Conditional Inequalities and the Shortest Common Superstring Problem

Uli Laube and Maik Weinard

Institut für Informatik
Johann Wolfgang Goethe-Universität Frankfurt am Main
Robert-Mayer-Straße 11-15
60054 Frankfurt am Main, Germany

e-mail: {laube,weinard}@thi.cs.uni-frankfurt.de

Abstract. We investigate the shortest common superstring problem (SCSSP). As SCSSP is APX-complete it cannot be approximated within an arbitrarily small performance ratio. One heuristic that is widely used is the notorious greedy heuristic. It is known, that the performance ratio of this heuristic is at least 2 and not worse than 4. It is conjectured that the greedy heuristic's performance ratio is in fact 2 (the greedy conjecture). Even the best algorithms introduced for SCSSP can only guarantee an upper bound of 2.5.

In [11] an even stronger version of the greedy conjecture is proven for a restricted class of orders in which strings are merged. We extend these results by broadening the class for which this stronger version can be established. We also show that the Triple inequality, introduced in [11] and crucial for their results, is inherently insufficient to carry the proof for the greedy conjecture in the general case. Finally we describe how linear programming can be used to support research along this line.

Keywords: Shortest Superstring, Greedy Heuristic, Performance Ratio

1 Introduction

Given a set of n strings $S = \{s_1, \ldots, s_n\}$, the shortest common superstring problem (SCSSP) is to find a string s such that each s_i is a substring of s, and such that s is as short as possible. We may assume without loss of generality, that no string s_i is a substring of another string s_j for $i \neq j$.

Apart from being an interesting problem in itself, SCSSP models parts of the reconstruction process in DNA-sequencing [4], since DNA-fragments can be described as strings. Data compression is another area where SCSSP is used [8].

SCSSP is proven to be NP-complete by Maier and Storer [6]. It is known from the work of Blum et al. [2] that SCSSP is APX-complete. According to Arora et al. [1] such problems do not have a polynomial-time approximation scheme, unless P=NP.

The crucial part of the problem is to find the best order in which the strings s_i should appear in the superstring. Once an order is fixed the superstring can be easily constructed by greedily pulling a string as far as possible over its predecessor in the given order, hence exploiting the greatest possible *overlap*.

Definition 1 *Consider two strings a and b. The* overlap *of a and b is the longest proper suffix of a that is also a proper prefix of b. Its length is denoted by $|a, b|$.*

For a given permutation π the superstring described by π (denoted s_π) has the length

$$|s_\pi| = \sum_{i=1}^{n} |s_i| - \sum_{i=1}^{n-1} |s_{\pi(i)}, s_{\pi(i+1)}|.$$

The *greedy heuristic* can be used to approximate the superstring s by repeatedly merging strings with maximal overlap until only one string is left:

1. Input: A set S of n strings.
2. while $|S| > 1$
   (a) Choose $a, b \in S$ with maximal overlap $|a, b|$ and $a \neq b$.
   (b) Let c be the partial superstring that is created by concatenating a and the suffix of b, that does not belong to (a, b).
   (c) Let $S := \big(S \setminus \{a, b\}\big) \cup \{c\}$.
3. Output: The one string left in S.

As we obtain a partial superstring in every step, the output will be a superstring for the strings in S.

It has been conjectured by Turner [10] that this greedy heuristic has an approximation factor of 2 (the *greedy conjecture*). A simple example given by Turner [10] establishes that the approximation factor of the greedy heuristic is no better than 2. Blum et al. [2] prove that the greedy heuristic achieves an approximation factor of 4 and these are still the best known upper and lower bounds for the performance of the greedy heuristic.

A 3-approximation algorithm derived from the greedy heuristic is presented by Blum et al. in the same paper. Since then a series of results has been published, improving the approximation factor to $2\frac{1}{2}$ [9]. This was achieved by developing more and more sophisticated algorithms.

However it is known, that the greedy heuristic has approximation factor 2, if one is interested in maximizing the total amount of overlap exploited (as opposed to minimizing the length of the resulting superstring). Moreover a straightforward variation of the greedy heuristic is able to find optimal cycle covers. Hence we seeked insight as to why the greedy superstring conjecture appears so hard to verify. Therefore we searched systematically for interesting instances of the problem by introducing *greedy orders*. In section 2 we will describe our approach and introduce the *mice game*, which resembles the task of proving the greedy superstring conjecture in a tricky way and allows to exploit *conditional linear inequalities* like the prominent Monge inequality.

First results of this approach are shown in Weinard and Schnitger [11]: It is possible to verify an even stronger version of the greedy conjecture for a restricted class of orders in which strings are merged. A second result shows that the established conditional inequalities are insufficient to prove the greedy conjecture even in the classical form. A new conditional inequality – called the Triple inequality – is introduced, that carries the proof for the stronger version. We will briefly describe these results in section 2.

In this paper we extend the results from [11]: We increase the number of greedy orders for which the stronger conjecture holds from $\Theta(2^n)$ to $\Theta(4^n)$. We further show that the Triple inequality [11] is inherently too weak to prove the greedy conjecture for the general case. These results will be presented in sections 3 and 4.

To support a systematical search for interesting instances of the problem we developed the software tool SINDBAD that turned out to be extremely helpful in achieving the results of [11] and the results of this paper. We will describe the major features of SINDBAD in section 5.

2 From SCSSP to the Mice Game

The difficult part of SCSSP is to find an optimal order in which to arrange the n strings. We can assume without loss of generality, that the greedy heuristic merges the strings into the order $s_1, s_2, \ldots, s_n$. The greedy heuristic achieves this by repeatedly merging two partial superstrings. We call the order in which the partial superstrings are merged the *greedy order*. We define the greedy order as a sequence of pairs (s_i, s_{i+1}) that indicate, in which order the ends of the partial superstrings are merged by the greedy heuristic.

The length of a superstring is the sum of the lengths of the strings in $S = \{s_1, s_2, \ldots, s_n\}$ minus the overlaps of the consecutive strings

$$|s_\pi| = \sum_{i=1}^{n} |s_i| - \sum_{i=1}^{n-1} \left| s_{\pi(i)}, s_{\pi(i+1)} \right| \tag{1}$$

given a permutation π, which defines the superstring. A *cycle cover* $\mathcal{C}_\sigma$ of the strings in S is a set of disjoint cycles $C_1, C_2, \ldots, C_r$. The length of a cycle cover is

$$|\mathcal{C}_\sigma| = \sum_{i=1}^{n} |s_i| - \sum_{i=1}^{n} \left| s_i, s_{\sigma(i)} \right| \tag{2}$$

for a bijection σ, which indicates the successor of string s_i in the cycle cover.

The classical greedy conjecture is

$$|s| \leq 2 \cdot |s^*|, \tag{3}$$

where $|s|$ is the length of the superstring defined by the greedy heuristic and $|s^*|$ is the length of the optimal superstring. In [11] a stronger version is proposed:

$$|s| \leq |s^*| + |\mathcal{C}^*|, \tag{4}$$

where $|\mathcal{C}^*|$ is the length of an optimal cycle cover. This conjecture is stronger because the length of the optimal cycle cover is not longer than $|s^*|$. For technical reasons – following [11] – our goal is to prove a variation of the stronger version:

$$\forall\, \pi \forall\, \sigma : |s^\circ| \leq |s^\circ_\pi| + |\mathcal{C}_\sigma| \tag{5}$$

where $|s^\circ|$ is the length of the superstring s when it is closed as a cycle. Therefore $|s^\circ| = |s| - |s_n, s_1|$ where $|s|$ is the length of the superstring that is determined by the

greedy order. We compare $|s^\circ|$ with the length of an alternative closed superstring s°_π plus the length of an alternative cycle cover $\mathcal{C}_\sigma$. Thus $|s^\circ_\pi| = |s_\pi| - \left|s_{\pi(n)}, s_{\pi(1)}\right|$.

In [11] it is shown that equation (5) implies equation (4). Replacing the terms in equation (5) with the expressions above and rearranging sums yields:

$$\forall\,\pi\forall\,\sigma : \sum_{i=1}^{n-1}\left|s_{\pi(i)}, s_{\pi(i+1)}\right| + \left|s_{\pi(n)}, s_{\pi(1)}\right| + \sum_{i=1}^{n}\left|s_i, s_{\sigma(i)}\right| \leq \sum_{i=1}^{n}|s_i| + \sum_{i=1}^{n-1}\left|s_i, s_{i+1}\right| + |s_n, s_1| \quad (6)$$

Hence we have to bound the $2n$ terms on the left-hand side by the $2n$ terms on the right-hand side. To achieve this we need to exploit properties of strings. Two trivial inequalities that can be used follow from the definition of the overlap. Let s_i, s_j be two strings with $i \neq j$ then $|s_i, s_j| < |s_i|$ and $|s_i, s_j| < |s_j|$ hold.

Definition 2 *Let s and u be strings and $|u| = p$. The string s is p-periodic if and only if s is a prefix of u^k for some k. If s is p_s-periodic and p_s is minimal, then p_s is a* minimum period *of s.*

A consequence of Definition 2 is the equality $|s_i, s_i| = |s_i| - |p_{s_i}|$ and hence $|s_i, s_i| < |s_i|$ follows. These simple inequalities are of course insufficient to prove equation (6).

Our approach is to use *conditional inequalities*, i.e. linear inequalities that hold whenever a condition, that is also a set of linear inequalities, holds. An example of a conditional inequality is the one observed by Gaspard Monge [7] in 1781.

Lemma 1 *Let a, b, c, d be strings. Given that $|a, d| \leq |a, b|$ and $|c, b| \leq |a, b|$ the inequality $|a, d| + |c, b| \leq |a, b| + |c, d|$ holds. Moreover the variant $|a, d| + |c, a| \leq |a| + |c, d|$ holds without prerequisites.*

In [11] another conditional linear inequality, the Triple inequality, is introduced.

Lemma 2 *Let a, b, c, d, x be strings. Given that $\max\{|a, x|, |x, b|\} \geq |a, b|, |x, d|, |c, x|$ then $|a, b| + |x, d| + |c, x| \leq |a, x| + |x, b| + |c, d| + |p_x|$ holds.*

In order to apply these conditional inequalities we need to fix the greedy orders because the greedy order provides a partial order on the overlaps. This order will establish the premises of some conditional inequalities thereby allowing us to exploit the conditioned inequality. As a consequence we have to prove (6) for all $g \in \mathcal{G}$, where $\mathcal{G}$ is the set of all greedy orders. Note that for a given n, there are $(n-1)!$ greedy orders. The following example illustrates these concepts.

Example 1 Let us use conditional inequalities and prove our inequality (6) for $n = 5$, a fixed greedy order $\big((s_1, s_2), (s_4, s_5), (s_3, s_4), (s_2, s_3)\big)$, an alternative superstring $s_5, s_4, s_1,\ s_3, s_2$ and $\big\{(s_1, s_5, s_3, s_2, s_1), (s_4, s_4)\big\}$ as a cycle cover. Hence $\pi = (5, 4, 1, 3, 2)$ and $\sigma(1) = 5, \sigma(2) = 1, \sigma(3) = 2, \sigma(4) = 4, \sigma(5) = 3$.

The partial order on the right of Fig.1 is the one induced by the given greedy order. At first the heuristic picks the overlap (s_1, s_2) and hence this overlap is larger than every other possible overlap. When, in the second step, greedy chooses to use (s_4, s_5) some of the other overlaps are no longer an option due to the choice in the

$$
\begin{aligned}
|s_3, s_2| &\leq |s_2| \\
|s_1, s_5| + |s_3, s_2| &\leq |s_1, s_2| + |s_3, s_5| \\
|s_1, s_3| + |s_2, s_1| &\leq |s_1| + |s_2, s_3| \\
|s_2, s_5| + |s_5, s_4| &\leq |s_2, s_4| + |s_5| \\
|s_2, s_4| &\leq |s_3, s_4| \\
|s_3, s_5| + |s_4, s_1| &\leq |s_3, s_1| + |s_4, s_5| \\
|s_3, s_1| + |s_5, s_3| &\leq |s_3| + |s_5, s_1| \\
|s_4, s_4| &\leq |s_4|
\end{aligned}
$$

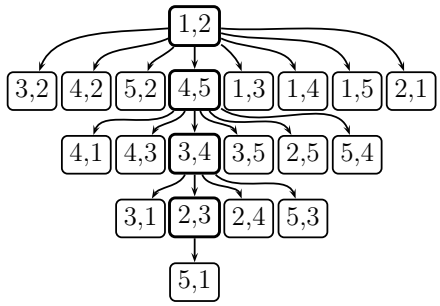


Figure 1: Proof and partial order

first round. Hence we can only exploit that (s_4, s_5) is larger than the overlaps still usable at the time. On the left we give a set of inequalities that sum up to equation (6) for this special case. The second and the sixth inequality are instances of the Monge inequality and their applicability can be verified with the partial order. (For example inequality 6 requires $|s_4, s_5| \geq |s_3, s_5|$ and $|s_4, s_5| \geq |s_4, s_1|$.)

This setup leads to the question: Which (conditional) inequalities should be used and which terms, that do not appear on either side of (6), should be introduced? For instance the introduction of the valid inequality $|s_1, s_5| + |s_4, s_1| \leq |s_1| + |s_4, s_5|$ in our example above would've made it impossible to complete the proof. We are implicitly asked to pick the *appropriate* linear inequalities. This task can be visualized as a game, that was first introduced in [11] and that is called the *mice game*.

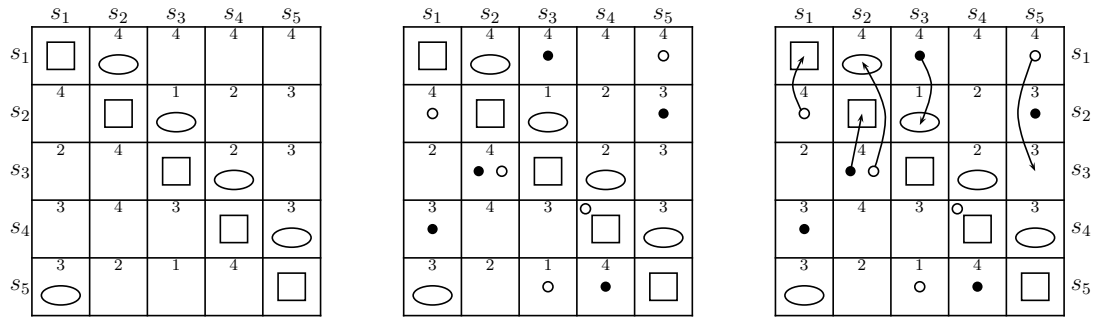
The mice game is played on a $n \times n$-board (see Figure 2(a)). The cells represent the length of the n^2 possible overlaps of the n strings. The cells on the diagonal have two interpretations: They represent the length of the self-overlap $|s_i, s_i|$ and the length of string $|s_i|$ itself.

We assign a *rank* to the cells of the board, based on the partial order of the overlaps as given by the greedy order. We assign a rank of $n - r$ to the cell whose pair was chosen as r^{th} pair and to all the cells whose pair was thereby eliminated as a possible choice for the greedy heuristic. Hence we know that an offdiagonal cell (s_i, s_{i+1}) represents a value at least as big as the value of every cell with equal or lower rank. (Note that the ranks correspond to the levels in Figure 1.)

Example 1 *continued.* Let us revisit our previous example. Figure 2(a) shows the board. The cells on the diagonal are divided into an inner and an outer part. The inner part represents the length of the string $|s_i|$ itself (the diagonal holes) and the outer part the length of the self-overlap of the string $|s_i, s_i|$. The cells with an ellipse will be refered to as the greedy holes.

The permutation π and the bijection σ determine the cells that contain the *mice* at the begining of the game. The cells are $\left|s_{\pi(i)}, s_{\pi(i+1)}\right|$, $\left|s_{\pi(n)}, s_{\pi(1)}\right|$ and $\left|s_i, s_{\sigma(i)}\right|$, the start-configuration of the game (Figure 2(b)). The mice shown as circles are placed according to the bijection that defines the cycle cover $\big\{(s_1, s_5, s_3, s_2, s_1), (s_4, s_4)\big\}$. The mice shown in black are placed according to the permutation that defines the alternative superstring s_5, s_4, s_1, s_3, s_2. For the course of the game the mice are indistinguishable, they are shown differently here to illustrate their origin.

A move is described by a set of start-cells and a set of end-cells. The move is justified by an inequality that guarantees that the sum of the lengths represented



(a) Board with ranks. (b) A start-configuration. (c) First three moves.

Figure 2: The board of the mice game

by the start-cells is not greater than the sum of the lengths represented by the end-cells. The moves inherit the name of the inequality that justifies them. Figure 2(c) shows the moves that correspond to the first three inequalities from the first part of Example 1. Firstly a diagonal insertion, secondly a greedy monge and thirdly a diagonal monge. (If we move a mouse from the outer to the inner part of a diagonal, using $|s_i, s_i| < |s_i|$ we call this *discarding a period.*) Applying the moves of all the inequalities listed in the example, brings all the mice into their holes, which is the objective of the game. A hole can only accomodate one mouse. Sucessfully moving the mice into their holes will be refered to as *winning the game.* Not used in this example is the Triple inequality, its intertrepation as a move in the game is shown below. The premises of the inequality are indicated by a sequence of arrows.

Figure 3: The horizontal and vertical Triple.

Winning the mice game hence corresponds to finding a derivation of the right-hand side (the end-configuration of the game) of equation (6), starting from the left-hand side (the start-configuration of the game). That is a proof of equation (6) for permutation π and bijection σ.

Note that we only use the properties of the strings described by the conditional linear inequalities. We do not have to care about the structure of the strings themselves, only their lengths and the lengths of their overlaps are sufficient in this case.

If we do this for every permutation π and bijection σ, we have proven the conjecture for a single greedy order (a single board). As we want to prove equation (6) for arbitrary n, we cannot handle all π and σ individually. We have to search for a winning strategy for the mice game, that is a set of rules according to which the mice should be moved. The *Rank Descending Algorithm* [11] is a winning strategy for the mice game for the restricted case of *linear greedy orders.* A linear greedy order is an order in which the greedy heuristic starts with an arbitrary overlap (s_i, s_{i+1}) and in later steps, when $s_j, \ldots, s_k$ is already created, either picks (s_{j-1}, s_j) or (s_k, s_{k+1}).

3 Extension of the Rank Descending Algorithm

We now introduce our extension of the simple Rank-Descending-Algorithm (RDA), the Duplex Rank-Descending-Algorithm (DPX-RDA), that will prove the greedy conjecture for $\Theta(4^n)$ greedy orders, thus squaring the number of orders covered in comparison to the simple RDA, that could only cover $\Theta(2^n)$ orders [11].

Definition 3 *A greedy order corresponds to one distinct distribution of ranks on the offdiagonal of the mice board.*

1. *A greedy order is linear, if there exists i such that $rank(1,2) < rank(2,3) < \ldots < rank(i,i+1) > rank(i+1,i+2) > \ldots > rank(n-1,n)$.*

2. *A greedy order is bilinear, if there exist i, k and w such that $rank(1,2) < rank(2,3) < \ldots < rank(i,i+1) > rank(i+1,i+2) > \ldots > rank(w,w+1) = 1 < rank(w+1,w+2) < \ldots < rank(k,k+1) > rank(k+1,k+2) > \ldots > rank(n-1,n)$*

The concept of subboards is essential for understanding both the RDA and the DPX-RDA. We extend the definition in [11] by introducing definitions 4.1b and 4.1c.

Definition 4 *1a. The subboard $Board_{i,j}$ (with $i \leq j$) is the set of all cells in the intersection of rows and columns $\{i \ldots j\}$.*

1b. The subboard $Board_{i,j}$ (with $i > j$) is the set of all cells in the intersection of rows and columns $\{1 \ldots j\} \cup \{i \ldots n\}$.

1c. For $i \leq j$ we say that $Board_{i,j}$ and $Board_{j+1,i-1}$ are complementary boards.

2. Let $B = Board_{i,j}$. The horizontal [vertical] frame of B is the set of all cells that belong to a row [column] of $Board_{i,j}$, but not to one of its columns [rows]. The frame of B is the union of its horizontal and vertical frame.

3. Let $B = Board_{i,j}$. We define $G_1(B)$ to be the greedy cell in position $(j, j+1)$, if existing, and $G_2(B)$ to be the greedy cell in position $(i-1, i)$, if existing. We further define the neighbouring diagonal cells, $N(G_1(B)) := (j+1, j+1)$ and $N(G_2(B)) := (i-1, i-1)$, if existing.

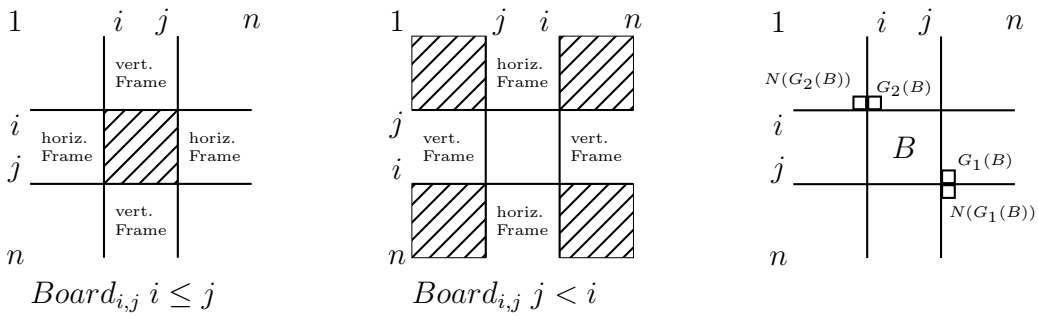


Figure 4: Complementary Boards and Frames

The Duplex-Rank Descending Algorithm

(1) The **input** consists of a superstring s°, a cycle cover $\mathcal{C}$ and a bilinear greedy sequence.

(2) **Preprocessing**

(2a) Place a mouse on position (u, v), if string s_v immediatly follows s_u in s° or in $\mathcal{C}$. If s_v is the immediate successor of s_u in both s° and $\mathcal{C}$, then (u, v) receives two mice. If a mouse is placed on a diagonal (u, u) it is placed in the outer part.

(2b) Let i, k and w be chosen according to the definition of bilinearity in Definition 3. Set $B_1 = Board_{i,i}$ and $B_2 = Board_{k,k}$

(2c) If (i, i) contains a mouse, then discard its period. Otherwise execute a diagonal monge in (i, i). If (k, k) contains a mouse, then discard its period. Otherwise execute a diagonal monge in (k, k).

(3) **Main Loop**

<table>
<tr><td colspan="6">while $B_1 \neq Board_{1,w}$ or $B_2 \neq Board_{w+1,n}$</td></tr>
<tr><td colspan="6">Let G be the highest ranked greedy cell among $G_1(B_1), G_2(B_1), G_1(B_2), G_2(B_2)$</td></tr>
<tr><td colspan="6">Does G contain a mouse?</td></tr>
<tr><td colspan="2">yes</td><td colspan="4">no</td></tr>
<tr><td colspan="2">Does $N(G)$ hold a mouse?</td><td colspan="4">Does $N(G)$ contain a mouse?</td></tr>
<tr><td>yes</td><td>no</td><td>no</td><td colspan="3">yes</td></tr>
<tr><td rowspan="3">Discard period in $N(G)$</td><td rowspan="3"></td><td rowspan="3">Greedy Monge in G</td><td colspan="2">Is the Triple in $G, N(G)$ legal?</td></tr>
<tr><td>no</td><td>yes</td></tr>
<tr><td>Greedy Monge in G</td><td rowspan="2">Triple</td></tr>
<tr><td colspan="2">Diagonal Monge in $N(G)$</td><td>Discard period in $N(G)$</td></tr>
<tr><td colspan="6">Extend the B_i, that G is incident to, to include G</td></tr>
</table>

By the preprocessing step (2) two subboards B_1, B_2 are provided that fulfill the following 5 invariants. In [11] the corresponding invariants for a single subboard were used.

I1 Every row and every column of the board contains 2 mice.

I2 Every hole in B_1 and B_2 is filled.

I3 No diagonal outside of B_1 and B_2 holds a mouse in the inner part.

I4 No diagonal contains more than one mouse.

I5.1 For all subboards B' with $B_1 \subseteq B' \neq Board_{1,n}$ the frame of B' is not empty.

I5.2 For all subboards B' with $B_2 \subseteq B' \neq Board_{1,n}$ the frame of B' is not empty.

It is easy to verify that these invariants hold after the preprocessing step. (I5 holds since s° is a single cycle.) We call a move legal if it does not violate any of the invariants. The main loop of the DPX-RDA grows the two subboards by systematically filling the greedy cell of highest rank that is not yet in B_1 or B_2. The body of the loop is essentially the same as in the simple RDA. Only the stop condition needs adjustment and the set of greedy cells to pick from is different. Note, that w is picked according to Definition 3.

Invariant I1 will be preserved as the moves used, leave the number of mice per row and column unchanged. Furthermore B_i only grows if new holes are filled. Hence I2 is established once the existence of the required moves is shown. If a mouse steps on a diagonal cell that is not about to be included into a B_i it only enters the outer part.

The proof of the existence of the moves required by the DPX-RDA, as well as maintaining invariants I4 and I5 is rather involved even for the simple RDA [11]. Luckily most of the observations establishing the existence of the moves and invariants I4 and I5 follow from [11]. In [11] the invariants are shown for one growing subboard B. These arguments remain valid and we only need to provide additional arguments to make sure that the two boards do not interfere with each other.

Hence the remaining arguments are organized as follows: For the existence of the moves we need Lemma 3. Once we have established that the moves exist, I4 holds, since it holds for both B_i individually. As to I5 we inherit from [11], that no move enlarging B_i will clear the frame of a subboard B' that includes B_i. Lemma 4 will guarantee that a move enlarging one of the B_i will not clear the frame of a board B' that includes the other B_i. Finally in Lemma 5 we argue that the preserved invariants together with the stop condition of the main loop yield a won game.

We call a mouse *free*, if it is not in a hole and not on a diagonal.

Lemma 3 *The rank of the greedy cell G, that is about to be filled at a given time, is high enough to dominate every free mouse on the board.*

Proof: When DPX-RDA tries to fill G, all the greedy cells of higher rank and their neighbouring diagonals are already filled by I2. By I1, there are only two mice in every row and column, no free mouse can be in the row or the column of a greedy cell already taken care of. In fact only the cell located in the bottom left corner of the B_i, that is about to be extendend, has a rank higher than G and could hold a mouse without violating I1. But in this case I5 would be violated, since the frame of B_i itself would be free of mice. ◇

Lemma 4 *If a move in G that extends B_i does not violate I5.i it will not violate I5.k either (with $i, k \in \{1, 2\}, i \neq k$).*

Proof: Assume the opposite, namely that the frame of a board B' with $B_k \subseteq B'$ gets cleared while no subboard $B'' \supseteq B_i$, whose frame gets cleared, exists. Observe that the frames of B' and $\overline{B'}$ are identical. But since B_i and B_k do not intersect, $B_i \subseteq \overline{B'}$ holds and we have a contradiction with $B'' = \overline{B'}$. ◇

Lemma 5 *At the end of the main loop of the DPX-RDA the game is won.*

Proof: All the invariants are preserved and we have $B_1 = Board_{1,w}$ and $B_2 = Board_{w+1,n}$. As all the holes in B_1 and B_2 are filled, the positions of the two mice in rows 1 to $w-1$ and $w+1$ to $n-1$ as well as columns 2 to w and $w+2$ to n are fixed. One of the mice in rows w and n as well as the columns 1 and $w+1$ is also accounted for. Only two possible arrangements for the last two mice do not violate I1: They are either on $(w, 1)$ and $(n, w+1)$ or on the winning position $(w, w+1)$ and $(n, 1)$. The first arrangement contradicts I5 for B_1 and B_2. ◇

It should be noted, that only Lemma 5 can not be extended to *tri-linear* or more complex greedy orders.

4 Limitations of the Triple Inequality

The Monge and the Triple inequalities (plus the trivial ones that correspond to insertions) are used in [11] to prove the strong version of the greedy conjecture (4) for linear greedy orders. In section 3 these inequalities are used to prove the strong version (5) for bilinear greedy orders. In [11] it is shown, that the Triple inequality is crucial, i.e. it is impossible to prove even the weaker classical greedy conjecture with just the Monge inequality.

We will now show that it is not possible to prove the classical greedy conjecture for arbitrary greedy orders with just the elementary inequalities, the Monge inequality *and* the Triple inequality. We do this by providing a 10×10 board that fulfills all these inequalities and still violates $|s| \leq 2 \cdot |s_\pi|$ for a given π.

	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8	s_9	s_{10}
s_1	**i+5**	$\boxed{5}$ **i+4**	$\boxed{7}$ i+3	$\boxed{8}$ i+1	$\boxed{5}$ 0	$\boxed{5}$ 0	$\boxed{5}$ 0	$\boxed{9}$ i+1	$\boxed{5}$ i+2	$\boxed{6}$ i+1
s_2	$\boxed{7}$ i+1	**3i+10**	$\boxed{7}$ **3i+9**	$\boxed{8}$ 3i+3	$\boxed{7}$ i	$\boxed{7}$ i	$\boxed{7}$ 2i+2	$\boxed{9}$ 3i+3	$\boxed{7}$ 3i+5	$\boxed{7}$ 3i+4
s_3	$\boxed{8}$ 0	$\boxed{8}$ 0	**4i+10**	$\boxed{8}$ **4i+4**	$\boxed{8}$ i	$\boxed{8}$ i	$\boxed{8}$ 3i	$\boxed{9}$ 4i+2	$\boxed{8}$ i	$\boxed{8}$ i
s_4	$\boxed{5}$ 0	$\boxed{7}$ 0	$\boxed{8}$ 0	**4i+5**	$\boxed{3}$ **i**	$\boxed{3}$ i	$\boxed{3}$ 0	$\boxed{9}$ i	$\boxed{4}$ i	$\boxed{6}$ i
s_5	$\boxed{3}$ 0	$\boxed{5}$ 0	$\boxed{7}$ 0	$\boxed{8}$ 0	**i+1**	$\boxed{1}$ **0**	$\boxed{2}$ 0	$\boxed{9}$ 0	$\boxed{4}$ 0	$\boxed{6}$ 0
s_6	$\boxed{2}$ 0	$\boxed{5}$ 0	$\boxed{7}$ 0	$\boxed{8}$ i	$\boxed{3}$ i	**i+1**	$\boxed{2}$ **0**	$\boxed{9}$ i	$\boxed{4}$ i	$\boxed{6}$ i
s_7	$\boxed{9}$ 0	$\boxed{9}$ 0	$\boxed{9}$ 0	$\boxed{9}$ 4i+2	$\boxed{9}$ i	$\boxed{9}$ i	**4i+6**	$\boxed{9}$ **4i+4**	$\boxed{9}$ i	$\boxed{9}$ i
s_8	$\boxed{4}$ 0	$\boxed{5}$ 0	$\boxed{7}$ i+2	$\boxed{8}$ 4i	$\boxed{4}$ i	$\boxed{4}$ i	$\boxed{9}$ 4i+2	**5i+6**	$\boxed{4}$ **i+1**	$\boxed{6}$ m
s_9	$\boxed{6}$ i+1	$\boxed{6}$ i+1	$\boxed{7}$ 3i+8	$\boxed{8}$ 3i+3	$\boxed{6}$ i	$\boxed{6}$ i	$\boxed{6}$ 2i+2	$\boxed{9}$ 3i+3	**4i+10**	$\boxed{6}$ **3i+6**
s_{10}	$\boxed{1}$ 0	$\boxed{5}$ i+2	$\boxed{7}$ 2i+4	$\boxed{8}$ 2i+2	$\boxed{3}$ 0	$\boxed{2}$ 0	$\boxed{4}$ i+1	$\boxed{9}$ 2i+2	$\boxed{6}$ 3i+6	**3i+8**
s.o.	0	i+1	0	i	0	i	3i	4i	3i+4	2i+2

This board describes the overlaps and lengths of 10 strings. A greedy order is given by the ranks in the little boxes. The diagonal shows the lengths of the strings and the extra row in the bottom indicates the overlaps of the strings with themselves. It can be checked exhaustively, that this board fulfills all the inequalities mentioned. The length of the strings on the diagonal sum up to $30i + 62$. The greedy locations on the offdiagonal accomodate a total value of $17i + 28$. We use $\pi = (1, 2, 10, 9, 3, 8, 7, 4, 6, 5)$ for the superstring s_π. The locations of the overlaps exploited by this superstring sum up to $24i + 28$. Subtracting the greedy overlaps from the lengths of the strings gives the length of the superstring of the greedy heuristic as $13i + 34$. The length of the alternative superstring s_π is $6i + 34$. As we may choose i arbitrarily, the ratio approaches $2\frac{1}{6}$.

This shows, that a proof of the greedy conjecture needs to exploit string properties beyond the Triple and the Monge inequality.

5 SINDBAD

SINDBAD is the name of a software tool that we developed to support the research on SCSSP. The introduction of greedy orders gives the greedy conjecture a shape with four quantifiers

$$\forall n\ \forall g \in \mathcal{G}_n\ \forall \pi\ \forall \sigma\ :\ |s^\circ| < |s^\circ_\pi| + |C_\sigma|.$$

For most instances (that is values of n, g, π and σ) the proof can be done with the established conditional inequalities. In order to proceed one must find instances, that cannot be covered with the means available. The use of calculation power is the logical consequence.

One important feature is SINDBAD's *enumeration mode*. It can automatically determine all valid moves on the board for a given greedy order and check whether the game on such a board can be won against a given alternative superstring π and a given cycle cover σ. To do this a linear program (LP), depending on the current configuration of the game is generated and evaluated. SINDBAD found the first instance of a game with linear greedy order for which the classical greedy conjecture can not be proven just with insertions and monges [11]. It was nessecary to check boards up to a size of 9×9 to encounter such an instance. The matrix from section 4 was also found using SINDBAD. (For more on the linear program and on how to define an *intermediate performance ratio* for arbitrary game configurations see subsection 5.1.)

SINDBAD also provides a *manual mode* that allows to play the mice game on arbitrary boards against arbitrary π and σ via a graphical interface. Playing the mice game manually allows interesting insights: In a game that can be won, there will still be legal moves that destroy the property, that the game can be won. Being able to locate these bad moves is extremely helpful when working on strategies. The invariants both of the RDA and of the DPX-RDA embody experience, as to what moves destroy the winning property of a game. In a game bound to be lost, one will probably still find valid moves. A *deadend* (that is a configuration with no legal move left) yields an interesting question about strings: The linear program assigns lengths to the strings and overlaps that agree with all the inequalities implemented. If a performance ratio above 2 is still possible, the question is, whether it is possible to construct a set of strings that behave like the solution of the linear program indicates. If so, a counter example would be found. If on the other hand one can pinpoint the reason why the construction of such strings is impossible, one has found a new property of strings that is guaranteed to solve yet unsolved instances of the problem. The Triple inequality was found in exactly that way. It seems fair to say that it would have been very hard to recognize the usefulness of the Triple inequality for our purposes without SINDBAD.

We also made it possible to implement game strategies into SINDBAD. Hence we could quickly find instances for which a certain strategy fails. If such a game can be won, but the strategy fails, one has a chance to improve the strategy. If it cannot be won according to the LP, one gets a hint on the limits of the moves established so far.

Further usefull features include the possibility to work with assumptions. For a given greedy order the partial order *activates* a set of conditional linear inequalities. By adding assumptions like $|a, b| \geq |c, d|$ the set of legal moves grows. If a game can be

won under this assumption and also under the assumption $|a, b| \leq |c, d|$, the instance is solved as well. SINDBAD supports the systematical search for these assumptions. SINDBAD can also assign different capacities to the holes and work with different numbers of mice. This makes it possible to work on weaker bounds for tough instances and on stronger bounds for easy instances.

5.1 SINDBAD's Linear Program

Writing a linear program to check the classical greedy conjecture on a given board and given π is simple: Let $x_{i,j}$ represent the lengths of the overlaps and l_i the lengths of the strings themselves. Let Q be the set of all inequalities that hold on the given board.

$$\begin{aligned} max &: \sum_{i=1}^{n} l_i - \sum_{i=1}^{n-1} x_{i,i+1} \\ \text{subject to} &: Q \cup \left\{ \sum_{i=1}^{n} l_i - \sum_{i=1}^{n-1} x_{\pi(i),\pi(i+1)} \leq 1 \right\} \\ \forall i, j &: x_{i,j}, l_i \geq 0 \end{aligned}$$

The linear program assigns values to all lengths and overlaps that comply with the constraints in Q and that produce a superstring s_π of length at most one. The objective function is the length of the greedy superstring. Hence the value of the optimal solution will be the best performance ratio that can be derived with the given set of inequalities Q.

From a linear programming perspective it is worthwhile noticing, that the dual problem [3, 5] of this implementation is a description of the mice game: There is a nonnegative variable y_q associated with every inequality $q \in Q$ (hence y_q is associated with a move in the mice game). A further nonnegative variable t associated with the inequality $\sum_{i=1}^{n} x_i - \sum_{i=1}^{n-1} x_{\pi(i),\pi(i+1)} \leq 1$ appears. As we don't have any inequalities in Q that include constants (i.e. all of them compare a linear combination of lengths and overlaps to 0), the objective function of the dual problem just depends on the variable t, that is to be minimized.

For every variable of the primal program (i.e. strings and overlaps, that is the cells of the miceboard) a constraint arises. For a cell c define its balance as follows:

$$bal(c) := \sum_{\substack{q \in Q \\ q \text{ moves a mouse into } c}} y_q - \sum_{\substack{q \in Q \\ q \text{ moves a mouse out of } c}} y_q. \tag{7}$$

If we interpret the variables y_q as the number of times move q is executed in a game, $bal(c)$ describes the balanced total of mice moving into and out of c during a game. The constraints of our dual problem are:

$$\begin{aligned} bal(c) - t &\leq -1 \text{ iff } c \text{ is a diagonal} \\ bal(c) &\leq 1 \text{ iff } c \text{ is a greedy cell} \\ bal(c) + t &\leq 0 \text{ iff } c \text{ is an initial mice position } (s_{\pi(i)}, s_{\pi(i+1)}) \\ bal(c) &\leq 0 \text{ otherwise.} \end{aligned}$$

(If a cell c should be a greedy cell *and* an initial mice position, its constraint is $bal(c) + t \leq 1$.) Remember, that the optimal solutions of primal and dual problem have the same value. Hence, we can verify that the performance ratio is upper bounded by 2, by giving a legal solution of the above inequalities with $t = 2$. That is, we need to provide a set of legal moves, that move 2 mice out of every initial position and respects the capacity constraints (1 for diagonal and offdiagonal, 0 for every other cell) — the mice game.

This dual representation also shows how the mice game can be adapted in order to prove bounds other than 2. If one is interested in proving a factor of 3 for instance, the game needs to be won with 3 mice on every initial position, a capacity of 2 for the diagonal holes and a capacity of 1 for the greedy holes. If we are interested in non integer performance ratios we can still use the game by scaling the number of mice and the capacities. To prove an upper bound of 2.5, we would play with 5 mice on the initial positions, and capacities 3 resp. 2 for diagonals and greedy cells.

The adaptions necessary to work with cyclicly closed superstrings are obvious. The cell $(n, 1)$ is treated as a greedy cell and $(\pi(n), \pi(1))$ is included as an initial position.

Crucial for our research was the ability to evaluate arbitrary game configurations, that is configurations that might arise during a game and that do not have a straight forward interpretation in terms of superstrings or cycle covers. Observe that the above representation indicates a total of $n \cdot t$ mice, if we work with cyclic closure. In the course of the game these mice (initially placed in groups of t on the initial positions) will spread over more cells and different numbers of mice will be on different cells of the board. In our dual representation the generalisation necessary is rather natural.

$$\begin{aligned} bal(c) - t &\leq -n \cdot t \cdot m(c) - 1 \text{ iff } c \text{ is a diagonal} \\ bal(c) &\leq -n \cdot t \cdot m(c) + 1 \text{ iff } c \text{ is a greedy cell} \\ bal(c) &\leq -n \cdot t \cdot m(c) \text{ otherwise,} \end{aligned}$$

where $m(c)$ indicates the percentage of mice on cell c. (Hence $\sum_c m(c) = 1$ and $n \cdot t \cdot m(c)$ is the number of mice on cell c at a given time.) Observe that we still describe the winning property of the game. To check how these adaptions are resembled in the primal version we need to put the above back into the shape of a linear program.

$$\begin{aligned} min &: t \\ bal(c) + (n \cdot m(c) - 1) \cdot t &\leq -1 \text{ iff } c \text{ is a diagonal} \\ bal(c) + (n \cdot m(c)) \cdot t &\leq 1 \text{ iff } c \text{ is a greedy cell} \\ bal(c) + (n \cdot m(c)) &\leq 0 \text{ otherwise,} \end{aligned}$$

Retransfering we find that we still have the variables l_i and $x_{i,j}$, we still have the constraints from set Q and the objective function is still to maximize the length of the greedy superstring $\sum_{i=1}^{n} l_i - \sum_{i=1}^{n-1} x_{i,i+1} - x_{n,1}$.

The adaption to arbitrary game configurations is only resembled in the one inequality, that initially bounded the superstring s_π. It is replaced by

$$\sum_{i=1}^{n} l_i - n \cdot \left(\sum_{c=(i,j)} m(c) \cdot x_{i,j} + \sum_{c \text{ is diagonal } i} m(c) \cdot l_i \right) \leq 1. \tag{8}$$

Observe that regular starting positions are embedded: In start configurations there are no mice on diagonals and every starting position holds 2 mice, hence a fraction of $\frac{1}{n}$ of all mice on the board.

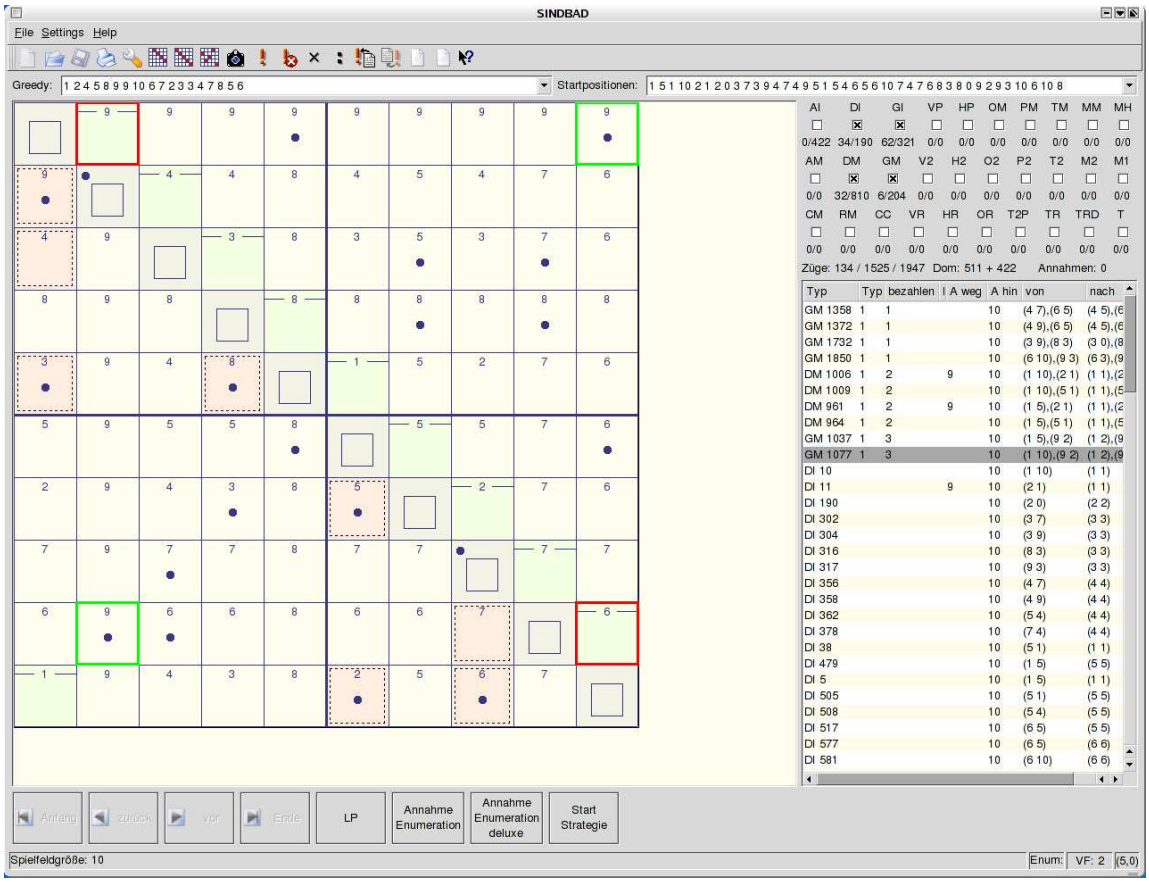


Figure 5: This is how SINDBAD looks like. SINDBAD is a KDE-Application written in C++. We experimented with an interior-point and a simplex based solver. The interior-point solver is a FORTRAN version of Csaba Mészáros' BPMPD solver. (`http://www.sztaki.hu/~meszaros/bpmpd`). The simplex based solver is `SoPlex` created by Roland Wunderling.

6 Conclusion

We have extended the class of greedy orders for which the greedy conjecture can be verified and proved that a proof for the general case is not possible without exploiting string properties beyond those used in [11]. Of course the conjecture for the general case remains *the* open problem. We believe that the stronger version of the greedy conjecture might turn out to be easier to prove. We are not aware of a counterexample for the stronger version for any greedy order. We do believe, that the approach via the dual problem (the mice game) can help focusing further research along this line as we can pinpoint unsolved instances with the help of computers. Hence we are quickly led to *good questions* about strings whose answers will cause progress in the work on the greedy conjecture.

7 Acknowledgements

We would like to thank Roland Wunderling for making SoPlex available [12].

References

[1] S. Arora, C. Lund, R. Motwani, M. Sudan and M. Szegedy: Proof verification and hardness of approximation problems. Journal of the ACM, **45**(3), 501–555, May 1998.

[2] A. Blum, T. Jiang, M. Li, J. Tromp and M. Yannakakis: Linear approximation of shortest superstrings. Journal of the ACM, **41**(4), 630–647, July 1994.

[3] Vašek Chvătal: Linear Programming. W. H. Freemann and Company, 1983.

[4] T. Jiang, M. Li: DNA Sequencing and String Learning. Mathematical Systems Theory (now Theory of Computing Systems), **29**(4), 387–405, July/August 1996.

[5] H. Karloff: Linear Programming. Birkhäuser, 1991.

[6] D. Maier and J. A. Storer: A Note on the Complexity of the Superstring Problem. In Proceedings of the 12th Annual Conference on Information Sciences and Systems, 52–56, 1978.

[7] G. Monge: Mémoire sur la théorie des déblais et des remblais. Historie de l'Academie Royale des Sciences, Année MDCCLXXXI, avec les Mémoires de Mathématique et de Physique, pour la même Année, Tirés des Registres de cette Académie, 666–704, 1781.

[8] J. A. Storer: Data Compression: Methods and Theory. Computer Science Press, 1988.

[9] Z. Sweedyk: A $2\frac{1}{2}$-Approximation Algorithm for Shortest Superstring. SIAM Journal on Computing, **29**(3), 954–986, December 1999.

[10] J. Turner: Approximation Algorithms for the Shortest Commom Superstring Problem. Information and Computation, **83**(1), 1–20, October 1989

[11] M. Weinard and G. Schnitger: On the Greedy Superstring Conjecture. In Proceedings of the 23rd Conference on Foundations of Software Technology and Theoretical Computer Science, Mumbai, India, LNCS 2914, 387–398, December 2003. Extended version:
`http://www.thi.informatik.uni-frankfurt.de/~weinard/ /indexE.html`

[12] R. Wunderling: Paralleler und Objektorientierter Simplex-Algorithmus. Ph.D. thesis, ZIB technical report TR 96-09, Berlin, 1996.
`http://www.zib.de/PaperWeb/abstracts/TR-96-09`