Reoptimization of Motif Finding Problem

Jhoirene B. Clemente, Jeffrey A. Aborot, Henry N. Adorna,

Abstract—**One of the approaches in solving NP-hard problems is through reoptimization. In this technique, we solve a locally modified instance of a problem by making use of known solution to its original instance instead of obtaining a solution from scratch. In this paper, we present a reoptimization of motif finding problem. Since the problem is showed to be self-reducible, we can use a self-reduction method to solve the reoptimization variant of the problem. Using the method, we have improved the approximation ratio of the algorithm solving the reoptimized version as compared to the non-reoptimized counterpart. Moreover, we showed that if a certain problem is self-reducible, any problem that obtains a polynomial-time reduction to it is also self-reducible.**

Index Terms—**motif discovery, reoptimization**

I. Introduction

Unless P=NP, most of the interesting discrete optimization problems are NP-Hard, that is we cannot find a polynomial-time algorithm which solves the problem on a Turing Machine. The most common approach in solving this kind of problems is to relax the condition of always finding the optimal solution for an instance and settle for "good enough" solutions. The kind of algorithms which are guaranteed to obtain a solution with a certain quality are called approximative algorithms. The goodness of these approximative algorithms are measured using an approximation ratio.

Definition 1 (Approximation Algorithm [9]):
An σ-approximation algorithm for an optimization problem is a polynomial-time algorithm that for all instances of the problem produces a solution whose value is within a factor of σ of the value of an optimal solution.

Problems where there exists an approximation algorithm solving it are called approximable problems. Suppose we have an instance of a minimization problem I with an optimal solution $Opt(I)$. Let $cost(I, Opt)$ be the cost of the optimal solution of the problem instance I. Note that, an exact algorithm that solves for a given minimization problem always obtains the minimum possible cost, but not for approximative algorithms. In order to asses the quality of the approximation algorithm, we have approximation ratio σ. An algorithm for a *minimization problem* is called σ-*approximative algorithm* for some $\sigma > 1$, if the algorithm obtains a maximum cost of $\sigma \cdot cost(I, Opt)$, for any input instance I. Meanwhile, an algorithm for a *maximization problem* is called σ-*approximative algorithm*, for some $\sigma < 1$, if the algorithm obtains a minimum cost of $\sigma \cdot cost(I, Opt)$, for any input instance I.

Improving the approximability of a problem involves improving the approximation ratio σ of a solution to the problem. One such group of algorithms that can guarantee the goodness of solution by a factor of ϵ are called Polynomial-Time Approximation Scheme (PTAS). Moreover, a group that can guarantee the goodness of the solution and the bound of the running time is called Fully Polynomial-Time Approximation Scheme (FPTAS) [9].

For real world problems, additional information about the problems we are solving are available, and so we may not have to solve them from scratch. One of the approaches is making use of apriori information, which can be a solution to a smaller input instance of a problem to solve a larger instance of it. This approach is called *reoptimization*. The idea was first mentioned in [7]. Reoptimization may help to improve the approximability of the problem or the running time of the solution to it. In fact, we can obtain a PTAS for a reoptimization variant of a problem given that the unmodified problem is approximable [10]. The formal definition of reoptimization is as follows.

Definition 2 (Reoptimization [10]):
Let $\Pi = (\mathcal{D}_\Pi, \mathcal{R}_\Pi, cost_\Pi, goal_\Pi)$ be an NP Optimization (NPO) problem, where $I \in \mathcal{D}_\Pi$ is an instance from the set of all valid instances of problem Π, $SOL \in \mathcal{R}_\Pi(I)$ is a solution from the set of feasible solutions of Π, $cost_\Pi(I, SOL)$ is a polynomial-time computable function that evaluates a certain SOL given I, and a $goal_\Pi \in \{max, min\}$ which identifies whether Π is a minimization or a maximization problem. Let $\mathcal{M} \subseteq \mathcal{D}_\Pi \times \mathcal{D}_\Pi$ be a binary relation (the modification).

The corresponding reoptimization problem

$$\mathbf{R}_\mathcal{M}(\Pi) = (\mathcal{D}_{\mathbf{R}_\mathcal{M}(\Pi)}, \mathcal{R}_{\mathbf{R}_\mathcal{M}(\Pi)}, cost_{\mathbf{R}_\mathcal{M}(\Pi)}, goal_{\mathbf{R}_\mathcal{M}(\Pi)})$$

consists of

1) a set of feasible instances defined as
$$\mathcal{D}_{\mathbf{R}_\mathcal{M}(\Pi)} = \{(I, I', SOL) : (I, I') \in \mathcal{M} \text{ and } SOL \in \mathcal{R}_\Pi(I)\};$$
we refer to I as the original instance and to I' as the modified instance
2) a feasibility relation defined as
$$\mathcal{R}_{\mathbf{R}_\mathcal{M}(\Pi)}((I, I', SOL)) = \mathcal{R}_\Pi(I')$$

To put it simply, given a problem instance and an optimal solution for it, we are to efficiently obtain an optimal solution for a locally modified instance of the problem [3]. Providing a local modification of a problem instance is answered by the so called *Postoptimality analysis* [3]. It answers the question of how much an instance can be modified such that the set of optimal solutions are unchanged. Reoptimization for several optimization problems already exists in the literature. Two reoptimization variants were presented for the shortest superstring problem in [2], which involves adding and removing a string and was shown to improve the approximation ratio of the current best approximation algorithm. Reoptimization and approximability for several graph problems involving

Manuscript received January 8, 2014; revised Jan 28, 2014.

All authors in this publication are members of Algorithms and Complexity Lab, Department of Computer Science, University of the Philippines Diliman

Jhoirene Clemente would like to thank Engineering Research and Development for Technology (ERDT) for the dissertation grant and PhD Scholarship

cliques, covers, and independent sets were also presented in [10]. It is also shown from the literature that not all problems with a reoptimization variant can help to improve the approximability of an algorithm [5].

The goal of this study is to define a reoptimization variant of the Motif Finding Problem (MFP) and to provide a solution to the defined reoptimization variant such that we can improve on either the quality of the solution or the running time of the algorithm which solves it. The motivation is driven by fact that DNA sequences from new species are being compared to previously known set of species. With reoptimization, there'll be no more need to perform sequence analysis from scratch, instead we make use of solution to previously known instances to improve the quality of the solution or the algorithm solving it.

It is first shown that MFP is *self-reducible*. Self-reduction method presented in [10] can then be used to solve the reoptimization variant of MFP. This method was also used in solving steiner tree problem in [1]. In showing the self-reducibility of MFP, we further identify the self-reducibility of problems which are polynomial-time reducible to a self-reducible problem as shown in Theorem 1.

This paper is outlined as follows. In Section II, the original definition of MFP that is based on consensus score presented in [4] is discussed. In this section, necessary notations and functions needed to define the modified version of MFP are also defined, which we will denote as mMFP. In Section III, proof of the self-reducibility of MFP is presented. In Section IV, reoptimization variant of mMFP and the self reduction method based on the type of modification applied are discussed. Finally, in Section V, contribution of this study is summarized.

II. Motif Finding Preliminaries

Motifs are significantly occurring patterns in the DNA, which may be transcription binding sites. Mapping these patterns are important for gene function discovery and building regulatory networks [6]. To formally define MFP as an optimization problem, let us go through some notations.

The input to the problem is a set of DNA strings denoted by the set $\mathcal{S} = \{S_1, S_2, \ldots, S_t\}$, which contains t sequences defined over the alphabet $\Sigma_{DNA} = \{a, c, g, t\}$ and $|S_i| = n$. The objective is to look for a pattern of length l among the t sequences.

Let us assume that the patten occurs exactly once per sequence. The set of feasible solutions to the problem is a set of starting position vectors denoted by $p = (a_1, a_2, \ldots, a_t)$ where each a_i corresponds to the starting position of the occurrence of the pattern in sequence S_i. The value of a_i may range from 1 up to $(n - l + 1)$.

Given a starting position vector p, we can obtain a set of ordered l-length strings from $\mathcal{S}$. Let $L(p, \mathcal{S}) = \{l_{a_1}, l_{a_2}, \ldots, l_{a_t}\}$ be the ordered set of strings obtained from $\mathcal{S}$ using p. Each l-length string l_{a_i}, is obtained from a_ith position of sequence S_i.

The set of strings $L(p, \mathcal{S})$ forms an alignment of t l-length strings. We define a profile matrix $F(p, \mathcal{S})$ with dimension $|\Sigma_{DNA}| \times l$. Each row in $F(p, \mathcal{S})$ corresponds to a symbol in Σ_{DNA}. Each column in $F(p, \mathcal{S})$ corresponds to position j in the alignment, where j ranges from 1 to l. The content $F(p, \mathcal{S})_{\gamma,j}$ is the frequency of symbol $\gamma \in \Sigma_{DNA}$ in column j of the alignment made by starting position vector p on $\mathcal{S}$. Let $max(p, \mathcal{S}, j)$ be the maximum of $F(p, \mathcal{S})$ in column j.

The cost of an alignment made by p on $\mathcal{S}$ is computed as follows.

$$cost(\mathcal{S}, p) = \sum_{j=1}^{l} max(p, \mathcal{S}, j) \tag{1}$$

Definition 3 (Motif Finding Problem (MFP) [4]):
INPUT:

- Pattern length l
- Set of t sequences $\mathcal{S} = \{S_1, S_2, \ldots, S_t\}$ defined over Σ_{DNA}, where $|S_i| = n$

OUTPUT: A starting position vector $p = (a_1, a_2, \ldots, a_t)$ such that $cost(\mathcal{S}, p)$ is maximum.

To show that MFP is self-reducible, we redefine the problem by modifying the cost function based from the definition of Median String Problem (MSP). Two problems are proven to be equivalent in the literature [4]. Hence, modification of the cost function doesn't change the set of feasible solution to either problems. The new cost function for MFP is computed as follows.

The cost function also evaluates a certain starting position vector p given a set of sequences $\mathcal{S}$. This cost function is derived from the formulation of MSP in [4]. MSP uses dissimilarity of strings to evaluate the closeness of the alignment made. Hence, the problem becomes a minimization problem. To use the cost function for MFP as a maximization problem, we use a similarity function instead.

Let us first define the similarity, denoted as $sim(v, w)$, between two strings v and w, where $|v| = |w| = l$. The $sim(v, w)$ is equal to the total number of symbol matches between the two strings when aligned. Formally, the similarity function is computed as

$$sim(v, w) = \sum_{j=1}^{l} g(v(j), w(j)),$$

where

$$g(a, b) = \begin{cases} 1 \text{ if } a = b \\ 0 \text{ o.w.} \end{cases}$$

To evaluate the similarity of a given l-length string v with respect to a set of l-strings obtained from p on $\mathcal{S}$, let us define

$$\overline{sim}(v, p) = \sum_{i=1}^{t} sim(v, l_{a_i})$$

be the similarity of a string to an alignment made by p on a given set of sequences $\mathcal{S}$.

We then need to assess a certain l-length string v for its feasibility being the pattern we are looking for. Given the set of sequences $\mathcal{S}$ and a string v the computing a certain cost involves minimization over all possible alignment in $\mathcal{S}$. MFP is a double minimization problem if we consider alignments of strings as the set of feasible solution instead of all possible starting position vectors. The cost of v on $\mathcal{S}$ is computed as follows.

$$c(\mathcal{S}, v) = \max_{\text{over all possible } p \text{ on } \mathcal{S}} \overline{sim}(v, p)$$

To extend the definition of cost to assess if an alignment is a possible motif occurrence on $\mathcal{S}$, we define

$$\overline{cost}(\mathcal{S}, p) = \sum_{i=1}^{t} c(\mathcal{S}, l_{a_i}). \tag{2}$$

Therefore, let us redefine MFP using the modified cost function.

Definition 4 (Modified Motif Finding Problem (mMFP)):

INPUT:

- Pattern length l
- Set of t sequences $\mathcal{S} = \{S_1, S_2, \ldots, S_t\}$ defined over Σ_{DNA}, where $|S_i| = n$

OUTPUT: A starting position vector $p = (a_1, a_2, \ldots, a_t)$ such that $\overline{cost}(\mathcal{S}, p)$ is maximum.

The equivalence of both cost in Equation 1 and a minimization variant of cost in Equation 2 presented in [4]. Since we can convert any cost function from maximization to a minimization counterpart, it follows the equivalence of two cost functions in Equation 1 and 2. For all input instances, the optimal solution found using the original cost function is the optimal solution found using the modified cost function and vice versa.

III. Self-reducibility

In order to use the self reduction methods presented in [10], we first show that the problem of interest (mMFP) is self-reducible.

Note that, self-reducibility in Definition 5 is different from the definition of general self-reducibility in [8], which we will no longer discuss in this paper. But note however that self-reducibility implies general self-reducibility.

Definition 5 (Self-reducibility [10]):
We will say that a problem Π is **self-reducible** if there is a polynomial-time algorithm, Δ, satisfying the following conditions.

1) Given an instance I and an atom α of a solution to I, Δ outputs an instance I_α. We require that the size of I_α is smaller than the size of I, i.e. $|I_\alpha| < |I|$. Let $\mathcal{R}(I|\alpha)$ represent the set of feasible solutions to I containing the atom α. We require that every solution SOL of I_α, i.e., $SOL \in \mathcal{R}(I_\alpha)$, has a corresponding $SOL \cup \{\alpha\} \in \mathcal{R}(I|\alpha)$ and that this correspondence is one-to-one.
2) For any set $H \in \mathcal{R}(I_\alpha)$ it holds that the
$$cost(I, H \cup \{\alpha\}) = cost(I, \alpha) + cost(I_\alpha, H).$$

An *atom*, as used in this context, constitutes a solution for a specific problem instance. For example, in a maximal clique problem, a solution is a clique with maximum number or vertices. Since a clique is composed of vertices, the atoms in this problem are vertices from the given input graph.

Given the definition of self reducibility of a problem. We prove that the following lemma is true.

Lemma 1: Modified Motif Finding Problem (mMFP) is *self-reducible*.

Proof: The set of input instances I of mMFP is a pair $I = (\mathcal{S}, l)$. A solution SOL to mMFP given instance I is an ordered set (equivalent to what is shown earlier as starting position vector p but can also be represented as an ordered set for the purpose of our discussion) $p = \{(1, a_1), (2, a_2), \ldots, (t, a_t)\} \in \mathcal{R}(\mathcal{S}, l)$, where the set of all atoms, defined by $Atoms(\mathcal{S}, l)$, contains all possible starting positions from $\mathcal{S}$. Hence, an atom α is a pair (i, a_i), but for the sake of simplicity, a_i refers to a part of a solution obtained from ith sequence starting at a_ith position.

Let us define a reduction function $\Delta(I, \alpha)$, which accepts a pair $(\mathcal{S}, l)$ and an atom a_i, i.e. $\Delta((\mathcal{S}, l), a_i)$ and produces a reduced instance I_α which we denote as $(\mathcal{S}_{a_i}, l)$.

The modified instance $(\mathcal{S}_{a_i}, l)$ is derived by removing one sequence S_i (corresponding to an atom a_i) from $\mathcal{S}$, i.e. $\mathcal{S}_{a_i}$ is $\mathcal{S} \setminus \{S_i\}$. We argue next that $\Delta(I, \alpha)$ follows the two properties stated in Definition 5.

1) For a $SOL \in \mathcal{R}(I)$ there is a corresponding $SOL_\alpha \cup \{\alpha\} \in \mathcal{R}(I|\alpha)$. Note that the solution $p = \{(1, a_1), (2, a_2), \ldots, (t, a_t)\}$ guarantees exactly one occurrence of the motif for each sequence. For any atom $\alpha = a_i$, a solution $p \in \mathcal{R}(\mathcal{S}, l)$ corresponds to $p_{a_i} \cup \{a_i\}$, i.e.
$$\{(1, a_1), (2, a_2), \ldots, (i-1, a_{i-1}), (i+1, a_{i+1}), \ldots, (t, a_t)\} \cup \{(i, a_i)\} \in \mathcal{R}(\mathcal{S}|a_i).$$
2) Clearly, the $cost(I, H \cup \{\alpha\}) = cost(I, \alpha) + cost(I_\alpha, H)$, for $H \in \mathcal{R}(I_\alpha)$ since
$$\overline{cost}(\mathcal{S}, p) = \sum_{j=1}^{t} c(\mathcal{S}, l_{a_j}) + c(\mathcal{S}, l_{a_i}), \text{ for } j \neq i.$$

■

Let us define the concept of *reducibility*, in particular which employs a polynomial-time transformation as defined below.

Definition 6 (Polynomial-Time Reduction):

We say that a language A is polynomial-time reducible to a language B ($A \leq_P B$), if $\exists$ a polynomial-time transformation f, which for every input

$$x \in A \leftrightarrow f(x) \in B.$$

The problem of finding motifs is modelled as a graph problem in [6]. Given a set of t sequences in MFP and a pattern length l, an edge weighted t-partite graph $G = (V, E, c_G)$ is obtained, where the problem is reduced to finding a maximum weighted clique on a t-partite graph. Note that the cost is c_G, so as not to be confused with our previous definition $c(\mathcal{S}, v)$. We revise the transformation in [6], instead of an edge weighted t-partite graph, we obtain a vertex weighted t-partite graph $G' = (V, E, c')$. The set of vertices is the same for both graphs G' and G. The transformation from the set of input instance of MFP is as follows.

Let l and $\mathcal{S} = \{S_1, S_2, \ldots, S_t\}$ be the set of given input instances for MFP. Each vertex $v \in V$ corresponds to an l-length string obtained from $\mathcal{S}$, i.e. we have $(n - l + 1)$ vertices from S_i and a total number of $(t(n-l+1))$ vertices for all t sequences. An edge e connects to vertices v and w if both vertices are not obtained from the same sequence S_i. The cost of each vertex given by c' is $c'(v) = c(\mathcal{S}, v)$, i.e.

equivalent to the computation of $c(\mathcal{S}, v)$ we've shown earlier where maximization is involved in the computation. This shows a reduction of MFP to Maximum Weighted Clique Problem (MWCP) on a vertex weighted graph. In fact, the cost of a clique that is equal to the sum of all its vertices is equal to the cost obtained from a starting position vector p using Equation 2. It is shown that MWCP has an exact reduction to Maximum Weighted Independent Set Problem (MWISP), since a clique on a graph is an independent set to the corresponding complement of the graph [10]. Moreover, in [10] the following lemma is presented.

Lemma 2: Maximum Weighted Independent Set Problem (MWISP) is *self-reducible*.

Since we proved earlier that mMFP is self-reducible and we know that there is a polynomial-time reduction from mMFP to MWISP, following from the above discussion, we are interested to know if the following theorem is true.

Theorem 1: If problem A is polynomial-time reducible to problem B($A \leq_P B$), and B is self-reducible, then A is self-reducible.

Proof:

Let $A = (\mathcal{D}_A, \mathcal{R}_A, cost_A, goal_A)$ and $B = (\mathcal{D}_B, \mathcal{R}_B, cost_B, goal_B)$ be two NPO problems. Let $I_A \in \mathcal{D}_A$, $SOL_A \in \mathcal{R}(I_A)$, where SOL_A is composed of atoms α_A. Similarly, let $I_B \in \mathcal{D}_B$, $SOL_B \in \mathcal{R}(I_B)$, where SOL_B is composed of atoms α_B.

Given that $A \leq_P B$, then by definition, there exist a polynomial-time computable function f such that for every instance I_A for problem A, $f(I_A)$ is an instance of problem B and for every solution SOL_A to A, $f(SOL_A) \in \mathcal{I}_\mathcal{B}$. Equivalently, as a decision problem, the polynomial-time reducibility implies

$$(I_A, SOL_A) \in (\mathcal{D}_A \times \mathcal{R}(\mathcal{I}_\mathcal{A}))$$

$$\leftrightarrow$$

$$(f(I_A), f(SOL_A)) \in (\mathcal{D}_B \times \mathcal{R}(\mathcal{I}_\mathcal{B}))$$

By definition of self-reducibility in Definition 5, if B is self-reducible then we can obtain a self-reduction function $\Delta_B(I_B, \alpha_B) = I_{\alpha_B}$ such that $|I_{\alpha_B}| < |I_B|$, and the following conditions hold

1) For $SOL_B \in \mathcal{R}(I_B)$ there is a corresponding $SOL_{\alpha_B} \cup \{\alpha_B\} \in \mathcal{R}(I_B|\alpha_B)$, where $SOL_{\alpha_B} \in \mathcal{R}(I_{\alpha_B})$.
2) For a subset of atoms $H_B \subseteq \mathcal{R}(I_{\alpha_B})$, $cost_B(I_B, H_B \cup \{\alpha_B\}) = cost_B(I_B, \alpha_B) + cost_B(I_{\alpha_B}, H_B)$.

We then need to show that problem A is also self-reducible, given that B is self-reducible. Then, there exists a self-reduction function $\Delta_A(I_A, \alpha_A) = I_{\alpha_A}$. Given the polynomial-time function f and the self-reduction function Δ_B, we realize $\Delta_A$using Δ_B through the following

$$\Delta_B(f(I_A), f(\alpha_A)) = f(I_{\alpha_A}),$$

which inherits the two conditions stated above. Moreover, the reduction function remains polynomial since f is polynomial-time computable. ■

IV. Reoptimization

Let us define a modification $\mathcal{M}$ for the input instances of mMFP. With the assumption that the length of the pattern we are looking for remains unchanged, the pair $(I, I') \in \mathcal{M}$ can be represented as $(\mathcal{S}, \mathcal{S}') \in \mathcal{M}$, where $\mathcal{S} = \{S_1, \ldots, S_{(t-1)}\}$ and $\mathcal{S}' = \mathcal{S} \cup \{S_t\}$. The modified instance $\mathcal{S}'$ is derived from the original instance by adding a new sequence S_t to $\mathcal{S}$. Note that, unlike the self reduction function Δ, the size of the modified instance may not be necessarily less than the size of the original instance, as what is the case here. Formally, the reoptimization variant of mMFP is defined as follows.

Definition 7 (mMFP Reoptimization ($\mathbf{R}_\mathcal{M}(mMFP)$)): INPUT: Original instance $(\mathcal{S}, l)$, solution p to $(\mathcal{S}, l)$, and a modified instance $\mathcal{S}'$ OUTPUT: Solution p' to $(\mathcal{S}', l)$, such that p' obtains the maximum $\overline{cost}(\mathcal{S}', p')$

A. Self Reduction Method

For self-reducible NPO problems, we can use the self reduction methods presented in [10]. The first method proposes an algorithm which employs a σ-approximation algorithm Alg. The general algorithm for any self-reducible NPO problem is also presented in [10]. Specifically for mMFP, S_{Alg} is shown in Algorithm 1.

Algorithm 1 Algorithm S_{Alg} employing algorithm Alg for mMFP

```
Input: Input instance $(\mathcal{S}, l)$, where $\mathcal{S} = \{S_1, S_2, \ldots, S_t\}$ such that $|S_i| = n$ and pattern length $l$

    for all $a_i \in Atoms((\mathcal{S}, l))$ do
        $p_{a_i} := Alg((\mathcal{S}_{a_i}, l), (i, a_i))$
    end for

Output: Solution $p_{a_i} \cup \{a_i\}$ with maximum $\overline{cost}(\mathcal{S}, p_{a_i} \cup \{a_i\})$
```

The algorithm outputs a starting position vector $p_{a_i} \cup \{a_i\}$, where $p_{a_i} \in \mathcal{R}(\mathcal{S}_{a_i})$, and it evaluates to the maximum $\overline{cost}(\mathcal{S}, p_{a_i} \cup \{a_i\})$ over all possible $a_i \in Atoms(\mathcal{S}, l)$. The total number of iterations is equal to $|Atoms(\mathcal{S}, l)| = t \cdot (n - l + 1)$. The running time of S_{Alg} for mMFP is equal to $(t \cdot (n - l + 1))$ times the running time of algorithm Alg. It is shown in [10] that the self reduction method S_{Alg}, whenever employed improves the approximation ratio σ as shown in the following lemma.

Lemma 3: If in any optimal solution $Opt \in R(I)$, there is an atom α with $cost(I, \alpha) \geq \delta Cost(I, Opt)$, then $\mathcal{S}_{Alg}$ is a $(\sigma - \delta(\sigma - 1))$-approximation algorithm.

Given an instance I for mMFP, the range of evaluating a solution and an atom are $0 \leq \overline{cost}(I, SOL) \leq lt^2$ and $0 \leq c(I, \alpha) \leq lt$ respectively, we can bound the cost of an atom α using the cost of the optimal solution for I, i.e. $cost(I, \alpha) \geq \delta cost(I, Opt)$, $lt \geq \delta lt^2$. Hence, S_{Alg} for mMFP is $(\sigma - \delta(\sigma - 1))$-approximation algorithm, for $\delta \leq 1/t$.

Note that using the defined reduction function Δ in Section II, the reduced instances $\mathcal{S}_{a_i}$ is a set of sequences obtained from $\mathcal{S} - \{S_i\}$. Regardless of the value of a_i for a specific sequence i, the reduced instance $\mathcal{S}_{a_i}$ remains the same.

Therefore, to guarantee that a simultaneous reduction always produce a modified instance, we restrict that i in a_i is distinct for the set of atoms to be removed. This observation is necessary to understand the next algorithm we will discuss.

Algorithm S^b_{Alg}, as shown below, is a generalization of S_{Alg} using a set of b atoms [10]. Suppose that in a given set of sequences, we already know several possible motif occurrences. Algorithm S^b_{Alg} takes advantage of the atoms corresponding to known motif occurrences, thus obtaining a much smaller instance for Alg to evaluate. The general algorithm using b atoms is presented in [10]. The corresponding algorithm for mMFP is shown below.

Algorithm 2 Algorithm S^b_{Alg} employing algorithm Alg for motif finding

```
Input: Set of sequences, pattern length $(\mathcal{S}, l)$, and integer $1 \leq b \leq t$
    for all $\{a_1, \ldots, a_b\} \subseteq Atoms((\mathcal{S}, l))$ do
        $\mathcal{S}_0 := \mathcal{S}$
        for $j := 1 \ldots b$ do
            $\mathcal{S}_j := \Delta(\mathcal{S}_{j-1}, a_j)$
        end for
        $\widetilde{SOL} := Alg(\mathcal{S}_b) \cup \{a_1, \ldots, a_b\}$
    end for
Output: Solution $\widetilde{SOL}$ with maximum $\overline{cost(\mathcal{S}, \widetilde{SOL})}$ over all considered $\{a_1, \ldots, a_b\} \subseteq Atoms(I)$
```

For mMFP, there is no need to explore all possible subset of b atoms, but only those that will modify the original instance b times. Moreover, if we will consider all possible subset $H \subseteq Atoms((\mathcal{S}, l))$, such that $|H| = b$, and $\forall\, \alpha \in H$, α is obtained from S_i, occurrence of any two atoms in the set H obtained from the same sequence contradicts our assumption that motifs occur exactly once per sequence. Hence, the set of atoms $\{a_1, \ldots, a_b\}$ in Algorithm 2, have the following restriction. All atoms a_i in the set should come from distinct sequences. For instance, if we let $b = t$, the restriction we impose lead to iterating over the set of all valid starting position vector in $\mathcal{S}$. Therefore, if we let $b = t$, the running time of Algorithm 2 becomes $(n - l + 1)^t$, which is the naive way of searching for the optimal solution. If that is the case, Alg will not be utilized in Algorithm 2, because $\mathcal{S}_b$ is always empty after b reductions. Note that, Algorithm 2 takes more iteration compared to Algorithm 1. Algorithm 2 has an advantage over Algorithm 1 when there exists a set of atoms which are known to be part of the solution. In such cases, we don't have to perform the first for-loop in Algorithm 2. Instead, we can already reduce the original instance given the set of atoms and obtain a solution for the reduced instance using Alg.

We say that $H \subseteq SOL \in \mathcal{R}(I)$ is $F(n)$-guessable for some instance I, if we can determine a set $\mathcal{G}$ with $|\mathcal{G}| \in O(F(n))$, such that $H \subseteq \mathcal{G}$ [10]. The set $\mathcal{G}$ is the set of guesses for H. A variant of Algorithm 2 is when we only iterate on a set of guesses $\mathcal{G}$ instead of all $Atoms((\mathcal{S}, l))$. Let us denote this algorithm $S^{\mathcal{G}}_{Alg}$. Based from Corollaries 4 and 5 in [10], algorithm $S^{\mathcal{G}}_{Alg}$ has the same approximation ratio with Algorithm 1. Also, if we can obtain a $F(n)$ guessable set of atoms H with cost $cost(I, H) \geq \delta cost(I, Opt)$, then the running time of $S^{\mathcal{G}}_{Alg}$ is $O(F(n) \cdot Time(Alg))$.

B. Modification

The previous algorithms presented are ideal for cases where we can identify or guess a set of expensive atoms. Without these atoms, the algorithms presented still need to exhaustively search for those atoms that can improve the approximation ratio of Alg. To further improve on these results, four major types of modifications were presented in [10]. These types of modification have corresponding greedy techniques on how to further improve the approximation ratio.

Note that, an original instance may be modified in several ways such that the set of feasible solutions are unchanged. These types of modification can be categorized as either *progressing* or *regressing* and *subtractive*. A progressive modification can be further categorize into two more categories, subtractive and additive, depending on whether its a maximization or minimization problem respectively [10]. Specifically for the reoptimization version in Definition 7, we argue that the type of modification $(\mathcal{S}, \mathcal{S}') \in \mathcal{M}$ is progressing and subtractive.

Lemma 4: The relation $(\mathcal{S}, \mathcal{S}') \in \mathcal{M}$ is progressing subtractive.

Proof: To support our claim, we need to show that $\mathcal{M}$ is progressing and is also subtractive. By definition of progressing modification in [10], we need to show that for every pair $(\mathcal{S}, \mathcal{S}') \in \mathcal{M}$,

$$\mathcal{R}(\mathcal{S}) \cap Atoms(\mathcal{S}') \subseteq \mathcal{R}(\mathcal{S}').$$

An atom is contained in $R(S')$, if it is in the ordered set

$$p' = \{(1, a_1), (2, a_2), \ldots, (t-1, a_{(t-1)}), (t, a_t)\}.$$

Note that, $\mathcal{R}(\mathcal{S}) \cap Atoms(\mathcal{S}')$ contains atom from the set of all feasible solution of $\mathcal{S}$, i.e. atom is in the ordered set $p = \{(1, a_1), (2, a_2), \ldots, (t-1, a_{(t-1)}\}$ and those that are contained in the set of atoms of the modified instance. Since all atoms in $\mathcal{R}(\mathcal{S})$ belongs to $Atoms(\mathcal{S}')$, we can ignore $Atoms(\mathcal{S}')$ in the relation and remove the possibility that the two sets may be equal. Hence, $\mathcal{R}(\mathcal{S}) \subset \mathcal{R}(\mathcal{S}')$.

By definition of subtractive progressing modification in [10], we need to show that there is an optimal solution to the modified instance $\mathcal{S}'$ which, after removing a part of it, becomes feasible for $\mathcal{S}$ and not less valuable: $\exists\ OPT' \in OPTIMA(\mathcal{S}')$, $\exists\ H' \subseteq OPT'$, and $OPT' \in \mathcal{R}(\mathcal{S})$ such that

$$cost(\mathcal{S}, OPT' \setminus H') \geq cost(\mathcal{S}', OPT' \setminus H')$$

Given that we have an optimal solution p'_{opt} for the modified instance $\mathcal{S}'$, if we remove some atoms H from it to become feasible for the original instance $\mathcal{S}$, the computed cost is equal for both instances.

$$\overline{cost}(\mathcal{S}, p'_{opt} \setminus H) = \overline{cost}(\mathcal{S}', p'_{opt} \setminus H)$$

In fact, for any set of atom H, it follows that $\overline{cost}(\mathcal{S}, H) = \overline{cost}(\mathcal{S}', H)$, regardless of the given input instance because computation of $\overline{cost}$ iterates over all atoms in H. ■

We say that a cost function is *pseudo-additive* if for any disjoint set of atoms H, $D \subseteq Atoms(I)$,

$$cost(I, H \cup D) \leq cost(I, H) + cost(I, D).$$

The cost function in our mMFP is additive, thus pseudo-additive because

$$\overline{cost}(\mathcal{S}, H \cup D) = \overline{cost}(\mathcal{S}, H) + \overline{cost}(\mathcal{S}, D),$$
$$\sum_{\alpha \in H \cup D} c(\mathcal{S}, \alpha) = \sum_{\beta \in H} c(\mathcal{S}, l_\alpha) + \sum_{\gamma \in D} c(\mathcal{S}, l_\gamma).$$

Given that we have categorized our modification $\mathcal{M}$ for mMFP as progressing subtractive (as claimed in Lemma 4) and its cost function as pseudo-additive. We have the following theorem, based from Theorem 1 in [10].

Theorem 2: There is a $1/(2-\sigma)$-approximation algorithm for $\mathbf{R}_{\mathcal{M}}(mMFP)$ that runs in $O(F(n)) \cdot Time(Alg)$ if H is $F(n)$-guessable.

The $1/(2-\sigma)$-approximation algorithm uses the given solution as a greedy solution for the modified instance. Based from the greedy algorithm in [10], it either returns the best among the given solution p or the solution obtained from using $S^{\mathcal{G}}_{Alg}(\mathcal{S}')$. Given that the set of atoms in $\mathcal{G}$ is $F(n)$-guessable, the running time for the algorithm solving mMFP is $O(F(n)) \cdot Time(Alg)$.

V. Conclusions

In this paper we have shown the self-reducibility of the motif finding problem. We also show that any problem that is polynomial-time reducible to a self-reducible problem is also self-reducible as shown in Theorem 1. Given that mMFP is self-reducible, we presented an algorithm which improves the goodness of the solution of a σ-approximation algorithm Alg to $1/(2-\sigma)$-approximation algorithm for $\mathbf{R}_{\mathcal{M}}(mMFP)$.

References

[1] D Bilò and A Zych. New advances in reoptimizing the minimum steiner tree problem. *Mathematical Foundations of Computer Science 2012*, pages 184–197, 2012.

[2] Davide Bilò, Hans-Joachim Böckenhauer, and Dennis Komm. Reoptimization of the Shortest Common Superstring Problem. *Combinatorial Pattern Matching*, 293:1–14, 2009.

[3] Juraj Hromkovič and Hans-Joachim Böckenhauer. On the Hardness of Reoptimization. *In Proc. of the 34th International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM 2008)*, 4910, 2008.

[4] Neil Jones and Pavel Pevzner. *An Introduction to Bioinformatics Algorithms*. 2004.

[5] Tobias Momke. *Algorithmic Approaches for Solving Hard Problems : Approximation and Complexity*. PhD thesis, Swiss Federal Institute of Technology Zurich, 2009.

[6] Pavel A Pevzner and Sing-Hoi Sze. Combinatorial approaches to finding subtle signals in DNA sequences. *Proceedings International Conference on Intelligent Systems for Molecular Biology ISMB International Conference on Intelligent Systems for Molecular Biology*, 8:269–278, 2000.

[7] MW Schäffter. Scheduling with forbidden sets. *Discrete Applied Mathematics*, 72:155–166, 1997.

[8] VV Vazirani. *Approximation Algorithms*. Springer-Verlag, Berlin, 2001.

[9] David P Williamson and David B Shmoys. *The Design of Approximation Algorithms*. Cambridge University Press, 2010.

[10] Anna Zych. *Reoptimization of NP-hard Problems*. PhD thesis, ETH Zurich, 2012.