Induced Acyclic Subgraphs With Optimized Endpoints

Moussa Abdenbi
Département d'informatique
Université du Québec à Montréal
Québec, Canada
Email: abdenbi.moussa@courrier.uqam.ca

Alexandre Blondin Massé
LACIM
Université du Québec à Montréal
Québec, Canada

Alain Goupil
Université du Québec à Trois-Rivières
Québec, Canada

***Abstract*—Given a lexicon, can we build a strategy for learning specialized vocabulary? If so, how to minimize its cost while maximizing its efficiency? We provide elements of answers to these questions, by using graph theory. A lexicon is represented by a directed graph of the *defining* relation between words and a *strategy* is an ordered sequence of vertices. We focus on two graphs properties, *acyclicity* which helps us to avoid words with cyclic definitions and *induced* to consider all the arcs between chosen words. We are interested in the *induced* and *acyclic* subgraphs of a directed graph containing a fixed set of vertices. To model cost minimization, we consider an optimization criteria based on the difference between the number of *sinks* and the number of *sources* of the subgraph, which represents words which appear in several definitions. We start with a study of the complexity of finding these subgraphs and we prove that it is a Non-deterministic Polynomial-time hard (NP-hard) problem. This observation leads us to provide two approximate solutions: a *greedy heuristic* and a local search enriched with *tabu restrictions*. Finally, we evaluate the efficiency of this graph based strategy comparing with psycholinguistic based strategies, on three digital dictionaries.**

Keywords–Directed graphs; learning strategies; computational linguistics.

I. INTRODUCTION

Acquiring specialized vocabulary, as well as learning a new language, often involves different types of learning approaches [1]. One such strategy is *dictionary look-up*, *i.e.*, learning new words by reading definitions from a dictionary or having someone explain, describe, characterize the new words with already known other words. Another complementary strategy is *direct sensorimotor experience*, *i.e.*, learning by seeing, hearing, smelling, tasting, touching, or interacting in any other way with the object referenced by the words. The effort of learning a word directly seems more *costly* than the one of learning a word by definition, since it involves trial and error [2]. Therefore, it is important to carefully choose the words to be learned directly to reduce the effort required to assimilate their meaning. In particular, psycholinguistic and pedagogical provide some criteria to determine these words [3]–[7].

In this paper, we are interested in a better approach based on computational linguistics to extract these subsets of words for different specialized vocabularies. The relation between the words in a lexicon or a dictionary definition and the meaning of the defined word, is modeled by a directed graph or *digraph*. So, it is quite natural that we turn to graph theory and the study of subgraphs, which has attracted constant interest and lead to several applications [8]–[11].

A *lexicon* can be described as a digraph, where a vertex is a node containing a word and the directed "defining" relation between words is represented by arcs. A *strategy* is an ordered sequence of vertices [2] [12]. We focus our attention on *induced and acyclic* subgraphs of digraphs, and we consider an optimization criterion based of the difference between the number of *sinks* vertices and the number of *sources* vertices. In [13], we produced a brief study of the subgraphs satisfying the constraints above and proved that finding these subgraphs is an NP-hard problem. In this paper, we expand this problem and add the constraint that the subgraphs must contain a fixed and arbitrary set of vertices from the digraph. This supplementary constraint provides a larger and more interesting family of problems. We believe that the subgraphs under investigation here are optimal learning strategies, according to the strategy cost defined in [2]. Indeed, the acyclicity constraint models the absence of circular definitions between words and the induced constraint mean that all links between selected words in the dictionary are to be considered. Beyond these natural interpretations, the choice of a fixed set of vertices allow to focus learning on a specific set of words or jargon. Therefore, unlike a complete learning strategy [2], a grounding kernel [14] or psycholinguistic strategies [3] [5], which aims at learning an entire dictionary, our subgraphs lead to targeted learning strategy. Indeed, through the constraint of a set of fixed words that we add, we can restrict and direct the learning strategy towards a specialized jargon. This will therefore reduce the learning of superfluous words. Which leads to effective and low cost learning strategies.

This paper is divided as follows. In Section II, we introduce definitions and notation about directed graphs. In Section III, we prove that the problem of finding subgraphs under the constraints described above is NP-hard. In Section IV, we present two metaheuristics to solve this problem: a *greedy heuristic* and a local search enriched with *tabu restrictions*. In Section V, we describe an experiment that supports the conjecture on learning strategies formulated above that gives affirmative and encouraging results. In Subsection V-B, we implement and test these algorithms on sets of digraphs. The performance of these algorithms is discussed and analyzed. Concluding remarks are presented in Section VI.

II. PRELIMINARIES

We recall some definitions from graph theory. See [15] for more details.

A *digraph* is an ordered pair $D = (V, A)$, where V is its set of *vertices* and $A \subseteq V \times V$ is its set of *arcs*. Given a vertex $u \in V$, its set of *predecessors* is defined by $N^-(u) = \{v \in V \mid (v, u) \in A\}$ and its set of *successors* is $N^+(u) = \{v \in V \mid (u, v) \in A\}$. The *indegree* of

 ISBN: 978-1-61208-780-1

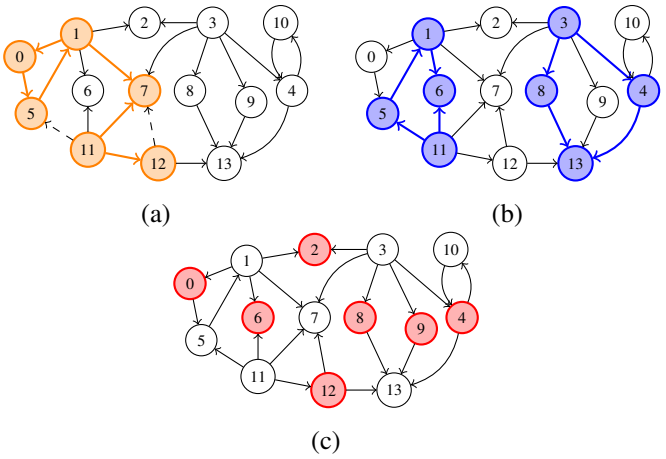

Figure 1. (a) In orange, a subgraph with a circuit $P = (0, 5, 1, 0)$. This subgraph is not induced because it does not contain the dashed arcs $(11, 5)$ and $(12, 7)$, despite that vertices $11, 5, 12, 7$ are contained in the subgraph. (b) In blue, a disconnected induced acyclic subgraph with an indirected path $P = (4, 3, 8, 13)$. Vertices 3 and 11 are sources, 6 and 13 are sinks. (c) In red, an independent set of size 7.

u is defined by $\deg^-_D(u) = |N^-(u)|$ and its *outegree* by $\deg^+_D(u) = |N^+(u)\}|$, where $|.|$ denotes set cardinality. An *(undirected) path* p from a vertex u_0 to a vertex u_k in D is a sequence of vertices $p = (u_0, u_1, \ldots, u_k)$ such that, for $0 \leq i \leq k-1$, $(u_i, u_{i+1}) \in A$ or $(u_{i+1}, u_i) \in A$. The path p is *directed* if $(u_i, u_{i+1}) \in A$, for every $0 \leq i \leq k-1$. We say that a directed path $p = (u_0, u_1, \ldots, u_k)$ is a *circuit* or a *cycle*, if $u_0 = u_k$. A digraph D is called *acyclic* if it does not contain any circuit, and it is called *connected* (or *weakly connected*) if there exists an undirected path between every pair of its vertices. A digraph $I = (V_I, A_I)$ is a *subgraph* of D, if $V_I \subseteq V$ and $A_I \subseteq (V_I \times V_I) \cap A$. For $U \subseteq V$, the *subgraph of D induced by U* is the subgraph $D[U] = (U, A_U)$, where $A_U = (U \times U) \cap A$. In words, for all vertices u and v in U, if we have an arc $(u, v) \in A$ then we must have $(u, v) \in A_U$. A vertex u is called a *source* of a subgraph I if $\deg^-_I(u) = 0$. Similarly, if $\deg^+_I(u) = 0$ then u is called a *sink* of I. We denote by $s(I)$ the number of sources in I and $t(I)$ its number of sinks. Finally, a set of vertices S of a digraph $D = (V, A)$ is called an *independent set*, if the induced subgraph $D[S] = (S, A_S)$ has $A_S = \emptyset$. In other words, for each $u \in S$, we have $\deg^+_{D[S]}(u) = \deg^-_{D[S]}(u) = 0$. Examples of induced and non-induced subgraphs appear in Figure 1, as well as an example of an independent set.

For the purposes of the next section, we state the following well-known NP-complete independent set problem,

Problem 1 ([16]): Given a specific digraph $D = (V, A)$ and a specific positive integer $i \leq |V|$, is there an independent set $S \subset V$ such that $|S| = i$?

III. Decision problem and NP-completeness

As mentioned before, we are interested in induced and acyclic subgraphs of digraphs. In order to identify these subgraphs, we define the following optimization criteria.

Definition 1: Given a digraph $D = (V, A)$ and a subset $M \subseteq V$, let $\mathcal{P}(V)$ be the set of all subsets of V and $\mathcal{I}_D(M, i)$ the family of all induced and acyclic subgraphs of D of size i containing M. We define the function Δ_D, with domain $\{0, 1, 2, \ldots, |D|\}$, by

$$\Delta_D(M, i) = \max\{t(I) - s(I) \mid I \in \mathcal{I}_D(M, i)\} \qquad (1)$$

with the convention $\max \emptyset = -\infty$. Given $I \in \mathcal{I}_D(M, i)$ we say that I has *optimized endpoints*, if $\Delta_D(M, i) = t(I) - s(I)$.

We are interested in induced and acyclic subgraphs with optimized endpoints of D.

Remark 1: The case $M = \emptyset$ is a specialization of the problem introduced in [13].

Now, we consider the following decision problem.

Problem 2: Given a digraph $D = (V, A)$, a set of vertices $M \subset V$ and two integers i and δ, does there exist an induced and acyclic subgraph of D of size i containing M, such that $t(I) - s(I) = \delta$?

We naturally associate with Problem 2 the following optimization problem.

Problem 3: Given a digraph $D = (V, A)$ and a set $M \subset V$ of vertices, what is the maximal value $\Delta_D(M, i)$ that can be realized by an induced and acyclic subgraph I of D of size i and containing M, for $i \in \{|M|, |M| + 1, \ldots, |D|\}$?

Before we go further, it is worth discussing the dual problem of maximizing the difference $s(I) - t(I)$. Consider the following optimization function:

$$\Lambda_D(M, i) = \max\{s(I) - t(I) \mid I \in \mathcal{I}_D(M, i)\}$$

It turns out that this optimization criterion is equivalent to that of Definition 1 under a slightly modified digraph. Proposition 1 formalizes the relationship between the functions $\Lambda_D(M, i)$ and $\Delta_D(M, i)$. We omit the proof because it is quite simple.

Proposition 1: Given a digraph $D = (V, A)$, for any subset $M \subset V$ and $i \in \{0, 1, \ldots, |D|\}$, we have

$$\Lambda_D(M, i) = \Delta_{D'}(M, i)$$

where $D' = (V, A')$ and $A' = \{(v, u) \mid (u, v) \in A\}$.

Now, we illustrate Problem 2 with an example. Let D be the digraph of size 14 illustrated in Figure 2 and let $M = \{1, 3, 11\}$, a subset of vertices of D depicted in gray. Positive instances of Problem 2 appear in (a) and (b) for respectively $i = 10$, $\delta = 4$ and $i = 11$, $\delta = 4$. It is not hard to prove that $\Delta_D(M, 10) = \Delta_D(M, 11) = 4$ so that $(10, \delta_1)$ and $(11, \delta_2)$ are negative instances of Problem 2 for $\delta_1, \delta_2 > 4$. Because of the two circuits $P_1 = (0, 5, 1, 0)$ and $P_2 = (4, 10, 4)$, the maximum size of an induced acyclic subgraph is $i = 12$. Therefore, there are only two solutions. Figure 2 (c) shows the first positive instances of Problem 2 with $i = 12$ and $\delta_1 = 2$. The second is obtained by replacing vertex 10 by vertex 4. Observe that $\Delta_D(\emptyset, 10) = 5 > \Delta_D(M, 10)$. This situation is depicted by the subgraph in blue in Figure 2 (d). It is easy to see that $\Delta_D(M, i) \leq \Delta_D(\emptyset, i)$ for any M and i. Exhaustive inspection shows that the function Δ_D of the graph in Figure 2 is given by Table I. Obviously, for $i < |M|$ we have $\mathcal{I}_D(M, i) = \emptyset$, thus, $\Delta_D(M, i) = -\infty$.

A. NP-completeness

Now, we state and prove the following theorem.

Theorem 1: Problem 2 is NP-complete.

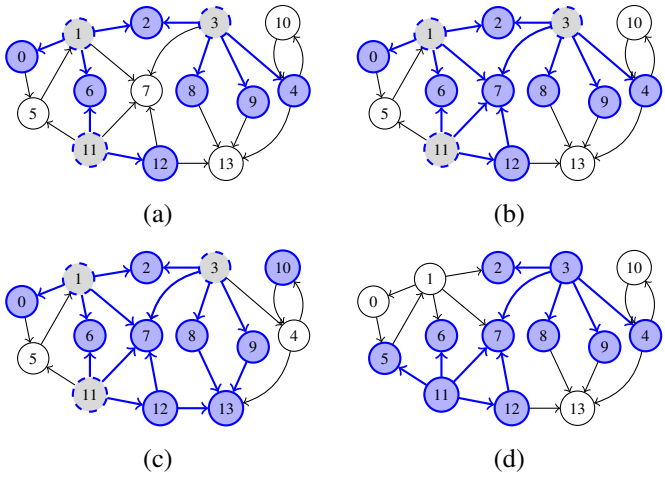

Figure 2. Induced acyclic subgraphs answering Problems 3 (a-c) with the set $M = \{1, 3, 11\}$ and (d) with the set $M = \emptyset$.

TABLE I. THE FUNCTION $\Delta_D(M, i)$ FOR THE DIGRAPH IN FIGURE 2 WITH $M = \{1, 3, 11\}$.

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$\Delta_D(M, i)$	$-\infty$	$-\infty$	$-\infty$	0	0	1	2	2	3	3	4	4	2	$-\infty$	$-\infty$

Proof: It is clear that Problem 2 is in the class NP. To show that it is NP-complete, we reduce Problem 1 to it.

We represent an instance of Problem 1 by the tuple (D, i) where $D = (V, A)$ is a digraph and $1 \leq i < |V|$ is an integer. An instance of Problem 2 is represented by the tuple (D, M, i, δ) where $D = (V, A)$ is a digraph, M is a set of m fixed vertices of D, i is the number of vertices parameter and δ is the sinks$-$sources difference parameter.

Consider the map f that associates with an instance (D, i) of Problem 1, the instance $(D', M, i+m, i-m)$ of Problem 2, where the digraph $D' = (V', A')$ is defined by

$$V' = V \cup M \quad \text{and} \quad A' = A \cup (M \times V)$$

Clearly, the map f is computable in polynomial time. Figure 3(a) and Figure 3(b) show an example of construction, using the map f, of the digraph $D' = (V', A')$ for a digraph $D = (V, A)$ and a set $M = \{m_1, m_2\}$.

For a positive instance (D, i) of Problem 1, there is a set $S \subset V$ with $|S| = i$ such that $D[S]$ contains no arcs. We consider the subgraph $I = D'[S \cup M] = (V_I, A_I)$ of D' induced by vertices of $S \cup M$. Obviously, I contains M and all vertices of M are sources, which makes all vertices of S sinks in I. Because there is no arc between vertices in M and also no arcs between vertices of S in I, therefore I is acyclic. Finally, $s(I) = m$ and $t(I) = i$, which makes $(D', M, i+m, i-m)$ a positive instance of Problem 2. Now, let $(D', M, i+m, i-m)$ be a positive instance of Problem 2. There is an induced and acyclic subgraph $I = (V_I, A_I)$ of D', such that $M \subset V_I$ with $|I| = i+m$ and $t(I) - s(I) = i-m$. It is easy to see that $s(I) > 0$ and $t(I) > 0$. Since I contains M, it is clear that only vertices from M can be sources, so, $s(I) = m$. Otherwise, there exists $v \in V_I - M$ such that $\deg_I^-(v) = 0$, which is impossible. Indeed, by construction, we have arcs from M towards v in D' and since I is induced and contains both M and v, these arcs are in I and so $\deg_I^-(v) > 0$. Therefore, the remaining i vertices are necessarily sinks in I and are contained in V. Indeed, since $t(I) - s(I) = i - m$ and $s(I) = m$, then $t(I) = i$. So, we construct an independent set S of size i by considering only sinks of I and so $S = V_I - M$. Indeed, we are sure that there is no arc between vertices of S. Otherwise, if there exist $u, v \in S$ with $(u, v) \in A'$, that is mean that u is not a sink ($\deg_I^+(u) > 0$ and $\deg_I^-(u) > 0$ because $u \notin M$). Which leads to a contradiction $t(I) - s(I) < i - m$. Finally, S is an independent set of size i in $D = (V, A)$, which makes (D, i) a positive instance of Problem 1. See Figure 3(c) and Figure 3(d) for the construction of positive instances between Problem 1 and Problem 2.

Therefore, Problem 1 $\leq$ Problem 2 and Problem 2 is NP-complete. ■

Remark 2: In virtue of Proposition 1 and the fact that the construction of D' from D takes a polynomial time (exactly, $\Theta(|A|)$), if we replace the optimization criterion by the difference between the number of sources and the number of sinks, the problem remains NP-complete.

IV. ALGORITHMS

Now that we have proved that Problem 2 is NP-complete, we conclude that Problem 3 is NP-hard and a polynomial algorithm to solve it is unlikely to exist. This motivates the use of approximate approaches among other resolution techniques. Thus we consider a *greedy algorithm* and a *tabu search* [17], two metaheuristics, to solve Problem 3.

In the following, let $D = (V, A)$ be a digraph, with $|V| = n$, $|A| = m$, and $M \subset V$ a subset of vertices of V. Let $I = (V_I, A_I)$ be an induced acyclic subgraph of D with $M \subseteq V_I$ and let $i = |V_I|$. Before presenting the algorithms, we assume that the following functions are implemented:

- DELTA(I) returns $t(I) - s(I)$, whose complexity is $\mathcal{O}(i + |A_I|)$.
- INDUCEDSUBGRAPH(D, E) returns the subgraph of D induced by the vertices of set E and has complexity $\mathcal{O}(n + m)$.
- NEIGHBORSVERTICES(D, I) returns all vertices u such that $u \in D - I$ and there exists $v \in V_I$ such that $(u, v) \in A$ or $(v, u) \in A$. This can be done in $\mathcal{O}(n + m)$.
- NEIGHBORHOOD(D, I, M, u) returns a set of induced and acyclic subgraphs by replacing, in turn, each vertex of $V_I - M$ by u. The complexity in that case is $\mathcal{O}((n + m) \times (i - |M|) \times (i + |A_I|))$.

The greedy strategy adopted is to add the most *interesting* vertices. Starting with the subgraph induced by M, $I = D[M]$ until we reach the desired size $|I| = i$. The function RANKVERTICES(D, I) assigns a score to each vertex $u \in D - I$ according to the variation they bring to the function DELTA(I) if we add them to I. Thus, the greater the value of DELTA(I) a vertex brings, the greater its interest. If two vertices bring the same variation, we choose the one with the greatest outdegree, which provides greater potential to generate sinks. Due to lack of space, we don't present the pseudo-code of algorithm, but we provide a full implementation of the greedy approach in [18].

For the tabu search algorithm, the main idea is to browse *neighborhoods* of an induced and acyclic subgraph I containing M, to increase the value DELTA(I). A *neighbor* of I is

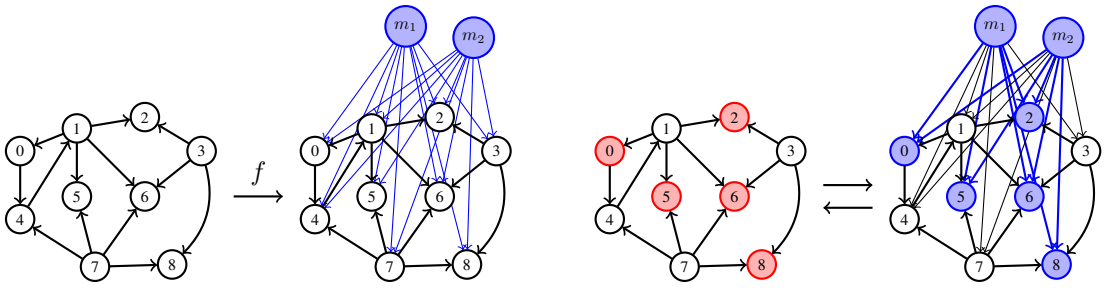

Figure 3. (a) A digraph $D = (V, A)$. (b) Construction of digraph $D' = (V', A')$ through the map f. (c) In red, a set S which makes $(D, 5)$ a positive instance of Problem 1. (d) In blue, an induced and acyclic subgraph which makes $(D', M, 7, 3)$ a positive instance of Problem 2. To build the digraph $D' = (V', A')$, we add vertices of $M = \{m_1, m_2\}$ to V to have V'. To build A', we add all possible arcs (u, v) to A, where $u \in M$ and $v \in V$. It's easy to see how we can build a positive instance of Problem 2 from a positive instance of Problem 1 and vice-versa.

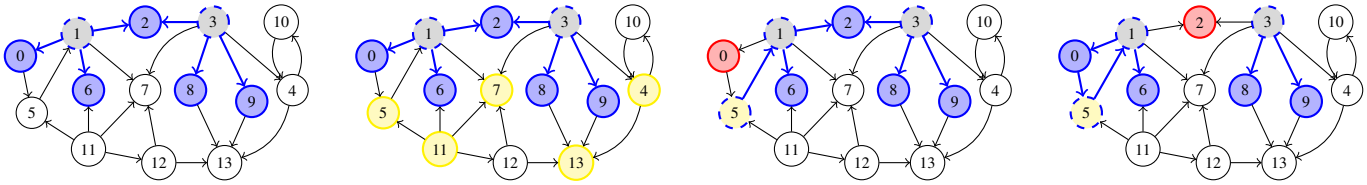

Figure 4. Let $D = (V, A)$ be the digraph represented above and $M = \{1, 3\}$ a subset of V. (a) In blue, an induced and acyclic subgraph $I = (V_I, A_I)$ of D containing M. Note that DELTA$(I) = 3$. (b) In yellow, vertices returned by NEIGHBORSVERTICES(D, I). The call to NEIGHBORHOOD$(D, I, M, 5)$ for example, will return I' the blue subgraph in (c) by switching vertices 5 and 0. Switching 5 and 2 results in the induced subgraph (d) which is rejected because it contains the circuit $P = (0, 5, 1, 0)$ so that NEIGHBORHOOD$(D, I, M, 5)$ will not consider this case. The remaining subgraphs built by replacement of vertices 6, 8 and 9 by 5 are all rejected because of the circuit $P = (0, 5, 1, 0)$.

an induced and acyclic subgraph I', such that $M \subset V_{I'}$, with $|I| = |I'| = i$ and a symmetric difference between V_I and $V_{I'}$ of exactly 2 vertices.

To generate some of these neighbors, we start by finding the neighboring vertices of I. The function NEIGHBORSVERTICES(D, I) finds these vertices. See Figure 4(b) for an example. For each neighbor u, we call NEIGHBORHOOD(D, I, M, u), which returns a set of induced and acyclic subgraphs by replacing each vertex $v \in V_I - M$ by u. In other words, it returns the subgraph INDUCEDSUBGRAPH$(D, (V_I - \{v\}) \cup \{u\})$ when it is acyclic.

See Algorithm 1 for the pseudo-code of this approach. Note that vertex returned by NEIGHBORSVERTICES(D, I) and the one inverted by the function NEIGHBORHOOD(D, I, M, u), are considered as our *tabu restriction*. We limit the number of prohibited vertices to two, thus this restriction does not penalize in the search for a better solution. It's implemented in Algorithm 1 by the queue T and lines 7 and 11. Note that, lines 12-15 are for the *aspiration criteria*. See Figure 4 for an illustrated example with a single call to NEIGHBORHOOD(D, I, M, u).

The complexity of Algorithm 1 is basically the complexity of the three functions DELTA(I), NEIGHBORSVERTICES(D, I) and NEIGHBORHOOD(D, I, M, u), which is $\mathcal{O}(n \times (i - |M|) \times (n + m)^2 \times (i + |A_I|)^2)$. The parameter *max_iterations* is the maximum number of loop laps allowed, and is an arbitrary multiple of n.

Algorithm 1 Tabu Search

```
1:  function TABUSEARCH(D, I : graph, M : set) : integer
2:      B, I' ← I let T be a queue with capacity 2
3:      repeat
4:          for u ∈ NEIGHBORSVERTICES(D, V_I') − T do
5:              for I'' ∈ NEIGHBORHOOD(D, I', M, u) do
6:                  if DELTA(I') < DELTA(I'') then
7:                      I' ← I'' add the removed vertex to T
8:                      if DELTA(B) < DELTA(I'') then
9:                          B ← I''
10:                     Exit from loop in line 5
11:             Add u to T
12:         v ← T.HEAD()
13:         for I'' ∈ NEIGHBORHOOD(D, I', M, v) do
14:             if DELTA(B) < DELTA(I'') then
15:                 B ← I''
16:     until B ≠ I' and max_iterations is reached
17:     return DELTA(B)
```

V. APPLICATIONS

We now turn back to our initial motivation in computational linguistics, by showing that the resulting structure drives a learning strategy in comparison with other psycholinguistic strategies. Then we evaluate the quality of the metaheuristics in terms of error, compared to an exact solution.

Note that we use algorithms with $M = \emptyset$ because the primary goal is to measure the correlation between cost of a strategy and our optimization criterion Δ_D. Also, the psycholinguistic strategies below are designed for learning an entire language, which differs from a targeted learning strategy when $M \neq \emptyset$.

 ISBN: 978-1-61208-780-1

A. Computational linguistics

To reinforce the hypothesis stated in Section I, we conducted an experiment to compare costs of learning strategies. We consider three digital dictionaries from the Wordsmyth linguistical educational project [19]. The Wordsmyth Learner's Dictionary-Thesaurus (WLDT), the Wordsmyth Illustrated Learner's Dictionary (WILD) and the Wordsmyth Children's Dictionary-Thesaurus (WCDT). And we use some computational linguistics definitions taken from [2] [20]. We construct a

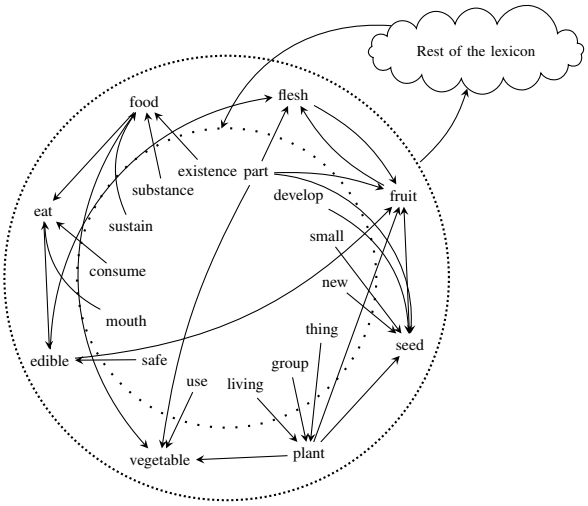

Figure 5. A partial subgraph of a digraph representing a dictionary [2].

digraph from a dictionary by considering each word as a vertex and we have an arc from word w_1 to word w_2 if and only if w_1 appears in the definition of w_2. Figure 5 depicts a subgraph of a digraph representing a dictionary. On a given dictionary D, we define a *learning strategy* of D as any *ordered sequence* S of words from D. We consider two ways of learning a new word,

- *Learning by definition.* We suppose that, if we already know the meaning of all the words occurring in the definition of word w, then we can learn the meaning of w simply by reading its definition. We consider the cost of such learning process to be 0.
- *Direct learning.* On the other hand, if we do not know the meaning of some words in the definition, we have the possibility to learn it *directly* with a cost of 1. For example, by seeing, hearing, smelling, tasting, touching, or interacting in any other way with the object referenced by the word.

In other words, the effort of learning a word directly, is more costly than the one of learning a word by definition.

Finally, we make the assumption that learning a dictionary efficiently amounts to learn its complete set of words at a minimal cost. This implies that a learning strategy is efficient if its associated cost is minimal, which is realized exactly when the number of words learned directly is minimal. Which is represented by the following two parameters in the Table II,

- *cost* : the total number of words learned directly.
- *effic.* : the efficiency, which is the ratio of the total number of words learned over the number of words learned through direct learning.

TABLE II. THE LEARNING COST AND EFFICIENCY OF STRATEGIES.

Dictionary		Subgraph strategy	Childes AOA	Freq. NGSL	Brysbaert AOA	Brysbaert Concret.	Brysbaert Freq.
WILD	size	50	1981	1169	1369	2417	1188
	cost	**3013**	3174	3079	3079	3207	3059
	effic.	**1.40**	1.33	1.37	1.37	1.32	1.38
WLDT	size	200	1580	697	1277	2366	957
	cost	**917**	1803	1088	1530	2485	1296
	effic.	**6.58**	3.34	5.54	3.94	2.42	4.65
WCDT	size	300	3316	1122	2879	5974	1995
	cost	**2431**	4366	2777	4016	6616	3315
	effic.	**12.23**	6.81	10.70	7.40	4.49	8.96

Under those definitions and assumptions, we implemented the algorithm *cost of a complete learning strategy* from [2] to compare strategies. We represented WILD dictionary by a graph with 4244 vertices and 59478 arcs, WLDT dictionary with 6036 vertices and 29735 arcs and WCDT dictionary with 20128 vertices and 107079 arcs. We focus our comparison on psycholinguistic strategies, from a database made available by Brysbaert [5] and the Child Language Data Exchange System (CHILDES) project [3]. Strategies are constructed [3]–[6] according to the following psycholinguistic criteria:

- Brysbaert-AOA: a set of words ordered according to the age of their acquisition.
- Brysbaert-Concreteness: a set of words ordered following their concreteness, from the most concrete to the most abstract.
- Brysbaert-Frequency: a set of words ordered by their rate of occurrence in a given corpus.
- Childes-AOA: a set of words from Child Language Data Exchange System (CHILDES) project are ordred following the age of their acquisition.

A last strategy that we considered, called NGSL/Frequency, is obtained from the NGSL [7],

- Frequency-NGSL: a set of words that are designed and ordered to help students learning English as a second language.

We compute some induced acyclic subgraphs with optimized endpoints on the three digital dictionaries. We consider vertices from the resulting subgraphs as a strategy. We order those vertices according to their outgoing degree, from the highest to the lowest to maximize the potential number of words learned by definition.

Since the size of the dictionary digraphs is important, it takes more computing time to obtain exact solutions. Therefore, to get as close as possible to the exact solution, we use algorithms above with the greedy solution as an initial subgraph input to tabu search.

Table II gives the results of cost computation. It is clear that the subgraphs strategies are better (values in bold) in terms of low cost and high efficiency. Moreover, the initial number of words (strategy size) required to learned in subgraphs based strategies are significantly smaller than the words in psycholinguistic strategies. Note that the acyclic constraint ensure that there is no cyclic words definition and so, no word appears in the definitions of the words that define it. The low cost and high efficiency support our assumption that our induced acyclic subgraphs with optimized endpoints are minimal directed learning strategies in a lexicon.

 ISBN: 978-1-61208-780-1

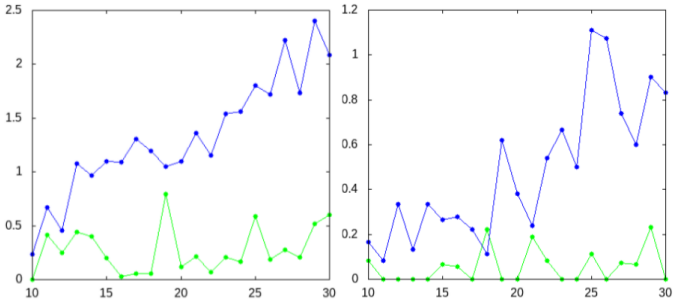

Figure 6. Average error for the greedy algorithm (in green) and the tabu search (in blue) for $M1$ (left) and $M2$ (right). X-axis is for digraphs sizes and Y-axis is for average error. The error is computed by subtracting the value of $\Delta_D(M,.)$ given by greedy or tabu algorithm from the exact value given by the naive algorithm. See [18] for more details.

B. Metaheuristics evaluation

Obviously, since Problem 2 is NP-complete, it is probable that all exact solutions to the associated optimization problem take at least an exponential time to be obtained. That being said, it is necessary to have an exact solution in order to calculate the error of metaheuristics and it is always interesting to observe the execution times of an exact solution. Therefore, we have implemented an exact algorithm based on a naive approach. Namely, we enumerate all possible subsets of V, then we construct the corresponding induced subgraphs and finally we verify the acyclicity. For each subgraph of size $|M| \leq i \leq |V|$, we choose the ones with the largest DELTA().

For sizes $10 \leq n \leq 30$ we generated 3 weakly connected digraphs. For each graph size n we generated two random sets $M1$ and $M2$ of sizes $n/3$ and $2n/3$ respectively.

Unsurprisingly, the heuristics are faster than the naïve algorithm, especially for small sets M. When the size of M is close to the size of D, the difference between the $\Delta_D(.,.)$ values given by the heuristics and the exact values is small. We explain this observation by the fact that a large part of the vertices is fixed by M, restricting the remaining choices and the error made by the heuristics.

Figure 6 (a) shows the average error for two random sets $M1$ and $M2$, with sizes $1/3 \times |D|$ and $2/3 \times |D|$ respectively.

For more details and a full Python implementation of the previous two algorithms, with benchmarks, see the GitLab repository [18]. Due to lack of space, these implementations are not included here.

VI. CONCLUSION

The problem studied in this paper is new. It is therefore difficult to compare our results to similar work in the literature. The results of linguistics experiment in Subsection V-A, show that, even if solutions given by tabu algorithm are approximate, they are best and lowest cost learning strategies. This result encourages us to further investigate linguistic applications. Mainly to push the experimentation on other digital dictionaries to establish the effectiveness of induced acyclic subgraphs with optimized endpoints as a learning strategy. Thereafter, extract targeted strategies with subsets M of vocabularies specialized in different fields. From graph theory point of view, we will also be interested in specializing to digraphs that satisfy one of our constraints: acyclicity. We believe that the solution to Problem 2 then becomes polynomial. It is also natural to investigate other optimization functions based on the number of sources and sinks, such as the ratio $t(I)/s(I)$ which provides additional information to those obtained from the difference $t() - s()$. We strongly suspect, in that case, that the problem remains NP-complete.

REFERENCES

[1] S. Harnad, "The symbol grounding problem," Physica D: Nonlinear Phenomena, vol. 42, no. 1, 1990, pp. 335 – 346.

[2] J.-M. Poulin, A. Blondin Massé, and A. Fonseca, "Strategies for learning lexemes efficiently: A graph-based approach," in The Tenth International Conference on Advanced Cognitive Technologies and Applications COGNITIVE 2018. Barcelona, Spain, 02 2018, pp. 18–22.

[3] B. MacWhinney, "The childes project: Tools for analyzing talk: Transcription format and programs," Child Language Teaching and Therapy, vol. Vol. 1, 3rd ed, no. 4, 2000, p. 366.

[4] M. Brysbaert and B. New, "Moving beyond kucera and francis: A critical evaluation of current word frequency norms and the introduction of a new and improved word frequency measure for american english," Behavior research methods, vol. 41, no. 4, 11 2009, pp. 977–990.

[5] V. Kuperman, H. Stadthagen-Gonzalez, and M. Brysbaert, "Age-of-acquisition ratings for 30,000 english words," Behavior Research Methods, vol. 44, no. 4, 12 2012, pp. 978–990. [Online]. Available: https://doi.org/10.3758/s13428-012-0210-4

[6] M. Brysbaert, A. Warriner, and V. Kuperman, "Concreteness ratings for 40 thousand generally known english word lemmas," Behavior research methods, vol. 46, no. 3, 10 2014, pp. 904–911.

[7] C. M. Browne, "A new general service list: The better mousetrap we've been looking for?" Vocabulary Learning and Instruction, vol. 3, no. 2, 2014, pp. 1–10.

[8] A. Boukerche, X. Cheng, and J. Linus, "A performance evaluation of a novel energy-aware data-centric routing algorithm in wireless sensor networks," Wireless Networks, vol. 11, no. 5, 2005, pp. 619–635.

[9] P. Erdős, M. Saks, and V. T. Sós, "Maximum induced trees in graphs," Journal of Combinatorial Theory, Series B, vol. 41, no. 1, 1986, pp. 61–79.

[10] L. Székely and H. Wang, "On subtrees of trees," Advances in Applied Mathematics - ADVAN APPL MATH, vol. 34, 01 2005, pp. 138–155.

[11] A. Blondin Massé, J. de Carufel, and A. Goupil, "Saturated fully leafed tree-like polyforms and polycubes," Journal of Discrete Algorithms, vol. 52-53, 2018, pp. 38 – 54, combinatorial Algorithms – Special Issue Devoted to Life and Work of Mirka Miller.

[12] M. Aronoff and J. Rees-Miller, The Handbook of Linguistics. John Wiley & Sons, 2002.

[13] M. Abdenbi, A. B. Massé, and A. Goupil, "Induced DAGs with maximal sinks-sources difference," Bordeaux Graph Workshop, 2019, accepted.

[14] G. Chicoisne, A. Blondin-Masse, O. Picard, and S. Harnad, "Grounding abstract word definitions in prior concrete experience," in Sixth Annual Conference on the Mental Lexicon, 2008, p. 1498, university of Alberta.

[15] R. Diestel, Graph theory, 4th ed., ser. Graduate Texts in Mathematics. Springer, Heidelberg, 2010, vol. 173.

[16] R. M. Karp, Reducibility among Combinatorial Problems. Boston, MA: Springer US, 1972, pp. 85–103.

[17] T. F. Gonzalez, Handbook of Approximation Algorithms and Metaheuristics (Chapman & Hall/Crc Computer & Information Science Series). Chapman & Hall/CRC, 2007.

[18] M. Abdenbi, Algorithms implementation, (accessed 2020), https://gitlab.com/moussa.abdenbi/implementation-cognitive.

[19] Wordsmyth Illustrated Learner's Dictionary and Wordsmyth Learner's Dictionary-Thesaurus, (accessed 2018), https://www.wordsmyth.net.

[20] A. B. Massé, G. Chicoisne, Y. Gargouri, S. Harnad, O. Picard, and O. Marcotte, "How is meaning grounded in dictionary definitions?" in Proceedings of the 3rd Textgraphs Workshop on Graph-Based Algorithms for Natural Language Processing, ser. TextGraphs-3. USA: Association for Computational Linguistics, 2008, p. 17–24.

 ISBN: 978-1-61208-780-1