Using Graph Distances for Named-Entity Linking[☆]

Roi Blanco[a], Paolo Boldi[b], Andrea Marino[b,*]

[a] *Yahoo Labs, London, United Kingdom*
[b] *Dipartimento di informatica, Università degli Studi di Milano*

Abstract

Entity-linking is a natural-language–processing task that consists in identifying strings of text that refer to a particular item in some reference knowledge base. When the knowledge base is Wikipedia, the problem is also referred to as *wikification* (in this case, items are Wikipedia articles). Entity-linking consists conceptually of many different phases: identifying the portions of text that may refer to an entity (sometimes called "entity detection"), determining a set of concepts (candidates) from the knowledge base that may match each such portion, and choosing one candidate for each set; the latter step, known as *candidate selection*, is the phase on which this paper focuses. One instance of candidate selection can be formalized as an optimization problem on the underlying concept graph, where the quantity to be optimized is the average distance between the selected items. Inspired by this application, we define a new graph problem which is a natural variant of the Maximum Capacity Representative Set. We prove that our problem is NP-hard for general graphs; we propose several heuristics trying to optimize similar easier objective functions; we show experimentally how these approaches perform with respect to some baselines on a real-world dataset. Finally, in the appendix, we show an exact linear time algorithm that works under some more restrictive assumptions.

Keywords: Entity Linking, Maximum Capacity Representative Set, Minimum Distance Representative

1. Introduction

Wikipedia[1] is a free, collaborative, hypertextual encyclopedia that aims at collecting articles on different (virtually, all) branches of knowledge. The usage of Wikipedia for automatically tagging documents is a well-known methodology, that includes in particular a task called *wikification* [1]. Wikification is a special instance of *entity-linking*: a textual document is given and within the document various fragments are identified (either manually or automatically) as being *(named) entities* (e.g., names of people, brands, places. . .); the purpose of entity-linking is assigning a specific reference (a Wikipedia article, in the case of wikification) as a tag to

[☆] The second and third authors were supported by the EU-FET grant NADINE (GA 288956).
[*] Corresponding author
Email addresses: `roi@yahoo-inc.com` (Roi Blanco), `paolo.boldi@unimi.it` (Paolo Boldi), `marino@di.unimi.it` (Andrea Marino)
[1] `http://en.wikipedia.org/`

each entity in the document.

Entity-linking happens typically in three stages: in a first phase (called “entity detection”), the linker identifies which parts of the text are likely to refer to an entity, then every entity is assigned to a set of candidate items, e.g., Wikipedia articles (the *candidate nodes* for that entity); finally a third phase consists in selecting a single node for each entity, from within the set of candidates [2]. The latter task, called *candidate selection*, is the topic on which this paper focuses.

To provide a concrete example, suppose that the target document contains the entity “jaguar” and the entity “jungle”. Entity “jaguar” is assigned to a set of candidates that contains (among others) both the Wikipedia article about the feline living in America and the one about the Jaguar car producer. On the other hand, “jungle” is assigned to the article about tropical forests and to the one about the electronic music genre. Actually, there are more than 30 candidates for “jaguar”, and about 20 for “jungle”.

In this paper, we study an instance of the candidate selection problem in which the selection takes place based on some cost function that depends on the average distance between the selected candidates, where the distance is measured on the Wikipedia graph[2]: the rationale should be clear enough—concepts appearing in the same text are related, and so we should choose, among the possible candidates for each entity, those that are more closely related to one another.

Getting back to the example above, there is an edge connecting “jaguar” the feline with “jungle” the tropical forest, whereas the distance between, say, the feline and the music genre is much larger.

The approach we assume here highlights the *collective* nature of the entity-linking problem, as mentioned already by Han et al. [3]: the accuracy of the selection can be improved by a global (rather than local) optimization of the choices. As Han et al. [3] observe, however, trying to optimize all-pair compatibility is a computationally hard problem.

In this paper we prove that the problem itself, even in the simple formulation we take into consideration, is NP-hard; however, we can still provide heuristics to solve real instances. Finally, we will show that the problem becomes efficiently solvable under some special assumptions.

This paper is an extended version of [4]. In this version several new heuristics are proposed: in particular, we show experimentally on a real-world dataset that one of these heuristics is more effective than the best one proposed in [4], and in fact it is more effective than other existing techniques, like applying Freebase [5, 6].

[2]The undirected graph whose vertices are the Wikipedia articles and whose edges represent hyperlinks between them.

2. Related Work

Named-entity linking (NEL) (also referred to as *named entity disambiguation*) grounds mentions of entities in text (*surface forms*) into some knowledge base[3] (e.g. Wikipedia, Freebase). Early approaches to NEL [1] make use of measures derived from the frequency of the keywords to be linked in the text and in different Wikipedia pages. These include *tf-idf*, χ^2 and *keyphraseness*, which stands for a measure of how much a certain word is used in Wikipedia links in relation to its frequency in general text. Cucerzan [7] employed the context in which words appear and Wikipedia page categories in order to create a richer representation of the input text and candidate entities. These approaches were extended by Milne and Witten [8] who combined commonness (i.e., prior probability) of an entity with its relatedness to the surrounding context using machine learning. Further, Bunescu [9] employed a *disambiguation* kernel which uses the hierarchy of classes in Wikipedia along with its word contents to derive a finer-grained similarity measure between the candidate text and its context with the potential named entities to link to. In this paper we will make use of the dataset that Kulkarni et al. adopted in [10], where they propose a general collective disambiguation approach, under the premise that coherent documents refer to entities from one or a few related topics. They introduce formulations that account for the trade-off between local spot-to-entity compatibility and measures of global coherence between entities. More recently, Han et al. [3] propose a graph-based representation which exploits the global interdependence of different linking decisions. The algorithm infers jointly the disambiguated named mentions by exploiting the graph.

It is worth remarking that NEL is a task somehow similar to Word Sense Disambiguation, which in essence has the goal of determining the right sense of a word given its context. In the case of WSD the role of the knowledge base would be played by Wordnet [11]. WSD is a problem that has been extensively studied and its explicitly connection with NEL was made by Hachey et al [12]. WSD has been an area of intense research in the past, so we will review here the approaches that are directly relevant to our work. Graph-based approaches to word sense disambiguation are pervasive and yield state-of-the-art performance [13]; however, its use for NEL has been restricted to ranking candidate named entities with different flavors of centrality measures, such as in-degree or PageRank [12].

Mihalcea [14] introduced an unsupervised method for disambiguating the senses of words using random walks on graphs that encode the dependencies between word senses.

Navigli and Lapata [15, 16, 17] present subsequent approaches to WSD using graph connectivity metrics, in which nodes are ranked with respect to their local *importance*, which is computed using centrality measures like in-degree, centrality, PageRank or HITS, among others.

[3]Some authors use "named-entity linking" to refer to the full process of detecting mentions, identifying possible candidates in the knowledge base and finally selecting the most appropriate candidate from the candidate set; other authors prefer to use it only for the last step, and refer to the whole process as "full named-entity linking".

Importantly, even if the experimental section of this paper deals with a NEL dataset exclusively, the theoretical findings could be as well applied to WSD-style problems. Our *greedy* algorithm can be seen as an adaptation of Navigli and Velardi's Structural Semantic Interconnections algorithms for WSD [15, 18]. The original algorithm receives an ordered list of words to disambiguate. The procedure first selects the *unambiguous* words from the set (the ones with only one synset), and then, for every ambiguous word, it iteratively selects the sense that is *closer* to the sense of already-disambiguated words, and adds the word to the unambiguous set. Unfortunately, this approach works in the case that a sufficiently connected amount of words is unambiguous; this is not the case in NEL and in our experimental set-up, where there could potentially exist hundreds of candidates for a particular piece of text.

3. Problem statement and NP-completeness

In this section we will introduce the general formal definition of the problem in the formulation we decided to take into consideration. We will make use of the classical graph notation: in particular, given an undirected graph $G = (V, E)$, we will denote with $G[W]$ the graph induced by the vertices in W, and with $d(u, v)$ the distance between the nodes u and v, that is, the number of edges in the shortest path from u to v (or the sum of the weights of the lightest path, if G is weighted).

If G is a graph and e is an edge of G, $G - e$ is the graph obtained by removing e from G; we say that e is a *bridge* if the number of connected components of $G - e$ is larger than that of G. A connected bridgeless graph is called *biconnected*; a maximal set of vertices of G inducing a biconnected subgraph is called a *biconnected component* of G.

We call our main problem *Minimum Distance Representative*, in short MinDR, and we define it as follows. Given an undirected graph $G = (V, E)$, possibly (edge-)weighted, and k subsets of its set of vertices, $X_1, \dots, X_k \subseteq V$, a feasible solution for MinDR is a sequence $(x_1, \dots, x_k)$ of vertices of G such that for any i, with $1 \leq i \leq k$, $x_i \in X_i$ (i.e., the solution contains exactly one element from every set, possibly with repetitions).

Given the instance $G, \{X_1, \dots, X_k\}$, the measure (the *distance cost*) of a solution $S = (x_1, \dots, x_k)$, is $f(S) = \sum_{i=1}^{k} \sum_{j=1}^{k} d(x_i, x_j)$, that is the sum of all distances over all the pairs of nodes in S. The goal is finding the solution of minimum distance cost, i.e., a feasible solution S such that $f(S)$ is minimum. For the sake of simplicity, we will refer to a solution as the multiset composed by its elements.[4]

We call the restriction of this problem, in which the sets of vertices in input $\{X_1, \dots, X_k\}$ are pairwise disjoint, MinDIR (Minimum Distance Independent Representative).

[4]We shall make free use of multiset membership, intersection and union with their standard meaning: in particular, if A and B are multisets with multiplicity function a and b, respectively, the multiplicity functions of $A \cup B$ and $A \cap B$ are $x \mapsto \max(a(x), b(x))$ and $x \mapsto \min(a(x), b(x))$, respectively.

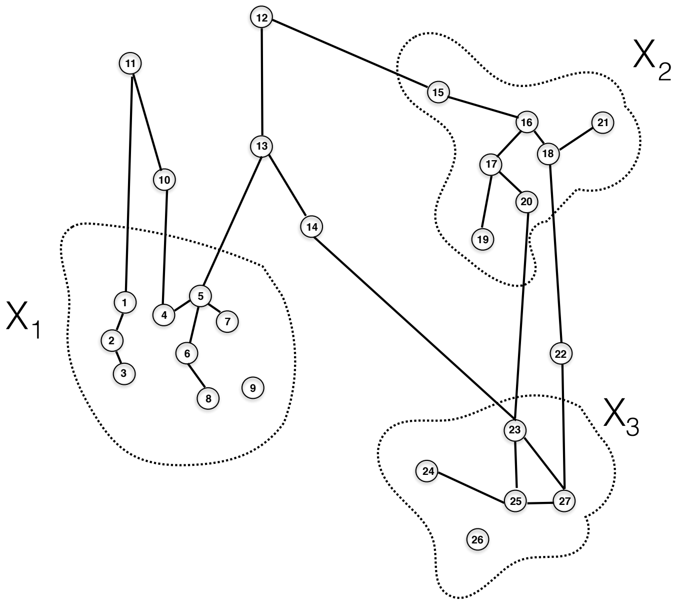

Figure 1: An instance of MinDIR: a graph G (for sake of simplicity, the graph is unweighted), and three subsets X_1, X_2, X_3 of its vertices.

An example of MinDIR is shown in Figure 1. Three possible solutions for this problem are $S = (8, 17, 25)$, $S' = (5, 16, 23)$ and $S'' = (5, 20, 23)$; their distance costs are $f(S) = 28$, $f(S') = 20$, $f(S'') = 16$, respectively. The latter is optimal.

3.1. NP-completeness of MinDR

The MinDIR problem is similar and related to the so-called Maximum Capacity Representative Set [19], in short MaxCRS. The Maximum Capacity Representative Set problem is defined as follows: given some disjoint sets $X_1, \ldots, X_m$ endowed with a nonnegative capacity $c(x, y)$ for any $x \in X_i$ and $y \in X_j$ (with $i \neq j$), a solution is a set $S = \{x_1, \ldots x_m\}$, such that, for any i, $x_i \in X_i$; such a solution is called *system of representatives*. The measure of a solution is the capacity of the system of representatives, that is $\sum_{x \in S} \sum_{y \in S} c(x, y)$, and the MaxCRS problem aims at *maximizing* it. The MaxCRS problem was introduced by Bellare [20], who showed that it is NP-complete and gave some non-approximability results. Successively, Serna et al. [21], presented tight inapproximability results for the problem.

The MinDIR problem differs from MaxCRS because we are dealing with distances instead of capacities, and therefore we ask for a minimum instead of a maximum. Nonetheless the following Lemma shows that also MinDIR problem is NP-complete.

Lemma 1. *The* MinDIR *(hence,* MinDR*) problem is NP-complete.*

Proof. We reduce MaxCRS to MinDIR. Given an instance of MaxCRS, $\{X_1, \ldots X_k\}$ and for any $i \neq j$, $x \in X_i$, and $y \in X_j$, a nonnegative capacity $c(x, y)$, we construct the instance of MinDIR $G, \{X_1, \ldots, X_k\}$; the vertices of G are $X_1 \cup \ldots \cup X_k$, and for any pair $x \in X_i$, $y \in X_j$ with $i \neq j$ we add a weighted edge between

x and y, i.e., for each pair for which MaxCRS defines a capacity we create a corresponding edge in G. In particular the weight of the edge between x and y is set to $\alpha - c(x, y)$, where $\alpha = 2 \max_{z \in X_i, t \in X_j, i \neq j} c(z, t)$.

Observe that for any pair of nodes $u \in X_i$ $v \in X_j$, with $i \neq j$, $d(u, v)$ in G is equal to the weight of (u, v), i.e., it is not convenient to pass through other nodes when going from u to v: in fact, for any path $u, z_1, \ldots, z_p, v$ from u to v in G, with $p \geq 1$, we always have $\alpha - c(u, v) \leq \alpha - c(u, z_1) + \ldots + \alpha - c(z_p, v)$, since $\alpha - c(u, v) \leq \alpha$. Moreover, observe that any solution in G has exactly one element for each set X_i: thus, we have $k(k-1)$ pairs of elements (x, y), whose distance is always given by the weight of the single edge (x, y), that is $\alpha - c(x, y)$.

Hence it is easy to see that MaxCRS admits a system of representatives whose capacity is greater than h, if and only if MinDIR admits a solution S such that $f(S)$ is less than $k(k-1)\alpha - h$.

Since MinDIR is a restriction of MinDR, we can conclude that also MinDR is NP-complete. $\square$

4. Heuristics for MinDR

As we observed in the previous section, the MinDR problem is NP-complete in general. In this section we provide several heuristics to solve the problem in practice.

Our heuristics for the original problem try to optimize similar easier objective functions still based on distances. For heuristics based on distances connectivity issues can get in the way, so some additional constraints for the input instances are enforced. In particular, we will consider a general connected MinDR instance, that is:

- a connected undirected (possibly weighted) graph $G = (V, E)$,
- k subsets of its set of vertices, $X_1, \ldots, X_k \subseteq V$,

with the additional assumption that $G[X_i]$ is connected for every i. Recall that a feasible solution is a sequence S of vertices of G, $x_1, \ldots, x_k$, such that for any i, with $1 \leq i \leq k$, we have $x_i \in X_i$; its (distance) cost is $f(S) = \sum_{i=1}^{k} \sum_{j=1}^{k} d(x_i, x_j)$.

We shall discuss several heuristics to approach this problem aiming at considering different (simpler) distance costs, being inspired by distance-based centrality measures.

Before describing the heuristics, we briefly explain the rationale behind the additional assumption (i.e., that every $G[X_i]$ be connected). In our main application (entity-linking) the structure of the graph within each X_i is not very important, and sometimes could be misleading: a very central node in a large candidate set may seem very promising (and may actually minimize the distance to the other sets) but can be blatantly wrong. It is pretty much like the distinction between nepotistic and non-nepotistic links in PageRank computation: the links *within* each host are not very useful in determining the importance of a page—on the contrary, they may be confusing, and are thus often disregarded.

Based on this observation, we can modify the structure of the graph within each set X_i to avoid this kind of trap. This is done by preserving the *external* links (those that connect vertices of X_i to the outside), but at the same time adding or deleting edges within each X_i in a suitable way. In our experiments, we considered two possible approaches:

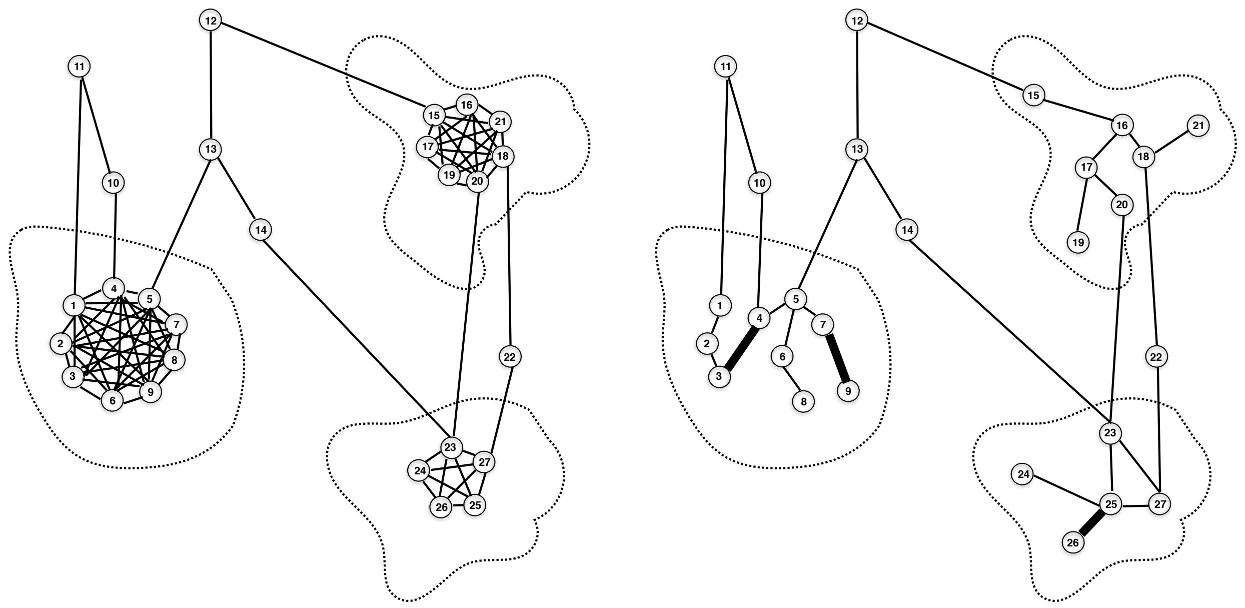
Figure 2: The graph of Figure 1 modified so that each $G[X_i]$ is connected. On the left, the maximally connected version; on the right the minimally connected one (the edges that have been added within each component to make it connected are highlighted: their choice is arbitrary).

- one consists in making $G[X_i]$ *maximally connected*, i.e., transforming it into a clique;
- the opposite approach makes $G[X_i]$ *minimally connected* by adding the minimum number of edges needed to that purpose; this can be done by computing the connected components of $G[X_i]$ and then adding enough edges to join them in a single connected component.

Both approaches guarantee that $G[X_i]$ is connected, so that all the heuristics described below can be applied; Figure 2 shows how the problem instance of Figure 1 should be modified to make the subgraphs connected. In the experimental section, we will run all the heuristics not requiring this assumption and compare the corresponding results also on the original graph.

4.1. Heuristics description

We propose a number of techniques all based on some notion of "set distance" $c(x, Y)$ (in some abstract sense) between a vertex x and a set of vertices Y. We call one such notion "set centrality" because it becomes a centrality measure (in the standard sense) when the set Y equals V. The heuristics we present have all the same approach, but differ for the notion of centrality they refer to [22].

Now given the graph G, and the k subsets of its nodes $X_1, \ldots, X_k$, each heuristic finds a solution S, $x_1, \ldots, x_k$, minimizing

$$\sum_{i=1}^{k} \sum_{j=1}^{k} c(x_i, X_j),$$

instead of minimizing $\sum_{i=1}^{k}\sum_{j=1}^{k} d(x_i, x_j)$. This corresponds to selecting for any set X_i the element $x_i^* \in X_i$ such that $\min_{j=1\ldots k} c(x_i^*, X_j)$ (or any such an element, if there are many).

In the following, we list the several definitions of $c(x, Y)$ we considered.

*Set closeness (*SetCloseness*).* In this case, we set $c(x, Y)$ equal to $\sum_{y\in Y:d(x,y)\neq\infty} d(x, y)$. In other words, we are selecting for each set X_i the node $x_i^* \in X_i$ such that the average distance from x_i^* to all the nodes in the other sets is minimum.

*Set eccentricity (*SetEccentricity*).* We set $c(x, Y)$ equal to $\max_{y\in Y:d(x,y)\neq\infty} d(x, y)$, that is we are selecting for each set X_i the node $x_i^* \in X_i$ such that the average (over all the sets Y_j) of the maximum distances from x_i^* to the nodes $y \in Y_j$ is minimum.

*Set harmonic (*SetHarmonic*).* We set $c(x, Y)$ equal to $-\sum_{y\in Y:x\neq y} 1/d(x, y)$. This corresponds to selecting for each set X_i, the node $x_i^* \in X_i$ such that the average harmonic distance to all the nodes in the other sets is maximum.

*Set hitting distance (*SetHitting*).* We set $c(x, Y)$ to the *hitting distance*: given a vertex x and a set of vertices Y, define the hitting distance of x to Y as $\min_{y\in Y} d(x, y)$. Hence the element $x_i^* \in X_i$ minimizing the average hitting distance is the candidate chosen for the set X_i in that solution.

*Set harmonic hitting distance (*SetHarmonicHitting*).* We set $c(x, Y)$ to $\min_{y\in Y:x\neq y} 1/d(x, y)$.

The main problem with these heuristics is related to their locality (optimization is performed separately for each X_i); moreover the worst-case complexity is $O(m \sum_i |X_i|)$, that reduces to $O(k \cdot m)$ only under the restriction that the sets X_i have $O(1)$ size.

Let us show, as an example, the values of $c(5, X_2)$ for the various heuristics introduced above, when G is the minimally connected instance shown in Figure 2 (right):

- SetCloseness: $c(5, X_2) = d(5, 15) + d(5, 16) + d(5, 17) + d(5, 18) + d(5, 19) + d(5, 20) + d(5, 21) = 3 + 4 + 5 + 5 + 6 + 4 + 6 = 33$;
- SetEccentricity: $c(5, X_2) = \max\{d(5, 15), d(5, 16), d(5, 17), d(5, 18), d(5, 19), d(5, 20), d(5, 21)\} = 6$;
- SetHarmonic: $c(5, X_2) = -(1/d(5, 15)+1/d(5, 16)+1/d(5, 17)+1/d(5, 18)+1/d(5, 19)+1/d(5, 20+1/d(5, 21))) \approx -1.56667$
- SetHitting: $c(5, X_2) = \min\{d(5, 15), d(5, 16), d(5, 17), d(5, 18), d(5, 19), d(5, 20), d(5, 21)\} = 3$;
- SetHarmonicHitting: $c(5, X_2) = \min\{1/d(5, 15), 1/d(5, 16), \ldots, 1/d(5, 21)\} = 1/6$.

5. Experiments

All our experiments were performed on a snapshot of the English portion of Wikipedia as of late February 2013; the graph (represented in the BVGraph format [23]) was symmetrized and only the largest component was retained. The resulting undirected graph has 3 685 351 vertices (87.2% of the vertices of the original graph) and 36 066 162 edges (99.9% of the edges of the original graph). Such a graph will be called the "Wikipedia graph" and referred to as G throughout this experimental section.

Competitors. Our experiments use actual real-world entity-linking problems for which we have human annotations, and tries to compare the accuracy of the methods proposed in Section 4 with a greedy baseline and a number of other heuristics that were proposed in the past for the problem under consideration.

More precisely, we compared our heuristics against the following techniques.

- The greedy baseline (Greedy) works as follows: it first chooses an index i at random, and draws an element $x_i \in X_i$ also at random. Then, it selects a vertex of $x_{i+1} \in X_{i+1}, x_{i+2} \in X_{i+2}, \dots, x_k \in X_k, x_1 \in X_1, \dots, x_{i-1} \in X_{i-1}$ (in this order) minimizing each time the sum of the distances to the previously selected vertices; the greedy algorithm continues doing the same also for $x_i \in X_i$ to get rid of the only element (the first one) that was selected completely at random.

- Two heuristics that we used for comparison have already been observed to be effective in practice [12] and are called Degree and PageRank: they respectively select the vertex with the largest degree and the one with the largest PageRank in each set.

- Three more heuristics that we considered correspond to classical centrality measures:

  - Closeness: select for each set X_i the node $x_i^* \in X_i$ such that $\sum_{v \in V : d(x_i^*, v) \neq 0} d(x_i^*, v)$ is minimum [24].

  - Eccentricity: select for each set X_i the node $x_i^* \in X_i$ such that $\max_{v \in V : d(x_i^*, v) \neq 0} d(x_i^*, v)$ is minimum [25].

  - Harmonic: select for each set X_i the node $x_i^* \in X_i$ such that $\sum_{v \in V} 1/d(x_i^*, v)$ is maximum [22].

- As further strong competitors of the same kind (namely, centrality measures for candidates), we have considered two heuristics based on the popularity score of the Freebase tool [5] provided by Google to assess the inherent relevance of a Wikipedia page. This quantity is computed as a function of its inbound and outbound link counts in Freebase and Wikipedia combined with a popularity score provided by Google; the exact formula is not public. In the case of Entity Linking, the idea of using Freebase popularity score to this aim has been proposed by Alhelbawy [6]; this work provides a complete system both for candidate generation and disambiguation: in the latter part they use a combination of Freebase popularity score and PageRank to extract from each set the right candidate. This is done on a graph where the nodes are pairs (mentions, entity) and there is an edge if the two entities are connected in Wikipedia. Even if we will show that when applied to this latter graph their objective function is less effective when the candidates are given (see Section 5.3), we have observed that their objective function is effective when applied to our graphs.

– The FREEBASE×PAGERANK heuristic uses the objective function in [6]: it queries the Google API for the Freebase score of each candidate page. For each set it selects the candidate whose Freebase score multiplied by its PageRank is maximum.

– The FREEBASE heuristic for each set simply selects the candidate with largest Freebase score.

It is worth noting that all the heuristic proposed, as well as the CLOSENESS, ECCENTRICITY, and HARMONIC baselines, require to run a Breadth First Search (in short, BFS) for any node in each X_i: in the case of our heuristics, the BFS can be stopped when all the vertices belonging to some X_j have been visited.

Preparing the dataset. The real-world entity-linking dataset has been taken from [10] which contains a large number of human-labelled annotations. For retrieving the candidates, we created an index over all Wikipedia pages with different fields (title, body, anchor text) and used a variant of BM25F [26] for ranking, returning the top 100 scoring candidate entities. Since the candidate selection method was the same for every graph-based method employed, this procedure should produce no bias in the experimental outcomes.

The 100 problem instances contained in the dataset have 11.73 entities on average (with a maximum of 53), and the average number of candidates per entity is 95.90 (with a maximum of 200). Each of the problem instances in the NEL dataset is annotated, and in particular, for every i there is a subset $X_i^* \subseteq X_i$ of *fair* vertices (that is, vertices that are good candidates for that set): typically $|X_i^*| = 1$. Note that, for every instance in the NEL dataset, we deleted the sets X_i such that X_i^* were not included in the largest connected component of the Wikipedia graph. The number of sets X_i deleted was never more than 2 (for two instances). We have not considered instances in which, after these modifications, we have just one set X_i: this situation happened in 5 cases. So the problem set on which we actually ran our algorithm contains 95 instances.

The experiments. For every instance, we ran all the heuristics described in Section 4 on the original symmetric graph. We also ran all our heuristics considering the maximal and minimal connection[5] approach. We ran our heuristics and compared them to each other and the baselines.

[5]To obtain the minimal connection of each $G[X_i]$, we chose to connect the vertex of maximum degree of its largest component with an (arbitrary) vertex of each of its remaining components.

In Section 5.1 we will give some evidence about the rationale behind our objective function, in Section 5.2 we will compare all the heuristics in finding a solution for MinDR, while in Section 5.3 we will analyze the effectiveness of the several heurisitics in finding a solution close to the groundtruth.

5.1. Soundness of the objective function

In order to understand whether the ground truth is in any sense related to distance, we have computed, for any instance, the objective function of the ground truth solution and we have compared this quantity with ten random feasible solutions (a solution composed by a random x_i for any X_i): in 90 cases over 95, we obtained that the ground truth solution had objective function less than or equal to the minimum objective function found among the ten random solutions; in all the remaining 5 cases, it was less than or equal to the average. We argue that there is an high correlation between a low distance cost and being a ground truth. Moreover in the following sections we will see that the methods that achieve smaller values of the objective function are also the ones that provide solutions closer to the ground truth. We can conclude that the distance cost is an important factor to be taken into account, although certainly we cannot guarantee it is the unique one.

5.2. Optimizing the objective function

In this Section we will compare all the heuristics in finding a solution of minimum distance cost. For any instance, when comparing the distance cost f of the solutions S_j returned by some algorithm A_j, we have computed the *distance-cost ratio* of A_j, defined as

$$\frac{f(S_j)}{\min_i f(S_i)} \cdot 100.$$

Intuitively this corresponds to the approximation ratio of each solution with respect to the best solution found by all the considered algorithms: hence, for every instance, the best algorithm has minimum distance-cost ratio and it equals 100.

In Table 1, we report the distance-cost ratio for all the heuristics. Each column corresponds to a different kind of connection and hence the distance cost is computed on a different graph.

For all the connection approaches the Greedy baseline and the SetCloseness heuristics seem to obtain more often a minimum distance cost solution. Another good option is the SetHitting heuristic, while the other methods seem to be more far away from a distance-optimal result.

In the last row of Table 1, we report the distance-cost ratio for the ground-truth solution given by the fair candidates. It seems that for any instance, the ground truth has distance cost averagely 17%-22% higher than the best solution we achieve by using the heuristics. This suggests that the optimization strategies we have are more than enough for the original problem and probably the objective function could be improved considering also other aspects. Moreover, as suggested by the following section, this should not cast doubt on the soundness of our objective function: the results of the next section will show that the three best

	Distance-cost ratio		
	Original Connection	Maximal Connection	Minimal Connection
Heuristic	Average (± Std Error)	Average (± Std Error)	Average (± Std Error)
SetCloseness	106.513 (± 1.943)	**102.063 (± 1,118)**	104.329 (± 2.135)
SetEccentricity	133.003 (± 3,040)	118.194 (± 1.613)	123.434 (± 2.046)
SetHarmonic	140.756 (± 3,932)	119.295 (± 1.673)	115.873 (± 2.363)
SetHitting	108.060 (± 4.286)	104.121 (± 1.269)	105.216 (± 2.222)
SetHarmonicHitting	142.488 (± 4.555)	122.376 (± 1.697)	131.391 (± 2.645)
Greedy	**105.867 (± 1.236)**	102.101(±0.323)	**103.874 (± 0.529)**
Degree	128.076 (± 4.760)	115.597(± 2.362)	115.229 (± 2.339)
PageRank	127.505 (± 4,791)	116.324(± 2.436)	114.326 (± 2.297)
Closeness	118.446 (± 2.886)	113.750(± 2.352)	114.331 (± 2.424)
Eccentricity	145.432 (± 4.927)	118.194(± 1.613)	137.676 (± 4,112)
Harmonic	128.076 (± 4.760)	115.860(± 2.362)	115.229 (± 2.334)
Freebase×PageRank	124.431 (± 3.649)	117.340 (± 2.340)	118.174 (± 3.253)
Freebase	138.224 (± 3.767)	122.221(± 2.321)	132.869 (± 3.464)
Ground truth	121.877 (± 2.162)	116.593 (± 1.774)	121.448 (± 2.062)

Table 1: Average distance-cost ratio among all the instances.

heuristics above, namely Greedy, SetCloseness, and SetHitting, are also the one providing more often solutions closer to the ground truth.

5.3. Comparing the solutions with the ground truth

Besides evaluating the distance cost of the solutions found by the various heuristics, we can compute how many of the elements found are fair: we normalize this quantity by k, so that 1.0 means that all the k candidates selected are fair. This quantity corresponds to the *precision* of a solution: note that in our case the precision is equal to the recall.

In Table 2 we report, for each heuristic, the average precision (across all the instances) along with the standard error. The variance of the results seems not to differ too much across all the methods. Notice that the precision of the Freebase heuristic is equal for all the kinds of connection, since it does not depend on the graph but just on the sets. The SetCloseness heuristic, ran by using the original connection (i.e., in the original symmetrized graph) outperforms all the other heuristics, including Freebase: it selects more than 53% of fair candidates. Another promising alternative is the SetHitting heuristic that seems to be the more effective in a minimal connection scenario.

It is worth observing that also the Greedy and Freebase×PageRank approaches seem to outperform the baseline techniques (Degree and PageRank) in all the scenarios. Moreover notice that our definitions for the set-centrality heuristics SetCloseness and SetEccentricity improve in this context the respective traditional definitions of classical centrality, namely Closeness and Eccentricity. This is also true of the SetHarmonic heuristic compared to Harmonic, but only for the original graph.

We remark that the SetCloseness heuristic always outperforms Freebase×PageRank in all the

	Precision		
	Original Connection	Maximal Connection	Minimal Connection
Heuristic	Average (± Std Error)	Average (± Std Error)	Average (± Std Error)
SetCloseness	**0.531 (± 0.023)**	**0.466 (±0.027)**	0.464 (± 0.027)
SetEccentricity	0.362 (± 0.024)	0.369 (± 0.025)	0.376 (± 0.026)
SetHarmonic	0.429 (± 0.023)	0.342 (± 0.023)	0.390 (± 0.023)
SetHitting	0.473 (± 0.027)	0.462 (± 0.027)	**0.471 (± 0.027)**
SetHarmonicHitting	0.346 (± 0.023)	0.329 (± 0.024)	0.331 (± 0.024)
Greedy	0.443 (±0.026)	0.436 (± 0.026)	0.435 (± 0.027)
Degree	0.399 (± 0.023)	0.412 (± 0.024)	0.395 (± 0.023)
PageRank	0.410 (± 0.026)	0.408 (± 0.025)	0.399 (± 0.024)
Closeness	0.430 (± 0.021)	0.434 (± 0.022)	0.425 (± 0.021)
Eccentricity	0.361 (± 0.024)	0.359 (±0.023)	0.360 (± 0.024)
Harmonic	0.399 (± 0.023)	0.413 (± 0.024)	0.395 (± 0.023)
Freebase×PageRank	0.443 (± 0.027)	0.449 (± 0.026)	0.425 (± 0.027)
Freebase		0.472 (± 0.028)	

Table 2: Precision, i.e. average fraction of fair candidates selected by each heuristic.

connectivity scenarios. We also ran this latter selection heuristic transforming the graph as described in [6] and providing the same problem set: the resulting average precision was 0.284.

Furthermore, we applied the Degree and PageRank based heuristics by using the same problem set but *in the original directed graph*; in this case, we did not symmetrize the graph and we did not enforce any connectivity of the subgraphs $G[X_i]$: the resulting average values for the precision (± standard error) are respectively 0.327 (±0.020) and 0.336 (±0.022), and they are both worse than the precision achieved by Degree and PageRank heuristics in Table 1. This seems to confirm that considering the undirected version of the graph improves the effectiveness of existing techniques.

Finally it is worth observing that enforcing some connectivity of the subgraphs $G[X_i]$ guarantees the applicability of our heuristics in a more suitable scenario but this modification does not affect heavily the corresponding precision and sometimes can improve the performance of some heuristic, like Degree, Closeness, and Harmonic (see third and fourth columns of Table 2 compared with the second one).

6. Conclusions and future work

Inspired by the entity-linking task in NLP, we defined and studied a new graph problem related to Maximum Capacity Representative Set and we proved that this problem is NP-hard in general (since our reduction does not preserve approximability, it remains an open problem to determine the exact approximability of the new problem we proposed). Moreover, we provided several heuristics to solve the problem testing our proposals on a real-world dataset. Finally we show (in the Appendix) that the problem can be solved efficiently in some special cases.

One of our heuristics, SetCloseness, turns out to be very effective, actually more effective than other methods previously proposed in the literature: not only it works better than a simple greedy approach using the same cost function adopted here, but it is even more effective than using the popularity score of Freebase.

Since Freebase is itself a knowledge graph like the Wikipedia graph, it would be interesting to see whether the proposed approaches are effective replacing the Wikipedia Graph with the Freebase Graph.

[1] R. Mihalcea, A. Csomai, Wikify!: Linking documents to encyclopedic knowledge, in: Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management, CIKM ’07, ACM, New York, NY, USA, 2007, pp. 233–242. doi:10.1145/1321440.1321475.

[2] R. Blanco, G. Ottaviano, E. Meij, Fast and space-efficient entity linking in queries, in: Proceedings of the Eight ACM International Conference on Web Search and Data Mining, WSDM ’15, ACM, New York, NY, USA, 2015.

[3] X. Han, L. Sun, J. Zhao, Collective entity linking in web text: A graph-based method, in: Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR ’11, ACM, 2011, pp. 765–774. doi:10.1145/2009916.2010019.

[4] R. Blanco, P. Boldi, A. Marino, Entity-linking via graph-distance minimization, in: Proceedings 3rd Workshop on GRAPH Inspection and Traversal Engineering, GRAPHITE 2014, Grenoble, France, 5th April 2014., 2014, pp. 30–43. doi:10.4204/EPTCS.159.4.

[5] K. D. Bollacker, R. P. Cook, P. Tufts, Freebase: A shared database of structured general human knowledge, in: Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence, July 22-26, 2007, Vancouver, British Columbia, Canada, 2007, pp. 1962–1963.

[6] A. Alhelbawy, R. J. Gaizauskas, Graph ranking for collective named entity disambiguation, in: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 2: Short Papers, 2014, pp. 75–80. doi:10.3115/v1/p14-2013.

[7] S. Cucerzan, Large-scale named entity disambiguation based on wikipedia data, in: In Proc. 2007 Joint Conference on EMNLP and CNLL, 2007, pp. 708–716.

[8] D. Milne, I. H. Witten, Learning to link with wikipedia, in: Proceedings of the 17th ACM Conference on Information and Knowledge Management, CIKM ’08, ACM, New York, NY, USA, 2008, pp. 509–518. doi:10.1145/1458082.1458150.

[9] R. C. Bunescu, M. Pasca, Using encyclopedic knowledge for named entity disambiguation, in: EACL, The Association for Computer Linguistics, 2006.

[10] S. Kulkarni, A. Singh, G. Ramakrishnan, S. Chakrabarti, Collective annotation of Wikipedia entities in web text, in: Knowledge Discovery and Data Mining, 2009, pp. 457–466. doi:10.1145/1557019.1557073.

[11] C. Fellbaum (Ed.), WordNet An Electronic Lexical Database, The MIT Press, Cambridge, MA ; London, 1998. doi:10.2307/417141.

[12] B. Hachey, W. Radford, J. R. Curran, Graph-based named entity linking with wikipedia, in: Proceedings of the 12th International Conference on Web Information System Engineering, WISE'11, Springer-Verlag, Berlin, Heidelberg, 2011, pp. 213–226. doi:10.1007/978-3-642-24434-6_16.

[13] R. Navigli, Word sense disambiguation: a survey, ACM COMPUTING SURVEYS 41 (2) (2009) 1–69. doi:10.1145/1459352.1459355.

[14] R. Mihalcea, Unsupervised large-vocabulary word sense disambiguation with graph-based algorithms for sequence data labeling, in: Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing, HLT '05, Association for Computational Linguistics, Stroudsburg, PA, USA, 2005, pp. 411–418. doi:10.3115/1220575.1220627.

[15] R. Navigli, P. Velardi, Structural semantic interconnections: A knowledge-based approach to word sense disambiguation, IEEE Trans. Pattern Anal. Mach. Intell. 27 (7) (2005) 1075–1086. doi:10.1109/TPAMI.2005.149.

[16] R. Navigli, M. Lapata, Graph connectivity measures for unsupervised word sense disambiguation, in: Proceedings of the 20th International Joint Conference on Artifical Intelligence, IJCAI'07, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2007, pp. 1683–1688.

[17] R. Navigli, M. Lapata, An experimental study of graph connectivity for unsupervised word sense disambiguation, IEEE Trans. Pattern Anal. Mach. Intell. 32 (4) (2010) 678–692. doi:10.1109/TPAMI.2009.36.

[18] M. Cuadros, G. Rigau, Knownet: A proposal for building highly connected and dense knowledge bases from the web, in: First Symposium on Semantics in Systems for Text Processing, 2008, pp. 71–84. doi:10.3115/1626481.1626488.

[19] P. Crescenzi, V. Kann, Approximation on the web: A compendium of np optimization problems, in: RANDOM, 1997, pp. 111–118. doi:10.1007/3-540-63248-4_10.

[20] M. Bellare, Interactive proofs and approximation: Reduction from two provers in one round., in: ISTCS, 1993, pp. 266–274. doi:10.1109/ISTCS.1993.253462.

[21] M. Serna, L. Trevisan, F. Xhafa, The approximability of non-boolean satisfiability problems and restricted integer programming, Theoretical Computer Science 332 (13) (2005) 123 – 139. doi:10.1016/j.tcs.2004.10.014.

[22] P. Boldi, S. Vigna, Axioms for centrality, Internet Mathematics 10 (3-4) (2014) 222–262. doi:10.1080/15427951.2013.865686.

[23] P. Boldi, S. Vigna, The WebGraph framework I: Compression techniques, in: Proc. of the Thirteenth International World Wide Web Conference (WWW 2004), ACM Press, Manhattan, USA, 2004, pp. 595–601. doi:10.1145/988672.988752.

[24] A. Bavelas, A mathematical model for group structures, Human Organization 7 (1948) 16–30.

[25] M. Borassi, P. Crescenzi, M. Habib, W. A. Kosters, A. Marino, F. W. Takes, On the solvability of the six degrees of kevin bacon game - A faster graph diameter and radius computation method, in: Fun with Algorithms - 7th International Conference, FUN 2014, Lipari Island, Sicily, Italy, July 1-3, 2014. Proceedings, 2014, pp. 52–63. doi:10.1007/978-3-319-07890-8_5.

[26] R. Blanco, P. Boldi, Extending bm25 with multiple query operators, in: Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '12, ACM, New York, NY, USA, 2012, pp. 921–930. doi:10.1145/2348283.2348406.

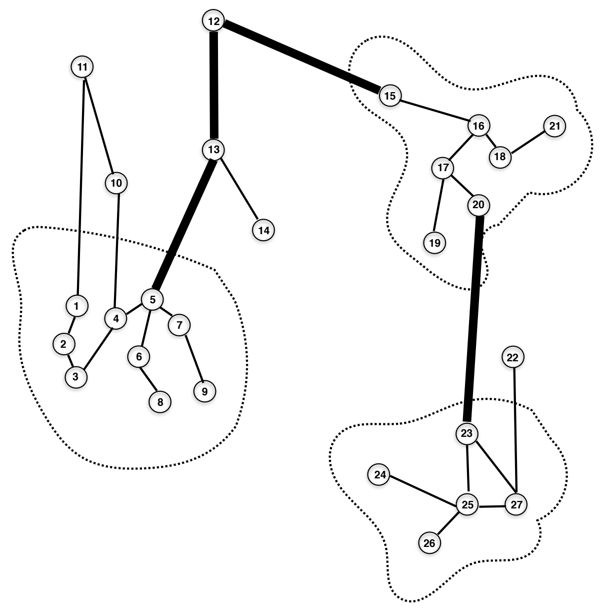

Figure A.3: The running example (of Figure 2, right) made decomposable after removing two edges. The highlighted edges are *useful*.

Appendix A. The decomposable case

In this section we will study the MinDR problem under some restrictive hypothesis and we will show that in this case a linear exact algorithm exists.

In particular, we assume that the graph G (possibly weighted) is such that:

- any set X_i induces a connected subgraph on G, i.e., $G[X_i]$ is connected,
- for any $i \neq j$, for any $x \in X_i$ and $y \in X_j$, x and y do not belong to the same biconnected component.

The problem, under these further restrictions, will be called *decomposable* MinDR. Note that the second condition implies that a decomposable MinDR is in fact an instance of MinDIR, because it implies that no two sets can have nonempty intersection.

The example of Figure 2 (right) is not decomposable; nonetheless, removing two edges we obtain a decomposable variant shown in Figure A.3.

Since in real world scenarios all the candidate sets very often lean on a same biconnected component, these hypothesis seems to be too strong to make the algorithm useful in practice; anyway, we think that this case is interesting from a theoretical point of view.

Let us consider an instance $(G, \{X_1, \dots X_k\})$ of the decomposable MinDR problem on a graph $G = (V, E)$. An edge $e = (x, y) \in E$ is called *useful* if it is a bridge, x and y do not belong to the same set X_i, and there are at least two indices i and j such that X_i and X_j are in different components of $G - e$ (since e is a bridge, the graph obtained removing the edge e from G is no more connected). In Figure 2 we highlighted the useful edges.

Appendix A.1. Decomposing the problem

The main trick that allows us to obtain a linear-time solution for the decomposable case is that we can actually decompose the problem (hence the name) through useful edges. First observe that, trivially:

Remark 1. *Let $e = (x, y)$ be a useful edge and let Z_x and Z_y be the two connected components of $G - e$ containing x and y, respectively. In G, all paths from any $x' \in Z_x$ to any $y' \in Z_y$ must include e.*

Moreover:

Remark 2. *Let $e = (x, y)$ be a useful edge. There cannot be an index i such that X_i has a nonempty intersection with two components of $G - e$.*

In fact, assume by contradiction that one such X_i exists, and let $u, w \in X_i$ be two vertices living in the two different components of $G - e$: since $G[X_i]$ is connected, there must be a path connecting u and w and made only of elements of X_i; because of Remark 1, this path passes through e, but this would imply that $x, y \in X_i$, in contrast with the definition of useful edge.

Armed with the previous observations, we can give the following further definitions. Let Y_x (respectively, Y_y) be the set of sets X_i such that $X_i \subseteq Z_x$ (respectively, $X_i \subseteq Z_y$); we denote the sets of nodes in Y_x and Y_y by $V(Y_x) \subseteq Z_x$ and $V(Y_y) \subseteq Z_y$, respectively.

By virtue of Remark 1, all the paths in G from any $x' \in V(Y_x)$ to any $y' \in V(Y_y)$ pass through e. This implies also that there is no simple cycle in the graph including both $x' \in V(Y_x)$ and $y' \in V(Y_y)$.

Given a solution S for MinDIR$(G, \{X_1, \dots, X_k\})$, and a useful edge (x, y), we have:

$$\begin{aligned} \sum_{x_i, x_j \in S} d(x_i, x_j) &= \sum_{x_i, x_j \in S \cap V(Y_x)} d(x_i, x_j) + \sum_{x_i, x_j \in S \cap V(Y_y)} d(x_i, x_j) + \\ & \quad 2 \sum_{x_i \in S \cap V(Y_x), x_j \in S \cap V(Y_y)} \left(d(x_i, x) + d(x, y) + d(y, x_j) \right). \end{aligned} \tag{A.1}$$

Indeed all the shortest paths from any $x_i \in S \cap V(Y_x)$ to any $x_j \in S \cap V(Y_y)$ pass through the useful edge (x, y) by Remark 1. Moreover, since the sets $X_1, \dots, X_k$ are disjoint, we have that $|S \cap V(Y_x)| = |Y_x|$ and $|S \cap V(Y_y)| = |Y_y|$, that is, a solution has exactly one element for each set in Y_x (respectively, Y_y). Hence we can rewrite the last summand of Equation A.1 as follows:

$$\sum_{x_i \in S\cap V(Y_x), x_j \in S \cap V(Y_y)} (d(x_i, x) + d(y, x_j) + d(x, y)) = |Y_y| \cdot \sum_{x_i \in S\cap V(Y_x)} d(x_i, x) + |Y_x| \cdot \sum_{x_j \in S\cap V(Y_y)} d(y, x_j) + |Y_x| \cdot |Y_y| \cdot d(x, y). \qquad (A.2)$$

By combining Equation A.1 and A.2, we can conclude that finding a solution for MinDIR$(G, \{X_1, \ldots, X_k\})$ can be decomposed into the following two subproblems:

1. find S_x minimizing $\sum_{x_i, x_j \in S\cap V(Y_x)} d(x_i, x_j) + 2\sum_{x_i \in S\cap V(Y_x)} |Y_y| d(x_i, x)$ in the instance $(G[Z_x], Y_x)$;
2. find S_y minimizing $\sum_{x_i, x_j \in S\cap V(Y_y)} d(x_i, x_j) + 2\sum_{x_j \in S\cap V(Y_y)} |Y_x| d(y, x_j)$ in the instance $(G[Z_y], Y_y)$.

Note that both instances are smaller than the original one because of the definition of a useful edge. The idea of our algorithm generalizes this principle; note that the new objective function we must take into consideration is slightly more complex than the original one: in fact, besides the usual all-pair–distance cost there is a further summand that is a weighted sum of distances from some fixed nodes, such as x for the instance $(G[Z_x], Y_x)$ and y for the instance $(G[Z_y], Y_y)$.

We hence define an extension of the MinDR problem, that we call ExtMinDR (for *Extended Minimum Distance Representatives*). In this problem, we are given:

- an undirected graph $G = (V, E)$ (possibly weighted)
- k subsets of its set of vertices, $X_1, \ldots, X_k \subseteq V$
- a multiset B of vertices, each $x \in B$ endowed with a weight $b(x)$.

A feasible solution for the ExtMinDR is a multiset $S = \{x_1, \ldots, x_k\}$ of vertices of G, such that for any i, with $1 \leq i \leq k$, $S \cap X_i \neq \emptyset$ (i.e., the set contains at least one element from every set). Its cost is

$$f(S) = \sum_{i=1}^{k}\sum_{j=1}^{k} d(x_i, x_j) + \sum_{i=1}^{k}\sum_{z\in B} b(z) d(x_i, z). \qquad (A.3)$$

The goal is finding the solution of minimum cost, i.e., a feasible solution S such that $f(S)$ is minimum. The original version of the problem is obtained by letting $B = \emptyset$.

We are now ready to formalize our decomposition through the following Theorem.

Theorem 1. *Let us be given a decomposable* ExtMinDR *instance* $(G, \{X_1, \ldots, X_k\}, B, b)$ *and a useful edge* $e = (t_0, t_1)$*. For every* $s \in \{0, 1\}$*, let* Z_s *be the connected component of* $G - e$ *containing* t_s*,* Y_s *be the set of sets* X_i *such that* $X_i \subseteq Z_s$ *and* $V(Y_s)$ *be the union of those* X_i*'s. Let also* B_s *be the intersection of* B *with* Z_s*. Define a new instance* $I_s = (G[Z_s], \{X_i, i \in Y_s\}, B_s \cup \{t_s\}, b_s)$ *where*

$$b_s(t_s) = 2|Y_{1-s}| + \sum_{z \in B_{1-s}} b(z) \text{ and } b_s(z) = b(z), \text{ for any } z \in B.$$

Then the cost $f(S)$ of an optimal solution S of the original problem is equal to

$$f(S_0) + f(S_1) + 2|Y_0||Y_1|d(t_0, t_1) + \sum_{s\in\{0,1\}} \left(|S \cap V(Y_s)| \cdot \sum_{z\in B\cap Z_{1-s}} b(z)d(t_s, z) \right)$$

where S_s is an optimal solution for the instance I_s.

Proof. We can rewrite the objective function in Equation A.3 as follows.

$$\begin{aligned}\sum_{x_i,x_j\in S} d(x_i,x_j) + \sum_{x_i\in S}\sum_{z\in B} d(x_i,z)b(z) \quad = \quad & \sum_{x_i,x_j\in S\cap V(Y_0)} d(x_i,x_j) + \sum_{x_i,x_j\in S\cap V(Y_1)} d(x_i,x_j) + \\ & 2|Y_1| \sum_{x_i\in S\cap V(Y_0)} d(x_i,t_0) + \sum_{x_i\in S\cap V(Y_0)}\sum_{z\in B} d(x_i,z)b(z) + \\ & 2|Y_0| \sum_{x_j\in S\cap V(Y_1)} d(t_1,x_j) + \sum_{x_i\in S\cap V(Y_1)}\sum_{z\in B} d(x_i,z)b(z) + \\ & 2|Y_0||Y_1|d(t_0,t_1). \qquad \text{(A.4)}\end{aligned}$$

Indeed, if $z \in B \cap Z_1$, for each node $x_i \in S \cap V(Y_0)$, we have $d(x_i, z) = d(x_i, t_0) + d(t_0, z)$ (and analogously, if $z \in B \cap Z_0$, for each node $x_i \in S \cap V(Y_1)$, we have $d(x_i, z) = d(x_i, t_1) + d(t_1, z)$). Hence:

$$\sum_{x_i\in S\cap V(Y_0)}\sum_{z\in B} d(x_i,z)b(z) \quad = \quad \sum_{x_i\in S\cap V(Y_0)}\sum_{z\in B\cap Z_0} d(x_i,z)b(z) + \sum_{x_i\in S\cap V(Y_0)}\sum_{z\in B\cap Z_1} d(x_i,t_0)b(z) + d(t_0,z)b(z)$$

and

$$\sum_{x_i\in S\cap V(Y_1)}\sum_{z\in B} d(x_i,z)b(z) \quad = \quad \sum_{x_i\in S\cap V(Y_1)}\sum_{z\in B\cap Z_1} d(x_i,z)b(z) + \sum_{x_i\in S\cap V(Y_1)}\sum_{z\in B\cap Z_0} d(x_i,t_1)b(z) + d(t_1,z)b(z).$$

It is worth observing that t_0 or t_1 might already belong to B: this is why we assumed that B is a multiset.

This means that Equation A.4 can be split in two independent parts corresponding to the followings.

$$f(S_0) = \sum_{x_i,x_j\in S\cap V(Y_0)} d(x_i,x_j) + \sum_{x_i\in S\cap V(Y_0)}\sum_{z\in B\cap Z_0} d(x_i,z)b(z) + \sum_{x_i\in S\cap V(Y_0)} d(x_i,t_0) \cdot \left(2|Y_1| + \sum_{z\in B\cap Z_1} b(z)\right)$$

$$f(S_1) = \sum_{x_i,x_j\in S\cap V(Y_1)} d(x_i,x_j) + \sum_{x_i\in S\cap V(Y_1)}\sum_{z\in B\cap Z_1} d(x_i,z)b(z) + \sum_{x_i\in S\cap V(Y_1)} d(x_i,t_1) \cdot \left(2|Y_0| + \sum_{z\in B\cap Z_0} b(z)\right)$$

We can conclude that, by adding t_s to $B \cap Z_s = B_s$, with weight equal to $b_s = 2|Y_{1-s}| + \sum_{z\in B\cap Z_{1-s}} b(z)$, $f(S)$ can be reduced to $f(S_0)$ and $f(S_1)$. □

For completeness, we need to consider the base case of an instance with just one set $(G, \{X_1\}, B, b)$: the solution in this case is just one node $x \in X_1$ and the objective function to be minimized is simply $\sum_{z\in B} d(x, z)b(z)$. The optimal solution can be found by performing a BFS from every $z_j \in B$ (in increasing order of j), maintaining for each node $y \in X_1$, $g(y) = \sum_{z_t\in B, t<j} d(x, z_t)b(z_t)$, and picking the node having maximum final $g(y)$. This process takes $O(|B| \cdot |E(G[X_1])|)$. It is worth observing that in our case the size of the multiset B is always bounded by k. Moreover since $\sum_{i=1}^{k} |E(G[X_i])| \leq |E(G)| = m$, the overall complexity for all these base cases is bounded by $O(k \cdot m)$.

Algorithm 1: UsefulEdge

```
Input: An instance $G, \{X_1, \ldots, X_k\}, B, b$
Output: A useful edge, or null
Pick a node $u$ of the set $X_i$ of the instance $G, \{X_1, \ldots, X_k\}, B, b$
Mark all the nodes as unseen
dfs[] $\leftarrow -1$,    visited $\leftarrow 0$,    usefulEdgeFound $\leftarrow$ $false$,    usefulEdge $\leftarrow$ $null$
DFS($u$,$-1$)
if usefulEdgeFound then
    return usefulEdge
else
    return $null$
end
```

Algorithm 2: DFS

```
Input: A node $u$, its parent $p$
Output: A pair ($t$,$Y$), where $t$ is an integer and $Y$ is a set of indices
if usefulEdgeFound then return $null$ ;
Mark $u$ as seen
dfs[$u$] $\leftarrow$ visited
visited $\leftarrow$ visited + 1
furthestAncestor $\leftarrow$ visited
$Y \leftarrow \emptyset$
if $t \in X_i$ then $Y \leftarrow Y \cup \{i\}$ ;
for $v \in N(u)$ s.t. $w \neq p$ do
    if $v$ is unseen then
        $(t', Y') \leftarrow$ DFS($v, u$)
        if $t' >$ dfs[$u$] and $\emptyset \neq Y' \neq \{1, \ldots, k\}$ then
            usefulEdgeFound $\leftarrow$ $true$
            usefulEdge $\leftarrow (u, v)$
            return $null$
        end
        furthestAncestor $\leftarrow$ min(furthestAncestor, $t'$)
        $Y \leftarrow Y \cup Y'$
    else
        furthestAncestor $\leftarrow$ min(furthestAncestor, dfs[$v$])
    end
end
return (furthestAncestor, $Y$)
```

Appendix A.2. Finding useful edges

For every instance with more than one set, given an useful edge e the creation of the subproblems as described above is linear, so we are left with the issue of finding useful edges. This task can be seen as a variant of the standard depth-first search of bridges, as shown in Algorithm 1 and 2. Recall that bridges can be found by performing a standard DFS that numbers the nodes as they are found (using a global counter, and keeping the DFS numbers in an array); every visit returns the index of the least ancestor reachable through a back edge while visiting the DFS-subtree rooted at the node where the visit starts from. Every time a DFS returns a value that is larger than the number of the node currently being visited, we have found a bridge.

The variant consists in returning not just the index of the least ancestor reachable, but also the set of indices i that are found while visiting the subtree. If the set of indices and its complement are both different from $\emptyset$ then the bridge is useful: at this point, a "rapid ascent" is performed to get out of the recursive

Algorithm 3: DecomposableMinDR

```
Input: A graph $G = (V, E)$, $X_1 \dots, X_k \subseteq V$, a weighted multiset $B$ of nodes in $V$, where each element in $B$ has a
       weight $b$. $G[X_i]$ is connected for every $i$ and moreover for all $i \neq j$ and $x \in X_i$, $y \in X_j$, the two vertices $x$ and $y$
       do not belong to the same biconnected component of $G$.
Output: A solution $S = \{x_1, \dots, x_k\}$ such that for any $i$, with $1 \leq i \leq k$, $x_i \in X_i$, minimizing
        $\sum_{i=1}^{h} \sum_{j=1}^{k} d(x_i, x_j) + \sum_{i=1}^{k} \sum_{z \in B} b(z) d(x_i, z)$
Find a useful edge $e = (x, y)$, if it exists, using Algorithm 1
if the useful edge does not exist then
    if $k \neq 1$ then
        Fail!
    end
    Output the element $x_1 \in X_1$ minimizing $\sum_{z \in B} b(z) d(x_1, z)$
else
    Let $Z_x$ (respectively $Z_y$) be the connected component of $T - e$ containing $x$ (respectively $y$) .
    Let $Y_x$ (respectively $Y_y$) be the indices $i$ such that $X_i \subseteq Y_x$ ($X_i \subseteq Y_y$, respectively)
    $B' \leftarrow B \cup \{x\}$ (multiset union) with $b(x) = 2|Y_y| + \sum_{z \in B \cap Z_y} b(z)$
    $B' \leftarrow B' \cap Z_x$ (multiset intersection)
    $S' \leftarrow$ DecomposableMinDR($T[Z_x], Y_x, B'$)
    $B'' \leftarrow B \cup \{x\}$ (multiset union) with $b(y) = 2|Y_x| + \sum_{z \in B \cap Z_x} b(z)$
    $B'' \leftarrow B'' \cap Z_y$ (multiset intersection)
    $S'' \leftarrow$ DecomposableMinDR($T[Z_y], Y_y, B''$)
    return $S' \cup S''$
end
```

procedure.

Appendix A.3. The final algorithm

Combining the observations above, we can conclude that the overall complexity of the algorithm (summarized in Algorithm 3) is $O(k \cdot m)$.