

# Content Search Through Comparisons

Technicolor Technical Report  
CR-PRL-2010-07-0002

First Publication Date: Apr. 28th, 2011

Amin Karbasi<sup>1</sup>, Stratis Ioannidis<sup>2</sup>, and Laurent Massoulié<sup>3</sup>

<sup>1</sup> Ecole Polytechnique Federale de Lausanne, Lausanne, Switzerland

<sup>2</sup> Technicolor, Palo Alto, USA

<sup>3</sup> Technicolor, Paris, France

**Abstract.** We study the problem of navigating through a database of similar objects using comparisons under heterogeneous demand, a problem closely related to small-world network design. We show that, under heterogeneous demand, the small-world network design problem is NP-hard. Given the above negative result, we propose a novel mechanism for small-world network design and provide an upper bound on its performance under heterogeneous demand. The above mechanism has a natural equivalent in the context of content search through comparisons, again under heterogeneous demand; we use this to establish both upper and lower bounds on content search through comparisons. These bounds are intuitively appealing, as they depend on the entropy of the demand as well as its doubling constant, a quantity capturing the topology of the set of target objects. Finally, we propose an adaptive learning algorithm for content search that meets the performance guarantees achieved by the above mechanisms.





## 1 Introduction

The problem we study in this paper is content search through comparisons. In short, a user searching for a target object navigates through a database in the following manner. The user is asked to select the object most similar to her target from small list of objects. A new object list is then presented to the user based on her earlier selection. This process is repeated until the target is included in the list presented, at which point the search terminates.

Searching through comparisons is typical example of *exploratory search* [20], the need for which arises when users are unable to state and submit explicit queries to the database. Exploratory search has several important real-life applications. An often-cited example [19, 18] is navigating through a database of pictures of humans in which subjects are photographed under diverse uncontrolled conditions. For example, the pictures may be taken outdoors, from different angles or distances, while the subjects assume different poses, are partially obscured, *etc.* Automated methods may fail to extract meaningful features from such photos, so the database cannot be queried in the traditional fashion. On the other hand, a human searching for a particular person can easily select from a list of pictures the subject most similar to the person she has in mind.

Users may also be unable to state queries because, *e.g.*, they are unfamiliar with the search domain, or do not have a clear target in mind. For example, a novice classical music listener may not be able to express that she is, *e.g.*, looking for a fugue or a sonata. She might however identify among samples of different musical pieces the closest to the one she has in mind. Alternatively, a user surfing the web may not know a priori which post she wishes to read; presenting a list of blog posts and letting the surfer identify which one she likes best can steer her in the right direction.

In all the above applications, the problem of content search through comparisons amounts to determining which objects to present to the user in order to find the target object as quickly as possible. Formally, the behavior of a human user can be modeled by a so-called *comparison oracle* [10]: given a target and a choice between two objects, the oracle outputs the one closest to the target. The goal is thus to find a sequence of proposed pairs of objects that leads to the target object with as few oracle queries as possible. This problem was introduced in [10] and has recently received considerable attention [16, 18, 19].

Content search through comparisons is also naturally related to the following problem: given a graph embedded in a metric space, how should one augment this graph by adding edges in order to minimize the expected cost of greedy forwarding over this graph? This is known as the *small-world network design* problem [7, 6] and has a variety of applications as, *e.g.*, in network routing. In this paper, we consider both problems under the scenario of *heterogeneous demand*. This is very interesting in practice: objects in a database are indeed unlikely to be requested with the same frequency. Our contributions are as follows:

- We show that the small-world network design problem under general heterogeneous demand is NP-hard. Given earlier work on this problem under homogeneous demand [6, 7], this result is interesting in its own right.
- We propose a novel mechanism for edge addition in the small-world design problem, and provide an upper bound on its performance.
- The above mechanism has a natural equivalent in the context of content search through comparisons, and we provide a matching an upper bound for the performance of this mechanism.
- We establish a lower bound for the any mechanism for solving the problem of content search through comparisons.
- Based on these results, we propose an adaptive learning algorithm for content search that, given access only to a comparison oracle, can meet the performance guarantees achieved by the above mechanisms.

To the best of our knowledge, we are the first to study the above two problems in a setting of heterogeneous demand. Our analysis is intuitively appealing because our upper and lower bounds relate the cost of content search to two important properties of the demand distribution, namely its *entropy* and its *doubling constant*. We thus provide performance guarantees in terms of the *bias* of the distribution of targets, captured by the entropy, as well as the *topology* of their embedding, captured by the doubling constant.

The remainder of this paper is organized as follows. In Section 2 we provide an overview of the related work in this area. In Sections 3 and 4 we introduce our notation and formally state the two problems that are the focus of this work, namely content search through comparisons and small-world network design. We present our main results in Section 5 and our adaptive learning algorithm in Section 6. Finally, we conclude in Section 7.

## 2 Related Work

Content search through comparisons is a special case of nearest neighbour search (NNS) [3, 12], where it is typical to assume that database objects are embedded in a metric space with a small intrinsic dimension. Krauthgamer and Lee [15] introduce navigating nets, a data structure for NNS in doubling metric spaces. Clarkson [3] considers a similar structure for objects embedded in a space satisfying a sphere-packing property, while Karger and Ruhl [13] study NNS under growth-restricted metrics. All three assumptions have formal connections to the doubling constant we consider in this paper. However, in these works, the underlying metric space is fully observable by the search mechanism, and the demand over target objects is homogeneous. Our work assumes access only to a comparison oracle while also dealing with heterogeneous demand.

NNS with access to a comparison oracle was first introduced by Lifshits *et al.* [10], and further explored by Lifshits and Zhang [16] and Tshopp and Diggavi [18, 19]. In contrast to [13, 15, 3], the above authors do not assume that objects are necessarily embedded in a metric space; instead, they only require

that a comparison oracle can rank any two objects in terms of their similarity to a given target. To provide performance guarantees on the search cost, Lifshits *et al.* introduce a *disorder constant* [10], capturing the degree to which object rankings violate the triangle inequality. This disorder constant plays roughly the same role in their analysis as the doubling constant does in ours. Nevertheless, these works also assume homogeneous demand. Our work introduces the notion of heterogeneity while assuming that a metric embedding exists.

An additional important distinction between [10, 16, 18, 19] and our work is the existence of a learning phase, during which explicit questions are placed to the comparison oracle. A data-structure is constructed during this phase, which is subsequently used to answer queries submitted to the database during a “search” phase. The above works establish different tradeoffs between the length of the learning phase, the space complexity of the data structure created, and the cost incurred during searching. In contrast, the learning scheme we consider in Section 6 is adaptive, and learning occurs while users search; the drawback lies in that our guarantees on the search cost are asymptotic. Again, the main advantage of our approach lies in dealing with heterogeneity.

The use of interactive methods (*i.e.*, that incorporate human feedback) for content search has a long history in literature. Arguably, the first oracle considered to model such methods is the so-called membership oracle [8], which allows the search mechanism to ask a user questions of the form “does the target belong to set  $A$ ” (see also our discussion in Section 3). Branson *et al.* [2] deploy such an interactive method for object classification and evaluate it on the *Animals with attributes* database. A similar approach was used by Geman *et al.* [9] who formulated shape recognition as a coding problem and applied this approach to handwritten numerals and satellite images. Having access to a membership oracle however is a strong assumption, as humans may not necessarily be able to answer queries of the above type for *any* object set  $A$ . Moreover, the large number of possible sets makes the cost of designing optimal querying strategies over large datasets prohibitive. In contrast, the comparison oracle model makes a far weaker assumption on human behavior—namely, the ability to compare different objects to the target—and significantly limits the design space, making search mechanisms using comparisons practical even over large datasets.

Small-world networks (also called navigable networks) have received a lot of attention since Kleinberg’s seminal paper [14]. Our work is closest to Fraigneaud *et al.* [7], [6], who identify conditions under which graphs embedded in a doubling metric space are navigable. Again, our approach to small-world network design differs by considering heterogeneous demand, an aspect absent from earlier work. Our work is most similar to Fraigneaud *et al.* [7], where a condition under which graphs embedded in a doubling metric space can be made navigable is identified. The same idea was explored in more general spaces by Fraigneaud and Giakkoupis [6]. Again, the main difference in our approach to small-world network design lies in considering heterogeneous demand, an aspect of small-world networks not investigated in earlier work.

### 3 Definitions and Notation

**Comparison Oracle.** Consider a set of objects  $\mathcal{N}$ , where  $|\mathcal{N}| = n$ , and a metric space  $(\mathcal{M}, d)$ , where  $d(x, y)$  denotes the distance between  $x, y \in \mathcal{M}$ . Assume that objects in  $\mathcal{N}$  are embedded in  $(\mathcal{M}, d)$ , *i.e.*, there exists a 1-to-1 mapping from  $\mathcal{N}$  to a subset of  $\mathcal{M}$ . The objects in  $\mathcal{N}$  may represent, for example, pictures in a database. The metric embedding is a mapping from the pictures to a set of attributes (*e.g.*, the person’s age, her eye color, *etc.*). The distance  $d$  then represents how “similar” objects are w.r.t. these attributes. In what follows, we abuse notation and write  $\mathcal{N} \subseteq \mathcal{M}$ , keeping in mind that database objects (the pictures) are in fact distinct from their embedding (their attributes).

Given an object  $z \in \mathcal{N}$ , we write  $x \preceq_z y$  if  $d(x, z) \leq d(y, z)$ , ordering thus objects according to their distance from  $z$ . Moreover, we write  $x \sim_z y$  if  $d(x, z) = d(y, z)$  and  $x \prec_z y$  if  $x \preceq_z y$  but not  $x \sim_z y$ . For a non-empty  $A \subseteq \mathcal{N}$ , let  $\min_{\preceq_z} A$  be the set of objects in  $A$  closest to  $z$ , *i.e.*,  $w \in \min_{\preceq_z} A \subseteq A$  if  $w \preceq_z v$  for all  $v \in A$ .

A *comparison oracle* [10] is an oracle that, given two objects  $x, y$  and a target  $t$ , returns the closest object to  $t$ . More formally,

$$\text{Oracle}(x, y, t) = \begin{cases} x & \text{if } x \prec_t y, \\ y & \text{if } x \succ_t y, \\ x \text{ or } y & \text{if } x \sim_t y. \end{cases} \quad (1)$$

Observe that if  $x = \text{Oracle}(x, y, t)$  then  $x \preceq_t y$ ; this does not necessarily imply however that  $x \prec_t y$ . This oracle “models” human users: a user interested in locating, *e.g.*, a target picture  $t$  within the database, can compare two pictures with respect to their similarity to this target but cannot associate a numerical value to this similarity. When the two pictures are equidistant from  $t$  the user’s decision is arbitrary. It is important to note here that although we write  $\text{Oracle}(x, y, t)$  to stress that a query always takes place with respect to some target  $t$ , in practice the target is hidden and only known by the oracle. Alternatively, following the “oracle as human” analogy, the human user has a target in mind and uses it to compare the two objects, but never discloses it until actually being presented with it.

Note that our oracle is weaker than one that correctly identifies the relationship  $x \sim_t y$  and, *e.g.*, returns a special character “=” once two such objects are proposed: to see this, observe that oracle (1) can be implemented by using this stronger oracle. Hence, all our results hold if we are provided with such an oracle instead.

**Entropy and Doubling Constant.** We denote by  $\mathcal{N} \times \mathcal{N}$  the set of all ordered pairs of objects in  $\mathcal{N}$ . For any ordered pair  $(s, t) \in \mathcal{N} \times \mathcal{N}$ , we call  $s$  the *source* and  $t$  the *target* of the pair. We consider a probability distribution  $\lambda$  over all ordered pairs of objects in  $\mathcal{N}$  which we call the *demand*. We refer to the marginal distributions  $\nu(s) = \sum_t \lambda(s, t)$  and  $\mu(t) = \sum_s \lambda(s, t)$ , as the *source* and *target* distributions, respectively. Moreover, we refer to the support of the

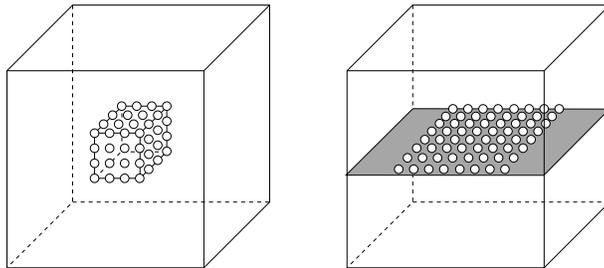
target distribution  $\mathcal{T} = \text{supp}(\mu) = \{x \in \mathcal{N} : \text{s.t. } \mu(x) > 0\}$  as the *target set* of the demand.

Let  $\sigma$  be a probability distribution over  $\mathcal{N}$ . We define the *entropy* and *max-entropy* of  $\sigma$ , respectively, as

$$H(\sigma) = \sum_{x \in \text{supp}(\sigma)} \sigma(x) \log \sigma^{-1}(x), \quad H_{\max}(\sigma) = \max_{x \in \text{supp}(\sigma)} \log \sigma^{-1}(x). \quad (2)$$

The entropy has strong connections with content search. More specifically, suppose that we have access to a so-called *membership oracle* [5] that answers queries of the following form: ‘‘Given a target  $t$  and a subset  $A \subseteq \mathcal{N}$ , does  $t$  belong to  $A$ ?’’ Let  $t$  be a random target selected with distribution  $\mu$ . To identify  $t$  one needs to submit at least  $H(\mu)$  queries, in expectation, to a membership oracle, and there exists an algorithm (Huffman coding) that identifies  $t$  with only  $H(\mu) + 1$  queries, in expectation (see, *e.g.*, [5]). In the worst case, which occurs when the target is the least frequently selected object, the algorithm requires  $H_{\max}(\mu) + 1$  queries to identify  $t$ . Our work identifies similar bounds assuming that one only has access to a comparison oracle, as defined in (1). Not surprisingly, the entropy  $H(\mu)$  also shows up in our performance bounds (Theorems 3 and 4).

For  $x \in \mathcal{N}$ , we denote by  $B_x(r) = \{y \in \mathcal{M} : d(x, y) \leq r\}$  the closed ball of radius  $r \geq 0$  around  $x$ . Given a probability distribution  $\sigma$  over  $\mathcal{N}$  and a set  $A \subset \mathcal{N}$  let  $\sigma(A) = \sum_{x \in A} \sigma(x)$ . We define the *doubling constant*  $c(\sigma)$  to be the minimum  $c > 0$  for which  $\sigma(B_x(2r)) \leq c \cdot \sigma(B_x(r))$ , for any  $x \in \text{supp}(\sigma)$  and any  $r \geq 0$ . Moreover, we say that  $\sigma$  is *c-doubling* if  $c(\mu) = c$ . Note that,



**Fig. 1.** Example of dependence of  $c(\sigma)$  on the topology of the support  $\text{supp}(\sigma)$ . When  $\text{supp}(\sigma)$  consists of  $n = 64$  objects arranged in a cube,  $c(\sigma) = 2^3$ . If, on the other hand, these  $n$  objects are placed on a plane,  $c(\sigma) = 2^2$ . In both cases  $\sigma$  is assumed to be uniform, and  $H(\sigma) = \log N$ .

contrary to the entropy,  $c(\sigma)$  depends on the topology of  $\text{supp}(\sigma)$ , determined by the embedding of  $\mathcal{N}$  in  $(\mathcal{M}, d)$ . This is illustrated in Fig. 1. In this example,  $|\mathcal{N}| = 64$ , and the set  $\mathcal{N}$  is embedded in a 3-dimensional cube. Assume that  $\sigma$  is the uniform distribution over the  $N$  objects; if these objects are arranged

uniformly in a cube, then  $c(\sigma) = 2^3$ ; if however these  $n$  objects are arranged uniformly in a 2-dimensional plane,  $c(\sigma) = 2^2$ . Note that, in contrast, the entropy of  $\sigma$  in both cases equals  $\log n$  (and so does the max-entropy). As we will see, search through comparisons depends not only on the entropy  $H(\mu)$  but also on the topology of  $\mu$ , as captured by  $c(\mu)$ .

**Table 1.** Summary of Notation

$\mathcal{N}$	Set of objects
$(\mathcal{M}, d)$	Metric space
$d(x, y)$	Distance between $x, y \in \mathcal{M}$
$x \preceq_z y$	Ordering w.r.t. distance from $z$
$x \sim_z y$	$x$ and $y$ at same distance from $z$
$\lambda$	The demand distribution
$\nu$	The source distribution
$\mu$	The target distribution
$\mathcal{T}$	The target set
$H(\sigma)$	The entropy of $\sigma$
$H_{\max}(\sigma)$	The max-entropy of $\sigma$
$B_x(r)$	The ball of radius $r$ centered at $x$
$c(\sigma)$	The doubling constant of $\sigma$

## 4 Problem Statement

We now formally define the two problems we study. The first is *content search through comparisons* and the second is the *small-world network design* problem.

### 4.1 Content Search Through Comparisons

Consider the object set  $\mathcal{N}$ . Although its embedding in  $(\mathcal{M}, d)$  exists, we are constrained by not being able to directly compute object distances; instead, we only have access to a comparison oracle. In particular, we define *greedy content search* as follows. Let  $t$  be a target and  $s$  an object serving as a starting point. The greedy content search algorithm proposes an object  $w$  and asks the oracle to select among  $s$  and  $w$  the object closest to  $t$ , *i.e.*, it evokes  $\text{Oracle}(s, w, t)$ . This is repeated until the oracle returns something other than  $s$ , say  $w'$ . If  $w' \neq t$ , the algorithm repeats these steps, now from  $w'$ . If  $w' = t$ , the search terminates.

Formally, for  $k = 1, 2, \dots$ , let  $x_k, y_k$  be the  $k$ -th pair of objects submitted to the oracle:  $x_k$  is the *current object*, which greedy content search is trying to improve upon, and  $y_k$  is the *proposed object*, submitted to the oracle for comparison with  $x_k$ . Let  $o_k = \text{Oracle}(x_k, y_k, t) \in \{x_k, y_k\}$  be the oracle's response,

and define the *history* of the search up to and including the  $k$ -th access as  $\mathcal{H}_k = \{(x_i, y_i, o_i)\}_{i=1}^k$ .

The source object is always one of the first two objects submitted to the oracle, *i.e.*,  $x_1 = s$ . Moreover,  $x_{k+1} = o_k$ , *i.e.*, the current object is always the closest to the target so far. The selection of the proposed object  $y_{k+1}$  is determined by the history  $\mathcal{H}_k$  and the object  $x_k$ . In particular, given  $\mathcal{H}_k$  and the current object  $x_k$  there exists a mapping  $(\mathcal{H}_k, x_k) \mapsto \mathcal{F}(\mathcal{H}_k, x_k) \in \mathcal{N}$  such that  $y_{k+1} = \mathcal{F}(\mathcal{H}_k, x_k)$ , where here we take  $x_0 = s \in \mathcal{N}$  (the source/starting object) and  $\mathcal{H}_0 = \emptyset$  (*i.e.*, before any comparison takes place, there is no history).

We call the mapping  $\mathcal{F}$  the *selection policy* of the greedy content search. In general, we allow the selection policy to be randomized; in this case, the object returned by  $\mathcal{F}(\mathcal{H}_k, x_k)$  is a random variable, whose distribution  $\Pr(\mathcal{F}(\mathcal{H}_k, x_k) = w)$  for  $w \in \mathcal{N}$  is fully determined by  $(\mathcal{H}_k, x_k)$ . Observe that  $\mathcal{F}$  depends on the target  $t$  only indirectly, through  $\mathcal{H}_k$  and  $x_k$ ; this is because  $t$  is only “revealed” when the search terminates. We say that a selection policy is *memoryless* if it depends on  $x_k$  but not on the history  $\mathcal{H}_k$ .

Our goal is to select an  $\mathcal{F}$  that minimizes the number of accesses to the oracle. In particular, given a source object  $s$ , a target  $t$  and a selection policy  $\mathcal{F}$ , we define the search cost  $C_{\mathcal{F}}(s, t) = \inf\{k : y_k = t\}$  to be the number of proposals to the oracle until  $t$  is found. This is a random variable, as  $\mathcal{F}$  is randomized; let  $\mathbb{E}[C_{\mathcal{F}}(s, t)]$  be its expectation. We thus define the following problem.

CONTENT SEARCH THROUGH COMPARISONS (CSTC): Given an embedding of  $\mathcal{N}$  into  $(\mathcal{M}, d)$  and a demand distribution  $\lambda(s, t)$ , select  $\mathcal{F}$  that minimizes the expected search cost  $\bar{C}_{\mathcal{F}} = \sum_{(s,t) \in \mathcal{N} \times \mathcal{N}} \lambda(s, t) \mathbb{E}[C_{\mathcal{F}}(s, t)]$ .

Note that, as  $\mathcal{F}$  is randomized, the free variable in the above optimization problem is the distribution  $\Pr(\mathcal{F}(\mathcal{H}_k, x_k) = w)$ .

## 4.2 Small-World Network Design

In the small-world network design problem, we assume that the objects in  $\mathcal{N}$ , embedded in  $(\mathcal{M}, d)$ , are connected to each other. The network formed by such connections is represented by a directed graph  $G(\mathcal{N}, \mathcal{L} \cup \mathcal{S})$ , where  $\mathcal{L} \cap \mathcal{S} = \emptyset$ ,  $\mathcal{L}$  is the set of *local* edges and  $\mathcal{S}$  is the set of *shortcut* edges. The edges in  $\mathcal{L}$  are typically assumed to satisfy the following property:

*Property 1.* For every pair of distinct objects  $x, t \in \mathcal{N}$  there exists an object  $u$  such that  $(x, u) \in \mathcal{L}$  and  $u \prec_t x$ .

*I.e.*, for any  $x$  and  $t$ ,  $x$  has a local edge leading to an object closer to  $t$ .

A comparison oracle can be used to route a message from  $s$  to  $t$  over the edges in graph  $G$ . In particular, given graph  $G$ , we define *greedy forwarding* [14] over  $G$  as follows. Let  $\Gamma(s)$  be the neighborhood of  $s$ , *i.e.*,  $\Gamma(s) = \{u \in \mathcal{N} \text{ s.t. } (s, u) \in \mathcal{L} \cup \mathcal{S}\}$ . Given a source  $s$  and a target  $t$ , greedy forwarding sends a message to neighbor  $w$  of  $s$  that is as close to  $t$  as possible, *i.e.*,  $w \in \min_{\prec_t} \Gamma(s)$ . If  $w \neq t$ , the above process is repeated at  $w$ ; if  $w = t$ , greedy forwarding terminates.

Property 1 guarantees that greedy forwarding from any source  $s$  will eventually reach  $t$ : there is always a neighbor closer to  $t$  than the object/node forwarding the message. Moreover, if the message is at  $x$ , the closest neighbor  $w$  can be found using  $|I(x)|$  queries to a comparison oracle.

The edges in  $\mathcal{L}$  are typically called “local” because they are usually determined by object proximity. For example, in the classical paper by Kleinberg [14], objects are arranged uniformly in a rectangular  $k$ -dimensional grid—with no gaps—and  $d$  is taken to be the Manhattan distance on the grid. Moreover, there exists an  $r \geq 1$  such that

$$\mathcal{L} = \{(x, y) \in \mathcal{N} \times \mathcal{N} \text{ s.t. } d(x, y) \leq r\}. \quad (3)$$

Assuming every position in the rectangular grid is occupied, such edges indeed satisfy Property 1. In this work, we *do not* require that edges in  $\mathcal{L}$  are given by any locality-based definition like (3); our only assumption is that they satisfy Property 1 though, for consistency, we also refer to edges in  $\mathcal{L}$  as “local”.

Our goal is to select the shortcut edges in  $\mathcal{S}$  so that greedy forwarding is as efficient as possible. In particular, assume that we can select no more than  $\beta$  shortcut edges, where  $\beta$  is a positive integer. For  $S$  a subset of  $\mathcal{N} \times \mathcal{N}$  such that  $|S| \leq \beta$ , we denote by  $C_S(s, t)$  the cost of greedy forwarding, in message hops, for forwarding a message from  $s$  to  $t$  given that  $\mathcal{S} = S$ . We allow the selection of shortcut edges to be random: the set  $\mathcal{S}$  can be a random variable over all subsets  $S$  of  $\mathcal{N} \times \mathcal{N}$  such that  $|S| \leq \beta$ . We denote the distribution of  $\mathcal{S}$  by  $\Pr(\mathcal{S} = S)$  for  $S \subseteq \mathcal{N} \times \mathcal{N}$  such that  $|S| \leq \beta$ . Given a source  $s$  and a target  $t$ , let  $\mathbb{E}[C_{\mathcal{S}}(s, t)] = \sum_{S \subseteq \mathcal{N} \times \mathcal{N}; |S| \leq \beta} C_S(s, t) \cdot \Pr(\mathcal{S} = S)$  be the expected cost of forwarding a message from  $s$  to  $t$  with greedy forwarding, in message hops.

We consider again a heterogeneous demand: a source and target object are selected at random from  $\mathcal{N} \times \mathcal{N}$  according to a demand probability distribution  $\lambda$ . The *small-world network design problem* can then be formulated as follows.

**SMALL-WORLD NETWORK DESIGN (SWND):** Given an embedding of  $\mathcal{N}$  into  $(\mathcal{M}, d)$ , a set of local edges  $\mathcal{L}$ , a demand distribution  $\lambda$ , and an integer  $\beta > 0$ , select a r.v.  $\mathcal{S} \subset \mathcal{N} \times \mathcal{N}$ , where  $|\mathcal{S}| \leq \beta$ , that minimizes  $\bar{C}_{\mathcal{S}} = \sum_{(s,t) \in \mathcal{N} \times \mathcal{N}} \lambda(s, t) \mathbb{E}[C_{\mathcal{S}}(s, t)]$ .

In other words, we wish to select  $\mathcal{S}$  so that the cost of greedy forwarding is minimized. Note again that the free variable of SWND is the distribution of  $\mathcal{S}$ .

## 5 Main Results

We now present our main results with respect to SWND and CSTC. Our first result is negative: optimizing greedy forwarding is a hard problem.

**Theorem 1.** *SWND is NP-hard.*

The proof of this theorem can be found in Appendix A. In short, the proof reduces DOMINATINGSET to the decision version of SWND. Interestingly, the

reduction is to a SWND instance in which (a) the metric space is a 2-dimensional grid, (b) the distance metric is the Manhattan distance on the grid and (c) the local edges are given by (3). Thus, SWND remains NP-hard even in the classic setup considered by Kleinberg [14].

The NP-hardness of SWND suggests that this problem cannot be solved in its full generality. Motivated by this, as well as its relationship to content search through comparisons, we focus our attention to the following version of the SWND problem, in which we place additional restrictions to the shortcut edge set  $\mathcal{S}$ . First,  $|\mathcal{S}| = |\mathcal{N}|$ , and for every  $x \in \mathcal{N}$  there exists *exactly one* directed edge  $(x, y) \in \mathcal{S}$ . Second, the object  $y$  to which  $x$  connects is selected independently at each  $x$ , according to a probability distribution  $\ell_x(y)$ . *I.e.*, for  $\mathcal{N} = \{x_1, x_2, \dots, x_n\}$ , the joint distribution of shortcut edges has the form:

$$\Pr(\mathcal{S} = \{(x_1, y_1), \dots, (x_n, y_n)\}) = \prod_{i=1}^n \ell_{x_i}(y_i). \quad (4)$$

We call this version of the SWND problem the *one edge per object* version, and denote it by 1-SWND. Note that, in 1-SWND, the free variables are the distributions  $\ell_x$ ,  $x \in \mathcal{N}$ .

For a given demand  $\lambda$ , recall that  $\mu$  is the marginal distribution of the demand  $\lambda$  over the target set  $\mathcal{T}$ , and that for  $A \subset \mathcal{N}$ ,  $\mu(A) = \sum_{x \in A} \mu(x)$ . Then, for any two objects  $x, y \in \mathcal{N}$ , we define the *rank* of object  $y$  w.r.t. object  $x$  as follows:

$$r_x(y) \equiv \mu(B_x(d(x, y))) \quad (5)$$

where  $B_x(r)$  is the closed ball with radius  $r$  centered at  $x$ .

Suppose now that shortcut edges are generated according to the joint distribution (4), where the outgoing link from an object  $x \in \mathcal{N}$  is selected according to the following probability:

$$\ell_x(y) \propto \frac{\mu(y)}{r_x(y)}, \quad (6)$$

for  $y \in \text{supp}(\mu)$ , while for  $y \notin \text{supp}(\mu)$  we define  $\ell_x(y)$  to be zero. Eq. (6) implies the following appealing properties. For two objects  $y, z$  that have the same distance from  $x$ , if  $\mu(y) > \mu(z)$  then  $\ell_x(y) > \ell_x(z)$ , *i.e.*,  $y$  has a higher probability of being connected to  $x$ . When two objects  $y, z$  are equally likely to be targets, if  $y \prec_x z$  then  $\ell_x(y) > \ell_x(z)$ . The distribution (6) thus biases both towards objects close to  $x$  as well as towards objects that are likely to be targets. Finally, if the metric space  $(\mathcal{M}, d)$  is a  $k$ -dimensional grid and the targets are uniformly distributed over  $\mathcal{N}$  then  $\ell_x(y) \propto (d(x, y))^{-k}$ . This is the shortcut distribution used by Kleinberg in [14]; (6) is thus a generalization of this distribution to heterogeneous targets as well as to more general metric spaces.

Our next theorem, whose proof is in Section 5.1, relates the cost of greedy forwarding under (6) to the entropy  $H$ , the max-entropy  $H_{\max}$  and the doubling parameter  $c$  of the target distribution  $\mu$ .

**Theorem 2.** *Given a demand  $\lambda$ , consider the set of shortcut edges  $\mathcal{S}$  sampled according to (4), where  $\ell_x(y)$ ,  $x, y \in \mathcal{N}$ , are given by (6). Then*

$$\bar{C}_{\mathcal{S}} \leq 6c^3(\mu) \cdot H(\mu) \cdot H_{\max}(\mu).$$

Note that the bound in Theorem 2 depends on  $\lambda$  only through the target distribution  $\mu$ . In particular, it holds for *any* source distribution  $\nu$ , and *does not require* that sources are selected independently of the targets  $t$ . Moreover, if  $\mathcal{N}$  is a  $k$ -dimensional grid and  $\mu$  is the uniform distribution over  $\mathcal{N}$ , the above bound becomes  $O(\log^2 n)$ , retrieving thus the classic result of Kleinberg [14].

Exploiting an underlying relationship between 1-SWND and CSTC, we can obtain an efficient selection policy for greedy content search. In particular,

**Theorem 3.** *Given a demand  $\lambda$ , consider the memoryless selection policy  $\Pr(\mathcal{F}(\mathcal{H}_k, x_k) = w) = \ell_{x_k}(w)$  where  $\ell_x$  is given by (6). Then*

$$\bar{C}_{\mathcal{F}} \leq 6c^3(\mu) \cdot H(\mu) \cdot H_{\max}(\mu).$$

The proof of this theorem is given in Section 5.2. Like Theorem 2, Theorem 3 characterises the search cost in terms of the doubling constant, the entropy and the max-entropy of  $\mu$ . This is very appealing, given (a) the relationship between  $c(\mu)$  and the topology of the target set and (b) the classic result regarding the entropy and accesses to a membership oracle, as outlined in Section 3.

A question arising from Theorems 2 and 3 is how tight these bounds are. Intuitively, we expect that the optimal shortcut set  $\mathcal{S}$  and the optimal selection policy  $\mathcal{F}$  depend both on the entropy of the target distribution and on its doubling constant. Our next theorem, whose proof is in Section 5.3, establishes that this is the case for  $\mathcal{F}$ .

**Theorem 4.** *For any integer  $K$  and  $D$ , there exists a metric space  $(\mathcal{M}, d)$  and a target measure  $\mu$  with entropy  $H(\mu) = K \log(D)$  and doubling constant  $c(\mu) = D$  such that the average search cost of any selection policy  $\mathcal{F}$  satisfies*

$$\bar{C}_{\mathcal{F}} \geq H(\mu) \frac{c(\mu) - 1}{2 \log(c(\mu))}. \quad (7)$$

Hence, the bound in Theorem 3 is tight within a  $c^2(\mu) \log(c(\mu)) H_{\max}$  factor.

### 5.1 Proof of Theorem 2

According to (6), the probability that object  $x$  links to  $y$  is given by  $\ell_x(y) = \frac{1}{Z_x} \frac{\mu(y)}{r_x(y)}$ , where  $Z_x = \sum_{y \in \mathcal{T}} \frac{\mu(y)}{r_x(y)}$  is a normalization factor bounded as follows.

**Lemma 1.** *For any  $x \in \mathcal{N}$ , let  $x^* \in \min_{\prec_x} \mathcal{T}$  be any object in  $\mathcal{T}$  among the closest targets to  $x$ . Then  $Z_x \leq 1 + \ln(1/\mu(x^*)) \leq 3H_{\max}$ .*

*Proof.* Sort the target set  $\mathcal{T}$  from the closest to furthest object from  $x$  and index objects in an increasing sequence  $i = 1, \dots, k$ , so the objects at the same distance

from  $x$  receive the same index. Let  $A_i, i = 1, \dots, k$ , be the set containing objects indexed by  $i$ , and let  $\mu_i = \mu(A_i)$  and  $\mu_0 = \mu(x)$ . Furthermore, let  $Q_i = \sum_{j=0}^i \mu_j$ . Then  $Z_x = \sum_{i=1}^k \frac{\mu_i}{Q_i}$ . Define  $f_x(r) : \mathbb{R}^+ \rightarrow \mathbb{R}$  as  $f_x(r) = \frac{1}{r} - \mu(x)$ . Clearly,  $f_x(\frac{1}{Q_i}) = \sum_{j=1}^i \mu_j$ , for  $i \in \{1, 2, \dots, k\}$ . This means that we can rewrite  $Z_x$  as  $Z_x = \sum_{i=1}^k (f_x(1/Q_i) - f_x(1/Q_{i-1}))/Q_i$ . By reordering the terms involved in the sum above, we get  $Z_x = f_x(\frac{1}{Q_k})/Q_k + \sum_{i=1}^{k-1} f_x(1/Q_i)(\frac{1}{Q_i} - \frac{1}{Q_{i+1}})$ . First note that  $Q_k = 1$ , and second that since  $f_x(r)$  is a decreasing function,  $Z_x \leq 1 - \mu_0 + \int_{1/Q_k}^{1/Q_1} f_x(r)dr = 1 - \frac{\mu_0}{Q_1} + \ln \frac{1}{Q_1}$ . This shows that if  $\mu_0 = 0$  then  $Z_x \leq 1 + \ln \frac{1}{\mu_1}$  or otherwise  $Z_x \leq 1 + \ln \frac{1}{\mu_0}$ .  $\square$

Given the set  $\mathcal{S}$ , recall that  $C_{\mathcal{S}}(s, t)$  is the number of steps required by the greedy forwarding to reach  $t \in \mathcal{N}$  from  $s \in \mathcal{N}$ . We say that a message at object  $v$  is in phase  $j$  if  $2^j \mu(t) \leq r_t(v) \leq 2^{j+1} \mu(t)$ . Notice that the number of different phases is at most  $\log_2 1/\mu(t)$ . We can write  $C_{\mathcal{S}}(s, t)$  as

$$C_{\mathcal{S}}(s, t) = X_1 + X_2 + \dots + X_{\log \frac{1}{\mu(t)}}, \quad (8)$$

where  $X_j$  are the hops occurring in phase  $j$ . Assume that  $j > 1$ , and let  $I = \{w \in \mathcal{N} : r_t(w) \leq \frac{r_t(v)}{2}\}$ . The probability that  $v$  links to an object in the set  $I$ , and hence moving to phase  $j - 1$ , is  $\sum_{w \in I} \ell_{v,w} = \frac{1}{Z_v} \sum_{w \in I} \frac{\mu(w)}{r_v(w)}$ . Let  $\mu_t(r) = \mu(B_t(r))$  and  $\rho > 0$  be the smallest radius such that  $\mu_t(\rho) \geq r_t(v)/2$ . Since we assumed that  $j > 1$  such a  $\rho > 0$  exists. Clearly, for any  $r < \rho$  we have  $\mu_t(r) < r_t(v)/2$ . In particular,  $\mu_t(\rho/2) < \frac{1}{2}r_t(v)$ . On the other hand, since the doubling parameter is  $c(\mu)$  we have  $\mu_t(\rho/2) > \frac{1}{c(\mu)}\mu_t(\rho) \geq \frac{1}{2c(\mu)}r_t(v)$ . Therefore,

$$\frac{1}{2c(\mu)}r_t(v) < \mu_t(\rho/2) < \frac{1}{2}r_t(v). \quad (9)$$

Let  $I_\rho = B_t(\rho)$  be the set of objects within radius  $\rho/2$  from  $t$ . Then  $I_\rho \subset I$ , so  $\sum_{w \in I} \ell_{v,w} \geq \frac{1}{Z_v} \sum_{w \in I_\rho} \frac{\mu(w)}{r_v(w)}$ . By triangle inequality, for any  $w \in I_\rho$  and  $y$  such that  $d(y, v) \leq d(v, w)$  we have  $d(t, y) \leq \frac{5}{2}d(v, t)$ . This means that  $r_v(w) \leq \mu_t(\frac{5}{2}d(v, t))$ , and consequently,  $r_v(w) \leq c^2(\mu)r_t(v)$ . Therefore,  $\sum_{w \in I} \ell_{v,w} \geq \frac{1}{Z_v} \frac{\sum_{w \in I_\rho} \mu(w)}{c^2(\mu)r_t(v)} = \frac{1}{Z_v} \frac{\mu_t(\rho/2)}{c^2(\mu)r_t(v)}$ . By (9), the probability of terminating phase  $j$  is uniformly bounded by

$$\sum_{w \in I} \ell_{v,w} \geq \min_v \frac{1}{2c^3(\mu)Z_v} \stackrel{\text{Lem. 1}}{\geq} \frac{1}{6c^3(\mu)H_{\max}(\mu)} \quad (10)$$

As a result, the probability of terminating phase  $j$  is stochastically dominated by a geometric random variable with the parameter given in (10). This is because (a) if the current object does not have a shortcut edge which lies in the set  $I$ , the greedy forwarding sends the message to one of the neighbours that is closer to  $t$  and (b) shortcut edges are sampled independently. Hence, given that  $t$  is the target object and  $s$  is the source object,

$$\mathbb{E}[X_j | s, t] \leq 6c^3(\mu)H_{\max}(\mu). \quad (11)$$

Suppose now that  $j = 1$ . By the triangle inequality,  $B_v(d(v, t)) \subseteq B_t(2d(v, t))$  and  $r_v(t) \leq c(\mu)r_t(v)$ . Hence,  $\ell_{v,t} \geq \frac{1}{Z_v} \frac{\mu(t)}{c(\mu)r_t(v)} \geq \frac{1}{2c(\mu)Z_v} \geq \frac{1}{6c(\mu)H_{\max}(\mu)}$  since object  $v$  is in the first phase and thus  $\mu(t) \leq r_t(v) \leq 2\mu(t)$ . Consequently,

$$\mathbb{E}[X_1|s, t] \leq 6c(\mu)H_{\max}(\mu). \quad (12)$$

Combining (8), (11), (12) and using the linearity of expectation, we get  $\mathbb{E}[C_S(s, t)] \leq 6c^3(\mu)H_{\max}(\mu) \log \frac{1}{\mu(t)}$  and, thus,  $\bar{C}_S \leq 6c^3(\mu)H_{\max}(\mu)H(\mu)$ .  $\square$

## 5.2 Proof of Theorem 3

The idea of the proof is very similar to the previous one and follows the same path. Recall that the selection policy is memoryless and determined by

$$\Pr(\mathcal{F}(\mathcal{H}_k, x_k) = w) = \ell_{x_k}(w).$$

We assume that the desired object is  $t$  and the content search starts from  $s$ . Since there are no local edges, the only way that the greedy search moves from the current object  $x_k$  is by proposing an object that is closer to  $t$ . Like in the SWND case, we are in particular interested in bounding the probability that the rank of the proposed object is roughly half the rank of the current object. This way we can compute how fast we make progress in our search.

As the search moves from  $s$  to  $t$  we say that the search is in phase  $j$  when the rank of the current object  $x_k$  is between  $2^j\mu(t)$  and  $2^{j+1}\mu(t)$ . As stated earlier, the greedy search algorithm keeps making comparisons until it finds another object closer to  $t$ . We can write  $C_{\mathcal{F}}(s, t)$  as

$$C_{\mathcal{F}}(s, t) = X_1 + X_2 + \dots + X_{\log \frac{1}{\mu(t)}},$$

where  $X_j$  denotes the number of comparisons done by comparison oracle in phase  $j$ . Let us consider a particular phase  $j$  and denote  $I$  the set of objects whose ranks from  $t$  are at most  $r_t(x_k)/2$ . Note that phase  $j$  will terminate if the comparison oracle proposes an object from set  $I$ . The probability that this happens is

$$\sum_{w \in I} \Pr(\mathcal{F}(\mathcal{H}_k, x_k) = w) = \sum_{w \in I} \ell_{x_k, w}.$$

Note that the sum on the right hand side depends on the distribution of shortcut edges and is independent of local edges. To bound this sum we can use (10). Hence, with probability at least  $1/(6c^3(\mu)H_{\max}(\mu))$ , phase  $j$  will terminate. In other words, using the above selection policy, if the current object  $x_k$  is in phase  $j$ , with probability  $1/(6c^3(\mu)H_{\max}(\mu))$  the proposed object will be in phase  $(j - 1)$ . This defines a geometric random variable which yields to the fact that on average the number of queries needed to halve the rank is at most  $6c(\mu)^3H_{\max}$  or  $\mathbb{E}[X_j|s, t] \leq 6c(\mu)^3H_{\max}$ . Taking average over the demand  $\lambda$ , we can conclude that the average number of comparisons is less than  $\bar{C}_{\mathcal{F}} \leq 6c^3(\mu)H_{\max}(\mu)H(\mu)$ .  $\square$

### 5.3 Proof of Theorem 4

Our proof amounts to constructing a metric space and a target distribution  $\mu$  for which the bound holds. Our construction will be as follows. For some integers  $D, K$ , the target set  $\mathcal{N}$  is taken as  $\mathcal{N} = \{1, \dots, D\}^K$ . The distance  $d(x, y)$  between two distinct elements  $x, y$  of  $\mathcal{N}$  is defined as  $d(x, y) = 2^m$ , where

$$m = \max \{i \in \{1, \dots, K\} : x(K-i) \neq y(K-i)\}.$$

We then have the following

**Lemma 2.** *Let  $\mu$  be the uniform distribution over  $\mathcal{N}$ . Then (i)  $c(\mu) = D$ , and (ii) if the target distribution is  $\mu$ , the optimal average search cost  $C^*$  based on a comparison oracle satisfies  $C^* \geq K \frac{D-1}{2}$ .*

Before proving Lemma 2, we note that Thm. 4 immediately follows as a corollary.

*Proof (of Lemma 2).* Part (i): Let  $x = (x(1), \dots, x(K)) \in \mathcal{N}$ , and fix  $r > 0$ . Assume first that  $r < 2$ ; then, the ball  $B(x, r)$  contains only  $x$ , while the ball  $B(x, 2r)$  contains either only  $x$  if  $r < 1$ , or precisely those  $y \in \mathcal{N}$  such that  $(y(1), \dots, y(K-1)) = (x(1), \dots, x(K-1))$  if  $r \geq 1$ . In the latter case  $B(x, 2r)$  contains precisely  $D$  elements. Hence, for such  $r < 2$ , and for the uniform measure on  $\mathcal{N}$ , the inequality

$$\mu(B(x, 2r)) \leq D\mu(B(x, r)) \tag{13}$$

holds, and with equality if in addition  $r \geq 1$ .

Consider now the case where  $r \geq 2$ . Let the integer  $m \geq 1$  be such that  $r \in [2^m, 2^{m+1})$ . By definition of the metric  $d$  on  $\mathcal{N}$ , the ball  $B(x, r)$  consists of all  $y \in \mathcal{N}$  such that  $(y(1), \dots, y(K-m)) = (x(1), \dots, x(K-m))$ , and hence contains  $D^{\min(K, m)}$  points. Similarly, the ball  $B(x, 2r)$  contains  $D^{\min(K, m+1)}$  points. Hence (13) also holds when  $r \geq 2$ .

Part (ii): We assume that the comparison oracle, in addition to returning one of the two proposals that is closer to the target, also reveals the distance of the proposal it returns to the target. We further assume that upon selection of the initial search candidate  $x_0$ , its distance to the unknown target is also revealed. We now establish that the lower bound on  $C^*$  holds when this additional information is available; it holds a fortiori for our more restricted comparison oracle.

We decompose the search procedure into phases, depending on the current distance to the destination. Let  $L_0$  be the integer such that the initial proposal  $x_0$  is at distance  $2^{L_0}$  of the target  $t$ , i.e.  $(x_0(1), \dots, x_0(K-L_0)) = (t(1), \dots, t(K-L_0))$ ,  $x_0(K-L_0+1) \neq t(K-L_0+1)$ . No information on  $t$  can be obtained by submitting proposals  $x$  such that  $d(x, x_0) \neq 2^{L_0}$ . Thus, to be useful, the next proposal  $x$  must share its  $(K-L_0)$  first components with  $x_0$ , and differ from  $x_0$  in its  $(K-L_0+1)$ -th entry. Now, keeping track of previous proposals made for which the distance to  $t$  remained equal to  $2^{L_0}$ , the best choice for the next proposal consists in picking it again at distance  $2^{L_0}$  from  $x_0$ , but choosing for its  $(K-L_0+1)$ -th entry one that has not been proposed so far. It is easy to see that, with this strategy, the number of additional proposals after  $x_0$  needed to

leave this phase is uniformly distributed on  $\{1, \dots, D-1\}$ , the number of options for the  $(K - L_0 + 1)$ -th entry of the target.

A similar argument entails that the number of proposals made in each phase equals 1 plus a uniform random variable on  $\{1, \dots, D-1\}$ . It remains to control the number of phases. We argue that it admits a Binomial distribution, with parameters  $(K, (D-1)/D)$ . Indeed, as we make a proposal which takes us into a new phase, no information is available on the next entries of the target, and for each such entry, the new proposal makes a correct guess with probability  $1/D$ . This yields the announced Binomial distribution for the numbers of phases (when it equals 0, the initial proposal  $x_0$  coincided with the target).

Thus the optimal number of search steps  $C$  verifies  $C \geq \sum_{i=1}^X (1 + Y_i)$ , where the  $Y_i$  are i.i.d., uniformly distributed on  $\{1, \dots, D-1\}$ , and independent of the random variable  $X$ , which admits a Binomial distribution with parameters  $(K, (D-1)/D)$ . Thus using Wald's identity, we obtain that  $\mathbb{E}[C] \geq \mathbb{E}[X]\mathbb{E}[Y_1]$ , which readily implies (ii).  $\square$

Note that the lower bound in (ii) has been established for search strategies that utilize the entire search history. Hence, it is *not* restricted to memoryless search.

## 6 Learning Algorithm

Section 5 established bounds on the cost of greedy content search provided that the distribution (6) is used to propose items to the oracle. Hence, if the embedding of  $\mathcal{N}$  in  $(\mathcal{M}, d)$  and target distribution  $\mu$  are known, it is possible to perform greedy content search with the performance guarantees provided by Theorem 3.

In this section, we turn our attention to how such bounds can be achieved if *neither* the embedding in  $(\mathcal{M}, d)$  *nor* the target distribution  $\mu$  are a priori known. To this end, we propose a novel adaptive algorithm that achieves the performance guarantees of Theorem 3 without access to the above information.

Our algorithm effectively learns the ranks  $r_x(y)$  of objects and the target distribution  $\mu$  as time progresses. It does not require that distances between objects are at any point disclosed; instead, we assume that it only has access to a comparison oracle, slightly stronger than the one described in Section 4.2.

It is important to note that our algorithm is *adaptive*: though we prove its convergence under a stationary regime, the algorithm can operate in a dynamic environment. For example, new objects can be added to the database while old ones can be removed. Moreover, the popularity of objects can change as time progresses. Provided that such changes happen infrequently, at a larger timescale compared to the timescale in which database queries are submitted, our algorithm will be able to adapt and converge to the desired behavior.

### 6.1 Demand Model and Probabilistic Oracle

We assume that time is slotted and that at each timeslot  $\tau = 0, 1, \dots$  a new query is generated in the database. As before, we assume that the source and

target of the new query are selected according to a demand distribution  $\lambda$  over  $\mathcal{N} \times \mathcal{N}$ . We again denote by  $\nu, \mu$  the (marginal) source and target distributions, respectively.

Our algorithm will require that the support of both the source and target distributions is  $\mathcal{N}$ , and more precisely that

$$\lambda(x, y) > 0, \text{ for all } x, y \in \mathcal{N}. \quad (14)$$

The requirement that the target set  $\mathcal{T} = \text{supp}(\mu)$  is  $\mathcal{N}$  is necessary to ensure learning; we can only infer the relative order w.r.t. objects  $t$  for which questions of the form  $\text{Oracle}(x, y, t)$  are submitted to the oracle. Moreover, it is natural in our model to assume that the source distribution  $\nu$  is at the discretion of our algorithm: we can choose which objects to propose first to the user/oracle. In this sense, for a given target distribution  $\mu$  s.t.  $\text{supp}(\mu) = \mathcal{N}$ , (14) can be enforced, *e.g.*, by selecting source objects uniformly at random from  $\mathcal{N}$  and independently of the target.

We consider a slightly stronger oracle than the one described in Section 4.1. In particular, we again assume that

$$\text{Oracle}(x, y, t) = \begin{cases} x & \text{if } x \prec_t y, \\ y & \text{if } x \succ_t y. \end{cases} \quad (15)$$

However, we further assume that if  $x \sim_t y$ , then  $\text{Oracle}(x, y, t)$  can return either of the two possible outcomes *with non-zero probability*. This is stronger than oracle in Section 4.1, where we assumed that the outcome will be arbitrary. We should point out here that this is still weaker than an oracle that correctly identifies  $x \sim_t y$  (*i.e.*, the human states that these objects are at equal distance from  $t$ ) as, given such an oracle, we can implement the above probabilistic oracle by simply returning  $x$  or  $y$  with equal probability.

## 6.2 Data Structures

For every object  $x \in \mathcal{N}$ , the database storing  $x$  also maintains the following associated data structures. The first data structure is a counter keeping track of how often the object  $x$  has been requested so far. The second data structure maintains an order of the objects in  $\mathcal{N}$ ; at any point in time, this total order is an “estimator” of  $\preceq_x$ , the order of objects with respect to their distance from  $x$ . We describe each one of these two data structures in more detail below.

*Estimating the Target Distribution* The first data structure associated with an object  $x$  is an estimator of  $\mu(x)$ , *i.e.*, the probability with which  $x$  is selected as a target. A simple method for keeping track of this information is through a counter  $C_x$ . This counter  $C_x$  is initially set to zero and is incremented every time object  $x$  is the target. If  $C_x(\tau)$  is the counter at timeslot  $\tau$ , then

$$\hat{\mu}(x) = C_x(\tau)/\tau \quad (16)$$

is an unbiased estimator of  $\mu(x)$ . To avoid counting to infinity a “moving average” (*e.g.*, and exponentially weighted moving average) could be used instead.

*Maintaining a Partial Order* The second data structure  $\mathcal{O}_x$  associated with each  $x \in \mathcal{N}$  maintains a total order of objects in  $\mathcal{N}$  w.r.t. their similarity to  $x$ . It supports an operation called `order()` that returns a partition of objects in  $\mathcal{N}$  along with a total order over this partition. In particular, the output of  $\mathcal{O}_x.\text{order}()$  consists of an ordered sequence of disjoint sets  $A_1, A_2, \dots, A_j$ , where  $\bigcup A_i = \mathcal{N} \setminus \{x\}$ . Intuitively, any two objects in a set  $A_i$  are considered to be at equal distance from  $x$ , while among two objects  $u \in A_i$  and  $v \in A_j$  with  $i < j$  the object  $u$  is assumed to be the closer to  $x$ .

Moreover, every time that the algorithm evokes `Oracle( $u, v, x$ )`, and learns, *e.g.*, that  $u \prec_x v$ , the data structure  $\mathcal{O}_x$  should be updated to reflect this information. In particular, if the algorithm has learned so far the order relationships

$$u_1 \prec_x v_1, \quad u_2 \prec_x v_2, \quad \dots, \quad u_i \prec_x v_i \tag{17}$$

$\mathcal{O}_x.\text{order}()$  should return the objects in  $\mathcal{N}$  sorted in such a way that all relationships in (17) are respected. In particular, object  $u_1$  should appear before  $v_1$ ,  $u_2$  before  $v_2$ , and so forth. To that effect, the data structure should also support an operation called  $\mathcal{O}_x.\text{add}(u, v)$  that adds the order relationship  $u \prec_x v$  to the constraints respected by the output of  $\mathcal{O}_x.\text{order}()$ .

A simple (but not the most efficient) way of implementing this data structure is to represent order relationships through a directed acyclic graph. Initially, the graph's vertex set is  $\mathcal{N}$  and its edge set is empty. Every time an operation  $\text{add}(u, v)$  is executed, an edge is added between vertices  $u$  and  $v$ . If the addition of the new edge creates a cycle then all nodes in the cycle are collapsed to a single node, keeping thus the graph acyclic. Note that the creation of a cycle  $u \rightarrow v \rightarrow \dots \rightarrow w \rightarrow u$  implies that  $u \sim_x v \sim_x \dots \sim_x w$ , *i.e.*, all these nodes are at equal distance from  $x$ .

Cycles can be detected by using depth-first search over the DAG [4]. The sets  $A_i$  returned by `order` are the sets associated with each collapsed node, while a total order among them that respects the constraints implied by the edges in the DAG can be obtained either by depth-first search or by a topological sort [4]. Hence, the `add()` and `order()` operations have a worst case cost of  $\Theta(n + m)$ , where  $m$  is the total number of edges in the graph.

Several more efficient algorithms exist in literature [11, 17, 1], the best [1] yielding a cost of  $O(n)$  for `order` and an aggregate cost of at most  $O(n^2 \log n)$  for any sequence of `add` operations. We stress here that any of these more efficient implementations could be used for our purposes. We refer the reader interested in such implementations to [11, 17, 1] and, to avoid any ambiguity, we assume the above naïve approach for the remainder of this work.

### 6.3 Greedy Content Search

Our learning algorithm implements greedy content search, as described in Section 4.1, in the following manner. When a new query is submitted to the database, the algorithm first selects a source  $s$  uniformly at random. It then performs

greedy content search using a memoryless selection policy  $\hat{\mathcal{F}}$  with distribution  $\hat{\ell}_x$ , *i.e.*,

$$\Pr(\mathcal{F}(\mathcal{H}_k, x_k) = w) = \hat{\ell}_{x_k}(w) \quad w \in \mathcal{N}. \quad (18)$$

Below, we discuss in detail how  $\hat{\ell}_x$ ,  $x \in \mathcal{N}$ , are computed.

When the current object  $x_k$ ,  $k = 0, 1, \dots$ , is equal to  $x$ , the algorithm evokes  $\mathcal{O}_{x_k}.\text{order}()$  and obtains an ordered partition  $A_1, A_2, \dots, A_j$  of items in  $\mathcal{N} \setminus \{x\}$ . We define

$$\hat{r}_x(w) = \sum_{j=1}^{i:w \in A_i} \hat{\mu}(A_j), \quad w \in \mathcal{N} \setminus \{x\}.$$

This can be seen as an “estimator” of the true rank  $r_x$  given by (5). The distribution  $\hat{\ell}_x$  is then computed as follows:

$$\hat{\ell}_x(w) = \frac{\hat{\mu}(w)}{\hat{r}_x(w)} \frac{1 - \epsilon}{\hat{Z}_x} + \frac{\epsilon}{n - 1}, \quad i = 1, \dots, n - 1, \quad (19)$$

where  $\hat{Z}_x = \sum_{w \in \mathcal{N} \setminus \{x\}} \hat{\mu}(w) / \hat{r}_x(w)$  is a normalization factor and  $\epsilon > 0$  is a small constant. An alternative view of (19) is that the object proposed is selected uniformly at random with probability  $\epsilon$ , and proportionally to  $\hat{\mu}(w_i) / \hat{r}_x(w_i)$  with probability  $1 - \epsilon$ . The use of  $\epsilon > 0$  guarantees that every search eventually finds the target  $t$ .

Upon locating a target  $t$ , any access to the oracle in the history  $\mathcal{H}_k$  can be used to update  $\mathcal{O}_t$ ; in particular, a call  $\text{Oracle}(u, v, t)$  that returns  $u$  implies the constraint  $u \preceq_t v$ , which should be added to the data structure through  $\mathcal{O}_t.\text{add}(u, v)$ . Note that this operation can take place only at the *end of the greedy content search*; the outcomes of calls to the oracle can be observed, but the target  $t$  is revealed only after it has been located.

Our main result is that, as  $\tau$  tends to infinity, the above algorithm achieves performance guarantees arbitrarily close to the ones of Theorem 3. Let  $\hat{\mathcal{F}}(\tau)$  be the selection policy defined by (18) at timeslot  $\tau$  and denote by

$$\bar{C}(\tau) = \sum_{(s,t) \in \mathcal{N} \times \mathcal{N}} \lambda(s, t) \sum_{s \in \mathcal{N}} \mathbb{E}[C_{\hat{\mathcal{F}}(\tau)}(s, t)]$$

the expected search cost at timeslot  $\tau$ . Then the following theorem holds:

**Theorem 5.** *Assume that for any two targets  $u, v \in \mathcal{N}$ ,  $\lambda(u, v) > 0$ .*

$$\limsup_{\tau \rightarrow \infty} \bar{C}(\tau) \leq \frac{6c^3(\mu)H(\mu)H_{\max}(\mu)}{(1 - \epsilon)}$$

where  $c(\mu)$ ,  $H(\mu)$  and  $H_{\max}(\mu)$  are the doubling parameter, the entropy and the max entropy, respectively, of the target distribution  $\mu$ .

*Proof.* Let  $\Delta_\mu = \sup_{x \in \mathcal{N}} |\hat{\mu}(x) - \mu(x)|$ . Observe first that, by the weak law of large numbers, for any  $\delta > 0$

$$\lim_{\tau \rightarrow \infty} \Pr(\Delta_\mu > \delta) = 0. \quad (20)$$

*i.e.*,  $\hat{\mu}$  converges to  $\mu$  in probability. The lemma below states, for every  $t \in \mathcal{N}$ , the order data structure  $\mathcal{O}_t$  will learn the correct order of any two objects  $u, v$  in finite time.

**Lemma 3.** *Consider  $u, v, t \in \mathcal{N}$  such that  $u \preceq_t v$ . Then, the order data structure in  $t$  evokes  $\mathcal{O}_t.\text{add}(u, v)$  after a finite time, with probability one.*

*Proof.* Recall that  $\mathcal{O}_t.\text{add}(u, v)$  is evoked if and only if a call  $\text{Oracle}(u, v, t)$  takes place and it returns  $u$ . If  $u \prec_t v$  then  $\text{Oracle}(u, v, t) = u$ . If, on the other hand,  $u \sim_t v$ , then  $\text{Oracle}(u, v, t)$  returns  $u$  with non-zero probability. It thus suffices to show that such, for large enough  $\tau$ , a call  $\text{Oracle}(u, v, t)$  occurs at timeslot  $\tau$  with a non-zero probability. By the hypothesis of Theorem 5,  $\lambda(u, t) > 0$ . By (19), given that the source is  $u$ , the probability that  $\hat{\mathcal{F}}(u) = v$  conditioned on  $\hat{\mu}$  is

$$\hat{\ell}_u(v) \geq \frac{\mu(v) - \Delta_\mu}{1 + (n-1)\Delta_\mu} \frac{1-\epsilon}{n-1} + \frac{\epsilon}{n-1} \geq \frac{\mu(v) - \Delta_\mu}{(1 + (n-1)\Delta_\mu)(n-1)}$$

as  $\hat{Z}_v \leq n-1$  and  $|\hat{\mu}(x) - \mu(x)| \leq \Delta_\mu$  for every  $x \in \mathcal{N}$ . Thus, for any  $\delta > 0$ , the probability that is lower-bounded by

$$\lambda(u, t) \Pr(\hat{\mathcal{F}}(u) = v) \geq \frac{\mu(v) - \delta}{1 + (n-1)\delta} \Pr(\Delta_\mu < \delta).$$

By taking  $\delta > 0$  smaller than  $\mu(v)$ , we have by (20) that there exists a  $\tau^*$  s.t. for all  $\tau > \tau^*$  the probability that  $\text{Oracle}(u, v, t)$  takes place at timeslot  $\tau$  is bounded away from zero, and the lemma follows.

Thus if  $t$  is a target then, after a finite time, for any two  $u, v \in \mathcal{N}$  the ordered partition  $A_1, \dots, A_j$  returned by  $\mathcal{O}_t.\text{order}()$  will respect the relationship between  $u, v$ . In particular for  $u \in A_i, v \in A_{i'}$ , if  $u \sim_t v$  then  $i = i'$ , while if  $u \prec_t v$  then  $i < i'$ . As a result, the estimated rank of an object  $u \in A_i$  w.r.t.  $t$  will satisfy

$$\hat{r}_t(u) = \sum_{x \in \mathcal{T}: x \prec_v u} \hat{\mu}(x) + \sum_{x \in \mathcal{N} \setminus \mathcal{T}: x \in A_{i'}, i' \leq i} \hat{\mu}(x) = r_t(u) + O(\Delta_\mu)$$

*i.e.* the estimated rank will be close to the true rank, provided that  $\Delta_\mu$  is small. Moreover, as in Lemma 1, it can be shown that

$$\hat{Z}_v \leq 1 + \log^{-1} \hat{\mu}(v) = 1 + \log^{-1} [\mu_v + O(\Delta_\mu)]$$

for  $v \in \mathcal{N}$ . From these, for  $\Delta_\mu$  small enough, we have that for  $u, v \in \mathcal{N}$ ,

$$\hat{\ell}_u(v) = [\ell_u(v) + O(\Delta_\mu)](1 - \epsilon) + \epsilon \frac{1}{n-1}.$$

Following the same steps as the proof of Theorem 2 we can show that, given that  $\Delta_\mu \leq \delta$ , the expected search cost is upper bounded by  $\frac{6c^3 HH_{\max}}{(1-\epsilon)+O(\delta)}$ . This gives us that

$$\bar{C}(\tau) \leq \left[ \frac{6c^3 HH_{\max}}{(1-\epsilon)} + O(\delta) \right] \Pr(\Delta_\mu \leq \delta) + \frac{n-1}{\epsilon} \Pr(\Delta_\mu > \delta)$$

where the second part follows from the fact that, by using the uniform distribution with probability  $\epsilon$ , we ensure that the cost is stochastically upper-bounded by a geometric r.v. with parameter  $\frac{\epsilon}{n-1}$ . Thus, by (20),

$$\limsup_{\tau \rightarrow \infty} \bar{C}(\tau) \leq \frac{6c^3 HH_{\max}}{(1-\epsilon)} + O(\delta).$$

As this is true for all small enough delta, the theorem follows.  $\square$

## 7 Conclusions

In this work, we initiated a study of CTSC and SWND under heterogeneous demands, tying performance to the topology and the entropy of the target distribution. Our study leaves several open problems, including improving upper and lower bounds for both CSTC and SWND. Given the relationship between these two, and the NP-hardness of SWND, characterizing the complexity of CSTC is also interesting. Also, rather than considering restricted versions of SWND, as we did here, devising approximation algorithms for the original problem is another possible direction.

Earlier work on comparison oracles eschewed metric spaces altogether, exploiting what were referred to as *disorder inequalities* [10, 16, 18]. Applying these under heterogeneity is also a promising research direction. Finally, trade-offs between space complexity and the cost of the learning phase vs. the costs of answering database queries are investigated in the above works, and the same trade-offs could be studied in the context of heterogeneity.

## References

1. BENDER, M., FINEMAN, J., AND GILBERT, S. A new approach to incremental topological ordering. In *SODA* (2009).
2. BRANSON, S., WAH, C., SCHROFF, F., BABENKO, B., WELINDER, P., PERONA, P., AND BELONGIE, S. Visual recognition with humans in the loop. In *ECCV* (4) (2010), pp. 438–451.
3. CLARKSON, K. L. Nearest-neighbor searching and metric space dimensions. In *Nearest-Neighbor Methods for Learning and Vision: Theory and Practice*, G. Shakhnarovich, T. Darrell, and P. Indyk, Eds. MIT Press, 2006, pp. 15–59.
4. CORMEN, T., LEISERSON, C., RIVEST, R., AND STEIN, C. *Introduction to Algorithms*, 2nd ed. MIT Press and McGraw-Hill, 2001.
5. COVER, T. M., AND THOMAS, J. *Elements of Information Theory*. Wiley, 1991.
6. FRAIGNIAUD, P., AND GIAKKOUPIS, G. On the searchability of small-world networks with arbitrary underlying structure. In *STOC* (2010).

7. FRAIGNIAUD, P., LEBHAR, E., AND LOTKER, Z. A doubling dimension threshold  $\theta(\log \log n)$  for augmented graph navigability. In *ESA* (2006).
8. GAREY, M. Optimal binary identification procedures. In *SIAM J. Appl. Math* (1972).
9. GEMAN, D., AND JEDYNAK, B. Shape recognition and twenty questions. Tech. rep., in Proc. Reconnaissance des Formes et Intelligence Artificielle (RFIA), 1993.
10. GOYAL, N., LIFSHITS, Y., AND SCHUTZE, H. Disorder inequality: a combinatorial approach to nearest neighbor search. In *WSDM* (2008).
11. HAEUPLER, B., KAVITHA, T., MATHEW, R., SEN, S., AND TARJAN, R. Incremental cycle detection, topological ordering, and strong component maintenance. In *SODA* (2008).
12. INDYK, P., AND MOTWANI, R. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *STOC* (1998), pp. 604–613.
13. KARGER, D., AND RUHL, M. Finding nearest neighbors in growth-restricted metrics. In *SODA* (2002).
14. KLEINBERG, J. The small-world phenomenon: An algorithmic perspective. In *STOC* (2000).
15. KRAUTHGAMER, R., AND LEE, J. R. Navigating nets: simple algorithms for proximity search. In *SODA* (2004).
16. LIFSHITS, Y., AND ZHANG, S. Combinatorial algorithms for nearest neighbors, near-duplicates and small-world design. In *SODA* (2009).
17. PEARCE, D., AND KELLY, P. Online algorithms for maintaining the topological order of a directed acyclic graph. Tech. rep., Imperial College, 2003.
18. TSCHOPP, D., AND DIGGAVI, S. N. Approximate nearest neighbor search through comparisons. 2009.
19. TSCHOPP, D., AND DIGGAVI, S. N. Facebrowsing: Search and navigation through comparisons. In *ITA workshop* (2010).
20. WHITE, R., AND ROTH, R. *Exploratory Search: Beyond the Query-Response Paradigm*. Morgan & Claypool, 2009.

## A Proof of Theorem 1

We first prove that the randomized version of SWND is no harder than its deterministic version. Define DETSWND to be the same as SWND with the additional restriction that  $\mathcal{S}$  is deterministic. For any random variable  $\mathcal{S} \subset \mathcal{N}$  that satisfies  $|\mathcal{S}| \leq \beta$ , there exists a deterministic set  $S^*$  s.t.  $|S^*| \leq \beta$  and  $\bar{C}_{S^*} \leq \bar{C}_{\mathcal{S}}$ . In particular, this is true for  $S^* = \arg \min_{S \in \mathcal{N}, |S| \leq \beta} C_S(s, t)$ . Thus, SWND is equivalent to DETSWND. In particular, any solution of DETSWND will also be a solution of SWND. Moreover, given a solution  $\mathcal{S}$  of SWND any deterministic  $S$  belonging to the support of  $\mathcal{S}$  will be a solution of DETSWND.

We therefore turn our attention on DETSWND. Without loss of generality, we can assume that the weights  $\lambda(s, t)$  are arbitrary non-negative numbers, as dividing every weight by  $\sum_{s, t} \lambda(s, t)$  does not change the optimal solution. The decision problem corresponding to DETSWND is as follows

DETSWND-D: Given an embedding of  $\mathcal{N}$  into  $(\mathcal{M}, d)$ , a set of local edges  $\mathcal{L}$ , a non-negative weight function  $\lambda$ , and two constants  $\alpha > 0$  and  $\beta > 0$ , is there a directed edge set  $S$  such that  $|S| \leq \beta$  and  $\sum_{(s, t) \times \mathcal{N} \times \mathcal{N}} \lambda(s, t) C_S(s, t) \leq \alpha$ ?

Note that, given the set of shortcut edges  $S$ , forwarding a message with greedy forwarding from any  $s$  to  $t$  can take place in polynomial time. As a result, DETSWND-D is in NP. We will prove it is also NP-hard by reducing the following NP-complete problem to it:

DOMINATINGSET: Given a graph  $G(V, E)$  and a constant  $k$ , is there a set  $A \subseteq V$  such that  $|A| \leq k$  and  $\Gamma(A) \cup A = V$ , where  $\Gamma(A)$  the neighborhood of  $A$  in  $G$ ?

Given an instance  $(G(V, E), k)$  of DOMINATINGSET, we construct an instance of DETSWND-D as follows. The set  $\mathcal{N}$  in this instance will be embedded in a 2-dimensional grid, and the distance metric  $d$  will be the Manhattan distance on the grid. In particular, let  $n = |V|$  be the size of the graph  $G$  and, w.l.o.g., assume that  $V = \{1, 2, \dots, n\}$ . Let

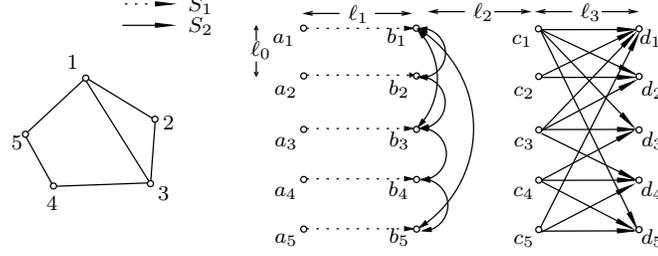
$$\ell_0 = 6n + 3, \quad \ell_3 = 6n + 3, \quad (21)$$

$$\ell_1 = n\ell_0 + 2 = 6n^2 + 3n + 2, \quad (22)$$

$$\ell_2 = \ell_1 + 3n + 1 = 6n^2 + 6n + 3. \quad (23)$$

We construct a  $n_1 \times n_2$  grid, where  $n_1 = (n - 1) \cdot \ell_0 + 1$  and  $n_2 = \ell_1 + \ell_2 + \ell_3 + 1$ . That is, the total number of nodes in the grid is  $N = [(n - 1) \cdot \ell_0 + 1] \cdot (\ell_1 + \ell_2 + \ell_3 + 1) = \Theta(n^4)$ . The object set  $\mathcal{N}$  will be the set of nodes in the above grid, and the metric space will be  $(\mathbb{Z}^2, d)$  where  $d$  is the Manhattan distance on  $\mathbb{Z}^2$ . The local edges  $\mathcal{L}$  is defined according to (3) with  $r = 1$ , *i.e.*, and any two adjacent nodes in the grid are connected by an edge in  $\mathcal{L}$ .

Denote by  $a_i$ ,  $i = 1, \dots, n$ , the node on the first column of the grid that resides at row  $(i - 1)\ell_0 + 1$ . Similarly, denote by  $b_i$ ,  $c_i$  and  $d_i$  the nodes on the



**Fig. 2.** A reduction of an instance of DOMINATINGSET to an instance of DETSWND-D. Only the nodes on the grid that have non-zero incoming or outgoing demands (weights) are depicted. The dashed arrows depict  $A_1$ , the set of pairs that receive a weight  $W_1$ . The solid arrows depict  $A_2$ , the set of pairs that receive weight  $W_2$ .

columns  $(\ell_1 + 1)$ ,  $(\ell_1 + \ell_2 + 1)$  and  $(\ell_1 + \ell_2 + \ell_3 + 1)$  the grid, respectively, that reside at the same row as  $a_i$ ,  $i = 1, \dots, n$ . These nodes are depicted in Figure 2. We define the weight function  $\lambda(i, j)$  over the pairs of nodes in the grid as follows. The pairs of grid nodes that receive a non-zero weight are the ones belonging to one of the following the sets:  $A_1 = \{(a_i, b_i) \mid i \in V\}$ ,  $A_2 = \{(b_i, b_j) \mid (i, j) \in E\} \cup \{(c_i, d_j) \mid (i, j) \in E\} \cup \{(c_i, d_i) \mid i \in V\}$ ,  $A_3 = \{(a_i, d_i) \mid i \in V\}$ . The sets  $A_1$  and  $A_2$  are depicted in Fig. 2 with dashed and solid lines, respectively. Note that  $|A_1| = n$  as it contains one pair for each vertex in  $V$ ,  $|A_2| = 4|E| + n$  as it contains four pairs for each edge in  $E$  and one pair for each vertex in  $V$ , and, finally,  $|A_3| = n$ . The pairs in  $A_1$  receive a weight equal to  $W_1 = 1$ , the pairs in  $A_2$  receive a weight equal to  $W_2 = 3n + 1$  and the pairs in  $A_3$  receive a weight equal to  $W_3 = 1$ .

For the bounds  $\alpha$  and  $\beta$  take

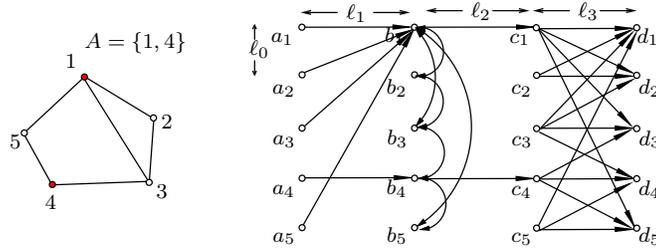
$$\alpha = 2W_1|A_1| + W_2|A_2| + 3|A_3|W_3 = (3n+1)(4|E|+n) + 5n \quad (24)$$

$$\beta = |A_2| + n + k = 4|E| + 2n + k. \quad (25)$$

The above construction can take place in polynomial time in  $n$ . Moreover, if the graph  $G$  has a dominating set of size no more than  $k$ , one can construct a deterministic set of shortcut edges  $\mathcal{S}$  that satisfies the constraints of DETSWND-D.

**Lemma 4.** *If the instance of DOMINATINGSET is a “yes” instance, then the constructed instance of DETSWND-D is also a “yes” instance.*

*Proof.* To see this, suppose that there exists a dominating set  $A$  of the graph with size  $|A| \leq k$ . Then, for every  $i \in V \setminus A$ , there exists a  $j \in A$  such that  $i \in \Gamma(j)$ , i.e.,  $i$  is a neighbor of  $j$ . We construct  $\mathcal{S}$  as follows. For every  $i \in A$ , add the edges  $(a_i, b_i)$  and  $(b_i, c_i)$  in  $\mathcal{S}$ . For every  $i \in V \setminus A$ , add an edge  $(a_i, b_j)$  in  $\mathcal{S}$ , where  $j$  is such that  $j \in A$  and  $i \in \Gamma(j)$ . For every pair in  $A_2$ , add this edge in  $\mathcal{S}$ . The size of  $\mathcal{S}$  is  $|\mathcal{S}| = 2|A| + (|V| - |A|) + |A_2| = |A| + n + 4|E| + n \leq 4|E| + 2n + k$ .



**Fig. 3.** A “yes” instance of DOMINATINGSET and the corresponding “yes” instance of DETSWND-D. The graph on the left is can be dominated by two nodes, 1 and 4. The corresponding set  $\mathcal{S}$  of shortcut contacts that satisfies the constraints of DETSWND-D is depicted on the right.

Moreover, the weighted forwarding distance is

$$\bar{C}_{\mathcal{S}}^w = \sum_{(i,j) \in A_1} W_1 C_{\mathcal{S}}(i,j) + \sum_{(i,j) \in A_2} W_2 C_{\mathcal{S}}(i,j) + \sum_{(i,j) \in A_3} W_3 C_{\mathcal{S}}(i,j).$$

We have  $\sum_{(i,j) \in A_2} W_2 C_{\mathcal{S}}(i,j) = W_2 |A_2|$  as every pair in  $A_2$  is connected by an edge in  $\mathcal{S}$ . Consider now a pair  $(a_i, b_i) \in A_1$ ,  $i \in V$ . There is exactly one edge in  $\mathcal{S}$  departing from  $a_i$  which has the form  $(a_i, b_j)$ , where where either  $j = i$  or  $j$  a neighbor of  $i$ . The distance of the closest local neighbor of  $a_i$  from  $b_i$  is  $\ell_1 - 1$ . The distance of  $b_j$  from  $b_i$  is at most  $n \cdot \ell_0$ . As  $\ell_1 - 1 = n\ell_0 + 2 - 1 > n\ell_0$  greedy forwarding will follow  $(a_i, b_j)$ . If  $b_j = b_i$ , then  $C_{\mathcal{S}}(a_i, b_i) = 1$ . If  $b_j \neq b_i$ , as  $j$  is a neighbor of  $i$ ,  $\mathcal{S}$  contains the edge  $(b_j, b_i)$ . Hence, if  $b_j \neq b_i$ ,  $C_{\mathcal{S}}(a_i, b_i) = 2$ . As  $i$  was arbitrary, we get that  $\sum_{(i,j) \in A_1} W_1 C_{\mathcal{S}}(i,j) \leq 2W_1 n$ .

Next, consider a pair  $(a_i, d_i) \in A_3$ . For the same reasons as for the pair  $(a_i, b_i)$ , the shortcut edge  $(a_i, b_j)$  in  $\mathcal{S}$  will be used by the greedy forwarding algorithm. In particular, the distance of the closest local neighbor of  $a_i$  from  $d_i$  is  $\ell_1 + \ell_2 + \ell_3 - 1$  and  $d(b_j, d_i)$  is at most  $\ell_2 + \ell_3 + n \cdot \ell_0$ . As  $\ell_1 - 1 > n\ell_0$ , greedy forwarding will follow  $(a_i, b_j)$ .

By the construction of  $\mathcal{S}$ ,  $b_j$  is such that  $j \in A$ . As a result, again by the construction of  $\mathcal{S}$ ,  $(b_j, c_j) \in \mathcal{S}$ . The closest local neighbor of  $b_j$  to  $d_i$  has  $\ell_2 + \ell_3 + d(b_j, b_i) - 1$  Manhattan distance from  $d_j$ . Any shortcut neighbor  $b_k$  of  $b_j$  has at least  $\ell_2 + \ell_3$  Manhattan distance from  $b_i$ . On the other hand,  $c_j$  has  $\ell_3 + d(b_j, b_i)$  Manhattan distance from  $d_i$ . As  $\ell_2 > 1$  and  $\ell_2 > n\ell_0 \geq d(b_j, b_i)$ , the greedy forwarding algorithm will follow  $(b_j, c_j)$ . Finally, as  $A_2 \subset \mathcal{S}$ , and  $j = i$  or  $j$  is a neighbor of  $i$ , the edge  $(c_j, d_i)$  will be in  $\mathcal{S}$ . Hence, the greedy forwarding algorithm will reach  $d_j$  in exactly 3 steps. As  $i \in V$  was arbitrary, we get that  $\sum_{(i,j) \in A_3} W_3 C_{\mathcal{S}}(i,j) = 3W_3 n$ . Hence,  $\bar{C}_{\mathcal{S}}^w \leq 2W_1 n + W_2 |A_2| + 3W_3 n = \alpha$  and, therefore, the instance of DETSWND-D is a “yes” instance.  $\square$

To complete the proof, we show that a dominating set of size  $k$  exists only if there exists a  $\mathcal{S}$  that satisfies the constraints in constructed instance of DETSWND-D.

**Lemma 5.** *If the constructed instance of DETSWND-D is a “yes” instance, then the instance of DOMINATINGSET is also a “yes” instance.*

*Proof.* Assume that there exists a set  $\mathcal{S}$ , with  $|\mathcal{S}| \leq \beta$  such that the augmented graph has a weighted forwarding distance less than or equal to  $\alpha$ . Then

$$A_2 \subseteq \mathcal{S}. \quad (26)$$

To see this, suppose that  $A_2 \not\subseteq \mathcal{S}$ . Then, there is at least one pair of nodes  $(i, j)$  in  $A_2$  with  $C_{\mathcal{S}}(i, j) \geq 2$ . Therefore,  $\bar{C}_{\mathcal{S}}^w \geq (3n + 1)(4|E| + n) + 5n + 1 > \alpha$  a contradiction.

Essentially, by choosing  $W_2$  to be large, we enforce that all “demands” in  $A_2$  are satisfied by a direct edge in  $\mathcal{S}$ . The next lemma shows a similar result for  $A_1$ . Using shortcut edges to satisfy these “demands” is enforced by making the distance  $\ell_1$  very large.

**Lemma 6.** *For every  $i \in V$ , there exists at least one shortcut edge in  $\mathcal{S}$  whose origin is in the same row as  $a_i$  and in a column to the left of  $b_i$ . Moreover, this edge is used during the greedy forwarding of a message from  $a_i$  to  $b_i$ .*

*Proof.* Suppose not. Then, there exists an  $i \in V$  such that no shortcut edge has its origin between  $a_i$  and  $b_i$ , or such an edge exists but is not used by the greedy forwarding from  $a_i$  to  $b_i$  (e.g., because it points too far from  $b_i$ ). Then, the greedy forwarding from  $a_i$  to  $b_i$  will use only local edges and, hence,  $C_{\mathcal{S}}(a_i, b_i) = \ell_1$ .

We thus have that  $\bar{C}_{\mathcal{S}}^w \geq \ell_1 + 2n - 1 + W_2|A_2| \stackrel{(22)}{=} 6n^2 + 5n + 1 + W_2|A_2|$ . On the other hand, by (24)  $\alpha = 5n + W_2|A_2|$  so  $\bar{C}_{\mathcal{S}}^w > \alpha$ , a contradiction.  $\square$

Let  $S_1$  be the set of all edges whose origin is between some  $a_i$  and  $b_i$ ,  $i \in V$ , and that are used during forwarding from this  $a_i$  to  $b_i$ . Note that Lemma 6 implies that  $|S_1| \geq n$ . The target of any edge in  $S_1$  must lie to the left of the  $2\ell_1 + 1$ -th column of the grid. This is because the Manhattan distance of  $a_i$  to  $b_i$  is  $\ell_1$ , so its left local neighbor lies at  $\ell_1 - 1$  steps from  $b_i$ . Greedy forwarding is monotone, so the Manhattan distance from  $b_i$  of any target of an edge followed subsequently to route towards  $b_i$  must be less than  $\ell_1$ .

Essentially, all edges in  $S_1$  must point close enough to  $b_i$ , otherwise they would not be used in greedy forwarding. This implies that, to forward the “demands” in  $A_3$  an *additional* set of shortcut edges need to be used.

**Lemma 7.** *For every  $i \in V$ , there exists at least one shortcut edge in  $\mathcal{S}$  that is used when forwarding a message from  $a_i$  to  $d_i$  that is neither in  $S_1$  nor in  $A_2$ .*

*Proof.* Suppose not. We established above that the target of any edge in  $S_1$  is to the left of the  $2\ell_1 + 1$  column. Recall that  $A_2 = \{(b_i, b_j) \mid (i, j) \in E\} \cup \{(c_i, d_j) \mid (i, j) \in E\} \cup \{(c_i, d_i) \mid i \in V\}$ . By the definition of  $b_i$ ,  $i \in V$ , the targets of the edges in  $\{(b_i, b_j) \mid (i, j) \in E\}$  lie on the  $(\ell_1 + 1)$ -th column. Similarly, the origins of the edges in  $\{(c_i, d_j) \mid (i, j) \in E\} \cup \{(c_i, d_i) \mid i \in V\}$  lie on the  $\ell_1 + \ell_2 + 1$ -th column. As a result, if the lemma does not hold, there is a demand in  $A_3$ , say  $(a_i, d_i)$ , that does not use any additional shortcut edges. This

means that the distance between the  $2\ell + 1$  and the  $\ell_1 + \ell_2 + 1$ -th column is traversed by using local edges. Hence,  $C_{\mathcal{S}}(a_i, d_i) \geq \ell_2 - \ell_1 + 1$  as at least one additional step is needed to get to the  $2\ell_1 + 1$ -th column from  $a_i$ . This implies that  $\bar{C}_{\mathcal{S}}^w \geq 2n + W_2|A_2| + \ell_2 - \ell_1 \stackrel{(23)}{=} W_2|A_2| + 5n + 1 > \alpha$  a contradiction.  $\square$

Let  $S_3 = \mathcal{S} \setminus (S_1 \cup A_2)$ . Lemma 7 implies that  $S_3$  is non-empty, while (26) and Lemma 6, along with the fact that  $|\mathcal{S}| \leq \beta = |A_2| + n + k$ , imply that  $|S_3| \leq k$ . The following lemma states that some of these edges must have targets that are close enough to the destinations  $d_i$ .

**Lemma 8.** *For each  $i \in V$ , there exists an edge in  $S_3$  whose target is within Manhattan distance  $3n + 1$  of either  $d_i$  or  $c_j$ , where  $(c_j, d_i) \in A_2$ . Moreover, this edge is used for forwarding a message from  $a_i$  to  $d_i$  with greedy forwarding.*

*Proof.* Suppose not. Then there exists an  $i$  for which greedy forwarding from  $a_i$  to  $d_i$  does not employ any edge fitting the description in the lemma. Then, the destination  $d_i$  can not be reached by a shortcut edge in either  $S_3$  or  $A_1$  whose target is closer than  $3n + 1$  steps. Thus,  $d_i$  is reached in one of the two following ways: either  $3n + 1$  steps are required in reaching it, through forwarding over local edges, or an edge  $(c_j, d_i)$  in  $A_2$  is used to reach it. In the latter case, reaching  $c_j$  also requires at least  $3n + 1$  steps of local forwarding, as no edge in  $A_2$  or  $S_3$  has an target within  $3n$  steps from it, and any edge in  $S_1$  that may be this close is not used (by the hypothesis). As a result,  $C_{\mathcal{S}}(a_i, d_i) \geq 3n + 2$  as at least one additional step is required in reaching the ball of radius  $3n$  centered around  $d_i$  or  $c_j$  from  $a_i$ . This gives us  $\bar{C}_{\mathcal{S}}^w \geq 5n + W_2|A_2| + 1 > \alpha$  a contradiction.  $\square$

When forwarding from  $a_i$  to  $d_i$ ,  $i \in V$ , there may be more than one edges in  $S_3$  fitting the description in Lemma 8. For each  $i \in V$ , consider the *last* of all these edges. Denote the resulting subset by  $S'_3$ . By definition,  $|S'_3| \leq |S_3| \leq k$ . For each  $i$ , there exists *exactly one* edge in  $S'_3$  that is used to forward a message from  $a_i$  to  $d_i$ . Moreover, recall that  $\ell_0 = \ell_3 = 6n + 3$ . Therefore, the Manhattan distance between any two nodes in  $\{c_1, \dots, c_n\} \cup \{d_1, \dots, d_n\}$  is  $2(3n + 1) + 1$ . As a result, the targets of the edges in  $S'_3$  will be within distance  $3n + 1$  of *exactly one* of the nodes in the above set.

Let  $A \subset V$  be the set of all vertices  $i \in V$  such that the unique edge in  $S'_3$  used in forwarding from  $a_i$  to  $d_i$  has a target within distance  $3n + 1$  of either  $c_i$  or  $d_i$ . Then  $A$  is a dominating set of  $G$ , and  $|A| \leq k$ . To begin with,  $|A| \leq k$  because each target of an edge in  $S'_3$  can be within distance  $3n + 1$  of only one of the nodes in  $\{c_1, \dots, c_n\} \cup \{d_1, \dots, d_n\}$ , and there are at most  $k$  edges in  $S'_3$ .

To see that it dominates the graph  $G$ , suppose that  $j \in V \setminus A$ . Then, by Lemma 8, the edge in  $S'_3$  corresponding to  $j$  is either pointing within distance  $3n + 1$  of either  $d_j$  or a  $c_i$  such that  $(c_i, d_j) \in A_2$ . By the construction of  $A$ , it cannot point in the proximity of  $d_j$ , because then  $j \in A$ , a contradiction. Similarly, it cannot point in the proximity of  $c_j$ , because then, again,  $j \in A$ , a contradiction. Therefore, it points in the proximity of some  $c_i$ , where  $i \neq j$  and  $(c_i, d_j) \in A_2$ . By the construction of  $A$ ,  $i \in A$ . Moreover, by the definition of  $A_2$ ,  $(c_i, d_j) \in A_2$  if and only if  $(i, j) \in E$ . Therefore,  $j \in \Gamma(A)$ . As  $j$  was arbitrary,  $A$  is a dominating set of  $G$ .  $\square$

## B An Upper Bound for Non-Metric Spaces

Similarity between objects is a well defined relationship even if the objects are not embedded in a metric space. More specifically, the notation  $x \prec_z y$  simply states that  $x$  is more similar to  $z$  than  $y$ .

If the only information given about the underlying space is the similarity between objects, then the maximum we can hope for is for each object  $x \in \mathcal{N}$  sort other objects  $\mathcal{N} \setminus y$  according to their similarity to  $x$ .

Given the demand  $\lambda$ , the target set  $\mathcal{T}$  is completely specified. For any  $y \in \mathcal{T}$  let us define the *rank* as follows:

$$r_x(y) = |\{z : z \in \mathcal{T}, z \prec_x y\}|.$$

We say that  $y \in \mathcal{T}$  is the  $k$ -th closest object to  $x$  if  $r_x(y) = k$ . First not that the rank is in general asymmetric, *i.e.*,  $r_x(y) \neq r_y(x)$ . Second, the triangle inequality is not satisfied in general, *i.e.*,  $r_x(y) \not\leq r_y(z) + r_z(x)$ . However the approximate inequality as introduced in [10] is always satisfied. More precisely, we say that the *disorder factor*  $D(\mu)$  is the smallest  $D$  such that we have the *approximate triangle inequality*

$$r_x(y) \leq D(r_z(y) + r_z(x)),$$

for all  $x, y, z \in \mathcal{T}$ . The factor  $D(\mu)$  basically quantifies the non-homogeneity of the underlying space when the only give information is order of objects. Let the selection policy for the non-metric space be defined as follows:

$$\Pr(\mathcal{F}(\mathcal{H}_k, x_k) = w) \propto \frac{1}{r_{x_k}(w)}, \quad (27)$$

for  $w \in \mathcal{T}$ . In case  $w \notin \mathcal{T}$  we define  $\Pr(\mathcal{F}(\mathcal{H}_k, x_k) = w)$  to be zero.

It is of high interest to see whether we can still navigate through the database when the characterization of the underlying space is unknown and only the similarity relationship between objects is provided. This is the main theme of the next theorem.

**Theorem 6.** *Consider the above selection policy. Then for any demand  $\lambda$ , the cost of greedy content search is bounded as*

$$\bar{C}_{\mathcal{F}} \leq 7D(\mu) \log^2 |\mathcal{T}|.$$

The proof of this Theorem is given below. Note again that the selection policy is memoryless. Furthermore, it is *universal* in a sense that using this selection policy for any kind of demands guarantees the search that only depends on the cardinality of target set and its disorder factor. For instance, this selection policy is useful when the target set is only known a priori and the demand is not fully specified.

*Proof.* The selection policy in the non-metric space scenario is given (27) which implies that only objects in the target set  $\mathcal{T}$  are going to be proposed by the

algorithm. Therefore, except for the starting point  $x_0 = s$ , the algorithms navigate only through the target set. The probability of proposing  $w \in \mathcal{T}$  when  $x_k$  is the current object of the search is given by

$$\Pr(\mathcal{F}(\mathcal{H}_k, x_k) = w) = \frac{1}{Z_{x_k}} \frac{1}{r_{x_k}(w)},$$

where  $Z_{x_k} = \sum_{w \in \mathcal{T}} r_{x_k}^{-1}(w)$ . Consequently,

$$Z_{x_k} = \begin{cases} H_{|\mathcal{T}|-1} & \text{if } x_k \in \mathcal{T}, \\ H_{|\mathcal{T}|} & \text{if } x_k \notin \mathcal{T}, \end{cases}$$

where  $H_n$  is the  $n$ -th harmonic number. Hence,  $z_{x_k} \leq 2 \log |\mathcal{T}|$ . As the search moves from  $s$  to  $t$  we say that the search is in phase  $j$  when the rank of the current object  $v \neq s$  with respect to  $t$  is  $2^j \leq r_t(v) \leq 2^{j+1}$ . Clearly, there are only  $\log |\mathcal{T}|$  different phases. The greedy search algorithm keeps proposing to the oracle until it finds another object closer to  $t$ . We can write  $C_{\mathcal{F}}(s, t)$  as

$$C_{\mathcal{F}}(s, t) = X_1 + X_2 + \cdots + X_{\log |\mathcal{T}|} + X_s,$$

where  $X_s$  denotes the number of comparisons done by oracle at the starting point until it goes to an object  $u \in \mathcal{T}$  such that  $r_s(u) \leq r_s(t)$ . As before  $X_j$  ( $j > 0$ ) is the number of comparisons done by oracle until it goes to the next phase.

We need to differentiate between the starting point of the process and the rest of it. Since unlike other objects proposed by the algorithm, the starting object  $s$  may not be in the target set. Let the rank of  $t$  with respect to  $s$  be  $k$ , *i.e.*,  $r_s(t) = k$ . Then, the probability that the greedy search algorithm proposes an object  $v \in \mathcal{T}$  such that  $r_s(v) \leq r_s(t)$  is  $\sum_{j=1}^k \frac{1}{H_{|\mathcal{T}|}^j} \leq \frac{1}{2 \log |\mathcal{T}|}$ . As a result  $\mathbb{E}[X_s | s, t] \leq 2 \log |\mathcal{T}|$ . This is the average number of comparisons performed by the oracle until the greedy search algorithm escapes from the starting object  $s$ .

Let the current object  $v \neq s$  be in phase  $j$ . We denote by

$$I = \left\{ u : u \in \mathcal{T}, r_t(u) \leq \frac{r_t(v)}{2} \right\},$$

the set of objects whose rank from  $t$  is at most  $r_t(v)/2$ . Clearly,  $|I| = r_t(v)/2$ . The probability that the greedy search proposes an object  $u \in I$  (and hence going to the next phase) is at least

$$\sum_{u \in I} \frac{1}{2 \log |\mathcal{T}|} \frac{1}{r_v(u)} \stackrel{(a)}{\geq} \frac{r_t(v)}{4 \log |\mathcal{T}| D(\mu)(r_t(u) + r_t(v))},$$

where in (a) we used the approximate triangle inequality. Since for  $u \in I$ , we have  $r_t(u) \leq r_t(v)/2$ , the probability of going from  $v$  to the next phase is at least  $6D \log |\mathcal{T}|$ . Therefore,  $\mathbb{E}[X_j | s, t] \leq 6D \log |\mathcal{T}|$ .

Using the linearity of expectation,

$$\mathbb{E}[C_{\mathcal{F}}(s, t)] \leq 6D \log^2 |\mathcal{T}| + 2 \log |\mathcal{T}| \leq 7D \log^2 |\mathcal{T}|.$$

The above conditional expectation does not depend on the demand  $\lambda$ . Hence, the expected search cost for any demand is bounded as  $\mathbb{E}[C_{\mathcal{F}}] \leq 7D \log^2 |\mathcal{T}|$ .  $\square$