Approximating optimal solution structure with edit distance and its applications

Antonina Kolokolova and Renesa Nizamee

Memorial University of Newfoundland
{kol,mrn271}@mun.ca

Abstract. An alternative notion of approximation arising in cognitive psychology, bioinformatics and linguistics is that of computing a solution which is structurally close to an optimal one. That is, an approximate solution is considered good if its distance from an optimal solution is small, for a distance measure such as Hamming distance or edit distance. There has been a modicum of work on approximating solution structure according to an arbitrary solution distance function [HMvRW07,vRW12]; several papers addressed the complexity of approximating a witness to an NP problem to within bounded Hamming distance [GHLP99,KS99,FLN00,SY13]. Most of these results are strong lower bounds.
In this paper, we extend Hamming distance inapproximability results of Sheldon and Young [SY13] to a setting where a solution is considered good if it is within a given edit distance bound from some optimal solution. We show several inapproximability results for this measure, in particular extending [SY13] $n/2 - n^{\epsilon}$ inapproximability $\forall \epsilon > 0$ with respect to Hamming distance to edit distance for SAT and VertexCover problems.
Finally, we show that these lower bounds for both edit distance and Hamming distance apply to two practical problems: Longest Common Subsequence and Weighted Sentence Alignment.

Keywords: Complexity, approximation, lower bounds, edit distance, Hamming distance, phrase alignment, longest common subsequence

1 Introduction

Approximation algorithms for NP-hard problems have been extensively studied for several decades. However, in essentially all this work the notion of approximation was defined as finding a solution with the *value* not too far from the value of an optimal solution. For example, saying that Max3SAT can be approximated within 7/8th of the optimal amounts to stating that there is an efficient algorithm which on an input ϕ finds an assignment β which satisfies at least 7/8th of the clauses satisfied by an optimal assignment α. Here, α and β are compared in terms of the number of clauses of ϕ satisfied by α and β, respectively. However, in some applications it can be more valuable to find an potential solution that resembles an optimal solution structurally. For example, β can be considered a good approximate solution if it agrees with some optimal α on a large fraction of variable assignments, even if it satisfies a small number of clauses.

Hamilton, Müller, van Rooij and Wareham [HMvRW07], who, to our knowledge, first formulated the notion of approximating solution structure, were motivated by cognitive psychology applications such as the Coherence problem. There, given a belief network consisting of a set of propositions (beliefs), a set of constraints on pairs of beliefs, and a weight function on the constraints, the goal is to find a truth assignment to propositions so that the sum of weights of violated constraints is minimized. From the point of view of cognitive psychology, according to them, it is more valuable to compute an assignment that shares many of the beliefs with the optimal assignment, than to compute a disparate assignment with a relatively high value [vRW12]. More precisely, let potential

solutions to the Coherence problem be encoded as a binary strings representing a truth assignment, as for the SAT problem. Then, a solution which coincides with the optimal solution on a significant fraction of beliefs is close to an optimal solution in Hamming distance. A relevant notion of approximation for this problem becomes structure approximation with respect to Hamming distance, that is, the problem of computing, in polynomial time, a string encoding a truth assignment which is a close as possible in Hamming distance to a string encoding an optimal assignment.

Hamming distance indeed gives rise to the most natural notion of "closeness in structure" of two solutions, and as such it has drawn the most attention [HMvRW07,KS99,FLN00,vRW12,SY13]. In this paper we will focus on another measure of distance that naturally arises in string problems: the edit distance. That is, two strings are considered within edit distance k if at most k insertion, deletion and replacement operations are sufficient to convert one into the other.

One of the tasks in some bioinformatics applications is computing a longest common subsequence of a set of strings encoding DNA sequences or protein signatures. For a constant number of strings this problem can be solved in polynomial time, however, it is NP-hard when the number of strings is not restricted. Thus, it would be useful to design an algorithm computing a string which coincides with the longest possible subsequence on a large number of positions. As before, this can be defined as computing a string which is within a small Hamming distance of an optimal solution. A different view of a "close enough" solution is to consider the output string as a "corrupted" encoding of a solution, with a number of "mistakes" which can include insertions, deletions and replacements of symbols. In this scenario, the edit distance is the measure of how many such mistakes a string contains in comparison to some optimal solution.

Yet another application of such edit distance approximation comes from the field of machine translation. Consider the task of computing a phrase alignment between two pieces of text, where phrases are not given in advance and any phrase can be paired with any phrase in another sentence, irrespective of the order (but preserving the property that each phrase is paired with exactly one translation phrase). Each potential pairing is given a weight, and the goal is to find an alignment consisting of a set of pairing covering all phrases, with the product of the weights of its constituent pairs maximized. As before, the Hamming distance already provides a good measure of closeness of two alignments. However, if the sentences are repetitive ("he said that she said that..."), and a phrase from the beginning of the first sentence is mapped to a phrase at the end of the second, it will shift the pairings of many phrases. However, intuitively, this is a fairly small mistake. Using the edit distance as a measure of closeness reflects this intuition.

However, attractive as the notion of approximating solution structure to within small Hamming or edit distances is, we show that computing such approximation can be as hard as solving the original problem. In particular, extending the techniques of Sheldon and Young [SY13], we show that for SAT and VertexCover computing a solution within edit distance $n/2-n^{\epsilon}$ from the optimal, for any $\epsilon > 0$, is NP-hard. We then apply these results to show that approximating solutions to Longest Common Subsequence and a Phrase Alignment problems within $n/2 - n^{\epsilon}$ with respect to both Hamming distance and edit distance, $\forall \epsilon > 0$, is NP-hard as well. Although these results are pessimistic, they can be used to pinpoint sources of intractability in these problems, suggesting a future direction for parameterized complexity analysis.

2 Previous work

The notion of approximating solution structure was introduced in [HMvRW07], with the motivation from the cognitive psychology. In their framework, an optimization problem description includes

a distance function $d(y, z)$, and an feasible solution y is considered good if $d(y, z)$ is small for some optimal solution z. Many results in [HMvRW07] are stated and proven for arbitrary metric integer-valued distance functions and optimization problems satisfying certain properties. In particular, they show that there are no NP-hard problems with a structure approximation analogue of FPTAS for an arbitrary function. Additionally, they state and prove a number of results for the Hamming distance function and problems such as MaxCut, SAT and MaxClique. In a follow-up paper [vRW12], this approach is applied to harmony maximization on Hopfield networks.

A related line of work considers decision problems, analyzing the complexity of computing a string within a given Hamming distance to a solution, that is, to a witness to a "yes"-instance of an NP problem. The reconstruction of a partially specified witness, considered in the 1999 paper by Gal, Halevi, Lipton and Petrank [GHLP99], is probably the first result along these lines. There, they show that it is possible to reconstruct a satisfying assignment to a formula from $N^{1/2+\epsilon}$ bits of a satisfying assignment of a related larger formula. Their main proof technique relies on erasure codes, and in addition to SAT they consider Graph Isomorphism, Shortest Lattice Vector and Clique/Vertex Cover/Independent set. In 1999, Kumar and Sivakumar [KS99] showed that for any NP problem there is a verifier with respect to which all solutions are Hamming-far from each other: make new witnesses to be encodings of witnesses to the original problem by some error-correcting code. The new verifier performs the decoding and checks the result using the original verifier. The list-decoding allows one to find a correct codeword encoding a witness from a string which is within $n/2 + n^{4/5+\gamma}$ Hamming distance from it. Following this, Feige, Langberg and Nissim [FLN00] show that some natural witnesses (e.g., binary strings directly encoding satisfying assignments for variants of SAT, encoding sequences of vertices for Clique/Vertex Cover, etc) are hard to approximate to within Hamming distance $n/2 - n^{\epsilon}$ for some ϵ dependent on the underlying error-correcting code. Guruswami and Rudra [GR08] improve this ϵ to $2/3 + \gamma$, but on the negative side argue that methods based on error-correcting codes can only give bounds up to $n/2 - O(\sqrt{n \log n})$.

The recent paper of Sheldon and Young [SY13] settles much of the Hamming distance approximation question, providing the lower bounds of $n/2 - n^{\epsilon}$ for any ϵ for many of the problems considered in [FLN00] and giving tight upper bounds of $n/2$ for several natural problems including Vertex Cover, as well as a surprising $n/2 + O(\sqrt{n \log n})$ lower bound for the universal NP-complete language. The latter result they extend to existence of such inapproximable verifiers for all paddable (in Berman-Hartmanis [BH77] sense) NP languages, improving on [KS99]. Their proof techniques avoid error-correcting codes altogether, relying instead on search-to-decision (Turing) reductions. For example, the $n/2 - n^{\epsilon}$ bound for SAT is achieved by showing how to determine the value of the first variable correctly assuming the existence of such approximation algorithm: add enough copies x_i of the first variable x, together with clauses $(x \leftrightarrow x_1)$ and $\forall i$, $(x_i \leftrightarrow x_{i+1})$, to overwhelm the n^{ϵ} factor, and recover the correct value for x from the approximate solution by taking the majority over x_i's. The proof of the better-than-$n/2$ bound for the universal language also relies on downwards self-reducibility, though the algorithm is somewhat more involved. Most of the lower bounds results of [SY13] have a slight generalization to randomized algorithms with good parameters.

3 Preliminaries

We follow [HMvRW07] for the structure approximation framework definition. In particular, a distance function is a integer-valued metric on feasible solutions, given as a part of the problem description.

Definition 1. *[HMvRW07] Let an optimization problem Π be a 4-tuple $\langle I, cansol, val, goal\rangle$, where I is a set of instances, cansol a set of feasible solutions, function $val(x, y)$ states the value of a solution $y \in cansol$ to an instance $x \in I$ and $goal \in \{MIN, MAX\}$ states whether Π is a minimization or a maximization problem. Let $optsol(x)$ be the set of optimal solutions to Π. Finally, let $d(y, z)$, be a (metric, integer-valued) distance function defined on encodings of $y, z \in cansol$.*

Then, a polynomial-time algorithm A is a $h(|x|)/d$ structure approximation (s-approx) algorithm for Π iff for every $x \in I$, the distance $d(A(x), optsol(x)) \leq h(|x|)$. Here, $d(y, optsol(x))$ is $\min_{z \in optsol(x)} d(y, z)$.

Note that this framework includes the approximation with respect to the ratio of values of the candidate feasible solution and an optimal solution as a special case. Let the distance function on solutions to an instance x be $d(y, z) = |log(val(x, y)/val(x, z))|$. This function is a metric: $d(y, y) = 0$, the symmetry comes from taking the absolute value and the triangle inequality is satisfied. If there is an approximation algorithm giving a factor $r(|x|)$ approximation, the corresponding $h(|x|) = |\log r(|x|)|$.

In the witness approximation setting of [KS99,FLN00,SY13], an approximation algorithm is allowed to return a string which is not necessarily a feasible solution, however in the [HMvRW07] setting it must return a solution that is feasible. Though not relevant for SAT where every binary string of length n encodes an assignment, for problems such as VertexCover the algorithm should return a string which does encode a vertex cover of the graph, albeit possibly one that is larger than the optimal.

There are two main views of a feasible solution (or a witness to an NP problem). In one, arbitrary encodings of the solutions are allowed: in partiular, in Kumar and Sivakumar [KS99], and in the paddable problems result of Sheldon and Young [SY13], a witness can be any polynomial-length string from which a polynomial-time verifier can extract and check the answer. In [KS99], witnesses are codewords of an error-correcting code which have, by virtue of being codewords, a large Hamming distance from each other. Another is natural (often also minimal) witnesses: for example, a satisfying assignment to a formula encoded as a binary string of length equal to the number of variables is a natural witness. If in the setting of Kumar and Sivakumar [KS99] only a small subset of binary strings could code solutions (i.e., only codewords), any binary string of length n codes an assignment; if an optimization problem is MaxSat, then each of these strings corresponds to a feasible solution. In practice, natural witnesses are often easier to approximate, in particular with respect to the Hamming distance; often a random string or a string of all 0s is guaranteed to be within $n/2$ from the optimal. By contrast, the general bound of Sheldon and Young [SY13] shows the existence of witnesses, i.e. encodings of solutions, that are hard to approximate even within the Hamming distance $n/2 + O(\sqrt{n \log n})$.

4 Approximating solution structure with edit distance measure

Consider $d_E(y, z)$ to be the *edit distance* between strings y and z, that is, the number of insert, replace and delete a symbol operations needed to convert y into z. This function, even though in some respect related to Hamming distance, nevertheless has a very different behaviour. For example, a string 01010101 and a string 10101010 have the maximal Hamming distance of $n = 8$, however their edit distance is just 2, corresponding to deleting a 0 in front and inserting it in the back of the string. For Hamming distance, a random string is expected to be within $n/2$ from any string, but it is not clear what the expected edit distance between two random strings is. If two strings

are far in the edit distance though, then in particular they are far in the Hamming distance. So lower bounds on edit distance approximability imply lower bounds for the Hamming distance, but the reverse is not immediate.

However, in case when one of the strings is a string of all 0s or all 1s then the two notions coincide, as long as the length of the approximating string is the same. Indeed, even edit distance with transpositions to a string of all 1s from any given string is equivalent to Hamming distance.

Lemma 1. *For any string x of length n, its Hamming distance to a string of n 1s is equal to the edit distance.*

Proof. The Hamming distance between x and the string of n 1s is the number of $0s$ in x; call it k. Now, since replacements are one of the operations counted in the edit distance, the edit distance is no greater than the Hamming distance k. Suppose that the edit distance is $k' < k$. Consider the corresponding sequence of k' operations converting the string of all 1s into x. Since $k' < k$, one of the operations is not replacement. Suppose that the first such operation in the sequence is a deletion. Then, as we assumed that $x = n$, there will be a corresponding insertion later in the sequence. It can be seen that the deletion removed a 1, and insertion introduced a 0. Now, modifying indices of the symbols after the deletion point, it is possible to simulate this pair of operations with one replacement. A similar argument holds for the first operation being an insertion. (Alternatively, it is sufficient to say that there are only two operations that increase the number of 0s in a string of all 1s: a replacement and an insertion. As any insertion has to have a corresponding deletion, the most efficient way to obtain a string with k 0s out of a string of all 1s is k replacements).

The following theorem from Sheldon and Young [SY13] provides the main technique for proving structure inapproximability results for Hamming distance.

Theorem 1. *[SY13] If for some $\epsilon > 0$ there is an algorithm A such that for any $\phi \in SAT$ $A(\phi)$ produces a truth assignment within the Hamming distance $n/2 - n^{\epsilon}$ of a satisfying assignment to ϕ, then P=NP.*

Proof. First, note that it is enough to have an algorithm determining the value of one variable. Let z_1 be the variable to be amplified. Construct a formula ϕ_1 consisting of ϕ together with clauses $(x_i \leftrightarrow x_{i+1})$ for new variables $x_1, \ldots, x_{n^{1/\epsilon}}$, as well as a clause $(z_1 \leftrightarrow x_1)$. These clauses state that all x_is are equivalent to z_1, so they can be considered copies of z_1. Now, if there is an algorithm that is guaranteed to return a witness within the $n/2 - n^{\epsilon}$ Hamming distance of a satisfying assignment, then such a string will be correct on majority of copies of z_1. Taking the majority of values of x_is thus gives the correct value of z_1. Repeating this process n times each time simplifying the formula using computed partial assignment results in a correct satisfying assignment.

This theorem together with lemma 1 gives the following corollary.

Theorem 2. *Unless P=NP, no algorithm can approximate the natural witness to SAT to within the edit distance $n/2 - n^{\epsilon}$ for any $\epsilon > 0$.*

Proof (Proof sketch). Note that a natural witness for an amplified instance from theorem 1 proof consists of either $n^{1/\epsilon}$ 0s or $n^{1/\epsilon}$ ones, together with $n-1$ symbols of arbitrary values for the rest of the variables; moreover, we can assume that all values of the copies of z_1 are together, for example forming the first $n^{1/\epsilon}$ positions of the string. Now, suppose there is an algorithm that approximates

the satisfying assignment above, with $n^{1/\epsilon}$ copies of z_1, to within edit distance $N/2 - N^\epsilon$ rather than Hamming distance, where $N = n + n^{1/\epsilon}$. Let y' be a string returned by the approximation algorithm and y the corresponding optimal solution. Consider only the first $n^{1/\epsilon}$ positions in y', ones corresponding to the copies of z_1. Without loss of generality, assume that $z_1 = 1$ in y. These positions can be changed to 0 (to obtain y') by either a replacement or an insertion/deletion pair moving values of the remaining $n-1$ variables into the first $n^{1/\epsilon}$ positions. But as discussed above, in this case the number of insert/delete pairs is at least as large as the number of replacements. Therefore, the same argument as for the Hamming distance applies, and bounding the edit distance between y and y' by $N - N^\epsilon$ means that majority of the copies of z_1 in y' have a correct value.

A similar argument can be used to show $n/2-n^\epsilon$ lower bound for the edit distance approximation of VertexCover; however, as it will involve a string of 1s and a string of 0s, the only edit distance operations allowed will be insertions, deletions and replacements. Recall that in the MinVertexCover the goal is to determine a minimal set of vertices such that every edge has at least one endpoint in the cover; the decision version VertexCover asks to determine if there is a cover of size at most k. A natural witness to VertexCover is a binary string of length $n = |V|$, where a bit corresponding to a vertex is 1 iff that vertex is in the cover, that is, a characteristic string of a set of vertices in the cover.

Theorem 3. *[SY13] If for some $\epsilon > 0$ there is an algorithm A such that for any $\langle G, k\rangle \in VertexCover$ $A(\langle G, k\rangle)$ produces a binary string within the Hamming distance $n/2-n^\epsilon$ of a characteristic string of a vertex cover of size at most k in G, then P=NP.*

Proof (Proof sketch). As in the proof of theorem 1, it is enough to determine whether any given vertex belongs to a vertex cover of size at most k. To construct an amplified graph G', make a copy v' of an arbitrary vertex v and add an even-length path on $2n^{1/\epsilon}$ vertices between v and its copy v'. Set $k' = k + n^{1/\epsilon}$. Now, as a minimal vertex cover of an even-length path consists of either all even or all odd vertices on the path, we say that the original v is in the $k + n^{1/\epsilon}$ cover if all even vertices are in that cover, otherwise v is not in the cover. Then the argument proceeds by showing that the majority of the vertices on the path will be correctly placed.

Theorem 4. *Unless P=NP, no algorithm can approximate the natural witness to VertexCover to within edit distance $n/2 - n^\epsilon$ for any $\epsilon > 0$.*

Proof. Consider the [SY13] construction described above, but with a specific ordering of positions encoding vertices in the witness. Let vertices $v_1, \ldots, v_n$ be the original vertices, v' a copy of a selected vertex e.g. of v_1, $u_1, \ldots, u_{n^{1/\epsilon}}$ be even-numbered vertices on the path from v to v' and $w_1, \ldots, w_{n^{1/\epsilon}}$ be the odd vertices on that path. Now, in the witness the first $n^{1/\epsilon}$ positions will correspond to the u-vertices, followed by v_is, in turn followed by the w_js. The length of this witness string is $N = n + 1 + 2n^{1/\epsilon}$.

Now, the same kind of argument as before applies. A minimal cover will be encoded by either a string of $n^{1/\epsilon}$ 0s followed by some string of length $n+1$ followed by $n^{1/\epsilon}$ 1s, or a similar string with 0s at the beginning and 1s at the end. If k is not the size of the minimal cover, there can be as many as $k - k_{min}$ extra vertices from the path in the cover, but by construction $2n^{1/\epsilon}$ should overwhelm this number of errors. Now, similarly to the SAT construction, we would like to argue that a sequence of $N/2 - N^\epsilon$ of arbitrary edit operations (inserts, deletes, replacements) would not result in any string that differs from the correct cover in more than $n^{1/\epsilon}$ positions on u-portion and w-portion of the string.

Consider a pair of insert/delete operations needed to convert a string encoding an optimal cover to an approximate string. Suppose, without loss of generality, that the optimal string starts with 1s and ends with 0s. Consider deleting a value from the u part of the string and inserting it into the w part. Now, the middle part of the string, corresponding to the v variables, could become maximally far from the minimal vertex cover at that point (i.e., if it was of the form 01010101), however we are only concerned with the u and w parts. The pair of insert-delete operations then introduces at most one 0 into the u part (by shifting the v part into it), and at most one 1 into the w part by insertion. Therefore, the "damage done" to these parts of the string is no more than from doing two replacements.

Therefore, if there exists a structure approximation algorithm for vertex cover that can consistently return a string within edit distance $n/2 - n^{\epsilon}$ from an optimal cover, then this algorithm can be used to determine exactly whether any given vertex is in the k-cover. By Turing/search-to-decision reduction, from there the actual cover can be computed. In this reduction, if a vertex was determined to be in the cover, then recurse on a graph without this vertex, and otherwise recurse on a graph without this vertex and all of its neighbours, adding the neighbours to the cover.

So far, we have discussed the complexity of approximating an NP witness, however in majority of practical problems it is approximating an optimal solution which is of interest. But since lower bounds on decision problems imply lower bounds on optimization problems, the results above give inapproximability of the optimization version of this problem, in particular MaxSAT and MinVertexCover.

5 Applications

One type of problems which naturally seem appropriate to analyze in the context of the edit distance approximation is alignment problems. Here we will look at two examples of NP-hard alignment problems, and show the Hamming distance and the edit distance inapproximability for them. The first example, phrase alignment, or, more specifically, weighted sentence alignment, comes from the field of machine translation; the second is the longest common subsequence problem, with its well-known bioinformatics applications.

5.1 Machine translation: phrase alignment

A phrase alignment problem arises in the field of machine translation. There, a sentence in one language needs to be aligned with its translation in another; though computationally tractable in the context of word-to-word translation, this problem becomes NP-hard when the segmentation of the sentences into phrases to be aligned is required. When the alignment is monotone, that is, the i^{th} phrase of the first sentence is mapped to the i^{th} phrase of the second, and the only computational task is the segmentation itself, the problem can be solved by a polynomial-time dynamic programming algorithm [DeN10]. So the interesting case is the phrase alignment where both the segmentation is present, and the order of the aligned phrases can be arbitrary.

Now, following DeNero and Klein [DK08], we formally define a *weighted sentence alignment (WSA)* problem as follows. Let e and f be sentences. The phrases in e are represented by a set $\{e_{ij}\}$, where e_{ij} is a sequence of words from in-between-word position i to j in e; f is represented by $\{f_{kl}\}$ in the same fashion. A link is an aligned pair of phrases (e_{ij}, f_{kl}). An alignment is a set of links such that every word (token), in either sentence, occurs in exactly one link (here, we treat each

occurrence of a word as a separate word). A weight function $\phi : \{(e_{ij}, f_{kl})\} \to \mathbb{R}$ assigns a weight to each link. A total weight of an alignment a, denoted $\phi(a)$, is a product of weights of its links. Now, an optimization version of the weighted sentence alignment problem asks, given (e, f, ϕ), to find an alignment with the maximum weight. A decision version of this problem can be stated as finding an alignment a of weight $\phi(a) \geq 1$.

Let us consider a restricted version of WSA corresponding to the weight function $\phi \in \{0, 1\}$ and the second sentence always segmented into words. It is easy to see that $3SAT \leq_p WSA$ reduction of DeNero and Klein [DK08] shows NP-hardness of this restricted problem of computing the segmentation of the first sentence. Thus, the witness to this problem can be encoded as a binary string of length $|e|$. Now, methods similar to the structure inapproximability proof for the VertexCover problem above can be used to show $n/2-n^\epsilon$ inapproximability of the solution structure of this witness both for Hamming distance and Edit Distance.

5.2 Bioinformatics: multiple longest common subsequence

The (multiple) longest common subsequence problem, LCS, is defined as follows: given a set of m strings $\{s_1, \ldots, s_m\}$ over an alphabet Σ, find the longest sequence of symbols which forms a subsequence of each of $s_1, \ldots, s_m$ (for the decision version, determine if there is a string with length at least a given k). This problem arises in several areas of bioinformatics, with looking for a substring of genome common to all given genomes being the simplest example. For a constant number m of strings, this problem is solvable in polynomial time; the classic dynamic programming algorithm for $m = 2$ is taught in many algorithms courses. However, already in 1978 Maier [Mai78] has shown that this problem is NP-hard for an arbitrary m.

The NP-hardness proof proceeds by reduction from VertexCover: for a graph on n vertices and m edges, construct an instance of LCS consisting of $m + 1$ strings. Set the first string to be the sequence t of all vertices $v_1 \ldots v_n$ (thus, $\Sigma = V$, the vertex set of G). Then, for every edge (v_i, v_j), introduce a string of length $2(n - 1)$ consisting of two copies of the sequence t of all vertices, except in the first copy the vertex v_i is missing, and in the second the vertex v_j, that is, a string $v_1 \ldots v_{i-1} v_{i+1} \ldots v_n v_1 \ldots v_{j-1} v_{j+1} \ldots v_n$ (assuming, without loss of generality, that $i < j$). With this construction, the longest common subsequence w corresponds to the set of vertices not in a minimal vertex cover of G, as w needs to be a subsequence of at least one half of each sequence encoding an edge; conversely, eliminating vertices in a minimal vertex cover from the first string leaves a sequence which is a subsequence of every s_i.

In order to talk about Hamming and edit distances, we need to specify an encoding of a witness: let us encode the solution by taking the shortest (or lexicographically first shortest) string s out of $\{s_1 \ldots s_m\}$, and representing the longest common subsequence as a characteristic string of length s encoding positions of s that are in the longest common subsequence. That is, if the shortest string is $ABABC$ and the longest common subsequence is ABC, then the witness can be 10011 (it is not a unique representation in general, but this does not matter for us). In the reduction above, this notation gives us a characteristic string of vertices not in the vertex cover, as t is the shortest string.

With this witness representation, we can translate the results for the Hamming and edit distance inapproximability of VertexCover to LCS. Consider an amplified instance of VertexCover from the construction in the proof of theorem 3, that is, an instance with an extra copy of a vertex and a path on $2n^{1/\epsilon}$ new vertices between the original vertex and its copy. As in the proof of theorem 4, rename the new vertices so that all even vertices on the path come first in the order, and all odd

vertices come last (it does not matter where the original vertices are, as long as the two blocks of even and odd vertices are contiguous; let us assume that they are between the even and odd blocks). Now, when the reduction to LCS above is applied to this graph, the string t encoding vertices will preserve their order. As the optimal solution must have either all even or all odd vertices from the path in the cover, either even or odd block of vertices will not be in the longest common subsequence, with the witness consisting of a block of $n^{1/\epsilon}$ 1s and a block of a $n^{1/\epsilon}$ 0s. By the same argument as for VertexCover, it is then possible to recover whether the amplified vertex is in the original vertex cover or not from a string within a Hamming or edit distance $N/2 + N^{\epsilon}$ from the witness, where $N = n + 2n^{1/\epsilon} + 1$.

6 Conclusions

In this paper, we considered structure approximation framework of [HMvRW07] for several problems with respect to the edit distance measure. We were able to extend [SY13] Hamming distance lower bounds to edit distance for several problems including SAT and VertexCover (which translates into bounds for Clique and Independent set as well via the usual reductions). These lower bounds are tight. We then considered two other problems in this framework, the Longest Common Subsequence and the Weighted Sentence Alignment, and showed similar lower bounds with respect to both Hamming distance and edit distance approximation.

The lower bounds are pessimistic though, and imply quite strong inapproximability for these problems. Is there any way to get around these bounds in practice? In fact, many practical problems have strong (constant) restrictions on some of the parameters, leading to efficient parameterized algorithms. Already [HMvRW07] introduced the notion of parameterized structure approximation setting, and proved, in particular, a lower bound on parameterized complexity of Longest Common Subsequence with respect to edit distance metric. This would be an interesting direction to investigate with respect to the parameters occurring in practical applications.

In this paper, we focused on lower bounds. Devising structure approximation algorithms is an interesting question on its own. Sheldon and Young [SY13] show that the algorithm for VertexCover based on rounding a linear programming relaxation produces a solution containing all vertices of some minimal vertex cover, and at most as many extraneous ones. Thus, if the minimal vertex cover has size k, the computed solution is a k-Hamming distance approximation. An interesting question is whether this can be extended to other linear programming algorithms, in particular rounding-based.

For the more theoretical direction, it would be interesting to see if there is a generic way to build a lattice of hardness implications for various distance functions. We conjecture that any such function with a certain "locality property" (that is, one "unit of change" only affects a small, though not necessarily constant) number of positions should be inapproximable by generalizing Hamming distance results. Alternatively, one wonders if there is a non-trivial, practically interesting distance function for which there is, indeed, a fast approximation algorithm for some NP-hard problem. In that respect, considering various distance measures and their interrelation with respect to computational problems is a promising area with a possibility for new approaches to problems from a wide variety of computational fields.

7 Acknowledgements

We are very grateful to Valentine Kabanets, Todd Wareham and Russell Impagliazzo for numerous discussions and suggestions, and to Venkat Guruswami for telling us about then-unpublished work of Sheldon and Young. We also thank the anonymous referees for their extensive comments and suggestions.

References

[BH77] Leonard Berman and Juris Hartmanis. On isomorphisms and density of NP and other complete sets. *SIAM Journal on Computing*, 6(2):305–322, 1977.

[DeN10] John Sturdy DeNero. *Phrase Alignment Models for Statistical Machine Translation*. PhD thesis, UC Berkeley, 2010.

[DK08] John DeNero and Dan Klein. The complexity of phrase alignment problems. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Short Papers*, pages 25–28. Association for Computational Linguistics, 2008.

[FLN00] Uriel Feige, Michael Langberg, and Kobbi Nissim. On the hardness of approximating NP witnesses. In *APPROX*, pages 120–131, 2000.

[GHLP99] Anna Gal, Shai Halevi, Richard J. Lipton, and Erez Petrank. Computing the partial solutions. In *14th Annual IEEE Conference on Computational Complexity (CCC'99)*, pages 34 – 45, 1999.

[GR08] Venkatesan Guruswami and Atri Rudra. Soft Decoding, Dual BCH Codes, and Better List-Decodable ε-Biased Codes. In *IEEE Conference on Computational Complexity*, pages 163–174, 2008.

[HMvRW07] Matthew Hamilton, Moritz Müller, Iris van Rooij, and Todd Wareham. Approximating solution structure. In Erik Demaine, Gregory Z. Gutin, Daniel Marx, and Ulrike Stege, editors, *Structure Theory and FPT Algorithmics for Graphs, Digraphs and Hypergraphs*, number 07281 in Dagstuhl Seminar Proceedings. Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany, Dagstuhl, Germany, 2007.

[KS99] Ravi Kumar and D. Sivakumar. Proofs, Codes, and Polynomial-Time Reducibilities. In *IEEE Conference on Computational Complexity*, pages 46–53, 1999.

[Mai78] David Maier. The complexity of some problems on subsequences and supersequences. *Journal of the ACM (JACM)*, 25(2):322–336, 1978.

[SY13] Daniel Sheldon and Neal E. Young. Hamming Approximation of NP Witnesses. *Theory of Computing*, 9(22):685–702, 2013.

[Vaz01] Vijay V. Vazirani. *Approximation algorithms*. Springer-Verlag New York, Inc., New York, NY, USA, 2001.

[vRW12] Iris van Rooij and Todd Wareham. Intractability and approximation of optimization theories of cognition. *Journal of Mathematical Psychology*, 56(4):232 – 247, 2012.