left(\frac{3}{2}-\frac{d}{2^{k-1}}\right)\left(\lambda(\mathcal{A}^{k-1}) - \frac{\Delta(\mathcal{A}^{k-1})}{2^{k-1}}\right) \\ & + \left(\frac{3}{2}-\frac{d}{2^k}\right)\left(\lambda(\mathcal{A}^{k}) - \frac{\Delta(\mathcal{A}^{k})}{2^{k}}\right) + \left(\frac{d}{2^{k-1}}-1\right)\left(\lambda(\mathcal{A}^{k+1}) - \frac{\Delta(\mathcal{A}^{k+1})}{2^{k+1}}\right). \end{aligned}$$

The first term in the expression on the right-hand side above is non-positive, since $d \le 2^k$ and $\Delta(\mathcal{A}^{k-2}) \le 2^{k-2}\lambda(\mathcal{A}^{k-2})$, by Lemma 12(i). So, dividing by d, we obtain

$$\begin{aligned} \frac{L}{d} \le {} & \left(\frac{3}{2d}-\frac{1}{2^{k-1}}\right)\left(\lambda(\mathcal{A}^{k-1}) - \frac{\Delta(\mathcal{A}^{k-1})}{2^{k-1}}\right) + \left(\frac{3}{2d}-\frac{1}{2^k}\right)\left(\lambda(\mathcal{A}^{k}) - \frac{\Delta(\mathcal{A}^{k})}{2^{k}}\right) \\ & + \left(\frac{1}{2^{k-1}}-\frac{1}{d}\right)\left(\lambda(\mathcal{A}^{k+1}) - \frac{\Delta(\mathcal{A}^{k+1})}{2^{k+1}}\right). \end{aligned} \tag{1}$$

If $2^{k-1} \le d \le 3\cdot 2^{k-2}$ then it follows from Lemma 12(i) that all three of the terms on the right-hand side of (1) are non-negative. Hence by Lemma 12(ii),

$$\begin{aligned} \frac{L}{d\rho} & \le \left(\frac{3}{2d}-\frac{1}{2^{k-1}}\right)2^{k-2} + \left(\frac{3}{2d}-\frac{1}{2^k}\right)2^{k-1} + \left(\frac{1}{2^{k-1}}-\frac{1}{d}\right)2^k \\ & = 1 + 2^{k-3}/d \\ & \le 5/4 \text{ (maximized when } d = 2^{k-1}). \end{aligned}$$

If $2^k \ge d > 3\cdot 2^{k-2}$ then the first term on the right-hand side of (1) is negative but the other two terms are non-negative by Lemma 12(i), so by Lemma 12(ii),

$$\begin{aligned} \frac{L}{d\rho} & \le + \left(\frac{3}{2d}-\frac{1}{2^k}\right)2^{k-1} + \left(\frac{1}{2^{k-1}}-\frac{1}{d}\right)2^k \\ & = 3/2 - 2^{k-2}/d \\ & \le 5/4 \text{ (maximized when } d = 2^{k}). \end{aligned}$$

This completes the proof in the case that G is a tree. If G is not a tree, then we remove edges from G until obtaining a shortest path tree. Note that removing these edges has no effect on $\lambda(v)$ and $d(v)$ for vertices v, so the lower bounds given by Lemma 7 remain unchanged. Hence we can apply the same argument as above, implementing the randomized deepening strategy on the shortest path tree of G, to obtain an approximation ratio asymptotically equal to 5/4.

□

We observe that the ratio 5/4 could be improved slightly by introducing some randomization into the definition of $\mathcal{A}_1, \ldots, \mathcal{A}_t$: that is, we could define $\mathcal{A}_i$ as the set of all edges whose length d satisfies $2^{i-1-\theta} \leq d < 2^{i-\theta}$, where θ is chosen according to some probability distribution on $[0,1]$. This would improve the approximation ratio from 5/4, but only marginally.

We may compare this result with analogous results from Koutsoupias et al. [30] in the context of pathwise search: for unweighted graphs they obtain algorithms that approximate the deterministic and randomized search ratio within a factor of 6 and 8.98 respectively. For expanding search, we easily obtain the optimal strategy in the deterministic case, and we obtain a 5/4-approximate strategy in the randomized case. Although our randomized strategy is somewhat more sophisticated, the difference must be in part due to the fact that expanding search is more straightforward to deal with than pathwise search.

We also note than in the case that G is a star, a careful analysis of the proof of Theorem 13 shows that the "asymptotically" in the statement of the theorem may be removed: in other words, the randomized deepening strategy approximates the optimal randomized search on a star by a factor of 5/4. Furthermore, it is a simple matter to extend it to a 5/4-approximation of the randomized search ratio for the analogous pathwise search problem (since both the lower and upper estimates are almost exactly doubled).

5 Star search.

In this section we consider problems related to the search ratio and randomized search ratio of a star graph. We have already shown in Section 4 that the optimal deterministic search simply visits the vertices in non-decreasing order of their distance from the root, and the randomized deepening strategy is a 5/4-approximation of the optimal randomized search. Here we show that the randomized search ratio of a star with n edges is largest when all the edges have the same length. We also show that if the lengths of the edges do not increase to quickly we can find the optimal randomized search.

Suppose that G is a star graph consisting of n edges $e_1, \ldots, e_n$ of lengths $d_1, \ldots, d_n$ with each e_i incident to the root, O and to a vertex v_i. We assume without loss of generality that $d_1 \leq d_2 \cdots \leq d_n$. An expanding search of such a graph corresponds to a permutation of the edges, or equivalently the vertices.

For $j = 1, \ldots, n$, let $\mu_j = \sum_{i=1}^{j} d_i$ be the total length of the first j edges and let $D_j = \sum_{i=1}^{j} d_i^2$ be the sum of the squares of the lengths of the first j edges. By Lemma 7, for each $k = 1, \ldots, n$,

$$\rho_S \geq \frac{\sum_{i,j \leq k} d_i d_j}{D_k} = \frac{1}{2}\left(1 + \frac{\mu_k^2}{D_k}\right). \tag{2}$$

Let $\pi_k = \frac{1}{2}\left(1 + \mu_k^2/D_k\right)$ be the right-hand side of (2).

5.1 A tight upper bound for the randomized search ratio of a star graph with n edges.

A natural question is whether the randomized search ratio for star graphs is exactly π_k for some $k \leq n$. For instance, it is easy to see that if the star only has two edges, this is the indeed the case and the bound helps us obtain optimal randomized strategies. However, it is not true in general even for a star with 3 edges.

Lemma 14. *For a star graph with 2 edges of lengths d_1 and d_2, the randomized search ratio ρ is given by*

$$\rho = \pi_2 = \frac{1}{2}\left(1 + \frac{(d_1 + d_2)^2}{d_1^2 + d_2^2}\right). \tag{3}$$

The optimal search strategy s is to search the edges in the order e_1, e_2 with probability proportional to d_2^2 and to search the edges in the order e_2, e_1 with probability proportional to d_1^2.

Moreover, there exist 3-edge stars for which $\rho \neq \pi_3$ and $\rho < \max_{k \leq n} \pi_k$.

Proof. Consider first a star with two edges. By (2) we only need to show that the search strategy s has randomized search ratio π_2. We have

$$\begin{aligned}\hat{T}(s, e_1) &= \frac{d_2^2}{d_1^2 + d_2^2}(1) + \frac{d_1^2}{d_1^2 + d_2^2}\left(\frac{d_1 + d_2}{d_1}\right) \\ &= \frac{1}{2}\left(1 + \frac{(d_1 + d_2)^2}{d_1^2 + d_2^2}\right) = \pi_2\end{aligned}$$

Similarly for $\hat{T}(s, e_2)$.

Consider now a star with three edges of lengths 1, 1 and 8. We have that $\pi_1 = 1, \pi_2 = 3/2, \pi_3 = 83/66$, thus $\max_k \pi_k = 3/2$. Consider a strategy s for the Searcher which searches edges e_1 and e_2 first in a random order, then searches e_3, so that $\hat{T}(s, e_1) = \hat{T}(s, e_2) = (1/2)(1 + 2) = 3/2$ and $\rho \leq \hat{T}(s, e_3) = 10/8 = 5/4 \leq 3/2 = \max_k \pi_k$. □

We now consider how large the deterministic and randomized search ratios can be for a star graph G with n edges. From Proposition 6(i), we have

$$\sigma(G) = \max_{j \leq n} \frac{\sum_{i \leq j} d_i}{d_j} \leq \max_{j \leq n} \frac{j d_j}{d_j} = n.$$

This upper bound is tight if and only if all edges have the same length. In this case it is easy to see that the randomized search ratio ρ is $(n + 1)/2$, and the optimal search strategy is to search the vertices in a uniformly random order. (The lower bound $\rho \geq (n + 1)/2$ comes from the uniform Hider strategy that picks every vertex with equal probability.)

In contrast, it is not so easy to see that $(n + 1)/2$ is the largest value that the randomized search ratio can attain for *any* star graph with n edges. We will show that this is indeed the case by inductively defining a particular randomized search strategy whose randomized search ratio is bounded above by $(n + 1)/2$. We thus prove that the bound is tight. We will also show in Section 5.2 that this strategy is optimal as long as the lengths of the edges do not increase too quickly.

For a given star graph G we inductively define a randomized search strategy s_k on the star graph G_k consisting of only the edges $e_1, \ldots, e_k$ with total length μ_k. Having defined the strategy s_k, we will define s_{k+1} as a randomized mix of two strategies, s_{k+1}^+ and s_{k+1}^-, which we define in Definition 15. The former strategy, s_{k+1}^+ searches the new edge e_{k+1} after searching the other edges, and works well in the case that the length of e_{k+1} is a lot larger than the previous edges. If e_{k+1} is not too large, then it is better to search it at some random point in the middle of s_k, which corresponds to the latter strategy, s_{k+1}^-.

Definition 15. *Suppose s_k has been defined for some $k = 1, \ldots, n-1$. Let s_{k+1}^+ and s_{k+1}^- be randomized search strategies on G_{k+1} defined by:*

(i) s_{k+1}^+: follow the strategy s_k on G_k and then search edge e_{k+1}.

(ii) s_{k+1}^-: choose a time t uniformly at random in $[0, \mu_k]$ and denote the edge that is being searched at time t by e. Follow the strategy s_k, but search edge e_{k+1} immediately before searching e.

Before giving the precise definition of s_k, we estimate the normalized expected search times $\hat{T}(s_{k+1}^+, v_i)$ and $\hat{T}(s_{k+1}^-, v_i)$ in terms of ρ_{s_k} for the vertices v_i with $i = 1, \ldots, k+1$.

First suppose $i \leq k$. Then clearly $\hat{T}(s_{k+1}^+, v_i) \leq \rho_{s_k}$ (with equality for some $i \leq k$) by definition of ρ_{s_k}. Under s_{k+1}^-, with probability $T(s_k, v_i)/\mu_k$ edge e_{k+1} is searched before e_i, so the expected search time of v_i is $T(s_k, v_i) + (T(s_k, v_i)/\mu_k)d_{k+1}$. Hence

$$\begin{aligned}\hat{T}(s_{k+1}^-, v_i) &= \frac{T(s_k, v_i) + (T(s_k, v_i)/\mu_k)d_{k+1}}{d_i} \\ &= \hat{T}(s_k, v_i)(1 + d_{k+1}/\mu_k) \\ &\leq \rho_{s_k}(1 + d_{k+1}/\mu_k).\end{aligned}$$

Now suppose $i = k+1$. Under s_{k+1}^+, the time taken to find the Hider is $\mu_k + d_{k+1}$, so $\hat{T}(s_{k+1}^+, v_{k+1}) = \mu_k/d_{k+1} + 1$. Under s_{k+1}^-, the expected search time is $\mu_k/2 + d_{k+1}$ minus a random correction error which depends upon which edge e is being searched under s_k at the random time t chosen uniformly in $[0, \mu_k]$. The edge e is e_i with probability d_i/μ_k, and in this case the expected value of the correction error is $d_i/2$. Hence the expected value of this correction error is $\sum_{i=1}^k (d_i/\mu_k) \cdot (d_i/2) = D_k/(2\mu_k)$. So we have

$$\begin{aligned}\hat{T}(s_{k+1}^-, v_{k+1}) &= \frac{\mu_k/2 + d_{k+1} - D_k/(2\mu_k)}{d_{k+1}} \\ &= \mu_k/(2d_{k+1}) + 1 - D_k/(2\mu_k d_{k+1}).\end{aligned}$$

To sum up, the expected search ratio for each combination of strategies can be bounded above by the payoffs in Table 2. We can now proceed to define s_n.

Table 2: Maximum value of $\hat{T}(s, v)$.

Search strategy, s	**Vertex,** v	
	v_i for some $i \leq k$	v_{k+1}
s_{k+1}^+	ρ_{s_k}	$\mu_k/d_{k+1} + 1$
s_{k+1}^-	$\rho_{s_k}(1 + d_{k+1}/\mu_k)$	$\mu_k/(2d_{k+1}) + 1 - D_k/(2\mu_k d_{k+1})$

Definition 16. *Let s_1 be the only strategy available on G_1. Suppose s_k has already been defined on G_k for some $k = 1, \ldots, n-1$. The strategy s_{k+1} is an optimal mixture of s_{k+1}^+ and s_{k+1}^- in the zero-sum game with payoff matrix given by Table 2.*

The search ratio of s_n can be calculated recursively, since the search ratio $\rho_{s_{k+1}}$ of s_{k+1} is at most the value of the game with payoff matrix given by Table 2, for each $k = 1, \ldots, n-1$. We use this to show that $\rho_{s_n} \leq (n+1)/2$.

Theorem 17. *The randomized search ratio ρ of star graph G with n edges is at most $(n+1)/2$, with equality if and only if all the edges have the same length.*

Proof. We have already pointed out that $\rho = (n+1)/2$ for the star whose edges all have the same length. To show that $\rho \leq (n+1)/2$ we use induction on the number of edges to show that $\rho_{s_n} \leq (n+1)/2$. It is clear that for $k = 1$, we have $\rho_{s_k} = 1 = (k+1)/2$, so assume that $\rho(s_k) \leq (k+1)/2$ for some $k > 1$ and we will show that $\rho_{k+1} \leq (k+2)/2 = k/2 + 1$.

First observe that if $d_{k+1} \geq 2\mu_k/k$ then the Searcher can ensure a payoff of no more than $k/2+1$ in the game in Table 2 just by using strategy s^+_{k+1}. This is because the payoff ρ_{s_k} against a vertex v_i with $i \leq k$ is no more than $(k+1)/2$ by the induction hypothesis and the payoff against v_{k+1} is $\mu_k/d_{k+1} + 1 \leq k/2$.

So assume that $d_{k+1} \leq 2\mu_k/k$, and note also that $d_{k+1} \geq \mu_k/k$, since the lengths of the edges are non-decreasing and d_{k+1} must be at least the average length of edges $e_1, \ldots, e_k$.

By the induction hypothesis, $\rho_{s_k} \leq (k+1)/2$, so the value of the game with payoff matrix given by Table 2 cannot decrease if we replace ρ_{s_k} with $(k+1)/2$ in the table. The value also does not decrease if we replace $-D_k$ by the maximum value it can take, which is $-\mu_k^2/k$ (that is, its value when $d_1, \ldots, d_k$ are all equal). In summary, $\rho_{s_{k+1}}$ is no more than the value of the game given in Table 3.

Table 3: Upper bounds for $\hat{T}(s, v)$.

Search strategy, s	**Vertex,** v	
	v_i for some $i \leq k$	v_{k+1}
s^+_{k+1}	$(k+1)/2$	$\mu_k/d_{k+1} + 1$
s^-_{k+1}	$(k+1)(1 + d_{k+1}/\mu_k)/2$	$\mu_k/(2d_{k+1}) + 1 - \mu_k/(2kd_{k+1})$

By assumption, against strategy s^+_{k+1}, the best response of the Hider (that is, the highest payoff) is given by choosing vertex v_{k+1}. We show that against strategy s^-_{k+1}, the Hider's best response is to choose a vertex v_i with $i \leq k$. This follows from writing the difference, δ between the payoffs in entries $(2,1)$ and $(2,2)$ of Table 3 as

$$\delta = (k-1)\frac{\mu_k}{2d_{k+1}}\left(\left(\frac{k+1}{k-1}\right)\left(\frac{d_{k+1}}{\mu_k}\right)^2 + \frac{d_{k+1}}{\mu_k} - 1/k\right).$$

The quadratic in (d_{k+1}/μ_k) inside the parentheses is increasing for positive values of d_{k+1}/μ_k, and when $d_{k+1}/\mu_k = 1/k$ the quadratic is positive. Since $d_{k+1}/\mu_k \geq 1/k$, we must have $\delta \geq 0$.

Hence the Hider does not have a dominating strategy in the game in Table 3. It is also clear that the Searcher does not have a dominating strategy, since it is better to search e_{k+1} last if and only if the Hider is at some v_i with $i \leq k$. Therefore the game in Table 3 has a unique equilibrium in proper mixed strategies (that is, the players both play each of their strategies with positive probability). The search ratio $\rho_{s_{k+1}}$ of s_{k+1} is bounded above by the value V of the game, which is easily verified to be

$$V = k/2 + 1 - \frac{\frac{k}{2}(d_{k+1}/\mu_k - 1/k)^2}{(d_{k+1}/\mu_k)^2 + 1/k}.$$

This is clearly at most $k/2+1$, with equality if and only if $\mu_k/d_{k+1} = k$. The theorem follows by induction, and equality is only possible if $d_1 = d_2 = \ldots = d_n$. □

5.2 An optimal strategy for edge lengths with small variance.

We give a particular class of star graphs for which s_n is optimal.

Theorem 18. *Suppose G is a star graph and*

$$d_{k+1} \leq \frac{2D_k\mu_k}{\mu_k^2 - D_k}, \tag{4}$$

for all $k = 2, \dots, n-1$. Then s_n is optimal and the search ratio $\rho(G)$ is

$$\rho(G) = \rho_{s_n} = \pi_n = \frac{1}{2}\left(1 + \frac{\mu^2}{D_n}\right).$$

Proof. We prove by induction on the number of edges that $\rho_{s_n} = \pi_n$, from which the theorem follows, by (2). For $n = 1$ or $n = 2$, there is no k satisfying $k = 2, \dots, n-1$, so condition (4) does not apply. For the case $n = 1$, it is clearly true that $\rho_{s_n} = \pi_n = 1$, since s_1 is the only search strategy available. For $n = 2$, the strategy s_2 is an optimal mixture of s_1^+ and s_1^-, that is a mixture of the only 2 pure strategies available to the Searcher. By Lemma 14, this mixture is optimal and $\rho_{s_2} = \pi_2$.

So we suppose that $\rho_{s_k} = \pi_k = (1+\mu_k^2/D_k)/2$ for some $k = 2, \dots, n-1$ and we show that $\rho_{s_{k+1}} = \pi_{k+1}$. It follows from (4) that against s_{k+1}^+, the best response of the Hider is to choose edge e_{k+1}. We show that the best response of the Hider against s_{k+1}^- is to choose some edge e_i with $i \le k$. Similarly to the proof of the Theorem 17, we calculate the difference δ between the payoffs in entries $(2,1)$ and $(2,2)$ of Table 2.

$$\begin{aligned}
\delta &= \rho_{s_k}\left(1 + \frac{d_{k+1}}{\mu_k}\right) - \frac{\mu_k}{2d_{k+1}} - 1 + \frac{D_k}{2\mu_k d_{k+1}} \\
&= \frac{1}{2}\left(1 + \frac{\mu_k^2}{D_k}\right)\left(1 + \frac{d_{k+1}}{\mu_k}\right) - \frac{\mu_k}{2d_{k+1}} - 1 + \frac{D_k}{2\mu_k d_{k+1}} \text{ (by the induction hypothesis)} \\
&= \frac{1}{2x}\left(\left(\frac{\mu_k^2}{D_k} + 1\right)x^2 + \left(\frac{\mu_k^2}{D_k} - 1\right)x + \left(\frac{D_k}{\mu_k^2} - 1\right)\right) \text{ (where } x = d_{k+1}/\mu_k\text{)}.
\end{aligned}$$

Now the coefficient of x^2 in the quadratic in parentheses above is clearly positive, and so is the coefficient of x, since $\mu_k^2 \ge D_k$. Hence for positive x, the quadratic is increasing in x. It follows from the inequality $D_k \le \mu_k d_{k+1}$ that $x \ge D_k/\mu_k^2$, so it is sufficient to show that δ is non-negative for $x = D_k/\mu_k^2$. Substituting this value of x into our expression for δ, we obtain:

$$\begin{aligned}
\delta &\ge \frac{1}{2x}\left(\left(\frac{\mu_k^2}{D_k} + 1\right)\frac{D_k^2}{\mu_k^4} + \left(\frac{\mu_k^2}{D_k} - 1\right)\frac{D_k}{\mu_k^2} + \left(\frac{D_k}{\mu_k^2} - 1\right)\right) \\
&= \frac{1}{2x}\left(\frac{D_k^2}{\mu_k^4} + \frac{D_k}{\mu_k^2}\right) \\
&\ge 0.
\end{aligned}$$

Hence the game in Table 2 has an equilibrium in proper mixed strategies, and it is a simple calculation to show that the value $\rho_{s_{k+1}}$ of the game is

$$\begin{aligned}
\rho_{s_{k+1}} &= \frac{1}{2}\left(1 + \frac{(\mu_k + d_{k+1})^2}{D_k + d_{k+1}^2}\right) \\
&= \frac{1}{2}\left(1 + \frac{\mu_{k+1}^2}{D_{k+1}}\right).
\end{aligned}$$

This completes the proof. □

We note that all our results on expanding search on star graphs can easily be extended to pathwise search, with very similar proofs.

6 Conclusion.

We have undertaken an analysis of expanding search, as defined by Alpern and Lidbetter [2], focusing on the search ratio, as introduced by Koutsoupias et al. [30] in the context of pathwise search in bounded domains. In contrast to [2], we have focused on computational and algorithmic issues of expanding search, an angle that is often neglected in the analysis of search games. For general graphs, we showed that computing the search ratio is NP-Complete, and we gave a $4\ln(4)$ approximation. Our main technical contribution is defining and analyzing explicit randomized search strategies that yield significant improvements to the approximation of the randomized search ratio of trees and unweighted graphs (namely, an approximation equal to $5/4$).

We believe that some of the techniques we introduced in this work can be applicable in the context of pathwise search. For instance, we believe that a variants of the randomized strategy presented in Section 4 will result in improved randomized search ratios for pathwise search in weighted trees.

We leave some open questions which we would like to see addressed by future work. Although we have showed that computing the search ratio of a graph is NP-hard, we do not have an equivalent result for computing the randomized search ratio (though we suspect such a result holds). It would be very interesting to improve upon the approximations of the search ratio and randomized search ratio for general weighted graphs; the latter, in particular, appears to be quite a difficult problem that we believe will require the introduction of new techniques and approaches. A related question is whether the $5/4$ approximation of the randomized search ratio for trees and unweighted graphs can be (significantly) improved.

Another direction for future work is related to the *continuous* model. In this model, the Hider may be located not only on a graph vertex, but on any given point across an edge. Optimal strategies that minimize the deterministic search ratio are relatively easy to obtain [4]; however, we do not yet have strategies that improve upon the straightforward approximations of the randomized search ratio.

Last, we note that the work that introduced the expanding search paradigm [2] raises several interesting optimization problems concerning the average search time of vertices of a graph (assuming expanding search). In particular, one can define the *expanding minimum latency problem*, as the problem of minimizing the total latency of a graph, assuming an expanding search of the graph. Is this problem NP-hard in general graphs? If yes, can one obtain constant-factor approximations? Is the general problem in the setting in which the search time of a vertex is weighted as hard as the unweighted variant? Answers to the above questions will help provide an almost-complete picture of the computability and approximability of expanding search across a variety of performance measures.

References

[1] S. Alpern, V. Baston, and S. Gal. Network search games with immobile hider, without a designated searcher starting point. *International Journal of Game Theory*, 37(2):281-302, 2008.

[2] S. Alpern and T. Lidbetter. Mining coal or Finding terrorists: The expanding search paradigm. *Operations Research*, 61(2):265-279, 2013.

[3] S. Angelopoulos. Further connections between contract-scheduling and ray-searching problems. In *Proc. of the 24th International Joint Conference on Artifcial Intelligence (IJCAI)*, pages 1516-1522, 2015.

[4] S. Angelopoulos, C. Dürr, and T. Lidbetter. The expanding search ratio of a graph. In *Proceedings of the 33rd Symposium on Theoretical Aspects of Computer Science (STACS)*, 2016. DOI:10.4230/LIPIcs.STACS.2016.9.

[5] S. Angelopoulos, A. López-Ortiz, and K. Panagiotou. Multi-target ray searching problems. *Theoretical Computer Science*, 540:2-12, 2014.

[6] S. Arora and G. Karakostas. Approximation schemes for minimum latency problems. *SIAM J. Comput.*, 32(5):1317-1337, 2003.

[7] G. Ausiello, S. Leonardi, and A. Marchetti-Spaccamela. On salesmen, repairmen, spiders, and other traveling agents. In *Algorithms and Complexity, 4th Italian Conference, CIAC* 2000, Rome, Italy, March 2000, Proceedings, pages 1-16, 2000.

[8] R. Baeza-Yates, J. Culberson, and G. Rawlins. Searching in the plane. *Information and Computation*, 106:234-244, 1993.

[9] V. Baston and K. Kikuta. Search games on networks with travelling and search costs and with arbitrary searcher starting points. *Networks*, 62(1):72-79, 2013.

[10] V. Baston and K. Kikuta. Search games on a network with travelling and search costs. *International Journal of Game Theory*, 44(2):347-365, 2015.

[11] A. Beck. On the linear search problem. *Naval Research Logistics*, 2:221-228, 1964.

[12] R. Bellman. An optimal search problem. *SIAM Review*, 5:274, 1963.

[13] D.S. Bernstein, T. J. Perkins, S. Zilberstein, and L. Finkelstein. Scheduling contract algorithms on multiple processors. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence (AAAI)*, pages 702-706, 2002.

[14] A. Blum, P. Chalasani, D. Coppersmith, W. R. Pulleyblank, P. Raghavan, and M. Sudan. The minimum latency problem. In *Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing (STOC)*, pages 163-171, 1994.

[15] P. Bose, J. De Carufel, and S. Durocher. Searching on a line: A complete characterization of the optimal solution. *Theoretical Computer Science*, 569:24-42, 2015.

[16] J. Byrka, F. Grandoni, T. Rothvoß, and L. Sanità. An improved lp-based approximation for steiner tree. In *Proc. of the 42nd ACM symposium on Theory of computing (ACM)*, pages 583-592, 2010.

[17] A. Dagan and S. Gal. Network search games, with arbitrary searcher starting point. *Networks*, 52(3):156- 161, 2008.

[18] E.D. Demaine, S.P. Fekete, and S. Gal. Online searching with turn cost. *Theoretical Computer Science*, 361:342-355, 2006.

[19] R. Fleischer, T. Kamphans, R. Klein, E. Langetepe, and G. Trippen. Competitive online approximation of the optimal search ratio. *SIAM Journal on Computing*, 38(3):881-898, 2008.

[20] S. Gal. A general search game. *Israel Journal of Mathematics*, 12:32-45, 1972.

[21] S. Gal. Minimax solutions for linear search problems. *SIAM J. on Applied Math.*, 27:17-30, 1974.

[22] S. Gal. Search games with mobile and immobile hider. *SIAM Journal on Control and Optimization*, 17(1):99-122, 1979.

[23] S. Gal. On the optimality of a simple strategy for searching graphs. *International Journal of Game Theory*, 29(4):533-542, 2001.

[24] M. X. Goemans and J. M. Kleinberg. An improved approximation ratio for the minimum latency problem. *Mathematical Programming*, 82:111-124, 1998.

[25] P. Jaillet and M. Stafford. Online searching. *Operations Research*, 49:234-244, 1993.

[26] M-Y. Kao and M.L. Littman. Algorithms for informed cows. In *Proceedings of the AAAI 1997 Workshop on Online Search*, 1997.

[27] M-Y. Kao, Y. Ma, M. Sipser, and Y.L. Yin. Optimal constructions of hybrid algorithms. *Journal of Algorithms*, 29(1):142-164, 1998.

[28] M-Y. Kao, J.H. Reif, and S.R. Tate. Searching in an unknown environment: an optimal randomized algorithm for the cow-path problem. *Inform. and Comp.*, 131(1):63-80, 1996.

[29] D. G. Kirkpatrick. Hyperbolic dovetailing. In *Proceedings of the 17th Annual European Symposium on Algorithms (ESA)*, pages 616-627, 2009.

[30] E. Koutsoupias, C.H. Papadimitriou, and M. Yannakakis. Searching a fixed graph. In *Proc. of the 23rd Int. Colloq. on Automata, Languages and Programming (ICALP)*, pages 280-289, 1996.

[31] T. Lidbetter. Search games with multiple hidden objects. *SIAM Journal on Control and Optimization*, 51(4):3056-3074, 2013.

[32] A. López-Ortiz and S. Schuierer. The ultimate strategy to search on m rays? *Theoretical Computer Science*, 261(2):267-295, 2001.

[33] A. López-Ortiz and S. Schuierer. On-line parallel heuristics, processor scheduling and robot searching under the competitive framework. *Theor. Comp. Sci.*, 310(1-3):527-537, 2004.

[34] A. McGregor, K. Onak, and R. Panigrahy. The oil searching problem. In Proc. of the 17th European Symposium on Algorithms (ESA), pages 504-515, 2009.

[35] L. Pavlovic. A search game on the union of graphs with immobile hider. *Naval Research Logistics*, 42(8):1177-1199, 1995.

[36] J. Reijnierse and J. Potters. Search games with immobile hider. *International Journal of Game Theory*, 21:385-394, 1993.

[37] S. Schuierer. A lower bound for randomized searching on m rays. In *Computer Science in Perspective*, pages 264-277, 2003.

[38] R. Sitters. The minimum latency problem is np-hard for weighted trees. In *Proceedings of the 9th Inernational Conference on Integer Programming and Combinatorial Optimization (IPCO)*, pages 230-239, 2002.

[39] B. von Stengel and R. Werchner. Complexity of searching an immobile hider in a graph. *Discrete Applied Mathematics*, 78(1):235-249, 1997.