# Approximation of Greedy Algorithms for Max-ATSP, Maximal Compression, Maximal Cycle Cover, and Shortest Cyclic Cover of Strings

Bastien Cazaux and Eric Rivals*

L.I.R.M.M. Université Montpellier II, CNRS U.M.R. 5506
161 rue Ada, F-34392 Montpellier Cedex 5, France
{cazaux, rivals}@lirmm.fr

**Abstract.** Covering a directed graph by a Hamiltonian path or a set of words by a superstring belong to well studied optimisation problems that prove difficult to approximate. Indeed, the *Maximum Asymmetric Travelling Salesman Problem* (Max-ATSP), which asks for a Hamiltonian path of maximum weight covering a digraph, and the *Shortest Superstring Problem* (*SSP*), which, for a finite language $P := \{s_1, \ldots, s_p\}$, searches for a string of minimal length having each input word as a substring, are both Max-SNP hard. Finding a short superstring requires to choose a permutation of words and the associated overlaps to minimise the superstring length or to maximise the compression of $P$. Hence, a strong relation exists between Max-ATSP and *SSP* since solving Max-ATSP on the Overlap Graph for $P$ gives a shortest superstring. Numerous works have designed algorithms that improve the approximation ratio but are increasingly complex. Often, these rely on solving the pendant problems where the cover is made of cycles instead of single path (Max-CC and SCCS). Finally, the greedy algorithm remains an attractive solution for its simplicity and ease of implementation. Its approximation ratios have been obtained by different approaches. In a seminal but complex proof, Tarhio and Ukkonen showed that it achieves 1/2 compression ratio for Max-CC. Here, using the full power of subset systems, we provide a unified approach for proving simply the approximation ratio of a greedy algorithm for these four problems. Especially, our proof for Maximal Compression shows that the Monge property suffices to derive the 1/2 tight bound.

## 1 Introduction

Given a set of words $P = \{s_1, \ldots, s_p\}$ over a finite alphabet, the *Shortest Superstring Problem* (*SSP*) 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 [4]. Recent progress in sequencing technologies have permitted an exponential increase in throughput, making acute the need for simple and efficient assembly algorithms. Two measures can be optimised for *SSP*: either the length of the superstring is minimised, or the compression is maximised (*i.e.*, $\|S\| - |u| := \sum_{s_i \in S} |s_i| - |u|$). Unfortunately, even for a binary alphabet, *SSP* is NP-hard [3] and MAX-SNP-hard relative to both measures [2]. Among many approximation algorithms, the best known fixed ratios are $2\frac{11}{23}$ for the superstring [10] and 3/4 for the compression [11]. A famous conjecture

* This work is supported by ANR Colib'read (ANR-12-BS02-0008) and Défi MASTODONS SePhHaDe from CNRS.

Bastien Cazaux and Eric Rivals: Approximation of Greedy Algorithms for Max-ATSP, Maximal Compression, Maximal Cycle Cover, and Shortest Cyclic Cover of Strings, pp. 148–161.
Proceedings of PSC 2014, Jan Holub and Jan Žďárek (Eds.), ISBN 978-80-01-05547-2 

states that a simple, greedy agglomeration algorithm achieves a ratio 2 for the superstring measure, while it is known to approximate tightly MC with ratio 1/2, but the later proofs are quite complex involving many cases of overlaps [13,14]. Figure 2 and Example 7 on page 153 illustrate the difference between two optimisation measures. The best approximation algorithms use the *Shortest Cyclic Cover of Strings* (SCCS) as a procedure, 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 $\|S\|$ [12,2].

These problems on strings can be viewed as problems on the Overlap Graph, in which the input words are the nodes, and an arc represents the asymmetric maximum overlap between two words. Figure 1 on p.150 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 Cyclic Cover* (Max-CC) and *Shortest Cyclic Cover of Strings* on the other. Both Max-ATSP and Max-CC have been extensively studied as essential computer science problems. Table 1 presents all these problems and their greedy approximation ratios.

| Type of cover | Input | | | | | |
|---|---|---|---|---|---|---|
| | directed graph | | | set of strings | | |
| | name | ratio | ref. | name | ratio | ref. |
| Hamiltonian path | *Maximum Asymmetric Travelling Salesman* | 1/3 y | [5,14] | *Maximal Compression* | 1/2 y | [13,14] |
| | | | | *Shortest Superstring* | 7/2 | [7] |
| Set of cycles | *Maximum Cyclic Cover* | Poly<br>1/2 y | [12]<br>here | *Shortest Cyclic Cover of Strings* | Poly 1 | [4] |

Table 1: The approximation performance of the greedy algorithm on the five optimisation problems considered here. The input is either a directed graph or a set of strings (in columns), while the type of cover can be a Hamiltonian path or a set of cycles (in lines). For each problem, the best greedy approximation ratio, its tightness, and the bibliographic reference are shown. Highlighted in blue: the approximation bounds for which we provide a proof relying on subset systems. The bound for *Maximum Cyclic Cover* was open. "Poly" means that the problem is solvable in polynomial time. A "y" after the bound means that it is tight.

**Our contributions**: Subset systems were introduced recently to investigate the approximation performances of greedy algorithms in a unified framework [8]. As mentioned earlier, the ratio of greedy for the five problems considered (except Max-CC) have been shown with different proofs and using distinct combinatorial properties. With subset systems, we investigate the approximation achieved by greedy algorithms on four of these problems in a unified manner, and provide new and simple proofs the results mentioned in Table 1. After introducing the required notation and concepts, we study the case of the Max-ATSP and Max-CC 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 3.1, before concluding.

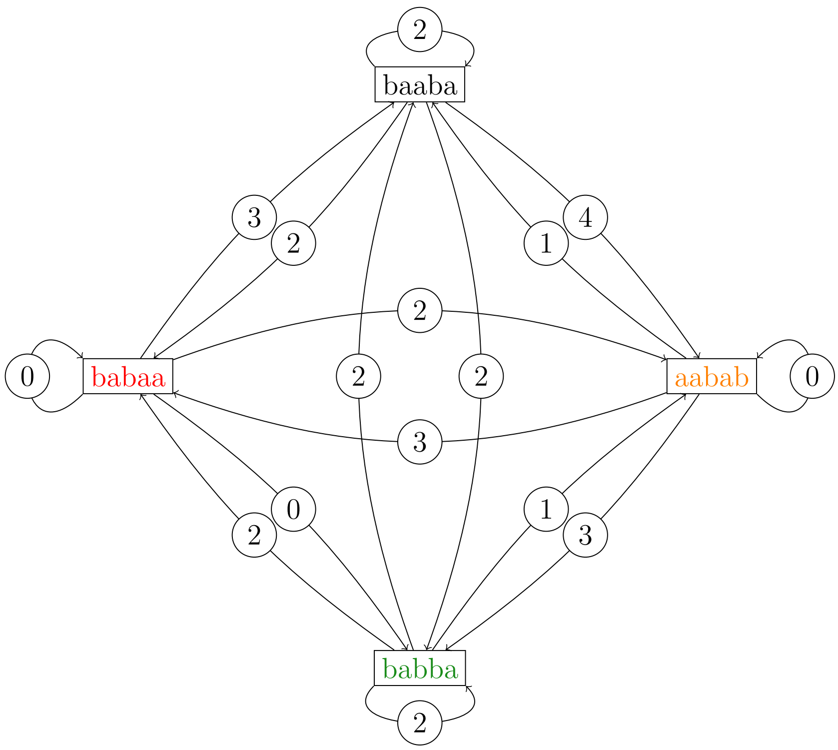


Figure 1: Example of an Overlap Graph for the input words $P :=$ {baaba, babaa, aabab, babba}.

### 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 $\Sigma^\star$, and $\epsilon$ 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 $1 \leq i \leq j \leq |x|$, $x[i]$ denotes the $i$-th letter of $x$, and $x[i\,;j]$ denotes the *substring* $x[i]x[i+1]\cdots x[j]$.

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 $\Sigma^\star$. We denote by $ov(s,t)$ the maximum overlap from $s$ over $t$; let $pr(s,t)$ be the prefix of $s$ such that $s = pr(s,t)\,.\,ov(s,t)$, then we denote the *agglomeration of $s$ over $t$* by $s \oplus t := pr(s,t)t$. 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 1.* Let $P := \{abbaa, baabb, aabba\}$. One has $ov(abbaa, baabb) = baa$ and $abbaa \oplus baabb = abbaabb$. Considering possible agglomerations of these words, we get $w_1 = abbaa \oplus baabb \oplus aabba = abbaabb \oplus aabba = abbaabba$, $w_2 = aabba \oplus abbaa \oplus baabb = aabbaa \oplus baabb = aabbaabb$ and $w_3 = baabb \oplus abbaa \oplus aabba = baabbaa \oplus aabba = baabbaabba$. Thus, $|w_1| = |pr(abbaa, baabb)| + |pr(baabb, aabba)| + |aabba| = |ab| + |b| + |aabba| = 2 + 1 + 5 = 8$, $\|P\| - |w_1| = 15 - 8 = 7$ and $|ov(abbaa, baabb)| + |ov(baabb, aabba)| = |baa| + |aabb| = 3 + 4 = 7$

### 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 is 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* are useful concepts to investigate how greedy algorithms can iteratively extend a current solution to a problem. A *subset system* is a pair $(E, \mathcal{L})$ comprising a finite set of elements $E$, and $\mathcal{L}$ a family of subsets of $E$ satisfying two conditions:

**(HS1)** $\mathcal{L} \neq \emptyset$,
**(HS2)** If $A' \subseteq A$ and $A \in \mathcal{L}$, then $A' \in \mathcal{L}$. *i.e.*, $\mathcal{L}$ is close by taking a subset.

Let $A, B \in \mathcal{L}$. One says that $B$ is an *extension of* $A$ if $A \subseteq B$ and $B \in \mathcal{L}$. A subset system $(E, \mathcal{L})$ is said to be $k$*-extendible* if for all $C \in \mathcal{L}$ and $x \notin C$ such that $C \cup \{x\} \in \mathcal{L}$, and for any extension $D$ of $C$, there exists a subset $Y \subseteq D \setminus C$ with $\#(Y) \leq k$ satisfying $D \setminus Y \cup \{x\} \in \mathcal{L}$.

The greedy algorithm associated with $(E, \mathcal{L})$ and a weight function $w$ is presented in Algorithm 1. Checking whether $F \cup \{e_i\} \in \mathcal{L}$ 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. Mestre has shown that a matroid is a 1-extendible subset system, thereby demonstrating that a subset system is a generalisation of a matroid [8, Theorem 1]. In addition, a theorem from Mestre links $k$-extendibility and the approximation ratio of the associated greedy algorithm.

**Theorem 2 (Mestre [8]).** *Let $(E, \mathcal{L})$ be a $k$-extendible subset system. The associated greedy algorithm defined for the problem $(E, \mathcal{L})$ with weights $w$ gives a $\frac{1}{k}$ approximation ratio.*

### 1.4 Definitions of problems and related work

**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

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

**Input** : $(E, \mathcal{L})$
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* $n$ **do**
  **if** $F \cup \{e_i\} \in \mathcal{L}$ **then** $F \leftarrow F \cup \{e_i\}$;
  ;
**return** $F$

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 3 (Max-ATSP).** *Let $G$ be a weighted directed graph.* Max-ATSP *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 [6]. However, the approximation ratio obtained by a simple greedy algorithm remains an interesting question, especially since other approximation algorithms are usually less efficient than a greedy one. In fact, Turner has shown a 1/3 approximation ratio for Max-ATSP [14, Thm 2.4]. As later explained, Max-ATSP is strongly related to the *Shortest Superstring Problem* and the *Maximal Compression* problems on strings.

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

**Definition 4 (Max Cyclic Cover).** *Let $G$ be a weighted directed graph.* Maximum Cyclic Cover *searches for a set of cycles of maximum weight that collectively cover $G$.*

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

### Superstring and Maximal Compression

**Definition 5 (Superstring).** *Let $P = \{s_1, s_2, \ldots, s_p\}$ be a set of $p$ strings of $\Sigma^\star$. 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 $\|S\| := \sum_{s_i \in S} |s_i|$. For any superstring $s'$, there exists a set $\{i_1, \ldots, i_p\} = \{1, \ldots, p\}$ such that $s' = s_{i_1} \oplus s_{i_2} \oplus \cdots \oplus s_{i_p}$, and then $\|S\| - |s'| = \sum_{j=1}^{p-1} |ov(s_{i_j}, s_{i_{j+1}})|$.

**Definition 6 (Shortest Superstring Problem (*SSP*)).** *Let $p$ be a positive integer and $P := \{s_1, s_2, \ldots, s_p\}$ be a set of $p$ strings over $\Sigma$. Find $s'$ a superstring of $P$ of minimal length.*

Two approximation measures can be optimised:

- the length of the obtained superstring, that is $|s'|$, or
- the compression of the input strings achieved by the superstring: $\|P\| - |s'|$.

The corresponding approximation problems are termed *Shortest Superstring Problem* in the first case, or *Maximal Compression* in the second.


(a) (b)

Figure 2: Consider the $P := \{abba, bbabab, ababa, babaa\}$. (a) The string *abbababaa* is a superstring of $P$ of length 9, the figure shows the order of the word of $P$ in the superstring. (b): the sum of the overlaps between adjacent words in *abbababaa* equals $\|P\| - |abbababaa| = 11$.

Figure 2 shows an input for Max-CC or *SSP*, with a superstring of length 9, which achieves a compression of 11.

*Example 7.* Let $P := \{a^k b, b^{k+1}, bc^k\}$ be a set of words; $w_g = a^k bc^k b^{k+1}$ is a superstring found by the greedy algorithm and $w_{opt} = a^k b^{k+1} c^k$ is an optimal superstring. Thus, the ratio of approximation is $\frac{|w_g|}{|w_{opt}|} = \frac{3k+2}{3k+1} \xrightarrow[k\to\infty]{} 1$ and the ratio of compression is $1/2$. In other words, a greedy superstring may be almost optimal in length, but its compression is only $1/2$.

Both *Maximal Compression* and *Shortest Superstring Problem* are NP-hard [3] and Max-SNP hard [2]. Numerous, complex algorithms have been designed for them, or their variants. Many are quite similar and use a procedure to find a *Maximum Cyclic Cover* of the input strings . The best known approximation ratio for the *Shortest Superstring Problem* was obtained in 2012 and equals $2\frac{11}{13}$ [10], although an optimal ratio of 2 has been conjectured in the 80's [13,2].

For the *Maximal Compression* problem, a recent algorithm gives a ratio of $3/4$ [11]. 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 [13]. 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 [14]. Both proofs are quite intricate and include many subcases [13]. 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 time greedy algorithm for the problem of Max Cyclic Covers on Strings.

**Definition 8 (Shortest Cyclic Cover of Strings (SCCS)).** *Let $p \in \mathbb{N}$ and let $P$ be a set of $p$ linear strings over $\Sigma$: $P := \{s_1, s_2, \ldots, s_p\}$. Find a set of cyclic strings of minimal cumulated length such that any input $s_i$, with $1 \le i \le p$, is a substring of at least one cyclic string.*

Several approximation algorithms for the *Shortest Superstring Problem* 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* [12,2,4]. This further indicates the importance of SCCS.

Both the *Maximal 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 $|ov(s_i, s_j)|$. Hence, the overlap graph is a complete graph with null or positive weights. A 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 $\rho$ approximation for Max-ATSP gives the same ratio for *Maximal Compression*. The same relation exists between the *Shortest Cyclic Cover of Strings* and *Maximum Cyclic Cover* on graphs. Indeed, SCCS optimises $\|P\| - \sum_j |c_j|$, where each $c_j$ is a cyclic string in the solution, and Max-CC 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). The performance of a greedy algorithm for the *Shortest Cyclic Cover of Strings* problem is declared to be open in [15], while a claim saying that greedy is an exact algorithm for this problem appears in [4].

## 2 Maximum Asymmetric Travelling Salesman and Maximum 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 9.** *Let $\mathcal{L}_S$ be the powerset of $E_G$. We define the pair $(E_G, \mathcal{L}_S)$ such that any $F$ in $\mathcal{L}_S$ satisfies*

*(L1) $\forall x, y$ and $z \in V_G$, $(x, z)$ and $(y, z) \in F$ implies $x = y$,*
*(L2) $\forall x, y$ and $z \in V_G$, $(z, x)$ and $(z, y) \in F$ implies $x = y$,*
*(L3) for any $r \in N^\star$, there does not exist any cycle $((x_1, x_2), \ldots, (x_{r-1}, x_r), (x_r, x_1))$ in $F$, where $\forall k \in \{1, \ldots, r\}, x_k \in V_G$.*

*Remark 10.*

- 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 $F \in \mathcal{L}_S$ and for any $v \in V_G$, the arc $(v, v)$ cannot belong to $F$, by Condition (L3) for $r = 1$.
- If in condition $(L_3)$, 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 13 remains valid. We will consider later the case where all cycles are allowed (*i.e.*, $r \in \emptyset$).

**Proposition 11.** $(E_G, \mathcal{L}_S)$ *is a subset system.*

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

Proposition 12 shows that the defined subset system is 3-extendible.

**Proposition 12.** $(E_G, \mathcal{L}_S)$ *is* 3-*extendible.*

*Proof.* Let $C \in \mathcal{L}_S$ and $e \notin C$ such that $C \cup \{e\} \in \mathcal{L}_S$. Let $D$ 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 $\mathcal{L}_S$.

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 $D$ is an extension of $C$, $D$ belongs to $\mathcal{L}_S$ and satisfies conditions (L1) and (L2). Hence, $\#(Y) \leq 3$.

It remains to show that $D \setminus Y \cup \{e\} \in \mathcal{L}_S$. As $C \cup \{e\} \in \mathcal{L}_S$, $C \cup \{e\}$ 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 $(z, y) \notin D \setminus C$. Therefore, for all $z \in V_G$, $(x, z)$ and $(z, y) \notin D \setminus Y$. Hence, $D \setminus Y \cup \{e\}$ satisfies conditions (L1) and (L2).

Now assume that $D \setminus Y \cup \{e\}$ violates Condition (L3). As $D \in \mathcal{L}_S$, $D$ satisfies condition (L3) and $D \setminus Y$ too. The only element who can generate a cycle is $e$. As $C \cup \{e\} \in \mathcal{L}_S$, $e$ 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. Another proof for this result originally published by [5] is given in [8, Theorem 6].

**Theorem 13.** *The greedy algorithm of* $(E_G, \mathcal{L}_S)$ *yields a* 1/3 *approximation ratio for Max-*ATSP*.*

*Proof.* By Proposition 12, $(E_G, \mathcal{L}_S)$ is 3-extendible. A direct application of Mestre's theorem (Theorem 2) yields the 1/3 approximation ratio for Max-ATSP.

*Case of the Maximum Cyclic Cover problem* If in condition $(L_3)$ we ask that $r \in \emptyset$, $(L_3)$ is not a constraint anymore and all cycles are allowed. This defines a new subset system, denoted by $(E_G, \mathcal{L}_\mathcal{C})$. As in the proof of Proposition 12, 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 $(E_G, \mathcal{L}_\mathcal{C})$ is 2-extendible and that the greedy algorithm achieves a 1/2 approximation ratio for the *Maximum Cyclic Cover* problem.

# 3 Maximal Compression and Shortest Cyclic Cover of Strings

Blum and colleagues [2] have designed an algorithm called `greedy` that iteratively constructs a superstring for both the *Shortest Superstring Problem* 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 [2], we need to prove that `greedy` computes exactly the same superstring as the greedy algorithm of the subset system of Definition 14. This proof is given in Appendix. Knowing that these two algorithms are equivalent in terms of output, the approximation ratio of Theorem 18 is valid for both of them.

From now on, let $P := \{s_1, s_2, \dots, s_p\}$ be a set of $p$ strings of $\Sigma^\star$.

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 $t$[1]. We set $E_P = \{s_i \odot s_j \mid s_i \text{ and } s_j \in P\}$. Hence, $E_P$ is the set of maximum overlaps between any two words of $S$.

**Definition 14 (Subset system for *Maximal Compression*).** *Let $\mathcal{L}_\mathcal{P}$ as the set of $F \subseteq E_P$ such that:*

*(L1)* $\forall s_i,\ s_j$ *and* $s_k \in S$, $s_i \odot s_k$ *and* $s_j \odot s_k \in F \Rightarrow i = j$, i.e. *for each string, there is only one overlap to the left*

*(L2)* $\forall s_i,\ s_j$ *and* $s_k \in S$, $s_k \odot s_i$ *and* $s_k \odot s_i \in F \Rightarrow i = j$, *and only one overlap to the right*

*(L3) for any* $r \in N^\star$, *there exists no cycle* $(s_{i_1} \odot s_{i_2},\ \dots,\ s_{i_{r-1}} \odot s_{i_r},\ s_{i_r} \odot s_{i_1})$ *in* $F$, *such that* $\forall k \in \{1, \dots, r\}, s_{i_k} \in S$.

For each set $F := \{s_{i_1} \odot s_{i_2}, \dots, s_{i_{p-1}} \odot s_{i_p}\}$ that is an inclusion-wise maximal element of $\mathcal{L}_\mathcal{P}$, we denote by $l(F)$ the superstring of $S$ obtained by agglomerating the input strings of $P$ according to the order induced by $F$:

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

First, knowing that *Maximal Compression* is equivalent to Max-ATSP on the Overlap Graph (see Section 1.4), we get a 1/3 approximation ratio for *Maximal Compression* as a corollary of Theorem 13. Another way to obtain this ratio is to show that the subset system is 3-extendible (the proof is identical to that of Proposition 12) and then use Theorem 2. However, the following example shows that the system $(E_P, \mathcal{L}_\mathcal{P})$ is not 2-extendible.

*Example 15.* The subset system $(E_P, \mathcal{L}_\mathcal{P})$ is not 2-extendible. Let $P := \{s_1, s_2, s_3, s_4, s_5\}$, $C := \emptyset$, $x := s_1 \odot s_2$. Then clearly $C \cup \{x\}$ belongs to $\mathcal{L}_\mathcal{P}$ and the set $D := \{s_1 \odot s_3,\ s_4 \odot s_2,\ s_5 \odot s_1,\ s_2 \odot s_5\}$ is an extension of $C$. However, when searching for a set $Y$ such that $Y$ included in $D \setminus C = D$ and such that $(D \setminus Y) \cup \{x\} \in \mathcal{L}_\mathcal{P}$ then $s_1 \odot s_3,\ s_4 \odot s_2$ must be removed to avoid violating $(L1)$ or $(L2)$, and at least one among $s_5 \odot s_1,\ s_2 \odot s_5$ must be removed to avoid violating $(L3)$. It follows that $\#(Y) \geq 3$.

[1] The notation $s \odot t$ represents the fact that $s$ can be aggregated with $t$ according to their maximal overlap. $ov(s, t)$ is a word representing a maximum overlap between $s$ and $t$. Hence, $s \odot t$ differs $ov(s, t)$.

To prove a better approximation ratio for the greedy algorithm, we will need the Monge inequality [9] adapted to word overlaps.

**Lemma 16.** *Let $s_1$, $s_2$, $s_3$ and $s_4$ be four different words satisfying $|ov(s_1, s_2)| \geq |ov(s_1, s_4)|$ and $|ov(s_1, s_2)| \geq |ov(s_3, s_2)|$. So we have :*

$$|ov(s_1, s_2)| + |ov(s_3, s_4)| \geq |ov(s_1, s_4)| + |ov(s_3, s_2)|.$$

When for three sets $A, B, C$, we write $A \cup B \setminus C$, it means $(A \cup B) \setminus C$. Let $A \in \mathcal{L}_\mathcal{P}$ and let $\textsc{opt}(A)$ denote an extension of $A$ of maximum weight. Thus, $\textsc{opt}(\emptyset)$ is an element of $\mathcal{L}_\mathcal{P}$ of maximum weight. The next lemma follows from this definition.

**Lemma 17.** *Let be $F \in \mathcal{L}_\mathcal{P}$ and $x \in E_P$, $w(\textsc{opt}(F \cup \{x\})) \leq w(\textsc{opt}(F))$.*

Now we can prove a better approximation ratio.

**Theorem 18.** *The approximation ratio of the greedy algorithm for the* Maximal Compression *problem is* $1/2$.

*Proof.* To prove this ratio, we revisit the proof of Theorem 2 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. 152). The structure of the proof is first to show for any element $x_i$ incorporated by the greedy algorithm, the inequality $w(\textsc{opt}(F_{i-1})) \leq w(\textsc{opt}(F_i)) + w(x_i)$, and second, to reason by induction on the sets $F_i$ starting with $F_0$.

One knows that $\textsc{opt}(F_{i-1})$ is an extension of $F_{i-1}$. By the greedy algorithm and by the definitions of $F_{i-1}$ and $x_i$, one gets $F_{i-1} \cup \{x_i\} \in \mathcal{L}_\mathcal{P}$. As $x_i \in E_P$, there exist $s_p$ and $s_o$ such that $x_i = s_p \odot s_o$. Like in the proof of Proposition 12, let $Y_i$ denote the subset of elements of $\textsc{opt}(F_{i-1}) \setminus F_{i-1}$ of the form $s_p \odot s_k$, $s_k \odot s_o$, or $s_k \odot s_p$, where $s_k \odot s_p$ belongs to a cycle in $\textsc{opt}(F_{i-1}) \cup \{x_i\}$. Thus, $\textsc{opt}(F_{i-1}) \setminus Y_i \cup \{x_i\} \in \mathcal{L}_\mathcal{P}$, and

$$\begin{aligned} w(\textsc{opt}(F_{i-1})) &= w(\textsc{opt}(F_{i-1}) \setminus Y_i \cup \{x_i\}) + w(Y_i) - w(x_i) \\ &\leq w(\textsc{opt}(F_i)) + w(Y_i) - w(x_i). \end{aligned}$$

Indeed, $w(\textsc{opt}(F_{i-1}) \setminus Y_i \cup \{x_i\}) \leq w(\textsc{opt}(F_i))$ because $\textsc{opt}(F_{i-1}) \setminus Y_i \cup \{x_i\}$ is an extension of $F_{i-1} \cup \{x_i\}$ and because $\textsc{opt}(F_i)$ 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 $F_j \cup \{y\} \notin \mathcal{L}_\mathcal{P}$, but $F_j \cup \{y\} \subseteq \textsc{opt}(F_{i-1}) \in \mathcal{L}_\mathcal{P}$, 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 $w(\textsc{opt}(F_{i-1})) \leq w(\textsc{opt}(F_i)) + w(x_i)$.

***Case*** 2 : $\#(Y_i) = 3$.
There exists $s_k$ and $s_{k'}$ such that $s_p \odot s_{k'}$ and $s_k \odot s_o$ are in $Y_i$. By Lemma 16, we have $w(x_i) + w(s_k \odot s_{k'}) \geq w(s_p \odot s_{k'}) + w(s_k \odot s_o)$. As $s_p \odot s_{k'}$ and $s_k \odot s_o$ belong to $\textsc{opt}(F_{i-1})$, one deduces $s_k \odot s_{k'} \notin \textsc{opt}(F_{i-1})$.

We get $\textsc{opt}(F_{i-1}) \setminus Y_i \cup \{x_i, s_k \odot s_{k'}\} \in \mathcal{L}_\mathcal{P}$. Indeed, as $Y_i \subseteq \textsc{opt}(F_{i-1})$, neither a right overlap of $s_k$, nor a left overlap of $s_{k'}$ can belong to $\textsc{opt}(F_{i-1})$. Furthermore, adding $s_k \odot s_{k'}$ to $\textsc{opt}(F_{i-1}) \setminus Y_i \cup \{x_i\}$ cannot create a cycle, since otherwise a cycle would have already existed in $\textsc{opt}(F_{i-1})$. This situation is illustrated in Figure 3.

We have $w(\textsc{opt}(F_{i-1}) \setminus Y_i \cup \{x_i,\ s_k \odot s_{k'}\}) \leq w(\textsc{opt}(F_{i-1} \cup \{x_i,\ s_k \odot s_{k'}\}))$, because $\textsc{opt}(F_{i-1}) \setminus Y_i \cup \{x_i,\ s_k \odot s_{k'}\}$ is an extension of $F_{i-1} \cup \{x_i,\ s_k \odot s_{k'}\}$ and $\textsc{opt}(F_{i-1} \cup \{x_i,\ s_k \odot s_{k'}\})$ is a maximum weight extension of $F_{i-1} \cup \{x_i,\ s_k \odot s_{k'}\}$. As $w(\textsc{opt}(F_{i-1} \cup \{x_i,\ s_k \odot s_{k'}\})) \leq w(\textsc{opt}(F_{i-1} \cup \{x_i\}))$, by Lemma 17 one gets:

$$\begin{aligned} w(\textsc{opt}(F_{i-1})) &= w(\textsc{opt}(F_{i-1}) \setminus Y_i \cup \{x_i,\ s_k \odot s_{k'}\}) + w(Y_i) - w(x_i) - w(s_k \odot s_{k'}) \\ &\leq w(\textsc{opt}(F_{i-1} \cup \{x_i,\ s_k \odot s_{k'}\})) + w(Y_i) - w(x_i) - w(s_k \odot s_{k'}) \\ &\leq w(\textsc{opt}(F_i)) + w(Y_i) - w(x_i) - w(s_k \odot s_{k'}). \end{aligned}$$

As $Y_i = \{s_p \odot s_{k'}, s_k \odot s_o, s_{k''} \odot s_p\}$, one obtains

$$\begin{aligned} w(\textsc{opt}(F_{i-1})) &\leq w(\textsc{opt}(F_i)) - w(s_k \odot s_{k'}) + w(Y_i) - w(x_i) \\ &\leq w(\textsc{opt}(F_i)) - w(s_k \odot s_{k'}) + w(s_p \odot s'_k) + w(s_k \odot s_o) + w(s_{k''} \odot s_p) - w(x_i) \\ &\leq w(\textsc{opt}(F_i)) + w(s_{k''} \odot s_p) \\ &\leq w(\textsc{opt}(F_i)) + w(x_i). \end{aligned}$$

Remembering that $\textsc{opt}(\emptyset)$ is an optimum solution, by induction one gets

$$\begin{aligned} w(\textsc{opt}(F_0)) &\leq w(\textsc{opt}(F_l)) + \sum_{i=1}^{l} w(x_i) \\ &\leq w(F_l) + w(F_l) \\ &\leq 2w(F_l). \end{aligned}$$

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

$$\frac{1}{2}(\|P\| - |s_{opt}|) \leq \|P\| - |l(F_l)|\,.$$

Finally, we obtain the desired ratio: the greedy algorithm of the subset system achieves an approximation ratio of $1/2$ for the *Maximal Compression* problem.

### 3.1 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 14, we can define a subset system for SCCS as the pair $(E_P, \mathcal{L}_\mathcal{C})$, where $\mathcal{L}_\mathcal{C}$ 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

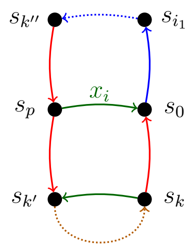


Figure 3: Impossibility to create a cycle by adding $s_k \odot s_{k'}$ to $\textsc{opt}(F_{i-1}) \setminus Y_i \cup \{x_i\}$, without having an already existing cycle in $\textsc{opt}(F_{i-1})$. Since we are adding $x_i$ to $\textsc{opt}(F_{i-1})$, we need to remove three elements in red: $s_{k''} \odot s_p$, $s_p \odot s_{k'}$, $s_k \odot s_o$.

overlaps is a set of cyclic strings containing the input words of $P$ as substrings. One can see that the proof of Theorem 18 giving the 1/2 ratio for MC can be simplified to show that the greedy algorithm associated with the subset system $(E_P, \mathcal{L}_\mathcal{C})$ achieves a 1/1 approximation ratio, in other words exactly solves SCCS.

**Theorem 19.** *The greedy algorithm of* $(E_P, \mathcal{L}_\mathcal{C})$ *exactly solves* Shortest Cyclic Cover of Strings *problem in polynomial time.*

## 4 Conclusion

Greedy algorithms are algorithmically simpler, and usually easier to implement than more complex approximation algorithms [7,2,6,10,11]. In this work, we investigated the approximation ratio of greedy algorithms on several well known problems using the power of subset systems. Our major result is to prove these ratios with a unified and simple line of proof. Moreover, this approach can likely be reused for variants of these problems [1]. 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. 9 on p. 154). For the general *Maximum Asymmetric Travelling Salesman Problem* problem, it achieves a 1/3 ratio, and a 1/2 ratio for the *Maximum Cyclic Cover* problem.

Today, the upper and lower bounds of approximation are still being refined for the *Shortest Superstring Problem* and *Maximal Compression* problems. It is important to know how good greedy algorithms are. Here, we have shown that the greedy algorithm solves the *Shortest Cyclic Cover of Strings* problem exactly, and gave an alternative proof of the 1/2 approximation ratio for *Maximal Compression*(Theorem 18). The latter is important for it shows that, beside the 3-extendibility, one only needs to consider the Monge property, to achieve this bound. It also illustrates how a combinatorial property that is problem specific can help to extend the approach of Mestre, while still using the theory of subset systems [8].

## References

1. M. Bläser and B. Manthey: *Approximating maximum weight cycle covers in directed graphs with weights zero and one.* Algorithmica, 42(2) 2005, pp. 121–139.
2. A. Blum, T. Jiang, M. Li, J. Tromp, and M. Yannakakis: *Linear approximation of shortest superstrings*, in ACM Symposium on the Theory of Computing, 1991, pp. 328–336.

3. J. Gallant, D. Maier, and J. A. Storer: *On finding minimal length superstrings.* Journal of Computer and System Sciences, 20 1980, pp. 50–58.
4. D. Gusfield: *Algorithms on Strings, Trees and Sequences*, Cambridge University Press, 1997.
5. T. A. Jenkyns: *The greedy travelling salesman's problem.* Networks, 9(4) 1979, pp. 363–373.
6. H. Kaplan, M. Lewenstein, N. Shafrir, and M. Sviridenko: *Approximation algorithms for asymmetric tsp by decomposing directed regular multigraphs.* J. of Association for Computing Machinery, 52(4) July 2005, pp. 602–626.
7. H. Kaplan and N. Shafrir: *The greedy algorithm for shortest superstrings.* Information Processing Letters, 93(1) 2005, pp. 13–17.
8. J. Mestre: *Greedy in Approximation Algorithms*, in Proceedings of 14th Annual European Symposium on Algorithms (ESA), vol. 4168 of Lecture Notes in Computer Science, Springer, 2006, pp. 528–539.
9. G. Monge: *Mémoire sur la théorie des déblais et des remblais*, in Mémoires de l'Académie Royale des Sciences, 1781, pp. 666–704.
10. M. Mucha: *Lyndon words and short superstrings*, in Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA, 2013, pp. 958–972.
11. K. E. Paluch: *Better approximation algorithms for maximum asymmetric traveling salesman and shortest superstring.* CoRR, abs/1401.3670 2014.
12. C. H. Papadimitriou and K. Steiglitz: *Combinatorial optimization : algorithms and complexity*, Dover Publications, Inc., 2nd ed., 1998, 496 p.
13. J. Tarhio and E. Ukkonen: *A greedy approximation algorithm for constructing shortest common superstrings.* Theoretical Computer Sciences, 57 1988, pp. 131–145.
14. J. S. Turner: *Approximation algorithms for the shortest common superstring problem.* Information and Computation, 83(1) Oct. 1989, pp. 1–20.
15. M. Weinard and G. Schnitger: *On the greedy superstring conjecture.* SIAM Journal on Discrete Mathematics, 20(2) 2006, pp. 502–522.


# Appendix

Here, we prove that the algorithm `greedy` defined by Tarhio and Ukkonen [13] and studied by Blum and colleagues [2] for the *Maximal Compression* problem, computes exactly the same superstring as the greedy algorithm of the subset system $(E_P, \mathcal{L}_\mathcal{P})$ (see Definition 14 on p. 156). This is to show that these two algorithms are equivalent in terms of output and that the approximation ratio of 1/2 of Theorem 18 is valid for both of them. Remind that the input, $P := \{s_1, s_2, \dots, s_p\}$, is a set of $p$ strings of $\Sigma^\star$.

**Proposition 20.** *Let $F$ be an maximal element for inclusion of $\mathcal{L}_\mathcal{P}$. Thus, there exists a permutation of the input strings, that is a set $\{i_1, \dots, i_p\} = \{1, \dots, p\}$ such that*

$$F = \{s_{i_1} \odot s_{i_2}, s_{i_2} \odot s_{i_3}, \dots, s_{i_{p-1}} \odot s_{i_p}\}.$$

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

Thus, let $(i_j)_{j\in I}$ be the sequence of elements of $P$ such that $i_1 = d_1$, for all $j \in I$ such that $j+1 \in I$, $s_{i_j} \odot s_{i_{j+1}} \in F$, and the size of $I$ is maximum. As $F$ has no cycle (condition L3), $I$ is finite; then let us denote by $t_1$ its largest element. We have for all $s_y \in P$, $s_{t_1} \odot s_y \notin F$. Hence, $\cup_{j\in I} i_j$ is the interval comprised between $s_{d_1}$ and $s_{t_1}$.

Assume that $F \setminus \{\cup_{j\in I} i_j\} \neq \emptyset$. We iterate the reasoning by taking the interval between $s_{d_2}$ and $s_{t_2}$ and so on until $F$ is exhausted. We obtain that $F$ is the set of intervals between $s_{d_i}$ and $s_{t_i}$. By the condition (L1) and (L2), $s_{t_1}$ (resp. $s_{d_2}$) is in the interval between $s_{d_j}$ and $s_{t_j} \Rightarrow j = 1$ (resp. $j = 2$). As $s_{t_1} \odot s_{d_2} \in E$, and $F \cup \{s_{t_1} \odot s_{d_2}\} \in \mathcal{L}_\mathcal{P}$, $F$ is not maximum, which contradicts our hypothesis.

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

For each set $F := \{s_{i_1} \odot s_{i_2}, \ldots, s_{i_{p-1}} \odot s_{i_p}\}$ that is a maximal element of $\mathcal{L}_\mathcal{P}$ 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 21.** *Let $F$ be the output of the greedy algorithm of subset system $(E_P, \mathcal{L}_\mathcal{P})$, 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 $e_i = s_j \odot s_k$. If $F \cup \{e_i\} \in \mathcal{L}_\mathcal{P}$, 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 $s_k$ with $s_j \oplus s_k$ in $E$.

The algorithm `greedy` from [13] can be seen as the greedy algorithm of the subset system $(E_P, \mathcal{L}_\mathcal{P})$. By the definition of the weight $w$, the later also answers to the *Maximal Compression* problem. Both algorithms are thus equivalent.