Applied Mathematical Sciences, Vol. 7, 2013, no. 97, 4821 - 4828
HIKARI Ltd, www.m-hikari.com
http://dx.doi.org/10.12988/ams.2013.36322

On the Shortest Common Parameterized Supersequence Problem

Anna Gorbenko

Department of Intelligent Systems and Robotics
Ural Federal University
620083 Ekaterinburg, Russia
gorbenko.ann@gmail.com

Vladimir Popov

Department of Intelligent Systems and Robotics
Ural Federal University
620083 Ekaterinburg, Russia
Vladimir.Popov@usu.ru

Abstract

In this paper, we consider an approach to solve the problem of the shortest common parameterized supersequence. This approach is based on an explicit reduction from the shortest common parameterized supersequence problem to the 3-satisfiability problem and the maximum 2-satisfiability problem.

Keywords: 3-satisfiability problem, maximum 2-satisfiability problem, shortest common supersequence, parameterized supersequence, **NP**-complete

Investigation of various problems on sequences of symbols forms a fundamental part of computer science (see e.g. [1] – [6]). One of the most important problems in analysis of sequences is the shortest common supersequence problem. There are a large number of different variants of this problem. In particular, we can mention the shortest common parameterized supersequence problem [7]. In this paper, we consider an approach to solve the shortest common parameterized supersequence problem (SCPS). Note that encoding various hard problems as instances of the satisfiability problem and solving them with efficient satisfiability algorithms has caused considerable interest (see e.g. [8] – [23]). An explicit reduction from SCPS to the satisfiability problem was proposed in [7]. In this paper, we consider an explicit reduction from SCPS to the 3-satisfiability problem and the maximum 2-satisfiability problem (MAX2SAT).

Note that

$$\begin{aligned}\alpha \;&\Leftrightarrow\; (\alpha \vee \beta_1 \vee \beta_2) \wedge (\alpha \vee \neg\beta_1 \vee \beta_2) \wedge \\ &\quad (\alpha \vee \beta_1 \vee \neg\beta_2) \wedge (\alpha \vee \neg\beta_1 \vee \neg\beta_2), \qquad (1)\end{aligned}$$

$$\begin{aligned}\vee_{j=1}^{l}\alpha_j \;&\Leftrightarrow\; (\alpha_1 \vee \alpha_2 \vee \beta_1) \wedge \\ &\quad (\wedge_{i=1}^{l-4}(\neg\beta_i \vee \alpha_{i+2} \vee \beta_{i+1})) \wedge (\neg\beta_{l-3} \vee \alpha_{l-1} \vee \alpha_l), \qquad (2)\end{aligned}$$

$$\alpha_1 \vee \alpha_2 \;\Leftrightarrow\; (\alpha_1 \vee \alpha_2 \vee \beta) \wedge (\alpha_1 \vee \alpha_2 \vee \neg\beta), \qquad (3)$$

$$\vee_{j=1}^{4}\alpha_j \;\Leftrightarrow\; (\alpha_1 \vee \alpha_2 \vee \beta_1) \wedge (\neg\beta_1 \vee \alpha_3 \vee \alpha_4) \qquad (4)$$

where $l > 4$.

Using relations (1) – (4) we can obtain an explicit transformation of function ξ from [7] in τ such that $\xi \Leftrightarrow \tau$ and τ is a 3-CNF. Clearly, τ give us an explicit reduction from SCPS to 3SAT.

Now we consider an explicit reduction from SCPS to MAX2SAT. In the decision version MAX2SAT can be formulated as following.

MAX2SAT:

INSTANCE: *Given a set U of variables, a collection $\mathbf{C}$ of clauses over U such that each clause $\mathcal{C} \in \mathbf{C}$ has at most two literals, and a positive integer k.*

QUESTION: *Is there a truth assignment for U that simultaneously satisfies at least k of the clauses in $\mathbf{C}$?*

Now, consider the following boolean function:

$$\begin{aligned}&x \wedge y \wedge z \wedge w \wedge (\neg x \vee \neg y) \wedge (\neg y \vee \neg z) \wedge \\ &(\neg z \vee \neg x) \wedge (x \vee \neg w) \wedge (y \vee \neg w) \wedge (z \vee \neg w). \qquad (5)\end{aligned}$$

It is easy to check that there is no way to satisfy all these clauses. Note that these ten clauses have the following interesting property:

Any truth assignment that satisfies $x \vee y \vee z$ can be extended to satisfy seven of them and no more, while the remaining truth assignment can be extended to satisfy only six of them
(see e.g. [24], proof of theorem 9.2.).

Using this property we can construct η by simply replacing in τ every occurrence of $x \vee y \vee z$ where x, y, and z are some literals by (5) where w is a new variable. It is easy to see that $\tau \Leftrightarrow \eta$. It is clear that η give us explicit reduction from SCPS to MAX2SAT.

There is a well known site on which posted solvers for SAT [25]. In addition to the solvers the site also represented a large set of test problems. This set includes a randomly generated problems of 3SAT. We create a generator of natural instances for SCPS. Also we use test problems from [25].

We use algorithms fgrasp and posit from [25]. Also we use a simple genetic algorithm (SGA) and design our own genetic algorithm (OA) for SAT which based on algorithms from [25].

Consider a boolean function $g(x_1, x_2, \dots, x_n) = \wedge_{i=1}^{m} \mathcal{C}_i$, where $m \geq 1$, and each of the $\mathcal{C}_i$ is the disjunction of one or more literals. Let $|\mathcal{C}_i|$ be a number of literals in $\mathcal{C}_i$. Let $occ(x_i, g)$ be a number of occurrences of x_i in g. Respectively, let $occ(\neg x_i, g)$ be a number of occurrences of x_i in g. For example, if $g = (x_1 \vee x_2) \wedge (\neg x_2 \vee x_3) \wedge (x_1 \vee x_4) \wedge (\neg x_1 \vee x_5) \wedge (x_1 \vee \neg x_4)$, then

$$occ(x_1, g) = 3, occ(\neg x_1, g) = 1,$$

$$occ(x_2, g) = 1, occ(\neg x_2, g) = 1,$$

$$occ(x_3, g) = 1, occ(\neg x_3, g) = 0,$$

$$occ(x_4, g) = 1, occ(\neg x_4, g) = 1,$$

$$occ(x_5, g) = 1, occ(\neg x_5, g) = 0.$$

We consider a number of natural principles that define importance of a variable x_i for satisfiability of boolean function g. These principles suggest us correct values of variables.

1. If
$$\min_{occ(x_i, \mathcal{C}_j) > 0} |\mathcal{C}_j| \leq \min_{occ(\neg x_i, \mathcal{C}_j) > 0} |\mathcal{C}_j|,$$
then $x_i = 1$.

2. If
$$\min_{occ(x_i,\mathcal{C}_j)>0} |\mathcal{C}_j| \geq \min_{occ(\neg x_i,\mathcal{C}_j)>0} |\mathcal{C}_j|,$$
then $x_i = 0$.

3. Given positive integers $p_1, p_2, \ldots, p_m$. If
$$\sum_{1\leq j\leq m, |\mathcal{C}_j|\leq p_j} occ(x_i, \mathcal{C}_j) \geq \sum_{1\leq j\leq m, |\mathcal{C}_j|\leq p_j} occ(\neg x_i, \mathcal{C}_j),$$
then $x_i = 1$.

4. Given positive integers $p_1, p_2, \ldots, p_m, q_1, q_2, \ldots, q_m$. If
$$\sum_{1\leq j\leq m, |\mathcal{C}_j|\leq p_j} occ(x_i, \mathcal{C}_j) \geq \sum_{1\leq j\leq m, |\mathcal{C}_j|\leq q_j} occ(\neg x_i, \mathcal{C}_j),$$
then $x_i = 1$.

5. Given positive integers $p_1, p_2, \ldots, p_m, q_1, q_2, \ldots, q_m$ and a set of rational numbers
$$\mathcal{Q} = \{\alpha_{i,u}, \beta_{i,v} \mid 1 \leq i \leq m, 1 \leq u \leq p_i, 1 \leq v \leq q_i\}.$$
If
$$\sum_{1\leq j\leq m, 1\leq u\leq p_j, |\mathcal{C}_j|=u} \alpha_{j,u} occ(x_i, \mathcal{C}_j) \geq \sum_{1\leq j\leq m, 1\leq v\leq q_j, |\mathcal{C}_j|=v} \beta_{j,v} occ(\neg x_i, \mathcal{C}_j),$$
then $x_i = 1$.

Based on these principles, we can consider the following five types of commands:
$$P_1, P_2, P_3[p_1, p_2, \ldots, p_m], P_4[p_1, p_2, \ldots, p_m, q_1, q_2, \ldots, q_m],$$
$$P_5[p_1, p_2, \ldots, p_m, q_1, q_2, \ldots, q_m, \mathcal{Q}].$$
Also we consider the following three commands for run algorithms: Try_fgrasp, Try_posit, and Try_ga, where Try_ga runs a simple genetic algorithm. Denote by $\mathcal{R}$ the set of commands of these eight types. Arbitrary element of $\mathcal{R}^*$ it is possible to consider as a program for finding values of variables of a boolean function. We assume that such programs are executed on a cluster. Execution of each of commands of type P_i reduces the number of variables of a boolean function by one. Execution of each of commands Try_fgrasp, Try_posit, and Try_ga consists in the run of corresponding algorithm for current

boolean function on a separate set of calculation nodes and the transition to the next command. Algorithms fgrasp [25] and posit [25] we run only on one calculation node. Genetic algorithms can be used in parallel execution. We use auxiliary genetic algorithms which determine the number of calculation nodes and, if necessary, scale commands P_3, P_4, P_5.

Initially, we selected a random subset of $\mathcal{R}^*$. We use a genetic algorithm to select a program from the current subset of $\mathcal{R}^*$ and a genetic algorithm for evolving the current subset of $\mathcal{R}^*$. The evolution of the current subset of $\mathcal{R}^*$ implemented on a separate set of calculation nodes. For every subsequent boolean functions it is used the current subset of $\mathcal{R}^*$ which is obtained by taking into account the results of previous runs.

We use heterogeneous cluster based on three clusters (Cluster USU, Linux, 8 calculation nodes; umt, Linux, 256 calculation nodes; um64, Linux, 124 calculation nodes) [26]. Each test was run on a cluster of at least 100 nodes.

Algorithms fgrasp and posit used only for 3SAT. Respectively, for SAT and MAX2SAT we use only simple genetic algorithm (SGA), and our algorithm (OA). Selected experimental results for 3SAT, SAT, and MAX2SAT are given in Tables 1 and 2.

time	fgrasp	posit	SGA	OA
average	31.8 min	33.9 min	37.4 min	2.7 min
max	22.8 h	26.1 h	28 h	5 h
best	1.4 min	45 sec	19 sec	17 sec

Table 1: Experimental results for 3SAT

time	SGA SAT	OA SAT	SGA MAX2SAT	OA MAX2SAT
average	32.5 min	28.9 min	39.3 min	4.8 min
max	27.1 h	21.2 h	43.4 h	9.1 h
best	23 sec	19 sec	11 sec	21 sec

Table 2: Experimental results for SAT and MAX2SAT

It is interesting to note that the performance of OA depends essentially on the number of letters from Π (see e.g. Table 3 and 4).

In experiments presented in Tables 1 – 4 OA used after 10 000 runs. The dependence of the performance of OA on the number of runs is shown in Table 5.

In this paper we describe an approach to solve the problem SCPS. This approach is based on constructing logical models for SCPS. Also we consider

time	10 %	30 %	60 %	90 %
average	17.2 min	2.1 min	1.9 min	8.4 min
max	4.6 h	2.6 h	56 min	5 h
best	41 sec	17 sec	20 sec	18 sec

Table 3: Experimental results for 3SAT runs of OA depending on the number of letters from Π

some genetic and local search algorithms for the problem. Our experimental results showed that our reduction to 3SAT with SAT-solver can be used as a test bed for intelligent algorithms for SCPS.

The main question that is of interest for future research is to study the dependence of complexity of solution on the distribution of letters from Π.

ACKNOWLEDGEMENTS. 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.

time	10 %	30 %	60 %	90 %
average	41.4 min	23.2 min	24.5 min	36.3 min
max	19.6 h	14.4 h	17.8 h	21.2 h
best	25 sec	19 sec	20 sec	23 sec

Table 4: Experimental results for SAT runs of OA depending on the number of letters from Π

time	100	500	1000	5000
average	44.6 min	35.8 min	33.6 min	5.3 min
max	29.2 h	27.1 h	15.4 h	6.4 h
best	11.8 h	4.3 min	26 sec	17 sec

Table 5: Experimental results for 3SAT runs of OA depending on the number of runs

References

[1] V. Yu. Popov, Computational complexity of problems related to DNA sequencing by hybridization, *Doklady Mathematics*, 72 (2005), 642-644.

[2] V. Popov, The approximate period problem for DNA alphabet, *Theoretical Computer Science*, 304 (2003), 443-447.

[3] V. Popov, The Approximate Period Problem, *IAENG International Journal of Computer Science*, 36 (2009), 268-274.

[4] V. Popov, Approximate Periods of Strings for Absolute Distances, *Applied Mathematical Sciences*, 6 (2012), 6713-6717.

[5] V. Popov, Sorting by prefix reversals, *IAENG International Journal of Applied Mathematics*, 40 (2010), 247-250.

[6] V. Popov, Multiple genome rearrangement by swaps and by element duplications, *Theoretical Computer Science*, 385 (2007), 115-126.

[7] A. Gorbenko and V. Popov, The Shortest Common Parameterized Supersequence Problem, *Applied Mathematical Sciences*, 7 (2013), 2373-2380.

[8] A. Gorbenko and V. Popov, The Problem of Finding Two Edge-Disjoint Hamiltonian Cycles, *Applied Mathematical Sciences*, 6 (2012), 6563-6566.

[9] A. Gorbenko and V. Popov, Footstep Planning for Humanoid Robots, *Applied Mathematical Sciences*, 6 (2012), 6567-6571.

[10] A. Gorbenko and V. Popov, Hamiltonian Alternating Cycles with Fixed Number of Color Appearances, *Applied Mathematical Sciences*, 6 (2012), 6729-6731.

[11] A. Gorbenko and V. Popov, Task-resource Scheduling Problem, *International Journal of Automation and Computing*, 9 (2012), 429-441.

[12] A. Gorbenko and V. Popov, Longest Common Parameterized Subsequences with Parameterized Common Substring, *Applied Mathematical Sciences*, 7 (2013), 2341-2345.

[13] A. Gorbenko and V. Popov, The Minimum k-Cover Problem, *Applied Mathematical Sciences*, 7 (2013), 2347-2352.

[14] A. Gorbenko and V. Popov, The Shortest Common Superstring Problem, *Applied Mathematical Sciences*, 7 (2013), 2353-2356.

[15] A. Gorbenko and V. Popov, Computational Experiments for the Problem of Footstep Planning for Humanoid Robots, *Applied Mathematical Sciences*, 7 (2013), 2357-2372.

[16] A. Gorbenko and V. Popov, The Minimum Test Collection Problem, *Applied Mathematical Sciences*, 7 (2013), 1191-1193.

[17] A. Gorbenko and V. Popov, The Farthest Substring Problem, *Applied Mathematical Sciences*, 7 (2013), 1209-1212.

[18] A. Gorbenko and V. Popov, The Swap Common Superstring Problem, *Applied Mathematical Sciences*, 7 (2013), 609-614.

[19] A. Gorbenko and V. Popov, The String Barcoding Problem, *Applied Mathematical Sciences*, 7 (2013), 615-622.

[20] A. Gorbenko and V. Popov, The c-Fragment Longest Arc-Preserving Common Subsequence Problem, *IAENG International Journal of Computer Science*, 39 (2012), 231-238.

[21] A. Gorbenko and V. Popov, The set of parameterized k-covers problem, *Theoretical Computer Science*, 423 (2012), 19-24.

[22] A. Gorbenko and V. Popov, On the Problem of Placement of Visual Landmarks, *Applied Mathematical Sciences*, 6 (2012), 689-696.

[23] A. Gorbenko and V. Popov, The Longest Common Parameterized Subsequence Problem, *Applied Mathematical Sciences*, 6 (2012), 2851-2855.

[24] C. H. Papadimitriou, Computational complexity, Addison Wesley, Reading, CA, 1994.

[25] http://people.cs.ubc.ca/∼hoos/SATLIB/index-ubc.html

[26] http://parallel.imm.uran.ru/mvc_now/hardware/ supercomp.htm

Received: June 15, 2013