On the Constrained Longest Common Subsequence Problem

Anna Gorbenko

Abstract—The problem of the longest common subsequence is a classical distance measure for strings. There have been several attempts to accommodate longest common subsequences along with some other distance measures. There are a large number of different variants of the problem. In this paper, we consider the constrained longest common subsequence problem for two strings and arbitrary number of constraints. In particular, we consider an explicit reduction from the problem to the satisfiability problem and present experimental results for different satisfiability algorithms. It should be noted that different regularities in experimentally obtained data reveal important information about the underlying physical system. In this paper, we consider the problem of systematic monitoring of passenger flows. In particular, we use constrained longest common subsequences for tracking the image features.

Index Terms—longest common subsequence, satisfiability problem, feature tracking, genetic algorithms.

I. INTRODUCTION

VARIOUS algorithms on sequences of symbols have been studied for a long time and now form a fundamental part of computer science (see e.g. [1]–[3]). One of the most important problems in analysis of sequences is the longest common subsequence problem. This problem has been extensively studied over the last thirty years (see [4]–[8]). There are a large number of applications of different variants of this problem (see e.g. [9]–[11]). In particular, we can mention robot self-awareness (see e.g. [12]–[19]), mining for interesting patterns (see e.g. [20], [21]), and automatic generation of recognition modules (see e.g. [22]).

In this paper, we consider the constrained longest common subsequence problem that was proposed in [23]. It should be noted that there are a number of efficient algorithms for the constrained longest common subsequence problem for two strings and one constraint (see e.g. [24]–[31]).

However, in general case, the constrained longest common subsequence problem is NP-hard [32], [33]. In particular, the NP-hardness and inapproximability of the constrained longest common subsequence problem for two strings and arbitrary number of constraints was proved in [32]. This paper is devoted to the consideration of efficient algorithms for the constrained longest common subsequence problem.

II. PRELIMINARIES

Let \( \Sigma = \{a_1, a_2, \ldots, a_m\} \) be a fixed alphabet. Given two strings \( S \) and \( T \) over \( \Sigma \), the string \( T \) is a subsequence of \( S \) if \( T \) can be obtained from \( S \) by deleting some letters from \( S \). Note that the order of the remaining letters of \( S \) should be preserved. The length of a string \( S \) is the number of letters in it. The length of a string \( S \) is denoted as \( |S| \). For simplicity, we use \( S[i] \) to denote the \( i \)th letter in the string \( S \), and \( S[i, j] \) to denote the substring of \( S \) consisting of the \( i \)th letter through the \( j \)th letter.

Given two strings \( S_1 \) and \( S_2 \), the classic longest common subsequence problem asks for a longest string \( T \) that is a subsequence of both \( S_1 \) and \( S_2 \). The decision version of the constrained longest common subsequence problem for two strings and arbitrary number of constraints can be formulated as following.

CONstrained Longest common subsequence PROblem (C-LCS-D):

Instance: Two strings \( S_1 \) and \( S_2 \) over \( \Sigma \), a set \( \{T_1, T_2, \ldots, T_n\} \) of strings over \( \Sigma \), a positive integer \( k \).

Question: Is there a string \( T \) over \( \Sigma \) such that

- \( |T| \geq k \);
- \( T \) is a common subsequence of \( S_1 \) and \( S_2 \);
- \( T_i \) is a subsequence of \( T \), for all \( 1 \leq i \leq n \)?

III. AN EXPLICIT REDUCTION FROM C-LCS-D TO THE SATISFIABILITY PROBLEM

The satisfiability problem was the first known NP-complete problem. Different variants of the satisfiability problem were considered. In particular, the 3-satisfiability problem (3SAT) is the problem of determining if the variables of a given Boolean function in conjunctive normal form with 3 variables per clause (3-CNF) can be assigned in such a way as to make the formula evaluate to true (see e.g. [34]).

Note that 3SAT is NP-complete. However, there are a large number of different efficient satisfiability algorithms. Encoding various hard problems as instances of the satisfiability problem and solving them with efficient satisfiability algorithms has caused considerable interest (see e.g. [35]–[38]). In this paper, we consider an explicit reduction from C-LCS-D to the satisfiability problem.

Let

\[
\varphi[p, 1] = \bigwedge_{1 \leq i \leq k} \bigvee_{1 \leq j \leq |S_p|} x[p, i, j],
\]

\[
\varphi[p, 2] = \bigwedge_{1 \leq i \leq k} \bigvee_{1 \leq j_1 < j_2 \leq |S_p|} (\neg x[p, i, j_1[1]] \lor \neg x[p, i, j_2[2]]),
\]

The work was partially supported by Analytical Departmental Program ”Developing the scientific potential of high school”, RFBR, research project No. 13-01-00048 a, and Ural Federal University development program with the financial support of young scientists.

(Advance online publication: 29 November 2013)
\[ \varphi[p,3] = \bigwedge_{1 \leq i[1] < i[2] \leq k, 1 \leq j[2] < j[1] \leq S_p} (\neg x[p, i[1], j[1]] \vee \neg x[p, i[2], j[2]]) \]

\[ \psi = \bigwedge_{1 \leq i \leq k, 1 \leq j[1] \leq S_1, 1 \leq j[2] < j[1] \leq k} (\neg x[1, i, j[1]]) \vee (\neg x[2, i, j[2]]) \]

\[ \rho[q,1] = \bigwedge_{1 \leq i \leq |T_q|, 1 \leq j \leq k} y[q, i, j] \]

\[ \rho[q,2] = \bigwedge_{1 \leq i \leq |T_q|, 1 \leq j[2] < j[1] \leq k} (\neg y[q, i, j[1]]) \vee (\neg y[q, i, j[2]]) \]

\[ \rho[q,3] = \bigwedge_{1 \leq i[1] < i[2] \leq |T_q|, 1 \leq j[2] < j[1] \leq k} (\neg y[q, i[1], j[1]]) \vee (\neg y[q, i[2], j[2]]) \]

\[ \tau[q] = \bigwedge_{1 \leq j \leq k, 1 \leq i \leq |T_q|, 1 \leq s \leq |S_1|, T_q[i] \neq S_1[i]} (\neg y[q, i, j]) \]

\[ \xi = (\bigwedge_{1 \leq i \leq 2} \varphi[i, j]) \wedge (\bigwedge_{1 \leq j \leq 3} \psi) \wedge (\bigwedge_{1 \leq i \leq n} \rho[i, j]) \wedge (\bigwedge_{1 \leq j \leq n} \tau[i]) \]

IV. MONITORING OF PASSENGER FLOWS

In this section, we consider the problem of systematic monitoring of passenger flows. In general, we can apply various face and body detectors to images for solution of this problem. However, low quality of data (see e.g. Figure 1) makes this task very difficult. To simplify this task, it is natural to use some method of tracking the image features. In particular, we can represent a sequence of features as a string.

We can consider strings of features of current and previous images and use longest common subsequence to establish a feature correspondence. However, successful feature tracking have different values for different types of features. In particular, features that extracted from the images of passengers have critical importance for solution of the problem of systematic monitoring of passenger flows. If we use classic longest common subsequences, then we may lose some important features (see e.g. Figure 2). In case of Figure 2, if we consider a classic longest common subsequence, then subsequence of features, which extracted from the back of the chair (white area), can absorb features of passenger. In this case, we lose corresponding passenger. Therefore, we use constraint longest common subsequences.

We consider image corners (see [40]), vertical edges, and color features (see [41]) as the set of features of the environment. Let

\[ B_1 = \{b_{1,1}, b_{1,2}, \ldots, b_{1,n}\} \]

be an alphabet of image corners. Let

\[ B_2 = \{b_{2,1}, b_{2,2}, \ldots, b_{2,n}\} \]

be an alphabet of vertical edges. Let

\[ B_3 = \{b_{3,1}, b_{3,2}, \ldots, b_{3,n}\} \]

be an alphabet of color features. In this case,

\[ B_1 \cup B_2 \cup B_3 \]

is the alphabet of features of the environment.

We use Haar cascades (see e.g. [42], [43]) for initial detection of passengers. Haar cascades allow us to obtain a set of various features, parts of faces, parts of bodies, pieces of clothing and so on. We classify these features based on their motion. This classification allows us to select areas of
Let $\mathcal{C}_1 = \{c_{1,1}, c_{1,2}, \ldots, c_{1,\gamma_1}\}$ be an alphabet of unusual patterns. Let $\mathcal{C}_2 = \{c_{2,1}, c_{2,2}, \ldots, c_{2,\gamma_2}\}$ be an alphabet of passenger color features.

Let $f$ be a feature. The set of pixels of the feature $f$ at time $t$ we denote by $S_f(t)$. We consider

$$\left( \max_{(x,y) \in S_f(t)} x, \min_{(x,y) \in S_f(t)} y \right)$$

as the coordinates of the feature $f$ at time $t$. Let $f \langle x(t), y(t) \rangle$ be a feature $f$ with coordinates $(x(t), y(t))$ at time $t$. We assume that $f_1 \langle x_1(t), y_1(t) \rangle < f_2 \langle x_2(t), y_2(t) \rangle$ if and only if

$$\{(x_1(t - 1) < x_2(t - 1)) \wedge (x_1(t) < x_2(t) + \epsilon),\}
\{(x_1(t - 1) = x_2(t - 1)) \wedge (y_1(t - 1) > y_2(t - 1)) \wedge (x_1(t) < x_2(t) + \epsilon),\}
\{(x_2(t) \geq x_1(t) + \epsilon),\}
\{(x_2(t) \geq x_1(t) + \epsilon),\}$$

where $\epsilon$ is a constant that depends on the resolution of the images. Under this assumption, we can construct the string

$$F(t) = f_{t,1} \langle x_{t,1}(t), y_{t,1}(t) \rangle f_{t,2} \langle x_{t,2}(t), y_{t,2}(t) \rangle \ldots$$

of all features at time $t$. 

(Fig. 2: An example of a loss of information.

Fig. 3: Areas of interest.

interest and identify these areas or sets of these areas as passengers (see e.g. Figure 3). After classification, we use unusual patterns and passenger color features as features for tracking.)
Let

\[ P(t) = \{ P_1(t), P_2(t), \ldots, P_n(t) \} \]

be a set of passengers at time \( t \). We create a set of strings of features of passengers. In particular, we assume that the string \( Z_j \) of features of \( P_j(t) \) is the longest subsequence of \( F(0) \) such that \( F(0)[i] \in P_j(t) \), for all \( i \) and \( j \).

We assume that

\[ f_1(x_1(t), y_1(t)) = f_2(x_2(t), y_2(t)) \]

if and only if \( f_1 = f_2 \). For feature tracking, we consider strings \( F(t-1) \) and \( F(t) \) and the set of constraints

\[ \{ Z_j \mid 1 \leq j \leq \alpha \} \].

If we can solve the constrained longest common subsequence problem, then we use constrained longest common subsequence to localize passengers. If we cannot solve the problem, we again use Haar cascades for initial detection of passengers and restart the process. Usage of constrained longest common subsequences allows us to minimize number of runs of very complicated process of classification.

In our experiments, we consider video files that have been received from one bus camera. We have considered the following parameters: resolution; infrared video and color video; number of passengers. We have created following data sets:

- \( \text{Set}[1] \): resolution 1280 × 800, infrared video, number of passengers < 5;
- \( \text{Set}[2] \): resolution 640 × 400, infrared video, number of passengers < 5;
- \( \text{Set}[3] \): resolution 1280 × 800, infrared video, 5 ≤ number of passengers < 10;
- \( \text{Set}[4] \): resolution 640 × 400, infrared video, 5 ≤ number of passengers < 10;
- \( \text{Set}[5] \): resolution 1280 × 800, infrared video, number of passengers < 15;
- \( \text{Set}[6] \): resolution 640 × 400, infrared video, number of passengers < 15;
- \( \text{Set}[7] \): resolution 1280 × 800, color video, number of passengers < 5;
- \( \text{Set}[8] \): resolution 640 × 400, color video, number of passengers < 5;
- \( \text{Set}[9] \): resolution 1280 × 800, color video, 5 ≤ number of passengers < 10;
- \( \text{Set}[10] \): resolution 640 × 400, color video, 5 ≤ number of passengers < 10;
- \( \text{Set}[11] \): resolution 1280 × 800, color video, number of passengers < 15;
- \( \text{Set}[12] \): resolution 640 × 400, color video, number of passengers < 15.

For any data set \( \text{Set}[i] \), let \( n_{f_{\text{LCS}}}(\text{Set}[i]) \) be the average number of frames before the loss of first passenger during longest common subsequence tracking, \( n_{f_{\text{LCSS}}}(\text{Set}[i]) \) be the average number of frames before the loss of first passenger during constrained longest common subsequence tracking. Selected experimental results are given in Table I.

### Table I
THE AVERAGE NUMBER OF FRAMES DURING TRACKING

<table>
<thead>
<tr>
<th>( i )</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>( n_{f_{\text{LCS}}}(\text{Set}[i]) )</td>
<td>19</td>
<td>16</td>
<td>10</td>
<td>7</td>
<td>4</td>
<td>3</td>
<td>24</td>
<td>20</td>
<td>11</td>
<td>8</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>( n_{f_{\text{LCSS}}}(\text{Set}[i]) )</td>
<td>304</td>
<td>112</td>
<td>94</td>
<td>57</td>
<td>36</td>
<td>27</td>
<td>743</td>
<td>481</td>
<td>366</td>
<td>154</td>
<td>217</td>
<td>125</td>
</tr>
</tbody>
</table>

### Table II
THE QUALITY OF PREDICTION

<table>
<thead>
<tr>
<th>( t )</th>
<th>10^7</th>
<th>10^6</th>
<th>10^5</th>
<th>10^4</th>
<th>10^3</th>
<th>10^2</th>
<th>10^1</th>
<th>10^0</th>
<th>10^-1</th>
<th>10^-2</th>
<th>10^-3</th>
<th>10^-4</th>
</tr>
</thead>
<tbody>
<tr>
<td>C-LCS</td>
<td>95%</td>
<td>98%</td>
<td>98%</td>
<td>99%</td>
<td>99%</td>
<td>99%</td>
<td>99%</td>
<td>99%</td>
<td>99%</td>
<td>99%</td>
<td>99%</td>
<td>99%</td>
</tr>
<tr>
<td>c-FLCS</td>
<td>91%</td>
<td>96%</td>
<td>97%</td>
<td>98%</td>
<td>98%</td>
<td>98%</td>
<td>98%</td>
<td>98%</td>
<td>98%</td>
<td>98%</td>
<td>98%</td>
<td>98%</td>
</tr>
<tr>
<td>LCSS</td>
<td>76%</td>
<td>83%</td>
<td>88%</td>
<td>96%</td>
<td>96%</td>
<td>96%</td>
<td>96%</td>
<td>96%</td>
<td>96%</td>
<td>96%</td>
<td>96%</td>
<td>96%</td>
</tr>
</tbody>
</table>

V. MINING FOR INTERESTING PATTERNS

It is well-known that feature selection is one of the most important problems of image processing (see e.g. [44], [45]). A common technique for feature selection is the discovery of frequent patterns.

Note that we can use fluents [46] to express temporal patterns. This approach allows us to consider different string problems to mine interesting patterns. Since different versions of the longest common subsequence problem frequently used to mine interesting patterns (see e.g. [9], [11], [47]–[49]), it is natural to use C-LCS to mine interesting patterns.

Mining for interesting patterns has a number of applications in robot self-awareness (see e.g. [9], [11]). In particular, we need some system of prediction of collisions to build robot with ability to anticipate the motions (see e.g. [16], [50], [51]).

The c-fragment longest arc-preserving common subsequence problem (c-FLCS) and the problem of the longest common subsequence over the set (LCSS) were used to create sets of interesting patterns for prediction of collisions (see [9], [11]). These sets were used by recurrent neural network for prediction of collisions of mobile robot. It is clear that we can apply C-LCS to create a set of interesting patterns for prediction of collisions. Let \( t \) be the size of training set. Selected experimental results are shown in Table II.
TABLE III

Experimental results for different test sets for monitoring of passenger flows

<table>
<thead>
<tr>
<th>solver</th>
<th>test</th>
<th>average time</th>
<th>max time</th>
<th>best time</th>
</tr>
</thead>
<tbody>
<tr>
<td>OA[1]</td>
<td>Test[1]</td>
<td>1.13 sec</td>
<td>2.19 min</td>
<td>0.04 sec</td>
</tr>
<tr>
<td>OA[2]</td>
<td>Test[1]</td>
<td>1.38 sec</td>
<td>1.28 min</td>
<td>0.06 sec</td>
</tr>
<tr>
<td>OA[3]</td>
<td>Test[1]</td>
<td>0.27 sec</td>
<td>42.14 sec</td>
<td>0.02 sec</td>
</tr>
<tr>
<td>OA[4]</td>
<td>Test[1]</td>
<td>0.08 sec</td>
<td>6.15 sec</td>
<td>0.012 sec</td>
</tr>
<tr>
<td>OA[5]</td>
<td>Test[1]</td>
<td>0.03 sec</td>
<td>3.19 sec</td>
<td>0.007 sec</td>
</tr>
<tr>
<td>GSAT</td>
<td>Test[1]</td>
<td>0.54 sec</td>
<td>1.17 min</td>
<td>0.05 sec</td>
</tr>
<tr>
<td>OA[1]</td>
<td>Test[2]</td>
<td>1.28 sec</td>
<td>2.43 min</td>
<td>0.12 sec</td>
</tr>
<tr>
<td>OA[2]</td>
<td>Test[2]</td>
<td>1.54 sec</td>
<td>1.57 min</td>
<td>0.09 sec</td>
</tr>
<tr>
<td>OA[3]</td>
<td>Test[2]</td>
<td>0.35 sec</td>
<td>47.32 sec</td>
<td>0.03 sec</td>
</tr>
<tr>
<td>OA[4]</td>
<td>Test[2]</td>
<td>0.24 sec</td>
<td>33.2 sec</td>
<td>0.021 sec</td>
</tr>
<tr>
<td>OA[5]</td>
<td>Test[2]</td>
<td>0.13 sec</td>
<td>19.8 sec</td>
<td>0.014 sec</td>
</tr>
<tr>
<td>GSAT</td>
<td>Test[2]</td>
<td>0.87 sec</td>
<td>59.13 sec</td>
<td>0.086 sec</td>
</tr>
<tr>
<td>OA[5]</td>
<td>Test[3]</td>
<td>2.52 sec</td>
<td>1.44 min</td>
<td>0.022 sec</td>
</tr>
<tr>
<td>GSAT</td>
<td>Test[3]</td>
<td>8.43 sec</td>
<td>5.12 min</td>
<td>0.121 sec</td>
</tr>
<tr>
<td>OA[4]</td>
<td>Test[4]</td>
<td>42.8 sec</td>
<td>6.27 min</td>
<td>0.043 sec</td>
</tr>
<tr>
<td>GSAT</td>
<td>Test[4]</td>
<td>1.83 min</td>
<td>19.73 min</td>
<td>1.15 sec</td>
</tr>
<tr>
<td>OA[1]</td>
<td>Test[5]</td>
<td>25.02 min</td>
<td>2.05 min</td>
<td>1.08 sec</td>
</tr>
<tr>
<td>GSAT</td>
<td>Test[5]</td>
<td>1.83 min</td>
<td>19.73 min</td>
<td>1.15 sec</td>
</tr>
<tr>
<td>OA[3]</td>
<td>Test[6]</td>
<td>47.3 min</td>
<td>3.69 hr</td>
<td>1.1 min</td>
</tr>
</tbody>
</table>

VI. SAT SOLVERS FOR C-LCS-D

We use genetic algorithms OA[1] (see [52]), OA[2] (see [53]), OA[3] (see [54]), OA[4] (see [55]), and OA[5] (see [56]) for the satisfiability problem to obtain optimal solutions of C-LCS-D. Also, we have considered GSAT with adaptive score function (see [57]).

We have used heterogeneous cluster (500 calculation nodes, Intel Core i7). Each test was runned on a cluster of at least 100 nodes. Note that due to restrictions on computation time (20 hours) we used savepoints.

In our experiments, we use real world data for monitoring of passenger flows. In particular, we consider two test sets,

- **Test[1]**: average length of strings = 150,
  average number of constraints = 7;
- **Test[2]**: average length of strings = 200,
  average number of constraints = 15.

Also, we consider four synthetic test sets for monitoring of passenger flows,

- **Test[3]**: average length of strings = 150,
  average number of constraints = 7;
- **Test[4]**: average length of strings = 200,
  average number of constraints = 15;
- **Test[5]**: average length of strings = 1000,
  average number of constraints = 100;
- **Test[6]**: average length of strings = 6000,
  average number of constraints = 200.

Selected experimental results are given in Table III.

We have considered real world data for mining for interesting patterns (see [9]). Selected experimental results are given in Table IV.

TABLE IV

Experimental results for mining for interesting patterns

<table>
<thead>
<tr>
<th>solver</th>
<th>average time</th>
<th>max time</th>
<th>best time</th>
</tr>
</thead>
<tbody>
<tr>
<td>OA[1]</td>
<td>47 sec</td>
<td>1.67 hr</td>
<td>4.8 sec</td>
</tr>
<tr>
<td>OA[2]</td>
<td>51 sec</td>
<td>1.83 hr</td>
<td>3.91 sec</td>
</tr>
<tr>
<td>OA[3]</td>
<td>45 sec</td>
<td>2.29 hr</td>
<td>7.53 sec</td>
</tr>
<tr>
<td>OA[4]</td>
<td>12 sec</td>
<td>19 sec</td>
<td>1.23 sec</td>
</tr>
<tr>
<td>OA[5]</td>
<td>4.2 sec</td>
<td>14.7 sec</td>
<td>3.6 sec</td>
</tr>
<tr>
<td>GSAT</td>
<td>49 sec</td>
<td>3.25 hr</td>
<td>1.82 sec</td>
</tr>
</tbody>
</table>

VII. A TASK-LEVEL ROBOT LEARNING FROM DEMONSTRATION

Robot task learning has received significant attention recently (see e.g. [58]). In particular, the longest common

(Advance online publication: 29 November 2013)
subsequence of the state sequences can be used for task
generalization (see e.g. [59]). We can assume that the longest
common subsequence of two demonstrations constitute the
generalized task model. The other actions can be considered
as alternative paths, noise, or alternative tasks. Also, the
longest common subsequence of the state sequences can
be used for task learning from demonstration. In particular,
we can consider task learning with one training example
prepared by a human. In this case, the robot’s state sequences
are processed to evaluate the robot’s performance given the
specific training example prepared by a human (see e.g. [60]).
It is natural to use demonstrations of different simple tasks
to learn a complex task. In this case, we need a common
subsequence of two demonstrations of the complex task such
that task models of simple tasks are subsequences of the
common subsequence of two demonstrations. It is clear that
we can use the constrained longest common subsequence for
solution of this problem.

In our experiments, we consider Neato XV-11 [61] with an
onboard computer and a camera (see Figure 4). We consider
a simple genetic algorithm that evolves a population of
sequences of motor primitives and tries to obtain a sequence
of motor primitives for given trajectory. At first, we assume
that we have only one human training example of some
trajectory $H$. We consider the robot’s state sequence $R$ and
use the length of the longest common subsequence of $H$ and
$R$ as the value of the fitness function for $R$. This
genetic algorithm we denote by $T_1$. Also we consider genetic
algorithm $T_2$ where we assume that we have human training
example of some trajectory $H$ and two human training
examples $H_1$ and $H_2$ of some parts of the trajectory. We
consider the robot’s state sequences $R$, $R_1$, and $R_2$ for $H$
and $H_1$, and $H_2$. Let $T_1$ be the longest common subsequence
of $H_1$ and $R_1$ where $i \in \{1, 2\}$. Let $T$ be the constrained longest
common subsequence of $H$ and $R$ for $\{T_1, T_2\}$. In $T_2$, we
use the length of $T$ as the value of the fitness function for
$R$.

Let $N_i(n)$ be the average number of generations of $T_i$ that
needed to obtain $H = R$ for $|H| = n$. It is clear that we can use

\[ N(n) = \frac{N_2(n)}{N_1(n)} \]

as a measure of the quality of $T_1$ and $T_2$. Selected experimental results are given in Table V.

<table>
<thead>
<tr>
<th>$n$</th>
<th>$10^2$</th>
<th>$10^3$</th>
<th>$10^4$</th>
<th>$10^5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N(n)$</td>
<td>0.43</td>
<td>0.37</td>
<td>0.12</td>
<td>0.03</td>
</tr>
</tbody>
</table>

It is easy to see that $T_2$ gives us better results. However,
for $T_2$ we need additional human training examples. Now we
consider the following genetic algorithm $T_3$. We consider human
training examples $H^{1}$, $H^{2}$, \ldots, $H^{k}$ for different tasks.
We assume that $T_3$ evolves a population of sequences of
motor primitives and tries to obtain a set of sequences of
motor primitives for trajectories $H^1$, $H^2$, \ldots, $H^k$. Let $R^i$ be
the robot’s state sequence for the trajectory $H^i$. Let $T_{i,j}$ be
the common subsequence of $H^i$, $R^i$, and $H^j$. Let $T^j$
be the constrained longest common subsequence of $H^j$
and $R^i$ for $\{T_{i,j} | i \neq j\}$. In $T_3$, use the length of $T^j$ as
the value of the fitness function for $R^i$. Let

\[ M(n) = \frac{N_3(n)}{N_1(n)} \]

where $N_3(n)$ be the average number of generations of $T_3$
that needed to obtain $H^j = R^j$ for $|H| = n$, $1 \leq j \leq k$.
Selected experimental results are given in Table VI.

<table>
<thead>
<tr>
<th>$n$</th>
<th>$10^2$</th>
<th>$10^3$</th>
<th>$10^4$</th>
<th>$10^5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M(n)$</td>
<td>0.56</td>
<td>0.18</td>
<td>0.041</td>
<td>0.0082</td>
</tr>
</tbody>
</table>

It is clear that $T_3$ demonstrates good performance and does
not require additional human training examples. However, $T_3$
can be used only in the case when we have many learning
tasks.

VIII. CONCLUSION

In this paper, we have considered the constrained longest
common subsequence problem for two strings and arbitrary
number of constraints. In particular, we have considered
applications of the constrained longest common subsequence
problem for monitoring of passenger flows and task-level
robot learning from demonstration.

We have proposed an explicit reduction from the
constrained longest common subsequence problem to the satis-
fiability problem. Also, we have presented experimental
results for different satisfiability algorithms. In particular, we
have considered synthetic test sets and real world data for
monitoring of passenger flows.

REFERENCES

[1] V. Popov, “Multiple genome rearrangement by swaps and by element
duplications,” Theoretical Computer Science, vol. 385, no. 1-3, pp. 115-
126, October 2007.
eterized complexity of sequence alignment and consensus,” Theoretical
Computer Science, vol. 147, no. 1-2, pp. 31-54, August 1995.
Anna Gorbenko was born on December 28, 1987. She received her M.Sc. in Computer Science from Department of Mathematics and Mechanics of Ural State University in 2011. She is currently a graduate student at Mathematics and Computer Science Institute of Ural Federal University and a researcher at Department of Intelligent Systems and Robotics of Mathematics and Computer Science Institute of Ural Federal University. She has (co-)authored 4 books and 90 papers.