Approximation of greedy algorithms for Max-ATSP, Maximal Compression, and Shortest Cyclic Cover of Strings
Bastien Cazaux, Eric Rivals

To cite this version:
Bastien Cazaux, Eric Rivals. Approximation of greedy algorithms for Max-ATSP, Maximal Compression, and Shortest Cyclic Cover of Strings. RR-14001, 2013, pp.12. lirmm-00932660

HAL Id: lirmm-00932660
https://hal-lirmm.ccsd.cnrs.fr/lirmm-00932660
Submitted on 17 Jan 2014

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Approximation of greedy algorithms for Max-ATSP, Maximal Compression, and Shortest Cyclic Cover of Strings

Bastien Cazaux and Eric Rivals
L.I.R.M.M. & Institut Biologie Computationnelle, University of Montpellier II, CNRS U.M.R. 5506
161 rue Ada, F-34392 Montpellier Cedex 5, France
cazaux,rivals@lirmm.fr

20th October 2013

Abstract

Given a directed graph $G$ with weights on its arcs, the Maximum Asymmetric Travelling Salesman Problem (Max-ATSP) asks for a Hamiltonian path of maximum weight covering $G$. Max-ATSP, a central problem in computer science, is known to NP-hard and hard to approximate. In the general case, when the Triangle Inequality is not satisfied, the best approximation ratio known to date equals $2/3$. Now consider the Overlap Graph for a set of finite words $P := \{s_1, \ldots, s_p\}$: the directed graph in which an arc links two words with a weight equals to the length of their maximal overlap. When Max-ATSP is applied to the Overlap Graph, it solves the Maximal Compression or Shortest Superstring problem, where one searches for a string of minimal length having each input word as a substring. Again these problems are hard to approximate. Both for Max-ATSP and for Maximal Compression, good approximation algorithms use a cover of the graph by a set of cycles or of the words by a set of cyclic strings. These questions are known as Maximal Directed Cyclic Cover (MDCC) and as Shortest Cyclic Cover of Strings (SCCS), and can be solved in polynomial time. However, among these four problems, the approximation ratio achieved by a simple greedy algorithm is known only for Maximal Compression. In a seminal but complex proof, Tarhio and Ukkonen showed that it achieves $1/2$ compression ratio. Taking advantage of the power of subset systems, we investigate the approximation of associated greedy algorithms for these four problems, and show they reach a ratio of $1/3$ for Max-ATSP, $1/2$ for Maximal Compression and for Maximal Cyclic Cover, and gives an exact solution for the Shortest Cyclic Cover of Strings. The proof for Maximal Compression is simpler than known ones. For these problems, greedy algorithms are easier to implement and often faster than existing approximation algorithms, an important fact since these problems have practical applications, for instance in data compression and computational biology.

1 Introduction

Given a set of words $P = \{s_1, \ldots, s_p\}$ over a finite alphabet, the Shortest Superstring (SS) or Maximal Compression (MC) problems ask for a shortest string $u$ that contains each of the given words as a substring. It is a key problem in data compression and in bioinformatics, where it models the question of sequence assembly. Indeed, sequencing machines yield only short reads that need to be aggregated according to their overlaps to obtain the whole sequence of the target molecule [6]. Two measures can be optimised for SS: either the length of the superstring is minimised, or the compression is maximised (i.e., $|S| - |u| := \sum_{i \in S} |s_i| - |u|$). Unfortunately, even for a binary alphabet, SS is NP-hard [5] and MAX-SNP-hard relative to both measures [3]. Among many approximation algorithms, the best known fixed ratios are $22/21$ for the superstring [9] and $2/3$ for the compression [7, 10]. A famous conjecture states that a simple, greedy agglomeration algorithm achieves a ratio $2$ for the superstring measure, while it is known to approximate the maximum compression with ratio $1/2$, but the later proofs are quite complex involving many cases of overlaps [12, 13]. The best approximation algorithms use the Shortest Cyclic Cover of Strings (SCCS) as a procedure,
Greedy approximation, Max-ATSP & compression  
B. Cazaux & E. Rivals

which asks for a set of cyclic strings of total minimum length that collectively contain the input words as substrings. The SCCS problem can be solved in polynomial time in \(|S|\) [11, 3].

These problems on strings can be viewed as problems on the Overlap Graph, where the input words are the nodes, and an arc represents the asymmetric maximum overlap between two words. Figure 2 on p. 12 displays an example of overlap graph. Covering the Overlap Graph with either a maximum weight Hamiltonian path or a maximum weight cyclic cover gives a solution for the problems of Maximal Compression or of Shortest Cyclic Cover of Strings, respectively. This expresses the relation between the Maximum Asymmetric Travelling Salesman Problem (Max-ATSP) and Maximal Compression on one hand, as well as between Maximum Directed Cyclic Cover (Max-DCC) and Shortest Cyclic Cover of Strings on the other. Both Max-ATSP and Max-DCC have been extensively studied as essential computer science problems.

Hereditary systems were introduced recently to investigate the approximation performances of greedy algorithms in a unified framework [8]. With their help, we investigate the approximation achieved by greedy algorithms on these four problems and provide the results mentioned in the abstract. After introducing the required notation and concepts, we study the case of the Max-ATSP and Max-DCC problems in Section 2, then we focus on the Maximal Compression problem in Section 3, and state the results regarding Shortest Cyclic Cover of Strings in Section 3.2, before concluding.

1.1 Sets, strings, and overlaps.

We denote by \(\#(\Lambda)\) the cardinality of any finite set \(\Lambda\).

An alphabet \(\Sigma\) is a finite set of letters. A linear word or string over \(\Sigma\) is a finite sequence of elements of \(\Sigma\). The set of all finite words over \(\Sigma\) is denoted by \(\Sigma^*\), and \(\varepsilon\) denotes the empty word. For a word \(x\), \(|x|\) denotes the length of \(x\). Given two words \(x\) and \(y\), we denote by \(xy\) the concatenation of \(x\) and \(y\). For every \(1 \leq i \leq j \leq |x|\), \(x[i : j]\) denotes the substring \(x[i : i+1] \ldots x[j]\).

A cyclic string or necklace is a finite string in which the last symbol precedes the first one. It can be viewed as a linear string written on a torus with both ends joined.

Overlaps and agglomeration  
Let \(s, t, u\) be three strings of \(\Sigma^*\). We denote by \(ov(s, t)\) the maximum overlap from \(s\) over \(t\); let \(r\) be prefix of \(s\) such that \(s = r \cdot ov(s, t)\), then we denote the agglomeration of \(s\) over \(t\) by \(s \oplus t := rt\). Note that neither the overlap nor the agglomeration are symmetrical. Clearly, one has \((s \oplus t) \oplus (t \oplus u) = (s \oplus t) \oplus u\).

Example: Let be three words \(abbaa, baabb\) and \(aabb\). \(ov(abbaa, baabb) = baa\) and \(abbaa \oplus baubb = abbaabb\). Considering possible agglomerations of these words, we get \(w_1 = abbaa \oplus baabb \oplus aabba = abbaabb \oplus aabba\), \(w_2 = aabba \oplus abbaa \oplus baabb = aabbaa \oplus baabb = aabbaabb\) and \(w_3 = baabb \oplus abbaa \oplus aabba = baabbbaa \oplus aabbaa = baabbbaabba\). Thus, \(|w_1| = |pr(abbaa, baabb)| + |pr(baabb, aabba)| + |aabba| = |ab| + |ba| + |aabba| = 2 + 1 + 5 = 8, \(|S| - |w_1| = 15 - 8 = 7\) and \(ov(abbaa, baabb)\) + \(ov(baabb, aabba)\) = \(baa\) + \(aabba\) = \(3 + 4 = 7\).

1.2 Notation on graphs

We consider directed graphs with weighted arcs. A directed graph \(G\) is a pair \((V_G, E_G)\) comprising a set of nodes \(V_G\), and a set \(E_G\) of directed edges called arcs. An arc is an ordered pair of nodes.

Let \(w\) be a mapping from \(E_G\) onto the set of non negative integers (denoted \(\mathbb{N}\)). The weighted directed graph \(G := (V_G, E_G, w)\) is a directed graph with the weights on its arcs given by \(w\).

A route of \(G\) is an oriented path of \(G\), that a subset of \(V_G\) forming a chain between two nodes at its extremities. A cycle of \(G\) is a route of \(G\) where the same node is at both extremities. The weight of a route \(r\) equals the sum of the weights of its arcs. For simplicity, we extend the mapping \(w\) and let \(w(r)\) denote the weight of \(r\).

We investigate the performances of greedy algorithms for different types of covers of a graph, either by a route or by a set of cycles. Let \(X\) be a subset of arcs of \(V_G\). \(X\) covers \(G\) if and only if each vertex \(v\) of \(G\) is the extremity of an arc of \(X\).
1.3 Subset systems, extension, and greedy algorithms

A greedy algorithm builds a solution set by adding selected elements from a finite universe to maximise a given measure. In other words, the solution is iteratively extended. Subset systems, which are also called hereditary systems in French, are useful concepts to investigate how greedy algorithms can iteratively extend a current solution to a problem. A subset system is a pair $(E, L)$ comprising a finite set of elements $E$, and $L$ a family of subsets of $E$ satisfying two conditions:

1. $(HS1)$ $L \neq \emptyset$.
2. $(HS2)$ If $A' \subseteq A$ and $A \in L$, then $A' \in L$.

A subset system is said to be $k$-extendible if for all $C \in L$ and $x \not\in C$ such that $C \cup \{x\} \in L$, and for any extension $D$ of $C$, there exists a subset $Y \subseteq D \setminus C$ with $|Y| \leq k$ satisfying $D \setminus Y \cup \{x\} \in L$.

The greedy algorithm associated with $(E, L)$ and a weight function $w$ is presented in Algorithm 1. Checking whether $F \cup \{e_i\} \in L$ consists in verifying the system’s conditions. In the sequel of this paper, we will simply use “the greedy algorithm” to mean the greedy algorithm associated to a subset system, if the system is clear from the context.

Theorem 1 (Mestre [8]). Let $(E, L)$ be a $k$-extendible subset system. The associated greedy algorithm defined for the problem $(E, L)$ with weights $w$ gives a $\frac{1}{k}$ approximation factor.

**Algorithm 1:** The greedy algorithm associated with the subset system $(E, L)$ and weight function $w$.

```
Input : $(E, L)$
The elements $e_i$ of $E$ sorted by increasing weight: $w(e_1) \leq w(e_2) \leq \ldots \leq w(e_n)$
$F \leftarrow \emptyset$
for $i = 1$ to $n$
  if $F \cup \{e_i\} \in L$ then $F \leftarrow F \cup \{e_i\}$
return $F$
```

1.4 Definitions of problems and related work.

1.4.1 Graph covers

Let $G := (V_G, E_G, w)$ be a weighted directed graph.

The well known Hamiltonian path problem on $G$ requires that the cover is a single path, while the Cyclic Cover problem searches for a cover made of cycles. We consider the weighted versions of these two problems, where the solution must maximise the weight of the path or the sum of the weights of the cycles, respectively. In a general case, the graph is not symmetrical, and the weight function does not satisfy the Triangle inequality. When a Hamiltonian path is searched for, the problem is known as the Maximum Asymmetric Travelling Salesman Problem or Max-ATSP for short.

**Definition 1.1** (Max-ATSP). Max-ATSP is the problem where one searches for a maximum weight Hamiltonian path on $G$.

Max-ATSP is an important and well studied problem. It is known to be NP-hard and hard to approximate (precisely, Max-SNP hard). The best known approximation ratio of $2/3$ is achieved by using a rounding technique on a Linear Programming relaxation of the problem [7]. However, the approximation ratio obtained by a simple greedy
algorithm remains an interesting open question, especially, since other approximation algorithms are usually less efficient than a greedy one. As later explained, Max-ATSP is strongly related to the Shortest Superstring and the Maximal Compression problems on strings.

If a set of cycles is needed as a cover the graph, the problem is called Maximum Directed Cyclic Cover. In the general setup, cycles made of singletons are allowed in a solution.

**Definition 1.2** (Max Directed Cyclic Cover). Maximum Directed Cyclic Cover is the problem where one searches for a set of cycles of maximum weight that collectively cover G.

To our knowledge, the performance of a greedy algorithm for Maximum Directed Cyclic Cover (Max-DCC) has not yet been established, although variants of Max-DCC with binary weights or with cycles of predefined lengths have been studied [1, 2].

### 1.4.2 Superstring and Maximal Compression

**Definition 1.3** (Superstring). Let \( P = \{s_1, s_2, \ldots, s_p\} \) be a set of \( p \) strings of \( \Sigma^* \). A superstring of \( P \) is a string \( s' \) such that \( s_i \) is a substring of \( s' \) for any \( i \) in \([1, p]\) .

Let us denote the sum of the lengths of the input strings by \( |S| := \sum_{i \in S} |s_i| \). For any superstring \( s' \), there exists a set \( \{i_1, \ldots, i_p\} = \{1, \ldots, n\} \) such that \( s' = s_{i_1} \oplus s_{i_2} \oplus \ldots \oplus s_{i_p} \), and then \( |S| - |s'| = \sum_{j=1}^{p-1} |ow(s_{i_j}, s_{i_{j+1}})| \).

**Definition 1.4** (Shortest Superstring Problem (SS)). **Input**: Let \( p \) be a positive integer and \( P := \{s_1, s_2, \ldots, s_p\} \) be a set of \( p \) strings over \( \Sigma \). **Question**: Find \( s' \) a superstring of \( P \) of minimal length.

Two approximation measures can be optimised:

- the length of the obtained superstring, that \( |s'| \), or
- the compression of the input strings achieved by the superstring: \( |S| - |s'| \).

The corresponding approximation problems are termed Shortest Superstring in the first case, or Maximal Compression in the second. These two problems have applications in data compression and in computational biology. Indeed, finding a superstring of a large set of short input sequences gives a solution to the DNA shotgun sequencing problem, which is encountered when one wishes to determine the complete nucleotidic sequence of a long gene, a chromosome, or a whole genome. The question of DNA assembly has attracted a lot of attention since the launch of Next Generation Sequencing technologies, which have revolutionised the process of DNA acquisition. These technologies have permitted an exponential increase in throughput, making acute the need for simple and efficient assembly algorithms.

Both the Maximal Compression and Shortest Superstring problems are NP-hard [5] and Max-SNP hard [3]. Numerous, complex algorithms have been designed for them, or their variants. Many are quite similar and use a procedure to find a Maximum Directed Cyclic Cover of the input strings. The best known approximation ratio for the Shortest Superstring was obtained in 2012 and equals \( 2 \frac{11}{13} \) [9], although an optimal ratio of 2 has been conjectured in the 80’s [12, 3].

For the Maximal Compression problem, two algorithms give a ratio of \( \frac{3}{2} \) [7, 10]. A seminal work gave a proof of an approximation ratio of 1/2 by an algorithm that iteratively updates the input set by agglomerating two maximally overlapping strings until one string is left [12]. This algorithm was termed greedy but does not correspond to a greedy algorithm for it modifies the original input set. We demonstrate in Appendix that this algorithm yields the same result than a greedy algorithm defined for an appropriate subset system. Another proof of this ratio was given in [13]. Both proofs are quite intricate and include many subcases [12]. Thanks to subset systems, we provide a much simpler proof of this approximation ratio for Maximal Compression by a greedy algorithm, as well as an optimal and polynomial greedy algorithm for the problem of Max Cyclic Covers on Strings.
**Definition 1.5** (Shortest Cyclic Cover of Strings (SCCS)). **Input:** Let \( p \in \mathbb{N} \) and \( P := \{s_1, s_2, \ldots, s_p\} \) be a set of \( p \) linear strings over \( \Sigma \). **Question:** Find a set of cyclic strings of minimal cumulated length such that any input \( s_i \) with \( 1 \leq i \leq p \), is a substring of at least one cyclic string.

Several approximation algorithms for the Shortest Superstring problem uses a procedure to solve SCCS on the instance, which is based on a modification of a polynomial time algorithm for the assignment problem \([11, 3, 6]\). This further indicates the importance of SCCS.

Both the Maximum Compression and the Shortest Cyclic Cover of Strings problems can be expressed as a cover of the Overlap Graph. In the Overlap Graph, the vertices represent the input strings, and an arc links \( s_i \) to \( s_j \) with weight \(|s_i \cap s_j|\). Hence, the overlap graph is a complete graph with null or positive weights. An Hamiltonian path of this graph provides a permutation of the input strings; by agglomerating these strings in the order given by the permutation one obtains a superstring of \( P \). Hence, the maximum weight Hamiltonian path induces a superstring that accumulates an optimal set of maximal overlaps, in other words a superstring that achieves maximal compression on \( P \). Thus, a \( p \) approximation for Max-ATSP gives the same ratio for Maximum Compression. The same relation exists between the Shortest Cyclic Cover of Strings and Maximum Directed Cyclic Cover on graphs. Indeed, SCCS optimises \(|P| - \sum_{c \in E} |c_j|\), where \( c_j \) is a cyclic string in the solution, and Max-DCC optimises the cumulated weight of the cycles of \( G \). With the Overlap Graph, a minimal cyclic string is associated to each graph cycle by agglomerating the input strings in this cycle. Thus, the cumulated weight of a set of graph cycles corresponds to compression achieved by the set of induced cyclic strings. In other words, Shortest Cyclic Cover of Strings could also be called the Maximal Compression Cyclic Cover of Strings problem (and seen as a maximisation problem). Here again, the performance of a greedy algorithm for the Shortest Cyclic Cover of Strings problem remains open \([14]\).

## 2 Maximum Asymmetric Travelling Salesman and Directed Cyclic Cover Problems

Let \( w \) be a mapping from \( E_G \) onto the set of non negative integers and let \( G := (V_G, E_G, w) \) be a directed graph with the weights on its arcs given by \( w \). We first define a subset system for Max-ATSP and its accompanying greedy algorithm.

**Definition 2.1.** We define the pair \((E_G, L_S)\) where \( E_G \) is the arc set of \( G \) and \( L_S \) the set of subsets, \( F \), of \( E_G \) satisfying:

\[
\begin{align*}
(L1) & \quad \forall x, y \text{ and } z \in V_G, (x, z) \text{ and } (y, z) \in F \text{ implies } x = y, \\
(L2) & \quad \forall x, y \text{ and } z \in V_G, (z, x) \text{ and } (z, y) \in F \text{ implies } x = y, \\
(L3) & \quad \text{for any } r \in \mathbb{N}^*, \text{ there does not exist any cycle } ((x_1, x_2), \ldots, (x_{r-1}, x_r), (x_r, x_1)) \text{ in } F, \text{ where } \forall k \in \{1, \ldots, r\}, x_k \in V_G.
\end{align*}
\]

**Remark:**

- In other words, for a subset \( F \) of \( E_G \), Condition \((L1)\) (resp. \((L2)\)) allows only one ingoing (resp. outgoing) arc for each vertex of \( G \).
- For all \( F \in L_S \) and for any \( v \in V_G \), the arc \((v, v)\) cannot belong to \( F \), by Condition \((L3)\) for \( r = 1 \).
- If in condition \((L3)\), one changes the set of forbidden values for \( r \), the subset system addresses a different problem. As the proofs in this section do not depend of \( r \), all results remain valid for these problems as well. For instance, with \( r \in \{1\} \), only cycles of length one are forbidden; the solution is either a maximal path or cyclic cover with cycles of length larger than one. The \( 1/3 \) approximation ratio obtained in Theorem \( 4 \) remains valid. We will consider later the case where all cycles are allowed (i.e., \( r \in \emptyset \)).

**Proposition 2.** \((E_G, L_S)\) is a subset system.
Proof. For (HS1), it suffices to note that \( \emptyset \in \mathcal{L}_6 \). For (HS2), we must show that each subset of an element of \( \mathcal{L}_8 \) is an element of \( \mathcal{L}_8 \). This is true since Conditions (L1), (L2), and (L3) are inherited by any subset of an element of \( \mathcal{L}_8 \).  

Next proposition shows that the defined subset system is 3-extendible.

**Proposition 3.** \((E_G, \mathcal{L}_8)\) is 3-extendible.

**Proof.** Let \( C \subseteq \mathcal{L}_8 \) and \( e \notin C \) such that \( C \cup \{ e \} \in \mathcal{L}_8 \). Let \( D \subseteq E \) be an extension of \( C \). One must show that there exists a subset \( Y \subseteq D \setminus C \) with \#(\( Y \)) \( \leq 3 \) such that \( D \setminus Y \cup \{ e \} \) belongs to \( \mathcal{L}_8 \).

As \( e \in E_G \), there exists \( x \) and \( y \) such that \( e = (x, y) \). Let \( Y \) be the set of elements of \( D \setminus C \) of the form \((x, z), (z, y), \) and \((z, x)\) for any \( z \in V_G \) where \((z, x)\) belongs to a cycle in \( D \cup \{ x \} \). As \( D \setminus C \) is 3-extendible and satisfies conditions (L1) and (L2), we know that for each \( z \in V_G \setminus \{ x, y \} \), the arcs \((x, z)\) and \((z, y)\) are not in \( C \).

By the definition of \( Y \), for each \( z \in V_G \), we have that \((x, z)\) and \((z, y)\) are not in \( C \). Therefore, for all \( z \in V_G \), \((x, z)\) and \((z, y)\) do not generate a cycle in \( C \cup \{ e \} \). Hence, \( D \setminus Y \cup \{ e \} \) satisfies conditions (L1) and (L2).

Now assume that \( D \setminus Y \cup \{ e \} \) violates Condition (L3). As \( D \subseteq \mathcal{L}_8 \), \( D \) satisfies condition (L3) and \( D \setminus Y \) too. The only element who can generate a cycle is \( e \). As \( C \cup \{ e \} \subseteq \mathcal{L}_8 \), \( e \) does not generate a cycle in \( C \cup \{ e \} \), which implies that it generates a cycle in \( D \setminus (C \cup Y) \). Hence, there exists \( z \in V_G \) such that \((z, x) \in D \setminus (C \cup Y) \), which contradicts the definition of \( Y \).  

Now we derive the approximation ratio of the greedy algorithm for Max-ATSP.

**Theorem 4.** The greedy algorithm of \((E_G, \mathcal{L}_8)\) yields a \( \frac{1}{3} \) approximation ratio for Max-ATSP.

**Proof.** By Proposition 3, \((E_G, \mathcal{L}_8)\) is 3-extendible. A direct application of Mestre’s theorem (Theorem 1) yields the \( \frac{1}{3} \) approximation ratio for Max-ATSP.

**Case of the Maximum Directed Cyclic Cover problem.** If in condition (L3) we ask that \( r = \emptyset \), (L3) is not a constraint anymore and all cycles are allowed. This defines a new subset system, denote by \((E_G, \mathcal{L}_C)\). As in the proof of Proposition 3, it suffices now to set \( Y := \{ (x, z), (z, y) \} \) (one does not need to remove an element of a cycle), and thus \#(\( Y \)) \( \leq 2 \). It follows that \((E_G, \mathcal{L}_C)\) is 2-extendible and that the greedy algorithm achieves a 1/2 approximation ratio for the Maximum Directed Cyclic Cover problem.

## 3 Maximal Compression and Shortest Cyclic Cover of Strings

Blum and colleagues [3] have designed an algorithm called greedy that iteratively constructs a superstring for both the Shortest Superstring and Maximal Compression problems. As mentioned in introduction, this algorithm is not a greedy algorithm per se. Below, we define a subset system corresponding to that of Max ATSP for the Overlap Graph, and study the approximation of the associated greedy algorithm. Before being able to conclude on the approximation ratio of the greedy algorithm of [3], we need to prove that greedy computes exactly the same superstring as the greedy algorithm of the subset system of Definition 3.1. This proof is given in Appendix. Knowing that these two algorithms are equivalent in terms of output, the approximation ratio of Theorem 7 is valid for both of them.

From now on, let \( P := \{ s_1, s_2, \ldots, s_p \} \) be a set of \( p \) strings of \( \Sigma^* \).

The subset system for Maximal Compression is similar to that of Max-ATSP. For any two strings \( s, t \), \( s \odot t \) represents the maximum overlap of \( s \) over \( t \).\footnote{\( s \odot t \) is a word.} We set \( E_P := \{ s_i \odot s_j \mid s_i, s_j \in P \} \). Hence, \( E_P \) is the set of maximum overlaps between any two words of \( S \).

**Definition 3.1 (Subset system for Maximal Compression).** Let \( \mathcal{L}_P \) as the set of \( F \subseteq E_P \) such that:

---

Theorem 7. The approximation ratio of the greedy algorithm for the Maximal Compression problem is $\frac{1}{2}$.

Proof. To prove this ratio, we revisit the proof of Theorem 1 in [8].

Let $x_1, x_2, \ldots, x_l$ denote the elements in the order in which the greedy algorithm includes them in its solution $F$, and let $F_0 := \emptyset, \ldots, F_l$ denote the successive values of the set $F$ during the algorithm, in other words $F_i := F_{i-1} \cup \{x_i\}$ (see Algorithm 1 on p. 3). The structure of the proof is first to show for any element $x_i$ incorporated by the greedy algorithm, the inequality $w(OPT(F_{i-1})) \leq w(OPT(F_i)) + w(x_i)$, and second, to reason by induction on the sets $F_i$ starting with $F_0$.

One knows that $OPT(F_0)$ is an extension of $F_0$. By the greedy algorithm and by the definitions of $F_{i-1}$ and $x_i$, one gets $F_{i-1} \cup \{x_i\} \in L_{EP}$. As $x_i \in E_P$, there exist $x_p$ and $s_p$ such that $x_i = s_p \cap s_p$. Like in the proof of Proposition 3, let $Y_i$ denote the subset of elements of $OPT(F_i) \setminus F_{i-1}$ of the form $s_p \cap s_k$ or $s_k \cap s_p$, where $s_k \cap s_p$ belongs to a cycle in $OPT(F_{i-1}) \cup \{x_i\}$. Thus, $OPT(F_{i-1}) \setminus Y_i \cup \{x_i\} \in L_{EP}$, and

$$w(OPT(F_{i-1})) = w(OPT(F_{i-1}) \setminus Y_i \cup \{x_i\}) + w(Y_i) - w(x_i),$$

$$\leq w(OPT(F_i)) + w(Y_i) - w(x_i).$$
Indeed, \(w(OPT(F_{i-1}) \setminus Y_i \cup \{x_i\}) \leq w(OPT(F_i))\) because \(OPT(F_{i-1}) \setminus Y_i \cup \{x_i\}\) is an extension of \(F_{i-1} \cup \{x_i\}\) and because \(OPT(F_i)\) is an extension of maximum weight of \(F_{i-1} \cup \{x_i\}\).

Now let us show by contraposition that for any element \(y \in Y_i\), \(w(y) \leq w(x_i)\). Assume that there exists \(y \in Y_i\) such that \(w(y) > w(x_i)\). As \(y \notin F_{i-1}\), \(y\) has already been considered by the greedy algorithm and not incorporated in the \(F\).

Hence, there exists \(j \leq i\) such that \(F_j \cup \{y\} \notin L_\rho\), but \(F_j \cup \{y\} \subseteq OPT(F_{i-1}) \in L_\rho\), which is a contradiction. Thus, we obtain \(w(y) \leq w(x_i)\) for any \(y \in Y_i\).

Now we know that \(#(Y_i)\leq 3\). Let us inspect two subcases.

**Case 1 : \(#(Y_i)\leq 2\).
We have \(w(Y) \geq 2w(x_i)\), hence \(w(OPT(F_{i-1})) \leq w(OPT(F_i)) + w(x_i)\).

**Case 2 : \(#(Y_i) = 3\).
There exists \(s_k\) and \(s_{k'}\) such that \(s_p \circ s_{k'}\) and \(s_k \circ s_o\) are in \(Y_i\). By Lemma 5, we have \(w(x_i) + w(s_k \circ s_{k'}) \geq w(s_p \circ s_{k'}) + w(s_k \circ s_o)\). As \(s_p \circ s_{k'}\) and \(s_k \circ s_o\) belong to \(OPT(F_{i-1})\), one deduces \(s_k \circ s_{k'} \notin OPT(F_{i-1})\).

We get \(OPT(F_{i-1}) \setminus Y_i \cup \{x_i, s_k \circ s_{k'}\} \in L_\rho\). Indeed, as \(Y_i \subseteq OPT(F_{i-1})\), neither a right overlap of \(s_k\) nor a left overlap of \(s_{k'}\) can belong to \(OPT(F_{i-1})\). Furthermore, adding \(s_k \circ s_{k'}\) to \(OPT(F_{i-1}) \setminus Y_i \cup \{x_i\}\) cannot create a cycle, since otherwise a cycle would have already existed in \(OPT(F_{i-1})\). This situation is illustrated in Figure 1.

We have \(w(OPT(F_{i-1}) \setminus Y_i \cup \{x_i, s_k \circ s_{k'}\}) \leq w(OPT(F_{i-1} \cup \{x_i, s_k \circ s_{k'}\}))\), because \(OPT(F_{i-1}) \setminus Y_i \cup \{x_i, s_k \circ s_{k'}\}\) is an extension of \(F_{i-1} \cup \{x_i, s_k \circ s_{k'}\}\) and \(OPT(F_{i-1} \cup \{x_i, s_k \circ s_{k'}\})\) is a maximum weight extension of \(F_{i-1} \cup \{x_i, s_k \circ s_{k'}\}\). As \(w(OPT(F_{i-1} \cup \{x_i, s_k \circ s_{k'}\})) \leq w(OPT(F_{i-1} \cup \{x_i\}))\), by Lemma 6 one gets:

\[
w(OPT(F_{i-1})) = w(OPT(F_{i-1}) \setminus Y_i \cup \{x_i, s_k \circ s_{k'}\}) + w(Y_i) - w(x_i) - w(s_k \circ s_{k'}),
\]

\[
\leq w(OPT(F_{i-1} \cup \{x_i, s_k \circ s_{k'}\})) + w(Y_i) - w(x_i) - w(s_k \circ s_{k'}),
\]

\[
\leq w(OPT(F_i)) + w(Y_i) - w(x_i) - w(s_k \circ s_{k'}).
\]

As \(Y_i = \{s_p \circ s_{k'}, s_k \circ s_o, s_{k'} \circ s_p\}\), one obtains

\[
w(OPT(F_{i-1})) \leq w(OPT(F_i)) - w(s_k \circ s_{k'}) + w(Y_i) - w(x_i),
\]

\[
\leq w(OPT(F_i)) - w(s_k \circ s_{k'}) + w(s_p \circ s_{k'}) + w(s_k \circ s_o) + w(s_{k'} \circ s_p) - w(x_i).
\]

\[
w(OPT(F_i)) + w(s_{k'} \circ s_p)
\]

\[
w(OPT(F_i)) + w(x_i).
\]

**Case 1**

Remembering that \(OPT(\emptyset)\) is an optimum solution, by induction one gets

\[
w(OPT(F_0)) \leq w(OPT(F_i)) + \sum_{i=1}^{l} w(x_i)
\]

\[
\leq w(F_i) + w(F_i)
\]

\[
\leq 2w(F_i).
\]

We can substitute \(w(OPT(F_i))\) by \(w(F_i)\) since \(F_i\) has a maximal weight by definition. Let \(s_{opt}\) be an optimal solution for Maximal Compression. \(||P|| - |s_{opt}| = w(OPT(\emptyset))\). As \(F_i\) is maximum, \(l(F_i)\) is the superstring of \(P\) output by the greedy algorithm and thus, \(||P|| - |l(F_i)| = w(F_i)\). Therefore,

\[
\frac{1}{2} (||P|| - |s_{opt}|) \leq ||P|| - |l(F_i)|.
\]

Finally, we obtain the desired ratio: the greedy algorithm of the subset system achieves an approximation ratio of \(\frac{1}{2}\) for the Maximal Compression problem.

\[\square\]
3.1 Equivalence between the greedy algorithm and algorithm greedy

Here, we prove that the algorithm greedy defined by Tarhio and Ukkonen [12] and studied by Blum and colleagues [3] for the Maximal Compression problem, computes exactly the same superstring as the greedy algorithm of the subset system \((E_p, L_p)\) (see Definition 3.1 on p. 6). This is to show that these two algorithms are equivalent in terms of output and that the approximation ratio of 1/2 of Theorem 7 is valid for both of them. Remind that the input, \(P := \{s_1, s_2, \ldots, s_p\}\), is a set of \(p\) strings of \(\Sigma^*\).

**Proposition 8.** Let \(F\) be an maximal element for inclusion of \(L_p\). Thus, there exists a permutation of the input strings, that is a set \(\{i_1, \ldots, i_p\} = \{1, \ldots, p\}\) such that

\[
F = \{s_{i_1} \oplus s_{i_1}, s_{i_2} \oplus s_{i_2}, \ldots, s_{i_p} \oplus s_{i_p}\}.
\]

**Proof.** By the condition (L3), cycles are forbidden in \(F\). Hence there exist \(s_{d_1}, s_x \in S\) such that \(s_{d_1} \oplus s_x \in F\), and for all \(s_x \in S\), \(s_{d_1} \oplus s_x \notin F\).

Thus, let \((i_j)_{j \in \mathbb{N}}\) be the sequence of elements of \(P\) such that \(i_1 = d_1\), for all \(j \in I\) such that \(j + 1 \in I\), \(s_{i_j} \oplus s_{i_{j+1}} \in F\), and the size of \(I\) is maximum. As \(F\) has no cycle (condition L3), \(I\) is finite; then let us denote by \(i_1\) its largest element. We have for all \(s_y \in P\), \(s_{i_1} \oplus s_y \notin F\). Hence, \(\cup_{j \in I} s_{i_j}\) is the interval comprised between \(s_{d_1}\) and \(s_{i_1}\).

Assume that \(F \setminus \{\cup_{j \in I} s_{i_j}\} \neq \emptyset\). We iterate the reasoning by taking the interval between \(s_{d_2}\) and \(s_{i_2}\) and so on until \(F\) is exhausted. We obtain that \(F\) is the set of intervals between \(s_{d_1}\) and \(s_{i_1}\). By the condition (L1) and (L2), \(s_{i_1}\) (resp. \(s_{d_1}\)) is in the interval between \(s_{d_j}\) and \(s_{i_j}\) ⇒ \(j = 1\) (resp. \(j = 2\)). As \(s_{i_1} \oplus s_{d_2} \in E\), and \(F \cup \{s_{i_1} \oplus s_{d_2}\} \in L_p\), \(F\) is not maximum, which contradicts our hypothesis.

We obtain that \(F \setminus \{\cup_{j \in I} s_{i_j}\} = \emptyset\), hence the result. □

For each set \(F := \{s_{i_1} \oplus s_{i_1}, \ldots, s_{i_p} \oplus s_{i_p}\}\) that is a maximal element of \(L_p\) for inclusion, remind that \(l(F)\) denotes the superstring of \(S\) obtained by agglomerating the input strings of \(P\) according to the order induced by \(F\):

\[
l(F) := s_{i_1} \oplus s_{i_2} \oplus \cdots \oplus s_{i_p}.
\]

The algorithm greedy takes from set \(P\) two words \(u\) and \(v\) having the largest maximum overlap, replaces \(u\) and \(v\) with \(a \oplus b\) in \(P\), and iterates until \(P\) is a singleton.

**Proposition 9.** Let \(F\) be the output of the greedy algorithm of subset system \((E_p, L_p)\), and \(S\) the output of Algorithm Greedy for the input \(P\). Then \(S = \{l(F)\}\).
Proof. First, see that for any $i$ between 1 and $p$, there exists $s_j$ and $s_k$ such that $e_i = s_j \oplus s_k$. If $F \cup \{e_i\} \in L_p$, then by Conditions (L1) and (L2), one forbids any other left overlap of $s_k$ or any other right overlap of $s_j$ are prohibited in the following. As cycles are forbidden by condition (L3), one will finally obtain the same superstring by exchanging the pair $s_j$ and $s_k$ in $E$.

The algorithm greedy from [12] can be seen as the greedy algorithm of the subset system $(E_P, L_P)$. By the definition of the weight $w$, the later also answers to the Maximal Compression problem. Both algorithms are thus equivalent.

3.2 Shortest Cyclic Cover of Strings

A solution for MC must avoid overlaps forming cycles in the constructed superstring. However, for the Shortest Cyclic Cover of Strings problem, cycles of any positive length are allowed. As in Definition 3.1, we can define a subset system for SCCS as the pair $(E_P, L_C)$, where $L_C$ is now the set of $F \subseteq E_P$ satisfying only condition (L1) and (L2). A solution for this system with the weights defined as the length of maximal overlaps is a set of cyclic strings containing the input words of $P$ as substrings. One can see that the proof of Theorem 7 giving the $1/2$ ratio for MC can be simplified to show that the greedy algorithm associated with the subset system $(E_P, L_C)$ achieves a $1/1$ approximation ratio, in other words exactly solves SCCS.

Theorem 10. The greedy algorithm of $(E_P, L_C)$ exactly solves Shortest Cyclic Cover of Strings problem in polynomial time.

4 Conclusion

In this work, we investigated the approximation performance of greedy algorithms on well known problems using the power of subset systems. Greedy algorithms are algorithmically simpler, and usually easier to implement than more complex approximation algorithms.

For instance, the upper and lower bounds of approximation are still being refined for the Shortest Superstring and Maximal Compression problems. It is important to know how good greedy algorithms are. Here, we gave a new and simpler proof of the $1/2$ approximation ratio for Maximal Compression, and have shown that the greedy algorithm solves the Shortest Cyclic Cover of Strings problem exactly.

For the cover of graphs with maximum weight Hamiltonian path or set of cycles, the subset system and its associated greedy algorithm, provides an approximation ratio for a variety of problems, since distinct kinds of cycles can be forbidden in the third condition of the subset system (see Def. 2.1 on p. 5). For the general Maximum Asymmetric Travelling Salesman Problem problem, it achieves a $1/3$ ratio, and a $1/2$ ratio for the Maximum Directed Cyclic Cover problem.

References


Figure 2: Example of an Overlap Graph for the input words $P := \{\text{baaba}, \text{babaa}, \text{aabab}, \text{babba}\}$. 