Volume 22 Number 4

ACTA CYBERNETICA

Szeged, 2016

Helmut Jürgensen
Department of Computer Science
Middlesex College
The University of Western Ontario
London, Canada
hjj@csd.uwo.ca

Zoltan Kato
Department of Image Processing
and Computer Graphics
Szeged, Hungary
kato@inf.u-szeged.hu

Alice Kelemenová
Institute of Computer Science
Silesian University at Opava
Opava, Czech Republic
Alica.Kelemenova@fpf.slu.cz

László Lovász
Department of Computer Science
Eötvös Loránd University
Budapest, Hungary
lovasz@cs.elte.hu

Gheorghe Păun
Institute of Mathematics of the
Romanian Academy
Bucharest, Romania
George.Paun@imar.ro

András Prékopa
Department of Operations Research
Eötvös Loránd University
Budapest, Hungary
prekopa@cs.elte.hu

Arto Salomaa
Department of Mathematics
University of Turku
Turku, Finland
asalomaa@utu.fi

László Varga
Department of Software Technology
and Methodology
Eötvös Loránd University
Budapest, Hungary
varga@ludens.elte.hu

Heiko Vogler
Department of Computer Science
Dresden University of Technology
Dresden, Germany
Heiko.Vogler@tu-dresden.de

Gerhard J. Woeginger
Department of Mathematics and
Computer Science
Eindhoven University of Technology
Eindhoven, The Netherlands
gwoegi@win.tue.nl

Acta Cybernetica **22** (2016) 715–733.

Nonlinear Symbolic Transformations for Simplifying Optimization Problems*

Elvira Antal[†‡] and Tibor Csendes[†]

Abstract

The theory of nonlinear optimization traditionally studies numeric computations. However, increasing attention is being paid to involve computer algebra into mathematical programming. One can identify two possibilities of applying symbolic techniques in this field. Computer algebra can help the modeling phase by producing alternate mathematical models via symbolic transformations. The present paper concentrates on this direction. On the other hand, modern nonlinear solvers use more and more information about the structure of the problem through the optimization process leading to hybrid symbolic-numeric nonlinear solvers.

This paper presents a new implementation of a symbolic simplification algorithm for unconstrained nonlinear optimization problems. The program can automatically recognize helpful transformations of the mathematical model and detect implicit redundancy in the objective function.

We report computational results obtained for standard global optimization test problems and for other artificially constructed instances. Our results show that a heuristic (multistart) numerical solver takes advantage of the automatically produced transformations.

New theoretical results will also be presented, which help the underlying method to achieve more complicated transformations.

Keywords: nonlinear optimization, reformulation, Mathematica

1 Introduction

Application of symbolic techniques to rewrite or solve optimization problems are a promising and emerging field of mathematical programming. Symbolic preprocessing of linear programming problems [11] is the classic example, this kind of transformation was implemented in the AMPL processor about twenty years ago

*This work was partially supported by the Grant TÁMOP-4.2.2.A-11/1/KONV-2012-0073.

[†]University of Szeged, Institute of Informatics, H-6720 Szeged, Árpád tér 2, Hungary, E-mail: antale@inf.u-szeged.hu, csendes@inf.u-szeged.hu

[‡]Kecskemét College, Faculty of Mechanical Engineering and Automation, Hungary

DOI: 10.14232/actacyb.22.4.2016.1

as an automatic "presolving" mechanism [7, 8]. A more recent example is the assistance of (mixed-) integer nonlinear programming solvers, as in the Reformulation-Optimization Software Engine of Liberti et al. [10]. In this field, the relaxation of some constraints or increasing the dimension of the problem could be reasonable to achieve feasibility.

However, as the work of Csendes and Rapcsák [5, 14] shows, it is also possible to produce equivalent models of an unconstrained nonlinear optimization problem via symbolic transformations automatically, while bijective transformations between the optima of the models are constructed. This method is capable of eliminating redundant variables or simplifying the problem in other ways.

The present paper is organized as follows. Section 2 summarizes briefly the results of Csendes and Rapcsák [5, 14] and presents a new, proper Mathematica implementation of their method together with a comparison with the earlier reported Maple prototype [2]. Section 3 presents computational results to show that the automatically produced transformations can help a traditional heuristic numeric solver, namely Global [4], to reduce computation times and function evaluations. Section 4 extends the theory of the mentioned method in two directions: describes parallel substitutions and introduces constraints into the model.

2 The Simplifier Method

2.1 Theoretical Background

We concentrate on unconstrained nonlinear optimization problems of the form

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} \quad f(\boldsymbol{x}), \tag{1}$$

where $f(\boldsymbol{x}) : \mathbb{R}^n \to \mathbb{R}$ is a smooth function given by a formula, i.e. a symbolic expression. "Expression" denotes a well-formed, finite combination of symbols (constants, variables, operators, function names and brackets), usually realized in computer algebra systems with a list (for example, a nested list of pointers in Mathematica [20]), or a directed acyclic graph [15]. In the subsequent description, vectors are denoted by boldface letters, sets by capital letters, and functions by small letters. The meaning of z_i depends on context: it denotes the i^{th} element of a vector $\boldsymbol{z}$ or an ordered set Z. We will use the function notation $v(\boldsymbol{z})$ to represent a v expression, which can contain any variable z_i, any real constant, and any function name.

The simplifier method aims to recognize, whether (1) could be transformed into an equivalent formulation, which is better in the following senses: the new formulation has fewer arithmetic operations to execute during evaluation, the dimension of the problem is less, or it is simpler to solve for another reason. Equivalent means here, that a bijective transformation can be given between the optima of the original and those of the transformed problem.

Csendes and Rapcsák [5] showed that an objective function $g(\boldsymbol{y})$ is equivalent to $f(\boldsymbol{x})$ in (1), if we get $g(\boldsymbol{y})$ by the following transformation:

- apply a substitution in $f(\boldsymbol{x})$:

$$y_i := h(\boldsymbol{x}), \quad 1 \leq i \leq n,$$

where $h(\boldsymbol{x})$ is a smooth function with a range $\mathbb{R}$, and $h(\boldsymbol{x})$ is strictly monotonic as a function of at least one variable x_i,

- rename the remaining variables:

$$y_j := x_j, \quad j = 1, \ldots, i-1, i+1, \ldots, n,$$

and

- omit the variables y_i without presence in the evolving objective function.

The term *appropriate substitution* will refer to an $y_i = h(\boldsymbol{x})$ substitution, where

- $h(\boldsymbol{x})$ satisfies the criteria being smooth, monotonic in at least one variable x_i, and its range is equal to $\mathbb{R}$,
- $h(\boldsymbol{x})$ covers (characterizes all occurrences of) at least one variable x_i, that is, x_i could be removed totally from the optimization problem by substituting $h(\boldsymbol{x})$ by y_i, and
- $y_i = h(\boldsymbol{x})$ is not a simple renaming, that is, $h(\boldsymbol{x}) \neq x_i$, $i = 1, \ldots, n$.

After applying a transformation with an appropriate substitution $y_i = h(\boldsymbol{x})$, $\boldsymbol{y}$ has at most the same dimension as $\boldsymbol{x}$. Redundant variables can be eliminated, if $h(\boldsymbol{x})$ covers two or more variables. In other words, we have the possibility to recognize whether the model can be formalized with a smaller set of variables. However, these are sufficient, but not necessary conditions for simplifier transformations.

For example, consider $f(x_1, x_2) = (x_1 + x_2)^2$. It is equivalent to minimize $g(y_1) = y_1^2$, and the optimal values of the original variables x_1 and x_2 can be set by the symbolic equation $y_1 = x_1 + x_2$, which is an appropriate substitution. In fact, we can handle an infinite number of global optimum points in this way, which is impossible for any numerical solver.

One of the main goals of the simplifier method is to find appropriate substitutions which would eliminate variables. Csendes and Rapcsák [5] with their Assertion 2 suggest to compute the partial derivatives $\partial f(\boldsymbol{x})/\partial x_i$, factorize them, and search for appropriate substitutions in the factors.

Assertion 2 [5]. *If the variables x_i and x_j appear everywhere in the expression of a smooth function $f(\boldsymbol{x})$ in a term $h(\boldsymbol{x})$, then the partial derivatives $\partial f(\boldsymbol{x})/\partial x_i$ and $\partial f(\boldsymbol{x})/\partial x_j$ can be factorized in the forms $(\partial h(\boldsymbol{x})/\partial x_i)\, p(\boldsymbol{x})$ and $(\partial h(\boldsymbol{x})/\partial x_j)\, q(\boldsymbol{x})$, respectively, and $p(\boldsymbol{x}) = q(\boldsymbol{x})$.*

If $\partial f(\boldsymbol{x})/\partial x_i$ cannot be factorized, then any appropriate substitution that is monotonic as a function of x_i is linear as a function of x_i.

For illustrative purposes, let us consider the following problem:

$$\min_{\boldsymbol{x}\in\mathbb{R}^3} 30 \cdot (5x_1 + e^{1+x_2}) + 20 \cdot x_3,$$
$$s.t.\ \ln(5x_1 + e^{1+x_2}) + x_3 \geq 5.$$

This example can be a tiny part of a process synthesis problem. Automatic model generator tools in the field of process synthesis produce several types of multiplicity and redundancy [6]. Among other redundancies, it is possible for some variables denoting chemical elements to appear exclusively in the chemical formula of a given material. In our example, x_1 and x_2 appear everywhere in the term $h(\boldsymbol{x}) = 5x_1 + e^{1+x_2}$. For the sake of simplicity, the constraint can be reformulated by adding a penalty term [9] to the objective, so we need to minimize

$$f(\boldsymbol{x}) = 30 \cdot (5x_1 + e^{1+x_2}) + 20 \cdot x_3 + \sigma \left(\ln(5x_1 + e^{1+x_2}) + x_3 - 5 - x_4\right)^2,$$

where x_1, x_2, x_3 are real variables based on physical parameters, σ is a penalty constant and x_4 is a slack variable. Due to Assertion 2, $\partial f(\boldsymbol{x})/\partial x_1$ can be transformed into the form $(\partial h(\boldsymbol{x})/\partial x_1) \cdot p(\boldsymbol{x})$, similarly $\partial f(\boldsymbol{x})/\partial x_2 = (\partial h(\boldsymbol{x})/\partial x_2) \cdot q(\boldsymbol{x})$, while $p(\boldsymbol{x}) = q(\boldsymbol{x})$. In our example $\partial h(\boldsymbol{x})/\partial x_1 = 5$, $\partial h(\boldsymbol{x})/\partial x_2 = e^{1+x_2}$, and

$$p(\boldsymbol{x}) = q(\boldsymbol{x}) = \frac{2\left(75x_1 + 15e^{1+x_2} + \sigma\left(\ln(5x_1 + e^{1+x_2}) + x_3 - 5 - x_4\right)\right)}{5x_1 + e^{1+x_2}}.$$

Based on the above result, we created a computer program to produce equivalent transformations automatically for the simplification of unconstrained nonlinear optimization problems. The naive implementation would realize the following steps:

1. Compute the gradient of the objective function.
2. Factorize the partial derivatives.
3. Collect appropriate substitutions which contain x_i, into a list l_i:
   a) Initialize l_i to be the empty set.
   b) If the factorization was successful for $\partial f(\boldsymbol{x})/\partial x_i$, then extend l_i with the respective integrals of the factors.
   c) Extend l_i with the subexpressions of $f(\boldsymbol{x})$ that are linear in x_i.
   d) Drop the elements of l_i which do not fulfill the conditions of an appropriate substitution (the elements of l_i need to be monotonic in x_i).
4. Create a list S by applying all proper combinations of the appropriate substitutions from $L = \bigcup l_i$, $i = 1, \ldots, n$ to $f(\boldsymbol{x})$.
5. Choose the least complex element of S to be the simplified objective function.
6. Solve the problem with the simplified objective function (if possible).

7. Give the solution of the original problem by executing inverse transformations.

Most of the required steps of the algorithm (partial differentiation, factorization, symbolic integration and substitution) are realized in modern computer algebra systems as reliable symbolic computation methods. On the other hand, our first implementation in Maple showed that even one of the market-leader computer algebra systems has serious deficiency to our requirements in point of substitution capability and interval arithmetic with infinite intervals [2].

Actually, the exact range calculation for a nonlinear function has the same complexity as computing the global minimum and maximum. However, naive interval inclusion can be applied to verify whether the range of a subexpression is equal to $\mathbb{R}$. Naive interval inclusion is exact for a single use expression (SUE, an expression that contains any variable at most once), but it might produce overestimation for more complex expressions [12]. The possible overestimation can lead to false-positive answers in the range calculation. In other words, L can contain some substitutions with a range which is not equal to $\mathbb{R}$. It means that an additional verification for the range of the produced non-SUE substitutions would be required. On the other hand, most of the substitutions produced in our tests were SUEs. As an alternative, real quantifier elimination [17, 19] would be an applicable symbolic technique for range calculation.

Naive interval inclusion can also be used for the monotonicity test. A real function $f : \mathbb{R}^n \to \mathbb{R}$ is monotone if any $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$ such that $\boldsymbol{x} \leq \boldsymbol{y}$ satisfy $f(\boldsymbol{x}) \leq f(\boldsymbol{y})$. Let us use the product order here: $(x_1, \ldots, x_n) \leq (y_1, \ldots, y_n)$ if $x_i \leq y_i$, $i = 1, \ldots, n$. In the discussed application we need to test whether a function $h_i(\boldsymbol{x})$ is strictly monotonic as a function of a variable x_i. Therefore we compute, whether the naive interval inclusion of the partial derivative $\partial h_i(\boldsymbol{x})/\partial x_i$ contains zero. This approach fits the mathematical definition of monotonicity and is expressive, as a strictly monotone function has a single region of attraction. Unfortunately, overestimation of the naive interval inclusion for non-SUEs can produce false-negative answers in the monotonicity test, so even some monotonic substitutions can be dropped.

Step 3 and Step 4 can be combined to speed up the process and also to ensure that a proper substitution set is applied to $f(\boldsymbol{x})$. We call a well ordered set H of appropriate substitutions *proper* if all the formulas $h_i(\boldsymbol{x}) \in H$ can be substituted by new variables at the same time in a function $f(\boldsymbol{x})$. That is, the expressions $\forall h_i(\boldsymbol{x}) \in H$ do not overlap in the computation tree of $f(\boldsymbol{x})$. Without this property, not all substitutions $y_i = h_i(\boldsymbol{x})$, $h_i(\boldsymbol{x}) \in H$ could be applied. For example, in $f(\boldsymbol{x}) = (x_1 + x_2 + x_3)^2$, the substitutions $y_1 = x_1 + x_2$ and $y_2 = x_2 + x_3$ would be also appropriate, but $H = \{x_1 + x_2, x_2 + x_3\}$ is not a proper substitution set because $x_1 + x_2$ and $x_2 + x_3$ both refer to the same occurrence of x_2. In fact, we prefer to choose the most complex $h(\boldsymbol{x})$ formula to eliminate a variable, so in this example, the substitution $y_3 = x_1 + x_2 + x_3$ should be accepted. At this point, and in Step 5, an easily applicable complexity definition for expressions is needed. In our implementation, an expression is said to be more complex than an other one if its representation (a list of pointers in Mathematica) is longer.

2.2 Implementation in Mathematica

Compared to our first implementation in the computer algebra system Maple [2], the new platform Mathematica has several advantages. First of all, the substitution procedures are much better, since the Mathematica programming language is based on term-rewriting. In other words, the capabilities of the basic substitution routine of Mathematica can be extended with regular expression based term-rewriting rules. We have written a specialized substitution routine in about 50 program lines. A dozen (delayed) rules are introduced, and we have combined four different ways for evaluating them, using the expanded, simplified, and also the factorized form of the formula. It is probably the most important part of the program, as simple refinements could have substantial influence on the result quality of the whole simplification process.

Mathematica has also better interval arithmetic implementation: this was crucial for quick and reliable range calculation on the expressions to be substituted. Naive interval inclusion for the enclosure of the ranges have been realized with the standard range arithmetic of Mathematica.

Furthermore, our new program supports the enumeration of all possible substitutions in Step 3, and it still keeps up in running time with the simpler Maple version, which has used simple heuristics to choose one possible substitution. The reasons for that are the application of adequate programming techniques, and some nice properties of the Mathematica system, such as automatic parallelization on list operations. However, a further study on an efficient substitution selection strategy would be welcome.

Let us mention that Mathematica tends to include more and more expert tools to ensure the possibility of writing efficient program code. For example, it has provided utilities to exploit the parallel computing abilities of the graphics processing unit (GPU) using CUDA or OpenCL technology since 2010.

Demonstration of the presented algorithm and our newest program codes will be available at the following homepage:

`http://www.inf.u-szeged.hu/~csendes/symbsimp/`

2.3 Improvements Compared to the Maple Version

Some simple examples follow to demonstrate the advantages of our new Mathematica program compared to the Maple version.

These examples were generated by us and were the problematic ones out of our self-made collection in the test phase of the Maple implementation, so it was obvious to use them to test the new program. See Table 1 for the Maple-based results and Table 2 for the results obtained by the Mathematica version. For the test cases not discussed, the two implementations gave the same output.

Remark, that the renaming convention of Csendes and Rapcsák [5] is slightly modified in the following tables. Simple renaming ($y_j := x_j$) is not applied in the hope that more elaborated substitutions are emphasized in this way.

Table 1: Results of some synthetic tests solved by our old, Maple based implementation

ID	Function f	Function g	Substitutions	Problem type	Result type
Sin2	$2x_3 \cdot \sin(2x_1 + x_2)$	$2y_1$	$y_1 = x_3 \cdot \sin(2x_1 + x_2)$	A	3
Exp1	$e^{x_1+x_2}$	e^{y_1}	$y_1 = x_1 + x_2$	A	1
Exp2	$2e^{x_1+x_2}$	$2y_1$	$y_1 = e^{x_1+x_2}$	A	3
Sq1	$x_1^2 x_2^2$	none	none	D	2
Sq2	$(x_1x_2 + x_3)^2$	y_1^2	$y_1 = x_1x_2 + x_3$	A	1
SqCos1	$(x_1x_2 + x_3)^2 - \cos(x_1x_2)$	$y_3^2 - \cos(y_1)$	$y_1 = x_1x_2$, $y_3 = y_1 + x_3$	A	1,3
SqExp2	$(x_1 + x_2)^2 + 2e^1 e^{x_1+x_2}$	$y_1^2 + 2e^1 e^{y_1}$	$y_1 = x_1 + x_2$	A	1
SqExp3	$(x_1 + x_2)^2 + 2e^{1+x_1+x_2}$	none	none	A	2

Table 2: Results of some synthetic tests solved by our new, Mathematica based implementation

ID	Function f	Function g	Substitutions	Problem type	Result type
Sin2	$2x_3 \cdot \sin(2x_1 + x_2)$	$2x_3 \sin(y_1)$	$y_1 = 2x_1 + x_2$	A	1
Exp1	$e^{x_1+x_2}$	e^{y_1}	$y_1 = x_1 + x_2$	A	1
Exp2	$2e^{x_1+x_2}$	$2e^{y_1}$	$y_1 = x_1 + x_2$	A	1
Sq1	$x_1^2 x_2^2$	none	none	D	2
Sq2	$(x_1x_2 + x_3)^2$	y_1^2	$y_1 = x_1x_2 + x_3$	A	1
SqCos1	$(x_1x_2 + x_3)^2 - \cos(x_1x_2)$	$y_1^2 - \cos(x_1x_2)$	$y_1 = x_1x_2 + x_3$	A	1
SqExp2	$(x_1 + x_2)^2 + 2e^1 e^{x_1+x_2}$	$y_1^2 + 2e^{1+y_1}$	$y_1 = x_1 + x_2$	A	1
SqExp3	$(x_1 + x_2)^2 + 2e^{1+x_1+x_2}$	$y_1^2 + 2e^{1+y_1}$	$y_1 = x_1 + x_2$	A	1

We apply again the evaluation codes from [2], which describe the quality of the results and the nature of the problem. These codes appear in the "Problem type" and "Result type" columns of Tables 1 and 2. The letters characterize the actual problem:

(A). Simplifying transformations are possible according to the presented theory.

(B). Simplifying transformations could be possible by extension of the presented theory.

(C). Some helpful transformations could be possible by extension of the presented theory, but they do not necessarily simplify the problem (e.g., since they increase dimensionality).

(D). We do not expect any helpful transformation.

The results are described by the second indicator: our program produced

1. proper substitutions,
2. no substitutions, or
3. incorrect substitutions.

As we had discussed earlier [2], Maple provides incorrect interval arithmetic, so we needed to use heuristics in the Maple implementation for range calculation. Also the algebraic substitution capabilities of Maple are weak. Almost all mistakes of the early implementation originated from these two reasons.

Code "2" is also used when only constant multipliers are eliminated.

In short, A1 means that proper substitution is possible and it has been found by the algorithm, while D2 means that as far as we know, no substitution is possible, and this was the conclusion of the program as well. The unsuccessful cases are those denoted by other codes.

The differences between the obtained results are explained next. For the *Sin2* test problem, a proper simplification was obtained by the new implementation, while the old one conveyed a more complex, but non-monotonic substitution.

In *Exp2*, $e^{x_1+x_2}$ does not have a range equal to $\mathbb{R}$, but the heuristic range calculation method used in Maple recognized it as an appropriate substitution. The range calculation subroutine in Mathematica proved to be better in this case.

In *Sq1*, the Maple implementation was not able to recognize the subexpression x_1x_2 in the expression $x_1^2x_2^2$, but was able to recognize $x_1x_2 + x_3$ in its square in *Sq2*. Since the second is not a multiplication type expression but a sum, it is represented in a different way. In Mathematica, regular expressions can be used to produce good substitutions, and our specialized substitution routine worked well for this problem. On the other hand, x_1x_2 is not monotone as a function of x_1 or x_2 for the whole search interval (supposed to be $\mathbb{R}$), so it cannot be chosen as an appropriate substitution.

Also for the *SqCos1* test problem, the new, Mathematica-based method applied a routine to check whether the substitution expression is monotone, so $y_1 = x_1x_2$ was eliminated from the substitution list.

In the cases *SqExp2-3*, also the weakness of the expression matching capability of Maple can be observed, as it was not able to recognize $x_1 + x_2$ in $e^{1+x_1+x_2}$, only in $e^1 e^{x_1+x_2}$.

2.4 Computational Test Results on Global Optimization Problems

Standard and frequently used global optimization test problems were used to study the capabilities and limitations of the symbolic simplification algorithm. The test set was extended in comparison to our earlier paper [2]. The description and even various implementations of the examined problems can be found in several resources and online collections. For example, the compact mathematical formulation and known optima of all of the mentioned problems can be found in Appendix A at Pal [13]. Our computational results are summarized in Table 3.

ID	Dim.	New variables	Substitutions	Problem type	Result type	Transform. time
BR	2	$\boldsymbol{y} = [x_1, y_1]$	$y_1 = x_2$ $-6 + (5/\pi)x_1$ $-0.129185x_1^2$	A	1	0.1092
Easom	2	$\boldsymbol{y} = \boldsymbol{x}$	none	D	2	0.1404
G5	5	$\boldsymbol{y} = \boldsymbol{x}$	none	D	2	2.7456
G7	7	$\boldsymbol{y} = \boldsymbol{x}$	none	D	2	36.5821
GP	2	$\boldsymbol{y} = \boldsymbol{x}$	none	D	2	0.4212
H3	3	$\boldsymbol{y} = \boldsymbol{x}$	none	D	2	16.3488
H6	6	Stopped.	Stopped.	D	2	1800 <
L1	1	$\boldsymbol{y} = \boldsymbol{x}$	none	D	2	0.0312
L2	1	$\boldsymbol{y} = \boldsymbol{x}$	none	D	2	0.6552
L3	2	$\boldsymbol{y} = \boldsymbol{x}$	none	D	2	2.3088
L5	2	$\boldsymbol{y} = \boldsymbol{x}$	none	D	2	15.3348
L8	3	$\boldsymbol{y} = [y_1, y_2, y_3]$	$y_1 = (x_1 - 1)/4$, $y_2 = (x_2 - 1)/4$, $y_3 = (x_3 - 1)/4$	A	1	13.1820
L9	4	$\boldsymbol{y} = [y_1, y_2, y_3, y_4]$	$y_1 = (x_1 - 1)/4$, $y_2 = (x_2 - 1)/4$, $y_3 = (x_3 - 1)/4$, $y_4 = (x_4 - 1)/4$	A	1	174.7047
L10	5	Stopped.	Stopped.	A	1	1800 <
L11	8	Stopped.	Stopped.	A	1	1800 <
L12	10	Stopped.	Stopped.	A	2	1800 <
L13	2	$\boldsymbol{y} = \boldsymbol{x}$	none	D	2	0.4992
L14	3	$\boldsymbol{y} = \boldsymbol{x}$	none	D	2	0.7800
L15	4	$\boldsymbol{y} = \boldsymbol{x}$	none	D	2	1.0296
L16	5	$\boldsymbol{y} = \boldsymbol{x}$	none	D	2	2.0904
L18	7	$\boldsymbol{y} = \boldsymbol{x}$	none	D	2	32.1517
Sch21 (Beale)	2	$\boldsymbol{y} = \boldsymbol{x}$	none	C	2	0.1248
Sch214						

(Powell)	4	$y = x$	none	D	2	0.0936
Sch218						
(Matyas)	2	$y = x$	none	D	2	0.0156
Sch227	2	$y = x$	none	D	2	0.0624
Sch25						
(Booth)	2	$y = x$	none	C	2	0.0312
Sch31	3	$y = x$	none	D	2	0.0468
Sch31	5	$y = x$	none	D	2	0.0936
Sch32	2	$y = [y_1, y_2]$	$y_1 = x_1 - x_2^2$, $y_2 = x_2 - 1$	A	1	0.0468
Sch32	3	$y = x$	none	D	2	0.0312
Sch37	5	$y = x$	none	D	2	0.0936
Sch37	10	$y = x$	none	D	2	0.2808
SHCB	2	$y = x$	none	D	2	0.0156
THCB	2	$y = x$	none	D	2	0.0156
Rastrigin	2	$y = x$	none	C	2	0.0936
RB	2	$y = [y_1, y_2]$	$y_1 = x_1^2 - x_2$, $y_2 = 1 - x_1$	A	1	0.0440
RB5	5	$y = [x_1, x_2, x_3, x_4, y_1]$	$y_1 = x_4^2 - x_5$	A	1	0.3080
S5	4	$y = x$	none	D	2	225.5018
S7	4	$y = x$	none	D	2	1,010.2431
S10	4	Stopped.	Stopped.	D	2	1800 <
R4	2	$y = x$	none	C	2	0.2964
R5	3	$y = [x_1, x_2, y_2]$	$y_1 = 3 + x_3$, $y_2 = (\pi/4)y_1$	A	1	13.8216
R6	5	$y = [x_1, x_2, x_3, x_4, y_2]$	$y_1 = 3 + x_5$, $y_2 = (\pi/4)y_1$	A	2	995.6883
R7	7	Stopped.	Stopped.	A	2	1800 <
R8	9	Stopped.	Stopped.	A	2	1800 <

Table 3: Results for the standard global optimization test functions

In the common cases, most of our new results are identical to what we have obtained with the earlier, Maple-based implementation. The two differences are reported here. For the *Schwefel-227* test problem, the Maple version gave the substitution $y_1 = x_1^2 + x_2^2 - 2x_1$. This expression characterizes all occurrences of x_2, but it is not monotonic in any variable, so the Mathematica version had not suggested it for substitution. For *Schwefel-32 (n=2)*, Mathematica found a good substitution, while Maple did not.

All transformations were performed with Mathematica 9.0 under time constraints. In those cases, in which the complete simplifier program had not stopped in 1800 seconds, the message "Stopped." was written to the New variables and Substitutions columns and "1800 <" to the Transformation time column of Table 3. The numerical tests ran on a computer with an Intel i5-3470 processor, 8 GB RAM and 64-bit operating system.

Most of the running time was used by the symbolic formula transformations of the extended substitution routine. In the problematic cases, usually symbolic factorization consumed 1800 seconds. While every transformation in Table 2 finished in less than 0.2 seconds, the running time for the standard test cases vary more. 24 of 45 test cases ran in one second, further 10 analysis required less than one

minute, but 7 test cases would require more than a half hour to finish.

Altogether, 45 well-known global optimization test problems were examined, and our Mathematica program offered equivalent transformations for 8 cases. In other words, our method suggested some simplification for 18% of this extended standard global optimization test set. The next section presents numerical results to demonstrate that these transformations could be useful for a global optimization solver.

3 On the Advantages of the Transformations

This section presents numerical test results to verify the usefulness of the transformations of Tables 2 and 3. We compare the results of a numerical global optimization solver for the minimization of the original and the transformed problem forms, for every cases of Tables 2 and 3 where our algorithm produced an equivalent transformation. The numerical indicators, as reached global optima values, running times, and function evaluation numbers are presented in Tables 4, 5, and 6. Boldface denotes the better options of related numbers.

Table 4: Optimal function values found by Global

	Original problem			Transformed problem		
ID	Fbest	Fmean	Fvar	Fbest	Fmean	Fvar
Exp1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Exp2	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Sq2	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
SqCos1	−1.0000	−0.7671	0.1899	−1.0000	**-0.9922**	**0.0700**
SqExp2	3.0000	3.0000	0.0000	3.0000	3.0000	0.0000
SqExp3	3.0000	3.0000	0.0000	3.0000	3.0000	0.0000
CosExp	−2.0000	**-1.6166**	0.4397	−2.0000	−1.5896	**0.2781**
BR	0.3979	0.3979	0.0000	0.3979	0.3979	0.0000
L8	0.0000	**2.1386**	**5.6861**	0.0000	2.2651	6.8558
L9	0.0000	2.4410	**8.7591**	0.0000	**2.3897**	10.1134
RB	0.0000	19.7318	1094.1167	0.0000	**0.0000**	**0.0000**
RB5	0.0000	1.8510	3.8703	0.0000	**1.8400**	**3.8677**
R5	0.0000	1.9017	26.3097	0.0000	**0.0000**	**0.0000**
R6	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Sch3.2	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Table 5: Number of function evaluations of Global

	Original problem		Transformed problem	
ID	NumEvalmean	NumEvalvar	NumEvalmean	NumEvalvar
Exp1	87	2,280	**51**	**188**
Exp2	102	2,489	**55**	**327**
Sq2	478	57,927	**52**	**38**
SqCos1	1467	463,242	**1,131**	**279,915**
SqExp2	155	4,903	**61**	**142**
SqExp3	151	5,282	**61**	**142**
CosExp	1,110	1,805,418	**631**	**154,691**
BR	136	1,504	**115**	**769**
L8	**785**	266,138	797	**242,593**
L9	2,606	1,838,627	**2,371**	**1,343,151**
RB	749	71,762	**127**	**976**
RB5	3,162	**693,878**	**2,634**	709,652
R5	**1,908**	1,644,365	2,957	**581,382**
R6	**6,001**	**223,269**	6,069	269,551
Sch3.2	119	1,290	**59**	**121**

Table 6: Running times of Global (seconds)

	Original problem		Transformed problem	
ID	Tmean	Tvar	Tmean	Tvar
Exp1	0.0289	0.0004	**0.0113**	**0.0000**
Exp2	0.0346	0.0005	**0.0124**	**0.0000**
Sq2	0.0919	0.0029	**0.0072**	**0.0000**
SqCos1	0.1822	0.0083	**0.1406**	**0.0047**
SqExp2	0.0270	0.0002	**0.0088**	**0.0000**
SqExp3	0.0264	0.0002	**0.0088**	**0.0000**
CosExp	0.2139	0.0429	**0.1623**	**0.0134**
BR	0.0241	0.0001	**0.0195**	**0.0000**
L8	0.0992	0.0046	**0.0990**	**0.0040**
L9	0.2771	0.0220	**0.2445**	**0.0150**
RB	0.0857	0.0009	**0.0241**	**0.0001**
RB5	0.2705	0.0050	**0.2089**	**0.0045**
R5	**0.2407**	0.0286	0.4567	**0.0159**
R6	0.7830	0.0136	**0.7785**	**0.0047**
Sch3.2	0.0187	0.0001	**0.0116**	**0.0000**

We performed 100 independent runs for every test cases, with the Matlab implementation of a general multi-start solver with a quasi-Newton type local search method, Global with the BFGS local search [4]. The tests were run on the same computer, which was described in Subsection 2.4, with 64-bit MATLAB R2011b. The parameters of Global were set to the defaults.

The solver needs an initial search interval for every variable, so we set the $[-100, 100]$ interval as initial box of every variable in our self-made test cases (Table 2), and the usual boxes for the standard problems (see for example Appendix A in [13]). In the transformed cases, the bounds were transformed appropriately.

Table 4 shows the optimal function values reached by Global with the above mentioned parameters. *Fbest* denotes the minimal optimum value, *Fmean* is the mean of the reached optimum values in average of the 100 runs, and *Fvar* denotes the variance of the reached minimum values. The real global optimum values were reached in every independent run for both of the original and the transformed forms in 8 of 15 cases. In the cases *RB* and *R5*, only the transformed form helped the solver to reach the minimum value in all 100 runs. Totally in 5 of 15 cases, the transformed form was easier to solve with Global, the two forms were equivalently difficult to solve in 8 cases, but in 2 cases the original form was slightly more favorable.

Let us compare the number of function evaluations required by the solver in a run (Table 5). *NumEvalmean* refers to the mean of the number of function evaluations, and *NumEvalvar* refers to the variance of the same indicator. The transformed problem needed fewer function evaluations in 12 of 15 cases, and in some cases it outperformed the original form very well. For example, Global needs only 17% of the original function evaluations for finding the optimum of the *Rosenbrock* problem in the transformed form, 80% of the original with the transformed *Branin*, and 11% of the original function evaluations with the transformed *Sq2* problem. The original form of *L8* and *R6* was slightly better for the solver in terms of function evaluations, and at *L8* also in founding minimum values. The original form of *R5* needed fewer function evaluations, but did not enable the solver to find the global optima in every run, while the transformed form did.

Regarding running times (Table 6), the transformed problem form allows Global to run a bit quicker than on the original problem form in almost every case. The average relative improvement in the running times of the whole test set is 31.5%. However, this indicator is better for our self-made problems (56.9%) and worse for the standard problems (9.3%).

We can conclude that the equivalent transformations of Table 2 and Table 3, which seem to be very simple, have a big influence on the performance of a traditional global optimization solver.

4 Theoretical Extension

We generalize the theoretical results of the papers of Csendes and Rapcsák [5, 14] to allow parallel substitutions and to cover constrained nonlinear optimization problems.

Let us start with an example for parallel substitutions. Consider the following objective function:

$$f(x_1, x_2, x_3) = (x_1 + x_2 + x_3)^2 + (x_1 - 2x_2 - 3x_3)^2.$$

It is equivalent to minimize $g(y_1, y_2) = y_1^2 + y_2^2$, which is a two-dimensional problem against the original three-dimensional one. Neither $y_1 = x_1 + x_2 + x_3$ nor $y_2 = x_1 - 2x_2 - 3x_3$ is appropriate in the earlier meaning, as they are smooth and monotonic in every variable, but none of the variables are covered by y_1 or y_2. However, y_1 together with y_2 characterizes all occurrences of x_1, x_2, and x_3, and $H = \{x_1+x_2+x_3, x_1-2x_2-3x_3\}$ is a proper set of substitutions, resulting dimension reduction of the aforementioned problem. The theoretical extension aims to handle this kind of parallel substitutions.

The original nonlinear optimization problem that will be simplified automatically now can include constraints, too:

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(\boldsymbol{x}) \\ c_i(\boldsymbol{x}) \quad & = 0 \\ c_j(\boldsymbol{x}) \quad & \leq 0, \end{aligned} \tag{2}$$

where $f(\boldsymbol{x}), c_i(\boldsymbol{x}), c_j(\boldsymbol{x}) : \mathbb{R}^n \to \mathbb{R}$ are smooth real functions, given by a formula, and $i = 1, \ldots, p_1$ and $j = p_1 + 1, \ldots, p_2$ are integer indexes.

The transformed constrained optimization problem will be

$$\begin{aligned} \min_{y \in \mathbb{R}^m} \quad & g(\boldsymbol{y}) \\ d_i(\boldsymbol{y}) \quad & = 0 \\ d_j(\boldsymbol{y}) \quad & \leq 0, \end{aligned} \tag{3}$$

where $g(\boldsymbol{y}) : \mathbb{R}^m \to \mathbb{R}$ is the transformed form of $f(\boldsymbol{x})$, and $d_i(\boldsymbol{y}), d_j(\boldsymbol{y}) : \mathbb{R}^m \to \mathbb{R}$ are again smooth real functions, the transformations of the constraints $c_i(\boldsymbol{x}), c_j(\boldsymbol{x})$, $i = 1, \ldots, p_1$ and $j = p_1 + 1, \ldots, p_2$.

Denote by X the set of variable symbols in the objective function $f(\boldsymbol{x})$ and in the constraint functions $c_k(\boldsymbol{x}), k = 1, \ldots, p_2$. Y will be the set of variable symbols in the transformed functions $g(\boldsymbol{y})$, and $d_k(\boldsymbol{y}), k = 1, \ldots, p_2$. Remark, that dimension increase is not allowed for the transformation steps, so $m \leq n$ and $|Y| \leq |X|$. At the beginning of the algorithm, $Y := X$.

Denote the set of the expressions on the left-hand side of the original constraints by C:

$$C := \{c_k(\boldsymbol{x}) : \mathbb{R}^n \to \mathbb{R}\,,\ k = 1, \ldots, p_2\}.$$

Denote by F the expression set of all subexpressions (all well-formed expressions that are part of the original expressions) of $f(\boldsymbol{x})$ and $c_k(\boldsymbol{x}) \in C$.

The crucial part of our algorithm is the *transformation step*. If an $H \subset F$ expression set covers a $V \subseteq X$ variable set (that is, none of $v \in V$ happens out of H in the original expression of $f(\boldsymbol{x})$ or $c_k(\boldsymbol{x}) \in C$), and $|H| \leq |V|$, then apply every substitutions, related to H, to $f(\boldsymbol{x})$ and C as follows. Substitute a new variable y_i in place of $h_i(\boldsymbol{x})$ for all $h_i(\boldsymbol{x}) \in H$ in $f(\boldsymbol{x})$ and also in every $c_k(\boldsymbol{x}) \in C$. Furthermore, let us update the variable set of the transformed problem: $Y := (Y \cup y_i) \setminus V$.

This will be referred to as a transformation step (corresponding to H). The special case $|H| = |V| = 1$, $p_1 = p_2 = 0$ belongs to the algorithm given by Csendes and Rapcsák [5] for the unconstrained case.

Further on, the notation $\boldsymbol{y} := H(\boldsymbol{x})$ will be used as an abbreviation for the following: $y_i := h_i(\boldsymbol{x})$, $i = 1, \ldots, |H|$.

The following assertion is a straightforward generalization of Assertion 1 in [5].

Assertion 1. *If a variable x_i appears in exactly one term, namely in $h(\boldsymbol{x})$, everywhere in the expressions of the smooth functions $f(\boldsymbol{x})$ and $c_k(\boldsymbol{x})$, $k = 1, \ldots, p_2$, then the partial derivatives of these functions related to x_i all can be written in the form $(\partial h(\boldsymbol{x})/\partial x_i)p(\boldsymbol{x})$, where $p(\boldsymbol{x})$ is continuously differentiable.*

Recall, that an ordered set H of substitutions is called *proper*, if all expressions $h_i(\boldsymbol{x}) \in H$ are such that they can be substituted by new variables at the same time. Ordering is required only for univocal indexing of the substitutions.

Theorem 1. *If H is proper and all $h_i(\boldsymbol{x}) \in H$ expressions are smooth and strictly monotonic as a function of every variable $v \in V \subseteq X$, the cardinality of H is less than or equal to the cardinality of V, and the domain of $h_i(\boldsymbol{x})$ is equal to $\mathbb{R}$ for all $h_i(\boldsymbol{x}) \in H$, then the transformation step corresponding to H simplifies the original problem in such a way that every local minimizer (maximizer) point $\boldsymbol{x}^*$ of the original problem is transformed to a local minimizer (maximizer) point $\boldsymbol{y}^*$ of the transformed problem.*

Proof. Consider first the sets of feasibility for the two problems. The substitution equations ensure that if a point $\boldsymbol{x}$ was feasible for the problem (2), then it remains feasible after the transformations for the new, simplified problem (3). The same is true for infeasible points.

Denote now a local minimizer point of $f(\boldsymbol{x})$ by $\boldsymbol{x}^*$, and let $\boldsymbol{y}^* := H(\boldsymbol{x}^*)$ be the transformed form of $\boldsymbol{x}^*$. As each $h_i(\boldsymbol{x}) \in H$ is strictly monotonic in at least one variable, all points from the $a = N(\boldsymbol{x}^*, \delta)$ neighborhood of $\boldsymbol{x}^*$ will be transformed into a $b = N(\boldsymbol{y}^*, \epsilon)$ neighborhood of $\boldsymbol{y}^*$, and $\forall x_j \notin a : y_j \notin b$. Both the objective functions $f(\boldsymbol{x})$ and $g(\boldsymbol{y})$, and the old and transformed constraint functions have the same value before and after the transformation. This fact ensures that each local minimizer point $\boldsymbol{x}^*$ will be transformed into a local minimizer point $\boldsymbol{y}^*$ of $g(\boldsymbol{y})$. The same is true for local maximizer points, by similar reasoning.

Additionally, $|H| \leq |V|$, so the construction of the transformation step ensures that the application of every substitution of H eliminates at least as many x_i

variables from the optimization model as the number of the new variables in every iteration. □

In contrast to Theorem 1, which gives sufficient conditions to have such a simplification transformation that will bring local minimizer points of the old problem to local minimizer points of the new one, the following theorem provides sufficient conditions to have a one-to-one mapping of the minimizer points.

Theorem 2. *If H is proper, and all $h_i(\boldsymbol{x}) \in H$ expressions are smooth, strictly monotonic as a function of every variable $v \in V \subseteq X$, the cardinality of H is less than or equal to the cardinality of V, and the domain and range of $h_i(\boldsymbol{x})$ are equal to $\mathbb{R}$ for all $h_i(\boldsymbol{x}) \in H$, then the transformation step corresponding to H simplifies the original problem in such a way that every local minimizer (maximizer) point $\boldsymbol{y}^*$ of the transformed problem can be transformed back to a local minimizer (maximizer) point $\boldsymbol{x}^*$ of the original problem.*

Proof. Since the substitution equations preserve the values of the constraint functions, each point $\boldsymbol{y}$ of the feasible set of the transformed problem (3) must be mapped from a feasible point $\boldsymbol{x}$ of the original problem (2): $\boldsymbol{y} = H(\boldsymbol{x})$. The same holds for infeasible points.

Denote a local minimizer point of (3) by $\boldsymbol{y}^*$. Now, since the ranges of the transformation functions in H are equal to $\mathbb{R}$, every local minimizer point $\boldsymbol{y}^*$ of the transformed problem (3) is necessarily a result of a transformation: let $\boldsymbol{x}^*$ be the back-transformed point: $\boldsymbol{y}^* = H(\boldsymbol{x}^*)$. Consider a neighborhood $N(\boldsymbol{y}^*, \delta)$ of $\boldsymbol{y}^*$, where every feasible point $\boldsymbol{y}$ has a greater or equal function value: $g(\boldsymbol{y}) \geq g(\boldsymbol{y}^*)$, and those feasible points $\boldsymbol{x}$ of (2), for which $H(\boldsymbol{x}) = \boldsymbol{y} \in N(\boldsymbol{y}^*, \delta)$. The latter set may be infinite if the simplification transformation decreases the dimensionality of the optimization problem. Anyway, there is a suitable neighborhood $N(\boldsymbol{x}^*, \delta')$ of $\boldsymbol{x}^*$ inside this set, for which the relation $f(\boldsymbol{x}) \geq f(\boldsymbol{x}^*)$ holds for all $\boldsymbol{x} \in N(\boldsymbol{x}^*, \delta')$ that satisfies the constraints of (2). In other words, $\boldsymbol{x}^*$ is a local minimum point of (2).

The argument for local maximizers is similar. □

Corollary 1 is an immediate consequence of Theorem 1:

Corollary 1. *If H is proper, all the $h_i(\boldsymbol{x}) \in H$ expressions are smooth and invertible as a function of every variable $v \in V \subseteq X$, and the cardinality of H is less than or equal to the cardinality of V, then the transformation step corresponding to H simplifies the original problem in such a way that every local optimum point $\boldsymbol{x}^*$ of the original problem is transformed to a local optimum point $\boldsymbol{y}^*$ of the transformed problem.*

5 Summary

This paper examines the possibility and ability of implementing equivalent transformations for nonlinear optimization problems as an automatic presolving phase of numerical global optimization methods.

An extensive computational test has been completed on standard global optimization test problems and on other often used global optimization test functions together with some custom made problems designed especially to test the capabilities of symbolic simplification algorithms. Maple and Mathematica based implementations were compared.

The test results show that most of the simplifiable cases were recognized by our new, Mathematica-based algorithm, and the substitutions were correct. Tests with a numerical solver, namely Global, were performed to check the usefulness of the produced transformations. The results show that the produced substitutions can improve the performance of this multi-start solver.

We have presented some new theoretical results on automatic symbolic transformations to simplify constrained nonlinear optimization problems. However, further investigations would be necessary to build an efficient branch and bound strategy into the algorithm at Step 3–4 to realize good running times for the described parallel substitutions.

As a natural extension of the present application, symbolic reformulations are promising for speeding up interval methods of global optimization. The overestimation sizes for interval arithmetic [1] based inclusion functions were investigated in optimization models [18]. Symbolic transformations seem to be appropriate for a proper reformulation. Obvious improvement possibilities in this field are the use of the square function instead of the usual multiplication (where it is suitable), the transformation along the subdistributivity law, and finding SUE forms. In fact, such transformations usually are performed by the default expression simplification mechanism [16] of an ordinary computer algebra system. The domain of calculation has an important role in this presolve approach, since important features of functions such as monotonicity change substantially within the domain where a function is defined.

References

[1] Alefeld, G., Herzberger, J. *Introduction to Interval Computation.* Academic Press, New York, 1983.

[2] Antal, E., Csendes, T., Virágh, J. Nonlinear Transformations for the Simplification of Unconstrained Nonlinear Optimization Problems. *Cent. Eur. J. Oper. Res.* 21(4):665–684, 2013.

[3] Avanzolini, G., Barbini, P. Comment on "Estimating Respiratory Mechanical Parameters in Parallel Compartment Models". *IEEE Trans. Biomed. Eng.* 29:772–774, 1982.

[4] Csendes, T., Pál, L., Sendín, J. O. H., Banga, J. R. The GLOBAL Optimization Method Revisited. *Optimization Letters* 2:445–454, 2008.

[5] Csendes, T., Rapcsák, T. Nonlinear Coordinate Transformations for Unconstrained Optimization. I. Basic Transformations. *J. Global Optim.* 3(2):213–221, 1993.

[6] Farkas, T., Rev, E., Lelkes, Z. Process flowsheet superstructures: Structural multiplicity and redundancy Part 1: Basic GDP and MINLP representations. *Computers and Chemical Engineering*, 29:2180–2197, 2005.

[7] Fourer, R., and Gay, D. M. Experience with a Primal Presolve Algorithm. In Hager, W. W., Hearn, D. W. and Pardalos, P. M., editors, *Large Scale Optimization: State of the Art*, pages 135–154. Kluwer Academic Publishers, Dordrecht, 1994.

[8] Gay, D. M. Symbolic-Algebraic Computations in a Modeling Language for Mathematical Programming. In Alefeld, G., Rohn, J. and Yamamoto, T., editors. *Symbolic Algebraic Methods and Verification Methods*, pages 99–106. Springer-Verlag, 2001.

[9] Grossmann, I. E. MINLP optimization strategies and algorithms for process synthesis. In *Proc. 3rd Int. Conf. on Foundations of Computer-Aided Process Design*, page 105., 1990.

[10] Liberti, L., Cafieri, S., Savourey, D. The Reformulation-Optimization Software Engine. *Mathematical Software – ICMS 2010, LNCS 6327*, pages 303–314, 2010.

[11] Mészáros, Cs., Suhl, U. H. Advanced preprocessing techniques for linear and quadratic programming. *OR Spectrum* 25(4):575–595, 2003.

[12] Neumaier, A. Improving interval enclosures. Manuscript. Available at `http://www.mat.univie.ac.at/~neum/ms/encl.pdf`

[13] Pál, L. *Global optimization algorithms for bound constrained problems.* Ph.D. thesis, University of Szeged, 2011.

[14] Rapcsák, T., Csendes, T. Nonlinear Coordinate Transformations for Unconstrained Optimization. II. Theoretical Background. *J. Global Optim.* 3(3):359–375, 1993.

[15] Schichl, H., Neumaier, A. Interval Analysis on Directed Acyclic Graphs for Global Optimization. *J. Global Optim.* 33(4):541–562, 2005.

[16] Stoutemyer, D. R. Ten commandments for good default expression simplification. *J. Symb. Comput.* 46:859–887, 2011.

[17] Sturm, T., Tiwari A. Verification and synthesis using real quantifier elimination. In *Proceedings of the 36th international symposium on Symbolic and algebraic computation (ISSAC '11)*, ACM, New York, NY, USA, 329–336, 2011.

[18] Tóth, B., Csendes, T. Empirical investigation of the convergence speed of inclusion functions. *Reliable Computing*, 11:253–273, 2005.

[19] Vajda, R. Effective Real Quantifier Elimination. In *J. Karsai, R. Vajda (eds.), Interesting Mathematical Problems in Sciences and Everyday Life - 2011*, University of Szeged, ISBN:978-963-306-109-1

[20] Wolfram Mathematica 9 Documentation Center, Mathematica Tutorial: Basic Internal Architecture. Available at `https://reference.wolfram.com/mathematica/tutorial/BasicInternalArchitecture.html`

Received 15th June 2015

Acta Cybernetica **22** (2016) 735–769.

The Structure of Rooted Weighted Trees Modeling Layered Cyber-security Systems

Geir Agnarsson[*], Raymond Greenlaw[†], and Sanpawat Kantabutra[‡]

Abstract

In this paper we consider the structure and topology of a layered-security model in which the containers and their nestings are given in the form of a rooted tree T. A *cyber-security model* is an ordered three-tuple $M = (T, C, P)$ where C and P are multisets of *penetration costs* for the containers and *target-acquisition values* for the prizes that are located within the containers, respectively, both of the same cardinality as the set of the non-root vertices of T. The problem that we study is to assign the penetration costs to the edges and the target-acquisition values to the vertices of the tree T in such a way that minimizes the total prize that an attacker can acquire given a limited *budget*. The attacker breaks into containers starting at the root of T and once a vertex has been broken into, its children can be broken into by paying the associated penetration costs. The attacker must deduct the corresponding penetration cost from the budget, as each new container is broken into. For a given assignment of costs and target values we obtain a *security system*. We show that in general it is not possible to develop an optimal security system for a given cyber-security model M. We define P- and C-models where the penetration costs and prizes, respectively, all have unit value. We show that if T is a rooted tree such that any P- or C-model $M = (T, C, P)$ has an optimal security system, then T is one of the following types: (i) a rooted path, (ii) a rooted star, (iii) a rooted 3-caterpillar, or (iv) a rooted 4-spider. Conversely, if T is one of these four types of trees, then we show that any P- or C-model $M = (T, C, P)$ does have an optimal security system. Finally, we study a duality between P- and C-models that allows us to translate results for P-models into corresponding results for C-models and vice versa. The results obtained give us some mathematical insights into how layered-security defenses should be organized.

Keywords: cyber-security model, duality, graph minors, rooted tree, security system, system attack, tree types, weighted rooted tree

[*]Department of Mathematical Sciences, George Mason University, Fairfax, VA 22030, E-mail: geir@math.gmu.edu

[†]Department of Cyber Sciences, United States Naval Academy, Annapolis, Maryland 21402, E-mail: greenlaw@usna.edu

[‡]The Theory of Computation Group, Computer Engineering Department, Chiang Mai University, Chiang Mai, 50200, Thailand, E-mail: sanpawat@alumni.tufts.edu

DOI: 10.14232/actacyb.22.4.2016.2

1 Introduction

According to [6], the global cyber-security market cost in 2017 is expected to top 120 billion US dollars. This site also reports that there are 18 victims of a cyber crime every single second! Other sources report similarly alarming and worsening statistics. There is agreement that the number of cyber attacks is increasing rapidly, and the consequences of such attacks are greater than ever on economics, national security, and personal data. Threats come from nation states with advanced cyber warfare commands, nation states having less technical capabilities but intent on doing harm, ideologically motivated groups of hackers or extremists, profit-seeking criminals, and others. As a result, quite a bit of work has been done where cyber-security systems, or more generally layered computer systems, are modeled as a fixed weighted trees. For example, in [1, 3, 4, 8, 10, 12] the authors consider finding *weight-constrained, maximum-density subtrees* and similar structures given a fixed weighting of a tree as part of the input. In these cases weights are specified on both vertices and edges. There has also been some research on *network fortification* and problems related to that topic. For example, in [13] stochastic linear programming games are studied and it is demonstrated how these can, among other things, model certain network fortifications. In [14] the problem of network interdiction is studied – how to minimize the maximum amount of flow an adversary/enemy can push through a given network from a source s to a sink t. There each edge/arc is provided with a fixed integer capacity and an integer resource (required to delete the edge/arc). This is a variation of the classical Max-Flow-Min-Cut Theorem. Although interesting in their own way, neither of these papers or related papers that we have found in the literature address directly what we study in this paper. To build secure systems requires first principles of security. "In other words, we need a *science of cyber-security* that puts the construction of secure systems onto a firm foundation by giving developers a body of laws for predicting the consequences of design and implementation choices" [11]. To this end, Schneider called for more models and abstractions to study cyber security [11]. This paper is a step in that direction. We hope that others will build on this work to develop even better and more realistic models, overcome the shortcomings of our model, as well as develop additional foundational results.

Building on the work done in [3], in this paper we study a layered-security model and strategies for assigning *penetration costs* and *target-acquisition values* so as to minimize the amount of damage an attacker can do to a system. That is, we examine *security systems*. The approach we take here is to assign weights to the vertices and edges of a tree in order to build a cyber defense that minimizes the amount of prize an attacker can accumulate given a limited budget. To the best of our knowledge this approach is new in that the usual approach is to consider a particular weighted tree as input. In [3] the following question was posed: Can one mathematically prove that the intuition of storing high-value targets deeper in the system and having higher penetration costs on the outer-most layers of the system results in the best or at least good security? In this paper we answer this question and obtain more general and specific results. We define three types of security

systems: *improved*, *good*, and *optimal*. We show that not all *cyber-security models* admit optimal security systems, but prove that paths and stars do. We define and study *P-* and *C-models* where all penetration costs, or all prizes, are set to one, respectively. We classify the types of trees that have optimal security systems for both P- and C-models. We then discuss a duality between P- and C-models, which provides a dictionary to translate results for P-models into corresponding results for C-models, and vice versa.

The outline of this article is as follows. In Section 2 we present the rationale for our layered-security model. In Section 3 we define the framework for security systems and present the definitions of improved, good, and optimal security systems, and state some related observations and examples. In Section 4 we explore optimal security systems and prove that they do not always exist, but they exist if and only if the underlying tree T of the given security system is either a path rooted at a leaf, or a star rooted at its center vertex. In Section 5 we define P- and C-models and show that any cyber-security model $M = (T, C, P)$ is equivalent to both a P-model M' and a C-model M''. We further show that if T is a rooted tree such that any P- or C-model M has an optimal security system, then T is one of the following four types: (i) a rooted path, (ii) a rooted star, (iii) a rooted 3-caterpillar, or (iv) a rooted 4-spider. In Section 6 we prove that if T is one of the four types of rooted trees mentioned above, then any P-model does indeed have an optimal security system. In Section 7 we define a duality between equivalence classes of P-models and equivalence classes of C-models that serves as a dictionary allowing us to obtain equivalent results for C-models from those of the P-models that were obtained in Section 6. In particular, we obtain Theorem 7.2 that completely classifies which P- and which C-models have optimal security systems. Conclusions and open problems are discussed in Section 8.

2 Rationale for Our Layered-Security Model

In defining our layered-security model to study defensive cyber security, we need to strike a balance between simplicity and utility. If the model is too simple, it will not be useful to provide insight into real situations; if the model is too complex, it will be too cumbersome to apply, and we may get bogged down in too many details. The model described in this paper is a step toward gaining a better understanding of a broad range of security systems in a graph-theoretical setting for a layered-security model.

Many systems contain layered security or what is commonly referred to as *defense-in-depth*, where valuable assets are hidden behind many different layers or secured in numerous ways. For example, a *host-based defense* might layer security by using tools such as signature-based vendor anti-virus software, host-based systems security, host-based intrusion-prevention systems, host-based firewalls, encryption, and restriction policies, whereas a *network-based defense* might provide defense-in-depth by using items such as web proxies, intrusion-prevention systems, firewalls, router-access control lists, encryption, and filters [9]. To break into such

a system and steal a valuable asset requires that several levels of security be penetrated, and, of course, there is an associated cost to break into each level, for example, money spent, time used, or the punishment served for getting caught.

Our model focuses on the layered aspect of security and is intended to capture the notion that there is a cost associated with penetrating each additional level of a system and that attackers have finite resources to utilize in a cyber attack. Defenders have the ability to secure targets using defense mechanisms of various strengths and to secure targets in desired locations and levels. We assume that the structure where targets will be stored, that is, the container nestings; is given as part of the input in the form of a rooted tree. In this way we can study *all* possible structures at a single time, as they can be captured in the definition of our problems. This methodology is as opposed to having the defender actually construct a separate defense structure for each input.

For any specific instance of a problem, a defender of a system will obviously consider the exact details of that system and design a layered-security approach to fit one's actual system. Similarly, a traveling salesman will be concerned about constructing a tour of *his* particular cities, not a tour of any arbitrary set of cities with any arbitrary set of costs between pairs of cities. Nevertheless, researchers have found it extremely helpful to consider a general framework in which to study the TRAVELING SALESMAN PROBLEM. And, in studying the general problem, insights have been gained into *all* instances of the problem. Thus, we believe it is worthwhile to consider having a fixed structure as part of our input, and this approach is not significantly different from that used in complexity theory to study problems [5, 7].

In this paper we focus on a static defense. We pose as an open problem the question of how to create a defense and an attack strategy if the defender is allowed to move targets around dynamically or redistribute a portion of a prize. We also consider the total prize as the sum of the individual values of the targets collected although one could imagine using other or more complex functions of the target values to quantify the damage done by an attacker. Our defensive posture is formed by assigning to the edges and vertices of the rooted tree in question the input-provided penetration costs and target-acquisition values, respectively. We formalize the model, the notion of a security system, and the concept of a system attack in the next section.

3 Cyber-Security Model and Security Systems

Let $\mathbb{N} = \{1, 2, 3, \ldots\}$, $\mathbb{Q}$ be the rational numbers, and $\mathbb{Q}_+$ be the non-negative rational numbers.

Definition 3.1. *A* cyber-security model (CSM) M *is given by a three-tuple* $M = (T, C, P)$, *where* T *is a directed tree rooted at* r *having* $n \in \mathbb{N}$ *non-root vertices,* C *is a multiset of* penetration costs $c_1, \ldots, c_n \in \mathbb{Q}_+$, *and* P *is a multiset of* target-acquisition-values *(or* prizes *for short)* $p_1, \ldots, p_n \in \mathbb{Q}_+$.

Remark. As mentioned right after Observation5.1, strictly speaking, we could have stated the above definition using the set $\mathbb{N}$ of natural numbers instead of non-negative rationals $\mathbb{Q}_+$ for possible penetrations costs and prizes. We do, however, prefer the most general definition we can discuss.

Throughout $V(T) = \{r, u_1, \ldots, u_n\}$, where r is the designated root that indicates the start of a *system attack*, and $E(T) = \{e_1, \ldots, e_n\}$ denotes the set of edges of T, where our labeling is such that u_i is always the head of the edge e_i. The prize at the root is set to 0. The penetration costs model the expense for breaking through a layer of security, and the target-acquisition-values model the amount of prize one acquires for breaking through a given layer and exposing a target. The penetration costs will be weights that are assigned to edges in the tree, and the target-acquisition-values, or the prizes, are weights that will be assigned to vertices in the tree.

Sometimes we do not distinguish a target from its acquisition value or prize, nor a container, which is a layer of security, from its penetration cost. Note that one can think of each edge in the rooted tree as another container, and as one goes down a path in the tree, as penetrating additional layers of security. We can assume that the number of containers and targets is the same. Since if we have a container housing another container (and nothing else), we can just look at this "double" container as a single container of penetration cost equal to the sum of the two nested ones. Also, if a container includes many prizes, we can just lump them all into a single prize, which is the sum of them all.

Recall that in a rooted tree T, each non-root vertex $u \in V(T)$ has exactly one parent, and that we assume the edges of T are directed naturally away from the root r in such a way that each non-root vertex has an in-degree of one. The root is located at *level* 0 of the tree. *Level* 1 of the tree consists of the children of the root, and, in general, *level* i of the tree consists of the children of those vertices at level $i-1$ for $i \geq 1$. We next present some key definitions about a CSM that will allow us to study questions about *security systems.*

Definition 3.2. *A* security system (SS) *with respect to a cyber-security model* $M = (T, C, P)$ *is given by two bijections* $c : E(T) \to C$ *and* $p : V(T) \setminus \{r\} \to P$. *We denote the security system by* (T, c, p).

A system attack (SA) *in a security system* (T, c, p) *is given by a subtree* τ *of* T *that contains the root* r *of* T.

- *The* cost *of a system attack* τ *with respect to a security system* (T, c, p) *is defined by*
$$\mathrm{cst}(\tau, c, p) = \sum_{e \in E(\tau)} c(e).$$
- *The* prize *of a system attack* τ *with respect to a security system* (T, c, p) *is defined by*
$$\mathrm{pr}(\tau, c, p) = \sum_{u \in V(\tau)} p(u).$$

- *For a given* budget $B \in \mathbb{Q}_+$ *the maximum prize* $\mathrm{pr}^\star(B,c,p)$ *with respect to* B *is defined by*

$$\mathrm{pr}^\star(B,c,p) := \max\{\mathrm{pr}(\tau,c,p) : \textit{for all system attacks } \tau \subseteq T, \textit{ where } \mathrm{cst}(\tau,c,p) \le B\}.$$

A system attack τ *whose prize is a maximum with respect to a given budget is called an* optimal attack.

The bijection c in Definition 3.2 specifies how difficult it is to break into the various containers, and the bijection p specifies the prize associated with a given container. Note that for any SS (T,c,p) we have $\mathrm{cst}(r,c,p) = 0 \le B \in \mathbb{Q}_+$. When $T = (\{r\},\varnothing)$, then $\mathrm{pr}^\star(B,c,p) = 0$ for any $B \in \mathbb{Q}_+$. When two bijections are given specifying a SS, we call the resulting weighted tree a *configuration of the CSM.* Any configuration represents a defensive posture and hence the name security system. Note that the CSM can be used to model any general security system and not just cyber-security systems. We are interested in configurations that make it difficult for an attacker to accumulate a large prize. It is natural to ask if a given defensive stance can be improved. Next we introduce the notion of an *improved security system* that will help us to address this question.

Definition 3.3. *Given a CSM* $M = (T,C,P)$ *and a SS* (T,c,p)*, an* improved security system (improved SS) with respect to (T,c,p) *is a SS* (T,c',p') *such that for any budget* $B \in \mathbb{Q}_+$ *we have* $\mathrm{pr}^\star(B,c',p') \le \mathrm{pr}^\star(B,c,p)$*, and there exists some budget* $B' \in \mathbb{Q}_+$ *such that* $\mathrm{pr}^\star(B',c',p') < \mathrm{pr}^\star(B',c,p)$.

Definition 3.3 captures the idea of a better placement of prizes and/or penetration costs so that an attacker cannot do as much damage. That is, in an improved SS one can never acquire a larger overall maximum prize with respect to any budget B; and furthermore, there must be at least one particular budget where the attacker actually does worse. Notice that there can be an improved SS (T,c',p'), where for some budget $B \in \mathbb{Q}_+$, there is a SA τ whose cost is less than or equal to B for both SSs such that $\mathrm{pr}(\tau,c',p') > \mathrm{pr}(\tau,c,p)$. In this case an attacker obtains a larger prize in the improved SS; and, of course, this situation is undesirable and means a weaker defense against this specific attack. We, however, are interested in improved SSs with respect to a given budget rather than a particular SA. Since we have exactly n penetration costs and n prizes to assign, it is difficult to imagine an improved SS for all but the most-restricted trees in which all SAs would be improved in the sense just described. Next, we formalize the notion of an *optimal security system.*

Definition 3.4. *Let* $M = (T,C,P)$ *be a given CSM. (i) For a budget* $B \in \mathbb{Q}_+$*, a SS* (T,c,p) *is* optimal w.r.t. B *if there is no other SS* (T,c',p') *for* M *such that* $\mathrm{pr}^\star(B,c',p') < \mathrm{pr}^\star(B,c,p)$*. (ii)* (T,c,p) *is* optimal *if it is optimal w.r.t. any budget* $B \in \mathbb{Q}_+$.

Notice that an optimal SS is not necessarily the best possible. We could define a *critically optimal security system* to be one where for every single SA the SS was

at least as good as all others and for at least one better. And, in a different context, these SSs might be interesting. However, in light of Theorem 4.1 in the following section, which shows that even an optimal SS may not exist for a given CSM, we do not pursue critically optimal SSs further in this paper. By Definitions 3.3 and 3.4 we clearly have the following.

Observation 3.1. *A SS (T, c, p) for a CSM $M = (T, C, P)$ is optimal if and only if no improved SS for (T, c, p) exists.*

We next introduce the concept of two closely-related configurations of a CSM, and this notion will give us a way to relate SSs.

Definition 3.5. *Given a CSM $M = (T, C, P)$, the two configurations (T, c, p), and (T, c', p') are said to be* neighbors *if*

1. *there exists an edge $(u, v) \in E(T)$ such that*

$$\begin{aligned} p'(v) &= p(u) \\ p'(u) &= p(v) \\ p'(w) &= p(w), \text{ otherwise, or} \end{aligned}$$

2. *there exist two edges $(u, v), (v, w) \in E(T)$ such that*

$$\begin{aligned} c'((u, v)) &= c((v, w)) \\ c'((v, w)) &= c((u, v)) \\ c'((x, y)) &= c((x, y)), \text{ otherwise.} \end{aligned}$$

The notion of neighboring configurations will be useful in developing algorithms for finding *good security systems*, which we define next.

Definition 3.6. *A* good security system (good SS) *is a SS (T, c, p) such that no neighboring configuration results in an improved security system.*

Given a SS (T, c, p) for a CSM M, a natural question to pose is whether a local change to the SS can be made in order to strengthen the SS, that is, make the resulting SS improved. In a practical setting one may not be able to redo the security of an entire system, but instead may be able to make local changes.

Suppose $(u, v) \in E(T)$ where $p(u) \geq p(v)$, and let p' be the prize assignment obtained from p by swapping the prizes on u and v, that is $p'(u) = p(v)$, $p'(v) = p(u)$, and $p'(w) = p(w)$ otherwise. If now τ is any SA, then $\mathrm{pr}(\tau, c, p') = \mathrm{pr}(\tau, c, p)$ if either both $u, v \in V(\tau)$ or neither u nor v are vertices of τ, or $\mathrm{pr}(\tau, c, p') \leq \mathrm{pr}(\tau, c, p)$ if $u \in V(\tau)$ and $v \notin V(\tau)$. In either case $\mathrm{pr}(\tau, c, p') \leq \mathrm{pr}(\tau, c, p)$ and therefore we have for any budget B that

$$\mathrm{pr}^\star(B, c, p') \leq \mathrm{pr}^\star(B, c, p). \tag{1}$$

Similarly, if $(u, v), (v, w) \in E(T)$ where $c((u, v)) \leq c((v, w))$, let c' be the cost assignment obtained from c by swapping the costs on the incident edges (u, v) and

(v,w) and leave all the other edge-costs unchanged, that is $c'((u,v)) = c((v,w))$, $c'((v,w)) = c((u,v))$ and $c'(e) = c(e)$ otherwise. If τ is a SA, then clearly we always have $\mathrm{pr}(\tau,c',p) = \mathrm{pr}(\tau,c,p)$. Also, if either both $(u,v),(v,w) \in E(\tau)$ or neither (u,v) nor (v,w) are edges in τ, then $\mathrm{cst}(\tau,c',p) = \mathrm{cst}(\tau,c,p)$, and if $(u,v) \in E(\tau)$ and $(v,w) \notin E(\tau)$, then $\mathrm{cst}(\tau,c',p) \geq \mathrm{cst}(\tau,c,p)$. In either case we have $\mathrm{cst}(\tau,c',p) \geq \mathrm{cst}(\tau,c,p)$. Hence, if B is any budget, then by mere definition we have that

$$\mathrm{pr}^\star(B,c',p) \leq \mathrm{pr}^\star(B,c,p). \tag{2}$$

By (1) and (2) we have the following proposition.

Proposition 3.1. *Let $M = (T,C,P)$ be a CSM. A SS given by (T,c,p) is a good SS if for all $(u,v),(v,w) \in E$ we have $c((u,v)) \geq c((v,w))$ and for all non-root vertices $u,v \in V(T)$ with $(u,v) \in E(T)$ we have $p(u) \leq p(v)$.*

Note that Proposition 3.1 says that on any root to leaf path in T the penetration costs occur in decreasing order and the prizes occur in increasing order.

From any configuration resulting from a SS (T,c,p) for a CSM, Proposition 3.1 gives a natural $O(n^2)$ algorithm for computing a good SS by repeatedly moving to improved neighboring configurations until no more such neighboring configurations exist. We can do better than this method by first sorting the values in C and P using $O(n \log n)$ time, and then conducting a breath-first search of T in $O(n)$ time. We can then use the breath-first search level numbers to define bijections c and p that meet the conditions of a good SS. We summarize in the following.

Observation 3.2. *Given a CSM $M = (T,C,P)$, there is an $O(n \log n)$ algorithm for computing a good SS for M.*

If we could eliminate the sorting step, we would have a more efficient algorithm for obtaining a good SS, or if we restricted ourselves to inputs that could be sorted in $O(n)$ time. Also, notice that a good SS has the heap property, if we ignore the root. However, in our case we cannot "choose" the shape of the heap, but we must use the structure that is given to us as part of our input.

Suppose that our SS (T,c,p) for M satisfies a *strict* inequality $p(u) > p(v)$ for some $(u,v) \in E(T)$, or that $c((u,v)) < c((v,w))$ for some incident edges $(u,v),(v,w) \in E(T)$. A natural question is whether the prize and cost assignments p' and c' as in (1) and (2) will result in an improved SS as in Definition 3.3. In Example 3.1 we will see that that is not the case.

CONVENTION: Let $T_p(\ell)$ denote the rooted tree whose underlying graph is a path on $2\ell+1$ vertices $V(T_p(\ell)) = \{r,u_1,\ldots,u_{2\ell}\}$ and directed edges

$$E(T_p(\ell)) = \{(r,u_1),(r,u_2),(u_1,u_3),(u_2,u_4),\ldots,(u_{2\ell-3},u_{2\ell-1}),(u_{2\ell-2},u_{2\ell})\}$$

rooted at its center vertex. We label the edges by the same index as their heads: $e_1 = (r,u_1)$, $e_2 = (r,u_2)$,..., $e_{2\ell-1} = (u_{2\ell-3},u_{2\ell-1})$, and $e_{2\ell} = (u_{2\ell-2},u_{2\ell})$, see Figure 1.

Example 3.1.

Let $(T_p(3), c, p)$ be a SS for a CSM M where

$$\begin{aligned} c(e_1, e_2, e_3, e_4, e_5, e_6) &:= (1, 1, 1, 1, 1, 2), \\ p(u_1, u_2, u_3, u_4, u_5, u_6) &:= (10, 2, 10, 3, 10, 40), \end{aligned}$$

where the penetration costs and the prizes have been simultaneously assigned in the obvious way. We see that for any budget $B \in \mathbb{Q}_+$ we have

$$\mathrm{pr}^\star(B, c, p) = \begin{cases} 10\lfloor B \rfloor & \text{for } 0 \leq B < 4, \\ 10\lfloor B \rfloor + 5 & \text{for } 4 \leq B \leq 7, \\ 75 & \text{for } 7 < B. \end{cases}$$

If now $p'(u_1, u_2, u_3, u_4, u_5, u_6) = (10, 3, 10, 2, 10, 40)$ is the prize assignment obtained from p by swapping the prizes on the neighboring vertices u_2 and u_4, and $c'(e_1, e_2, e_3, e_4, e_5, e_6) = (1, 1, 1, 2, 1, 1)$ be the edge-cost assignment obtained from c be swapping the costs of the incident edges e_4 and e_6, then

$$\mathrm{pr}^\star(B, c, p') = \mathrm{pr}^\star(B, c', p) = \mathrm{pr}^\star(B, c, p),$$

for any non-negative budget $B \in \mathbb{Q}_+$, showing that locally swapping either prize assignments on adjacent vertices, or edge-costs on incident edges, does not necessarily improve the SS.

In Theorem 4.1 in Section 4, we show that there are CSMs for which no optimal SS exists. In such cases obtaining a locally optimal SS, as defined in Definition 3.6, may provide us with a reasonable defensive posture.

4 Optimal Security Systems

One of the most natural and important questions to consider for a given CSM M is whether an optimal SS exists and if it does, what it would look like. Unfortunately, Theorem 4.1 shows that there are small and simple CSMs for which no optimal SS exists. Still we would like to know for what CSMs optimal SSs do exist, and, if possible, have a way to find these optimal SSs efficiently. Corollary 4.1 and Theorem 4.2 show that optimal SSs exist for CSMs $M = (T, C, P)$ when T is a path or a star, respectively. These theorems also yield $O(n \log n)$ algorithms for producing optimal SSs in these cases. But, these results are not satisfying, as they are limited. In Sections 5, 6, and 7 we study *P-* and *C-models* and completely characterize the types of trees that have optimal SSs.

We begin with a lemma showing that all optimal SSs must have the highest penetration costs assigned to the edges involving the root and level-one vertices.

Lemma 4.1. *Let $M = (T, C, P)$ be a CSM, where T rooted at r contains at least one non-root vertex. Let $V_1 \subseteq T(V)$ denote the level-one vertices of T, and let C_L be the multiset of the largest $|V_1|$ values in C. If an optimal (T, c, p) SS for M, exists, then $c(e) \in C_L$ for $e \in \{(r, v) \mid v \in V_1\}$.*

Proof. Suppose we have an optimal SS (T, c, p) that does not meet the conditions of the lemma. Let $c_s \notin C_L$ be the smallest penetration cost assigned by c to an edge between the root r and a vertex $v_s \in V_1$, that is, $c((r, v_s)) = c_s \leq c((r, v))$ for all $v \in V_1 - \{v_s\}$. Let $e_s = (r, v_s)$ and let e_l be an edge not between the root and a level-one vertex where $c(e_l) \in C_L$. We know that such an edge exists because (T, c, p) does not meet the conditions of the lemma. To show that (T, c, p) cannot be an optimal SS, we define a SS (T, c', p) by letting $c'(e_s) = c(e_l)$, $c'(e_l) = c(e_s)$, and $c'(e) = c(e)$ otherwise. Notice that for the budget $B = c_s$, we have $\text{pr}^\star(B, c, p) = p(v_s) > 0 = \text{pr}^\star(B, c', p)$. This fact contradicts that (T, c, p) is an optimal SS. □

If an optimal SS exists, Lemma 4.1 tells us something about its form. In the next theorem we show that there are CSMs for which no optimal SS exists.

Theorem 4.1. *There is a CSM $M = (T, C, P)$ for which no optimal security system exists.*

Proof. Consider $M = (T, \{1, 2, 3\}, \{1, 2, 3\})$, Where T is the tree given by $V(T) = \{r, u_1, u_2, u_3\}$ and $E(T) = \{e_1, e_2, e_3\}$ where $e_1 = (r, u_1)$, $e_2 = (r, u_2)$, and $e_3 = (u_1, u_3)$. By Lemma 4.1 we know that an optimal SS (T, c, p) has $c(e_3) = 1$, and we can further assume that $p(u_3) = 3$. By considering the budget of $B = 2$, we can also assume the prize of the head of the edge of cost 2 to by 1. Therefore, we have only two possible optimal SSs for M: (T, c, p) with $c(e_1, e_2, e_3) = (3, 2, 1)$ and $p(u_1, u_2, u_3) = (2, 1, 3)$, or (T, c', p') with $c'(e_1, e_2, e_3) = (2, 3, 1)$ and $p'(u_1, u_2, u_3) = (1, 2, 3)$, see Figure 2. Since $\text{pr}^\star(3, c, p) = 2$ and $\text{pr}^\star(3, c', p') = 4$, we see that (T, c', p') is not optimal, and since $\text{pr}^\star(4, c, p) = 5$ and $\text{pr}^\star(4, c', p') = 4$, we see that (T, c, p) is not optimal either. Hence, no optimal SS for M exists. □

Although Theorem 4.1 showed that there are CSMs for which no optimal SS exists, we are interested in finding out for which trees T optimal SSs do exist. We should point out that the values of the weights in C and P also play an important role in whether or not an optimal SS exists for a given tree. In the next theorem we show that an optimal SS exists for CSMs in which the tree in the model is a path, and this result is independent of the values of the weights in C and P.

Consider a CSM $M = (T, C, M)$ where T is a path rooted at a leaf, so

$$V(T) = \{u_0, u_1, \ldots, u_n\}, \quad E(T) = \{e_1, \ldots, e_n\}, \tag{3}$$

where $u_0 = r$ and $e_i = (u_{i-1}, u_i)$, for each $i \in \{1, \ldots, n\}$. For a SS (T, c, p) for M, then for convenience let $p_i = p(u_i)$ and $c_i = c(e_i)$ for each i. If we have $p_i \leq p_{i+1}$ and $c_i \geq c_{i+1}$ for each $i \in \{1, \ldots, n-1\}$ (so the prizes are ordered increasingly and the edge-costs decreasingly as we go down the path from the root), then by Proposition 3.1 the SS (T, c, p) is a good SS as in Definition 3.6. But, we can say slightly more here when T is a path, in terms of obtaining an improved SS as in Definition 3.3.

Lemma 4.2. *Let $M = (T, C, M)$ be a CSM where T is a path with its vertices and edges labeled as in (3).*

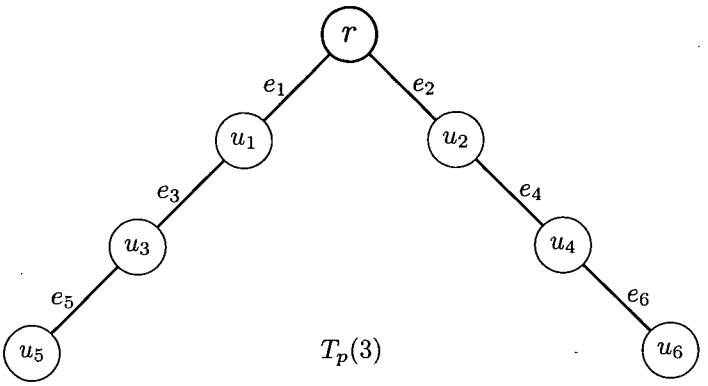


Figure 1: $T_p(3)$ is a path on seven vertices rooted at its center.

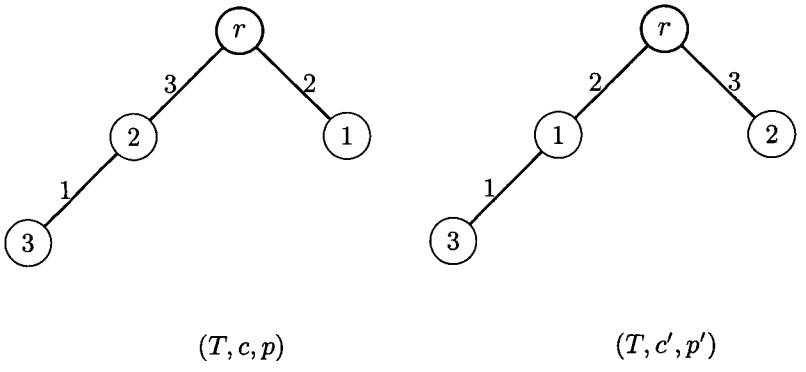


Figure 2: Only two possible SSs for $M = (T, \{1, 2, 3\}, \{1, 2, 3\})$.

(i) If (T, c, p) is a SS for M and there is an i with $p_i > p_{i+1}$ and $c_{i+1} > 0$, then the SS (T, c, p') where p' is obtained by swapping the prizes on u_i and u_{i+1} is an improved SS.

(ii) If (T, c, p) is a SS for M and there is an i with $c_i < c_{i+1}$, then the SS (T, c', p) where c' is obtained by swapping the edges costs on e_i and e_{i+1} is an improved SS.

Proof. By Proposition 3.1 we only need to show (i) there is a budget B' such that $\mathrm{pr}^\star(B', c, p') < \mathrm{pr}^\star(B', c, p)$ and (ii) a budget B'' such that $\mathrm{pr}^\star(B'', c', p) < \mathrm{pr}^\star(B'', c, p)$. For each j let $\tau_j = T[e_1, \ldots, e_j]$ be the rooted sub-path of T that contains the first j edges of T.

For $B' = c_1 + \cdots + c_i$ we clearly have

$$\begin{aligned} \mathrm{pr}^\star(B', c, p') &= \mathrm{pr}(\tau_i, c, p') \\ &= p_1 + \cdots + p_{i-1} + p_{i+1} \\ &< p_1 + \cdots + p_i \\ &= \mathrm{pr}(\tau_i, c, p) \\ &= \mathrm{pr}^\star(B', c, p), \end{aligned}$$

showing that (T, c, p') is an improved SS for M.

Likewise, we have

$$\begin{aligned} \mathrm{pr}^\star(B', c', p) &= \mathrm{pr}(\tau_{i-1}, c', p) \\ &= p_1 + \cdots + p_{i-1} \\ &< p_1 + \cdots + p_i \\ &= \mathrm{pr}(\tau_i, c, p) \\ &= \mathrm{pr}^\star(B', c, p), \end{aligned}$$

showing that (T, c', p) is also an improved SS for M. □

Given any SS (T, c, p) for M as in Lemma 4.2 when T is a rooted path, by bubble sorting the prizes and the edge costs increasingly and decreasingly respectively, as we go down the path T from the root, we obtain by Lemma 4.2 a SS (T, c', p') such that for any budget B we have $\mathrm{pr}^\star(B, c', p') \leq \mathrm{pr}^\star(B, c, p)$. We therefore have the following corollary.

Corollary 4.1. *If $M = (T, C, M)$ is a CSM where T is a rooted path with its vertices and edges labeled as in (3), then there is an optimal SS for M, and it is given by assigning the penetration costs to the edges and the prizes to the vertices in a decreasing order and increasing order respectively from the root.*

We now show that an optimal SS exists for $M = (T, C, P)$ when T is a star. Let T be a star with root r and non-root vertices $u_1, \ldots, n_n$ and edges $e_i = (r, u_i)$ for $i = 1, \ldots, n$. Suppose the costs and prizes are given by $C = \{c_1, \ldots, c_n\}$ and $P = \{p_1, \ldots, p_n\}$. When considering an arbitrary security system (T, c, p) where $c(u_i) = c_i$ and $p(e_i) = p_i$ for each i, we can without loss of generality assume the edge-costs to be in an increasing order $c_1 \leq \cdots \leq c_n$.

Lemma 4.3. *Suppose T is a star and (T, c, p) is a SS as above. If p' is another prize assignment obtained from p by swapping the prizes p_i and p_j where $i < j$ and $p_i \leq p_j$, then for any budget B we have* $\mathrm{pr}^\star(B, c, p) \leq \mathrm{pr}^\star(B, c, p')$.

Proof. Let B be a given budget and $\tau \subseteq T$ an optimal attack with respect to p, so $\mathrm{pr}(\tau, c, p) = \mathrm{pr}^\star(B, c, p)$. We consider the following cases.

CASE ONE: If both of u_i and u_j are in τ, or neither of them are, then we haef $\mathrm{pr}^\star(B, c, p) = \mathrm{pr}(\tau, c, p) = \mathrm{pr}(\tau, c, p') \leq \mathrm{pr}^\star(B, c, p')$.

CASE TWO: If $u_i \in V(\tau)$ and $u_j \notin V(\tau)$, then $\mathrm{pr}^\star(B, c, p) = \mathrm{pr}(\tau, c, p) \leq \mathrm{pr}(\tau, c, p) - p_i + p_j = \mathrm{pr}(\tau, c, p') \leq \mathrm{pr}^\star(B, c, p')$.

CASE THREE: If $u_i \notin V(\tau)$ and $u_j \in V(\tau)$, then $\tau' = (\tau - u_j) \cup u_i$ is a rooted subtree of T with $c(\tau') = c(\tau) - c_j + c_i \leq B$ and is therefore within the budget B. Hence, $\mathrm{pr}^\star(B, c, p) = \mathrm{pr}(\tau, c, p) = \mathrm{pr}(\tau', c, p') \leq \mathrm{pr}^\star(B, c, p')$.

Therefore, in all cases we have $\mathrm{pr}^\star(p, c, B) \leq \mathrm{pr}^\star(p', c, B)$. □

Since any permutation is a composition of transpositions, we have the following theorem as a corollary.

Theorem 4.2. *Let $M = (T, C, P)$ be a CSM where T is a star rooted at its center vertex. Then there is an optimal SS for M, and it is given by assigning the prizes to the vertices in the same increasing order as the costs are assigned increasingly to the corresponding edges.*

For rooted trees on n non-root vertices, Corollary 4.1 and Theorem 4.2 give rise to natural sorting-based $O(n \log n)$ algorithms for computing optimal SSs. Notice that in an optimal SS in a general tree, the smallest prize overall must be assigned to a level-one vertex u which has the largest penetration cost assigned to its corresponding edge, (r, u), to the root. And, furthermore, we cannot say more than this statement for arbitrary trees as the next assignment of a prize will depend on the relative values of the penetration costs, prizes, and structure of the tree. In view of the fact that optimal SSs do not exist, except for paths and stars as we will see shortly in Observation 5.1, we turn our attention to restricted CSMs and classify them with respect to optimal SSs.

5 Specific Security Systems, P-Models, and C-Models

In this section we extend CSMs to include penetration costs and prizes of value zero. For a CSM $M = (T, C, P)$ with no optimal SS and a rooted super-tree $T^\dagger$ of which T is a rooted subtree, we can always assign the prize of zero to the nodes in $V(T^\dagger) \setminus V(T)$ and likewise the penetration cost of zero to the edges in $E(T^\dagger) \setminus E(T)$, thereby obtaining a CSM $M^\dagger = (T^\dagger, C^\dagger, E^\dagger)$ that also has no optimal SS. Note that if T is the rooted tree in the proof of Theorem 4.1, then the only rooted trees that do not have T as a rooted subtrees are paths rooted at one of their leaves or stars rooted at their center vertices. Hence, by the example provided in the proof of Theorem 4.1, we have the following observation.

Observation 5.1. *If T is a rooted tree, such that for any multisets C and P of penetration costs and prizes, respectively, the CSM $M = (T, C, P)$ has an optimal SS, then T is either a path rooted at one of its leaves, or a star rooted at its center vertex.*

In light of Observation 5.1, we seek some natural restrictions on our CSM M that will guarantee it having an optimal SS. Since both the penetration costs and the prizes of $M = (T, C, P)$ take values in $\mathbb{Q}_+$ we can, by an appropriate scaling, obtain an equivalent CSM where both the costs and prizes take values in $\mathbb{N} \cup \{0\}$, that is, we may assume $c(e) \in \mathbb{N} \cup \{0\}$ and $p(u) \in \mathbb{N} \cup \{0\}$ for every $e \in E(T)$ and $u \in V(T)$, respectively.

First, we consider the restriction on a CSM $M = (T, C, P)$ where C consists of a single penetration-cost value, that is, $C = \{1, 1, \ldots, 1\}$ consists of n copies of the unit penetration cost one. From a realistic point of view, this assumption seems to be reasonable; many computer networks consist of computers with similar password/encryption security systems on each computer (that is, the penetration cost is the same for all of the computers), whereas the computers might store data of vastly distinct values (that is, the prizes are distinct).

CONVENTION: In what follows, it will be convenient to denote the multiset containing n (or an arbitrary number of) copies of 1 by I. In a similar way, we will denote by $\mathbf{1}$ the map that maps each element of the appropriate domain to 1. As the domain of $\mathbf{1}$ should be self-evident each time, there should be no ambiguity about it each time.

Definition 5.1. *A* P-model *is a CSM $M = (T, I, P)$ where T has n non-root vertices and where I is constant, consisting of n copies of the unit penetration cost.*

Consider a SS (T, c, p) of a CSM $M = (T, C, P)$. We can obtain an *equivalent* SS $(T', \mathbf{1}, p')$ of a P-model $M' = (T', I, P')$ in the following way: for each edge $e = (u, v) \in E(T)$ with penetration cost $c(e) = k \in \mathbb{N}$ and prizes $p(u), p(v) \in \mathbb{N}$ of its head and tail, respectively, replace the 1-path (u, e, v) with a directed path of new vertices and edges $(u, e_1, u_1, e_2, u_2, \ldots, u_{k-1}, e_k, v)$ of length k. We extend the penetration cost and prize functions by adding zero-prize vertices where needed, that is, $\mathbf{1}(f) = 1$ for each $f \in E(T')$, and we let

$$p'(u) = p(u), \quad p'(v) = p(v), \text{ and } p'(u_1) = p'(u_2) = \cdots = p'(u_{k-1}) = 0.$$

In this way we obtain a SS (T', c', p') of a P-model $M' = (T', I, P')$. We view the vertices $V(T)$ of positive prize as a subset of $V(T')$ (namely, those vertices of T' with positive prize).[1]

Recall that T is a *rooted contraction* of T' if T is obtained from T' by a sequence of simple contractions of edges, and where any vertex contracted into the root remains the root. This means precisely that T is a rooted *minor* of T' [2, p. 54].

[1]Note that there are some redundant definitions on the prizes of the vertices when considering incident edges, but the assignments do agree, as they have the same prize values as in T.

Proposition 5.1. *Any SS (T, c, p) of a CSM $M = (T, C, P)$ is equivalent to a SS $(T', \mathbf{1}, p')$ of a P-model $M' = (T', I, P')$ where (i) T is rooted minor of T', and (ii) $p'(u) = p(u)$ for each $u \in V(T) \subseteq V(T')$, and $p'(u) = 0$, otherwise.*

Proof. (Sketch) Given a budget $B \in \mathbb{Q}_+$, clearly any optimal attack τ on a SS (T, c, p) with $\mathrm{pr}(\tau, c, p) = \mathrm{pr}^\star(B, c, p)$ has an equivalent attack τ' on a SS $(T', \mathbf{1}, p')$ of the same cost $\mathrm{cst}(\tau', \mathbf{1}, p') = \mathrm{cst}(\tau, c, p)$ and hence within the budget B, where τ' is the smallest subtree of T' that contains all of the vertices of τ. By construction, we also have that $\mathrm{pr}(\tau', \mathbf{1}, p') = \mathrm{pr}(\tau, c, p) = \mathrm{pr}^\star(B, c, p)$ since all of the vertices from τ are in τ' and have the same prize there, and the other vertices in τ' have prize zero. This shows that $\mathrm{pr}^\star(B, c, p) \leq \mathrm{pr}^\star(B, \mathbf{1}, p')$.

Conversely, an optimal attack τ' on $(T', \mathbf{1}, p')$ with $\mathrm{pr}(\tau', \mathbf{1}, p') = \mathrm{pr}^\star(B, \mathbf{1}, p')$ yields an attack τ on (T, c, p) by letting τ be the subtree of T induced by the vertices $V(\tau') \cap V(T)$. In this way $\mathrm{pr}(\tau, c, p) = \mathrm{pr}(\tau', \mathbf{1}, p')$ and $\mathrm{cst}(\tau, c, p) \leq \mathrm{cst}(\tau', \mathbf{1}, p')$, since some of the vertices of τ' might have zero prize, as they are not in τ. By definition of $\mathrm{pr}^\star(\cdot)$ we have that $\mathrm{pr}^\star(B, \mathbf{1}, p') \leq \mathrm{pr}^\star(B, c, p)$. Hence, the SS (T, c, p) and $(T', \mathbf{1}, p')$ are equivalent. □

Secondly, and dually, we can restrict our attention to the case where the multiset of prizes P consists of a single unit prize value, so $P = I = \{1, 1, \ldots, 1\}$ consists of n copies of the unit prize.

Definition 5.2. *A* C-model *is a CSM $M = (T, C, I)$, where T has n non-root vertices and where I is constant, consisting of n copies of the unit prize.*

As before, consider a SS (T, c, p) of a CSM $M = (T, C, P)$. We can obtain an equivalent SS $(T'', c'', \mathbf{1})$ of a C-model $M'' = (T'', C'', I)$ in the following way: for each edge $e = (u, v) \in E(T)$ with penetration cost $c(e) = k \in \mathbb{N}$ and prizes $p(u), p(v) \in \mathbb{N}$ of its head and tail, respectively, replace the 1-path (u, e, v) with a directed path of new vertices and edges $(u, e, u_1, e_1, u_2, \ldots, u_{k-1}, e_{k-1}, v)$ of length k. We extend the penetration cost and prize functions by adding zero-cost edges where needed, that is, $\mathbf{1}(w) = 1$ for every $w \in V(T'')$, and we let

$$c''(e) = c(e) \quad \text{and } c''(e_1) = c''(e_2) = \cdots = c''(e_{k-1}) = 0.$$

In this way we obtain a SS $(T'', c'', \mathbf{1})$ of a C-model $M'' = (T'', C'', I)$, where the multiset of prizes consists of a single unit prize value ($\sum_{u \in V(T) \setminus \{r\}} p(u)$ copies of it). We also view the edges $E(T)$ of positive penetration cost as a subset of $E(T'')$ (namely, those edges of T'' with positive penetration cost). We also have the following proposition that is dual to Proposition 5.1.

Proposition 5.2. *Any SS (T, c, p) of a CSM $M = (T, C, P)$ is equivalent to a SS $(T'', c'', \mathbf{1})$ of a C-model $M'' = (T'', C'', I)$, where (i) T is rooted minor of T'', and (ii) $c''(e) = c(e)$ for each $e \in E(T) \subseteq E(T'')$, and $c''(e) = 0$, otherwise.*

Proof. (Sketch) Suppose we are given a budget $B \in \mathbb{Q}_+$ and an optimal attack τ on a SS (T, c, p) with $\mathrm{pr}(\tau, c, p) = \mathrm{pr}^\star(B, c, p)$. Here $(T'', c'', \mathbf{1})$ has an equivalent attack

τ'', where τ'' is the largest subtree of T'' that contains all of the edges of τ and no other edges of T. Note that $\text{cst}(\tau'', c'', \mathbf{1}) = \text{cst}(\tau, c, p)$ since all of the additional edges of τ'' that are not in $V(\tau)$ have zero penetration cost, and so τ'' is within the budget B. Also, by construction we have $\text{pr}(\tau'', c'', \mathbf{1}) = \text{pr}(\tau, c, p) = \text{pr}^\star(B, c, p)$. This result shows that $\text{pr}^\star(B, c, p) \leq \text{pr}^\star(B, c'', \mathbf{1})$.

Conversely, consider an optimal attack τ'' on $(T'', c'', \mathbf{1})$ with $\text{pr}(\tau'', c'', \mathbf{1}) = \text{pr}^\star(B, c'', \mathbf{1})$. By the optimality of τ'', every leaf of τ'' is a tail of an edge of T, since otherwise we can append that edge (of zero penetration cost), and thereby obtain an attack with a prize strictly more than $\text{pr}(\tau'', c'', \mathbf{1})$, a contradiction. The edges $E(\tau'') \cap E(T)$ induce a subtree τ of T of the same cost $\text{cst}(\tau, c, p) = \text{cst}(\tau'', c'', \mathbf{1})$; and moreover, τ'' is, by its optimality, the largest subtree of T'' that contains exactly all of the edges of τ, and so $\text{pr}(\tau, c, p) = \text{pr}(\tau'', c'', \mathbf{1}) = \text{pr}^\star(B, c'', \mathbf{1})$. This result shows that $\text{pr}^\star(B, c'', \mathbf{1}) \leq \text{pr}^\star(B, c, p)$. This proves that the SS (T, c, p) and $(T'', c'', \mathbf{1})$ are equivalent. □

We now present some examples of both P- and C-models that will play a pivotal role in our discussion to come.

Definition 5.3. *Let $T(2)$ denote the rooted tree given as follows:*

$$\begin{aligned} V(T(2)) &= \{r, u_1, u_2, u_3, u_4, u_5\}, \\ E(T(2)) &= \{(r, u_1), (r, u_2), (u_1, u_3), (u_2, u_4), (u_2, u_5)\}. \end{aligned}$$

Note that $T(2)$ has all of its non-root vertices on two non-zero levels. Similarly, let $T(3)$ denote the rooted tree given as follows:

$$\begin{aligned} V(T(3)) &= \{r, u_1, u_2, u_3, u_4\}, \\ E(T(3)) &= \{(r, u_1), (r, u_2), (u_2, u_3), (u_3, u_4)\}. \end{aligned}$$

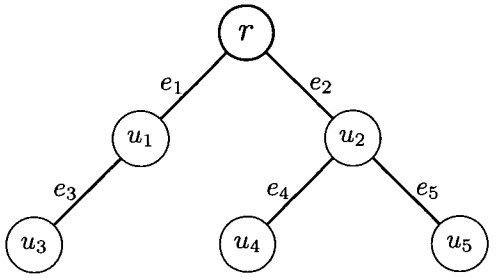
Note that $T(3)$ has all of its vertices on three non-zero levels.

CONVENTION: For convenience we label the edges of both $T(2)$ and $T(3)$ with the same index as their heads (see Figures 3 and 4):

$$\begin{aligned} T(2) &: e_1 = (r, u_1),\ e_2 = (r, u_2),\ e_3 = (u_1, u_3),\ e_4 = (u_2, u_4),\ e_5 = (u_2, u_5). \\ T(3) &: e_1 = (r, u_1),\ e_2 = (r, u_2),\ e_3 = (u_1, u_3),\ e_4 = (u_3, u_4). \end{aligned}$$

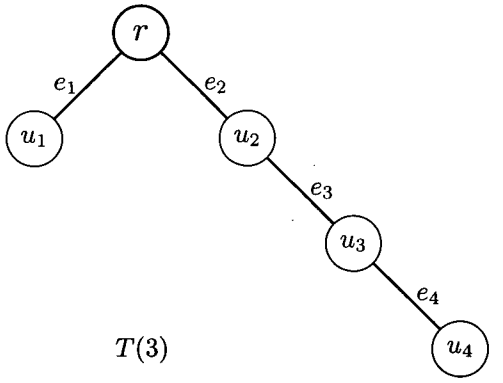
Example 5.1.
Consider a P-model (with $c = \mathbf{1}$) on the rooted tree $T(2)$, where the prize values are given by $P = \{0, 1, 2, 2, 3\}$.

Prize Assignment (A): Consider the case where the prizes have been simultaneously assigned to the non-root vertices of $T(2)$ by $p(u_1, u_2, u_3, u_4, u_5) := (0, 1, 3, 2, 2)$ in the obvious way. We will use a similar shorthand notation later for the bijection c. In this case we see that for budgets of $B = 2, 3$, we have $\text{pr}^\star(2, \mathbf{1}, p) = 3$ and $\text{pr}^\star(3, \mathbf{1}, p) = 5$, respectively.



$T(2)$

Figure 3: $T(2)$ has all of its non-root vertices on two non-zero levels.



$T(3)$

Figure 4: $T(3)$ has all of its non-root vertices on three non-zero levels.

Prize Assignment (B): Consider now the case where the prizes have been simultaneously assigned to the non-root vertices of $T(2)$ by $p'(u_1, u_2, u_3, u_4, u_5) := (1, 0, 3, 2, 2)$. In this case we see that for the same budgets of $B = 2, 3$ as in (A), we have $\mathrm{pr}^\star(2, \mathbf{1}, p') = 4$ and $\mathrm{pr}^\star(3, \mathbf{1}, p') = 4$, respectively.

From these assignments we see that for budget $B = 2$, the SS in (A) is better than the one in (B), and for $B = 3$, the SS in (B) is better than the one in (A).

Example 5.2.

Consider a P-model on the rooted tree $T(3)$, where the prize values are given by $P = \{0, 0, 1, 1\}$.

Prize Assignment (A): Consider the case where the prizes have been simultaneously assigned to the non-root vertices of $T(3)$ by $p(u_1, u_2, u_3, u_4) := (0, 0, 1, 1)$. In this case we see that for budgets of $B = 1, 3$, we have $\mathrm{pr}^\star(1, \mathbf{1}, p) = 0$ and $\mathrm{pr}^\star(3, \mathbf{1}, p) = 2$, respectively.

Prize Assignment (B): Consider now the case where the prizes have been simultaneously assigned to the non-root vertices of $T(3)$ by $p'(u_1, u_2, u_3, u_4) := (1, 0, 0, 1)$. In this case we see that for the same budgets of $B = 1, 3$ as in (A), we have $\mathrm{pr}^\star(1, \mathbf{1}, p') = 1$ and $\mathrm{pr}^\star(3, \mathbf{1}, p') = 1$, respectively.

From these assignments we see that for budget $B = 1$, the SS in (A) is better than the one in (B), and for $B = 3$, the SS in (B) is better than the one in (A).

Considering the budget $B = 1$ for the P-model in Example 5.1, we see that in order for a prize assignment to be optimal we must have the prizes of u_1 and u_2 to be 0 and 1. Considering further $B = 2$ we see that an optimal prize assignment in this case must be p or p' as in Example 5.1, or p'' where $p''(u_1, u_2, u_3, u_4, u_5) := (1, 0, 2, 3, 2)$. Since $\mathrm{pr}^\star(B, \mathbf{1}, p'') = \mathrm{pr}^\star(B, \mathbf{1}, p)$ for any B, we see that the P-model in Example 5.1 has no optimal SS. As the P-model in Example 5.2 can be analysed in the same way, we have the following observation.

Observation 5.2. *For general prize values P, neither of the P-models $M = (T(2), I, P)$ nor $M = (T(3), I, P)$ have optimal SSs.*

We will now consider the dual cases of the C-models.

Example 5.3.

Consider a C-model (with $p = \mathbf{1}$) on the rooted tree $T(2)$, where the penetration costs are given by $C = \{0, 1, 1, 2, 3\}$.

Cost Assignment (A): Consider the case where the penetration costs have been simultaneously assigned to the edges of $T(2)$ by $c(e_1, e_2, e_3, e_4, e_5) := (3, 2, 0, 1, 1)$. In this case we see that for budgets of $B = 2, 4$, we have $\mathrm{pr}^\star(2, c, \mathbf{1}) = 1$ and $\mathrm{pr}^\star(4, c, \mathbf{1}) = 3$, respectively.

Cost Assignment (B): Consider now the case where the penetration costs have been assigned to the edges of $T(2)$ by $c'(e_1, e_2, e_3, e_4, e_5) := (2, 3, 0, 1, 1)$. In this case we see that for the same budgets of $B = 2, 4$ as in (A), we have $\mathrm{pr}^\star(2, c', \mathbf{1}) = 2$ and $\mathrm{pr}^\star(4, c', \mathbf{1}) = 2$, respectively.

From these assignments we see that for budget $B = 2$, the SS in (A) is better than the one in (B), and for $B = 4$, the SS in (B) is better than the one in (A).

Example 5.4.

Consider now a C-model on the rooted tree $T(3)$, where the penetration costs are given by $C = \{0, 0, 1, 1\}$.

Cost Assignment (A): Consider the case where the penetration costs have been simultaneously assigned to the edges of $T(3)$ by $c(e_1, e_2, e_3, e_4) := (1, 1, 0, 0)$. In this case we see that for budgets of $B = 0, 1$, we have $\mathrm{pr}^\star(0, c, \mathbf{1}) = 0$ and $\mathrm{pr}^\star(1, c, \mathbf{1}) = 3$, respectively.

Cost Assignment (B): Consider now the case where the penetration costs have been assigned to the edges of $T(3)$ by $c'(e_1, e_2, e_3, e_4) := (0, 1, 1, 0)$. In this case we see that for the same budgets of $B = 0, 1$ as in (A), we have $\mathrm{pr}^\star(0, c', \mathbf{1}) = 1$ and $\mathrm{pr}^\star(1, c', \mathbf{1}) = 2$, respectively.

From these assignments we see that for budget $B = 0$, the SS in (A) is better than the one in (B), and for $B = 1$, the SS in (B) is better than the one in (A).

In a similar way as we obtained Observation 5.2, we see from the previous two examples the following.

Observation 5.3. *For general penetration costs C, neither of the C-models $M = (T(2), C, I)$ nor $M = (T(3), C, I)$ have optimal SSs.*

Remark 5.1. (i) Note that in Examples 5.1 and 5.3 involving the rooted tree $T(2)$, we have that the prize assignments to the non-root vertices and cost assignments to the corresponding edges sum up to a constant vector for both assignments *(A)* and *(B)*:

$$\begin{aligned}
(A) \quad &: \quad p(u_1, u_2, u_3, u_4, u_5) + c(e_1, e_2, e_3, e_4, e_5) \\
&\quad = (0, 1, 3, 2, 2) + (3, 2, 0, 1, 1) = (3, 3, 3, 3, 3), \\
(B) \quad &: \quad p'(u_1, u_2, u_3, u_4, u_5) + c'(e_1, e_2, e_3, e_4, e_5) \\
&\quad = (1, 0, 3, 2, 2) + (2, 3, 0, 1, 1) = (3, 3, 3, 3, 3),
\end{aligned}$$

and similarly for the rooted tree $T(3)$:

$$\begin{aligned}
(A) \quad &: \quad p(u_1, u_2, u_3, u_4) + c(e_1, e_2, e_3, e_4) = (0, 0, 1, 1) + (1, 1, 0, 0) = (1, 1, 1, 1), \\
(B) \quad &: \quad p'(u_1, u_2, u_3, u_4) + c'(e_1, e_2, e_3, e_4) = (1, 0, 0, 1) + (0, 1, 1, 0) = (1, 1, 1, 1).
\end{aligned}$$

This duality is not a coincidence and will discussed in more detail in Section 7. (ii) Although special cases of Theorems 6.1, 6.2, 7.3 and 7.4, it is an easy combinatorial exercise to see that both a C- or P-model $M = (T, C, P)$, where T is a proper rooted subtree of either $T(2)$ or $T(3)$ does indeed have an optimal SS, and so $T(2)$ and $T(3)$ are the smallest rooted trees, in either model, with no optimal SS. This point will also be discussed and stated explicitly in Sections 6 and 7.

Consider now a given rooted tree T and another rooted tree $T^\dagger$ containing T as a rooted subtree, so $T \subseteq T^\dagger$. Assume that the P-model $M = (T, I, P)$ has no optimal SS. Extend M to a P-model on $T^\dagger$ by adding a zero prize for each vertex in $V(T^\dagger) \setminus V(T)$, so $P^\dagger = P \cup Z$, where Z is the multiset consisting of $|V(T^\dagger)| - |V(T)|$ copies of 0. In this case we have the following.

Observation 5.4. *If $M = (T, I, P)$ is a P-model with no optimal SS, and $T^\dagger$ contains T as a rooted subtree, then the P-model $M^\dagger = (T^\dagger, I, P^\dagger)$ has no optimal SS.*

Proof. (Sketch) For any budget consisting of $B = m$ edges and a SS $(T, \mathbf{1}, p)$, there is a rooted subtree τ of T with m edges such that $\mathrm{pr}(\tau, \mathbf{1}, p) = \mathrm{pr}^\star(m, \mathbf{1}, p)$. Let $\mathbf{1}$ and $p^\dagger$ be the obvious extensions of $\mathbf{1}$ and p to $T^\dagger$, by letting $\mathbf{1}(e) = 1$ for all $e \in E(T^\dagger)$ and $p^\dagger(u) = 0$ for any $u \in V(T^\dagger) \setminus V(T)$. If τ' is a rooted subtree of $T^\dagger$ with m edges, then $\tau' \cap T$ is a rooted subtree of both T and $T^\dagger$ on m or fewer edges. Since any vertex of $V(\tau') \setminus V(T)$ has zero prize, we have

$$\mathrm{pr}(\tau', \mathbf{1}, p^\dagger) = \mathrm{pr}(\tau' \cap T, \mathbf{1}, p^\dagger) = \mathrm{pr}(\tau' \cap T, \mathbf{1}, p) \leq \mathrm{pr}^\star(m, \mathbf{1}, p),$$

with equality for $\tau' = \tau$ since $\tau \subseteq T \subseteq T^\dagger$. Hence, $\mathrm{pr}^\star(m, \mathbf{1}, p^\dagger) = \mathrm{pr}^\star(m, \mathbf{1}, p)$, and we conclude that if $M = (T, I, P)$ has no optimal SS, then neither does $M^\dagger = (T^\dagger, I, P^\dagger)$. □

Dually, assume that we have a C-model $M = (T, C, I)$ that has no optimal SS, and similarly, let $T^\dagger$ be a rooted subtree containing T as a rooted subtree. Extend M to a C-model on $T^\dagger$ by adding penetration costs of ∞[2] for each edge of $T^\dagger$ that is not in T, so $C^\dagger = C \cup Y$, where Y is the multiset consisting of $|E(T^\dagger)| - |E(T)|$ copies of ∞.

Observation 5.5. *If $M = (T, C, I)$ is a C-model with no optimal SS, and $T^\dagger$ contains T as a rooted subtree, then the C-model $M^\dagger = (T^\dagger, C^\dagger, I)$ has no optimal SS.*

Proof. (Sketch) The proof is similar to the one for Observation 5.4. For any budget $B \in \mathbb{Q}_+$ and a SS $(T, c, \mathbf{1})$ of M, there is a rooted subtree τ of T with m edges such that $\mathrm{pr}(\tau, c, \mathbf{1}) = \mathrm{pr}^\star(B, c, \mathbf{1})$. Let $c^\dagger$ and $\mathbf{1}$ be the obvious extensions of c and $\mathbf{1}$ to $T^\dagger$, by letting $c^\dagger(e) = \infty$ for all $e \in E(T^\dagger) \setminus E(T)$. If τ' is a rooted subtree of $T^\dagger$ within the attacker's budget of $B < \infty$, then every edge of τ' must be in T, and so $\tau' \subseteq T \subseteq T^\dagger$. Since $c^\dagger$ agrees with c on the edges of T we have

$$\mathrm{pr}(\tau', c^\dagger, \mathbf{1}) = \mathrm{pr}(\tau', c, \mathbf{1}) \leq \mathrm{pr}^\star(B, c, \mathbf{1}),$$

with equality for $\tau' = \tau$. Hence, $\mathrm{pr}^\star(B, c^\dagger, 1) = \mathrm{pr}^\star(B, c, \mathbf{1})$, and we conclude that if $M = (T, C, I)$ has no SS, then neither does $M^\dagger = (T^\dagger, C^\dagger, I)$. □

By Observations 5.2, 5.3, 5.4, and 5.5 we have the following corollary.

[2] Where here we can choose ∞ to be the number of edges of T plus one, that is, a large number exceeding any sensible attack budget.

Corollary 5.1. *If T is a rooted tree such that any P- or C-model $M = (T, C, P)$ has an optimal SS, then T contains neither $T(2)$ nor $T(3)$ as rooted subtrees.*

Let T be a rooted tree such that any CSM $M = (T, C, P)$ has an optimal SS. Assume further that T is not a path rooted at one of its two leaves. If T has at least three non-zero levels (we consider the root r to be the unique level-0 vertex), then T must contain $T(3)$ as a rooted subtree and hence, by Corollary 5.1, there is a CSM $M = (T, C, P)$ with no optimal SS, contradicting our assumption on T. Consequently, T has at most two non-zero levels.

If T has at most two non-zero levels, and it has two leaves of distance four apart (with the root r being midways between them), then neither parent of the leaves is of degree three or more, because then T has $T(2)$ as a rooted subtree. And, so again, by Corollary 5.1, there is a CSM $M = (T, C, P)$ with no optimal SS. This observation again contradicts our assumption on T. As a result, either (i) T has a diameter of three and is obtained by attaching an arbitrary number of leaves to the end vertices of a single edge and then rooting it at one of the end-vertices of the edge, or (ii) T has diameter of four and each level-one vertex has degree at most two.

Recall that a *caterpillar tree* is a tree where each vertex is within distance one of a central path, and that a *spider tree* is a tree with one vertex of degree at least three and all other vertices of degree at most two.

Definition 5.4. *A* rooted path *is a path rooted at one of its two leaves.*

A rooted star *is a star rooted at its unique center vertex.*

A 3-caterpillar *is a caterpillar tree of diameter three.*

A rooted 3-caterpillar *is a 3-caterpillar rooted at one of its two center vertices.*

A 4-spider *is a spider tree of diameter four with its unique center vertex of degree at least three.*

A rooted 4-spider *is a 4-spider rooted at its unique center vertex.*

By Corollary 5.1 and the discussion just before Definition 5.4, we therefore have the following main theorem of this section.

Theorem 5.1. *If T is a rooted tree such that any P- or C-model $M = (T, C, P)$ has an optimal SS, then T is one of the following types: (i) a rooted path, (ii) a rooted star, (iii) a rooted 3-caterpillar, or (iv) a rooted 4-spider.*

It remains to be seen whether or not a rooted 3-caterpillar or a rooted 4-spider T is such that any P- or C-model $M = (T, C, P)$ has an optimal SS. This item will be the main topic of the next two sections.

6 P-models with Optimal Security Systems

In this section we prove that if T is one of the four types of rooted trees mentioned in Theorem 5.1, then any P-model $M = (T, I, P)$ indeed has an optimal SS. The

C-models will be discussed in Section 7. We already have that any P-model $M = (T, I, P)$ (in fact, any CSM $M = (T, C, P)$), where T is a rooted path or a rooted star, does have an optimal SS, so it suffices to consider rooted 3-caterpillars and rooted 4-spiders.

Let T be a rooted 3-caterpillar on vertices $\{r, u_1, \ldots, u_n\}$ with edges given by

$$E(T) = \{(r, u_1), \ldots, (r, u_k), (u_1, u_{k+1}), \ldots, (u_1, u_n)\}, \tag{4}$$

where $2 \leq k \leq n-1$. As before, we label the edges by the index of their heads, so $e_i = (r, u_i)$ for $i \in \{1, \ldots, k\}$ and $e_i = (u_1, u_i)$ for $i \in \{k+1, \ldots, n\}$. Our first result is the following.

Theorem 6.1. *Let $M = (T, I, P)$ be a P-model where T is a rooted 3-caterpillar and $P = \{p_1, \ldots, p_n\}$ is a multiset of possible prizes indexed increasingly $p_1 \leq p_2 \leq \cdots \leq p_n$. Then the SS $(T, \mathbf{1}, p)$, where $p(u_i) = p_i$ for each $i \in \{1, \ldots, n\}$ is an optimal SS for M.*

Proof. Let $B = m \in \{0, 1, \ldots, n\}$ be the attacker's budget, that is the number of edges an adversary can afford to penetrate. We want to show that $\mathrm{pr}^\star(m, \mathbf{1}, p) \leq \mathrm{pr}^\star(m, \mathbf{1}, p')$ for any prize assignment p' to the vertices of the rooted 3-caterpillar T.

Let $\tau \subseteq T$ be a rooted subtree of T on m edges with $\mathrm{pr}(\tau, \mathbf{1}, p) = \mathrm{pr}^\star(m, \mathbf{1}, p)$. There are two cases we need to consider.

FIRST CASE: $e_1 \in E(\tau)$. Since all the leaves are connected to one of the end-vertices of $e_1 = (r, u_1)$, the remaining $m-1$ edges of τ must be incident to the $m-1$ maximum prize vertices, and so $\mathrm{pr}^\star(m, \mathbf{1}, p) = \mathrm{pr}(\tau, \mathbf{1}, p) = p_n + p_{n-1} + \cdots + p_{n-m+2} + p_1$. If p' is another prize assignment to the vertices of T, then $p'(u_1) = p_c$, where $c \in \{1, \ldots, n\}$. Therefore, $\mathrm{pr}^\star(m, \mathbf{1}, p') \geq \mathrm{pr}(\tau', \mathbf{1}, p')$, where τ' is a rooted subtree of T that contains e_1 and contains all the remaining $m-1$ maximum prizes, and so

$$\mathrm{pr}(\tau', \mathbf{1}, p') = \begin{cases} p_n + p_{n-1} + \cdots + p_{n-m+1} & \text{if } c \in \{n-m+1, \ldots, n\}, \\ p_n + p_{n-1} + \cdots + p_{n-m+2} + p_c & \text{if } c \notin \{n-m+1, \ldots, n\}. \end{cases}$$

In either case we have $\mathrm{pr}(\tau', \mathbf{1}, p') \geq p_n + p_{n-1} + \cdots p_{n-m+2} + p_1 = \mathrm{pr}^\star(m, \mathbf{1}, p)$, and so $\mathrm{pr}^\star(m, \mathbf{1}, p') \geq \mathrm{pr}^\star(m, \mathbf{1}, p)$ in this case.

SECOND CASE: $e_1 \notin E(\tau)$. For this case to be possible we must have $m \leq k-1$, since otherwise e_1 must be in τ. Secondly, we must have that τ contains all the maximum prize vertices on level one and so $\mathrm{pr}^\star(m, \mathbf{1}, p) = \mathrm{pr}(\tau, \mathbf{1}, p) = p_k + p_{k-1} + \cdots + p_{k-m+1}$. In particular, we must have

$$p_k + p_{k-1} + \cdots + p_{k-m+1} \geq p_n + p_{n-1} + \cdots + p_{n-m+2} + p_1,$$

since a tree containing e_1 does not have a greater total prize than τ. If p' is another prize assignment to the vertices of T, then let $\{\ell_1, \ldots, \ell_k\}$ be the indices of the prizes assigned to vertices on level one by p', that is, $\{p_{\ell_1}, \ldots, p_{\ell_k}\} = \{p'(u_1), \ldots, p'(u_k)\}$

as multisets. If now τ' is the rooted subtree of T with m edges containing the m vertices with the largest prizes, then, since $p_{\ell_i} \geq p_i$ for each $i \in \{1, \ldots, k\}$, we have

$$\begin{aligned} \mathrm{pr}^\star(m, \mathbf{1}, p') &\geq \mathrm{pr}(\tau', \mathbf{1}, p') \\ &= p_{\ell_k} + p_{\ell_{k-1}} + \cdots + p_{\ell_{k-m+1}} \\ &\geq p_k + p_{k-1} + \cdots + p_{k-m+1} \\ &= \mathrm{pr}^\star(m, \mathbf{1}, p), \end{aligned}$$

in this case as well. This completes the proof that the SS (T, p) is optimal. □

Now, let T be a rooted 4-spider on vertices $\{r, u_1, \ldots, u_n\}$ with edges given by

$$E(T) = \{(r, u_1), \ldots, (r, u_k), (u_1, u_{k+1}), (u_2, u_{k+2}), \ldots, (u_{n-k}, u_n)\}, \tag{5}$$

where $n/2 \leq k \leq n-2$. As before, the edges are labeled by the index of their heads: $e_i = (r, u_i)$ for $i \in \{1, \ldots, k\}$ and $e_i = (u_{i-k}, u_i)$ for $i \in \{k+1, \ldots, n\}$. Our second result is the following.

Theorem 6.2. *Let $M = (T, I, P)$ be a P-model, where T is a rooted 4-spider and $P = \{p_1, \ldots, p_n\}$ is a multiset of possible prizes indexed increasingly $p_1 \leq p_2 \leq \cdots \leq p_n$. Then the SS $(T, \mathbf{1}, p)$, where $p(u_i) = p_i$ for $i \in \{1, \ldots, k\}$ and $p(u_i) = p_{n+k+1-i}$ for $i \in \{k+1, \ldots, n\}$ is an optimal SS for M.*

Before we prove Theorem 6.2, we need a few lemmas that will come in handy for the proof.

Lemma 6.1. *Let T be a 4-spider presented as in (5) and $m \in \mathbb{N}$. Let p be a prize assignment on $V(T)$ such that $p_i = p(u_i) \leq p(u_j) = p_j$, where u_i is on level one and u_j is a leaf of T. If p' is the prize assignment obtained from p by swapping the prizes of u_i and u_j, then $\mathrm{pr}^\star(m, \mathbf{1}, p) \leq \mathrm{pr}^\star(m, \mathbf{1}, p')$.*

Proof. If $j = k + i$, so u_j is the unique child of u_i, then the lemma holds by (1). Hence, we can assume that u_j is not a child of u_i. Let $\tau \subseteq T$ be a max-prize rooted subtree on m edges, so $\mathrm{pr}(\tau, \mathbf{1}, p) = \mathrm{pr}^\star(m, \mathbf{1}, p)$. We now consider the following cases.

If either both u_i and u_j are vertices of τ, or neither of them are, then clearly $\mathrm{pr}^\star(m, \mathbf{1}, p) = \mathrm{pr}(\tau, \mathbf{1}, p) = \mathrm{pr}(\tau, \mathbf{1}, p') \leq \mathrm{pr}^\star(m, \mathbf{1}, p')$.

If $u_i \in V(\tau)$ and $u_j \notin V(\tau)$, then

$$\mathrm{pr}^\star(m, \mathbf{1}, p) = \mathrm{pr}(\tau, \mathbf{1}, p) \leq \mathrm{pr}(\tau, \mathbf{1}, p) - p_i + p_j = \mathrm{pr}(\tau, \mathbf{1}, p') \leq \mathrm{pr}^\star(m, \mathbf{1}, p).$$

If $u_i \notin V(\tau)$ and $u_j \in V(\tau)$, then, since u_i is on level one and u_j is a leaf of τ, we have that $\tau' = (\tau - u_j) \cup u_i$ is also a rooted subtree of T on m vertices and $\mathrm{pr}^\star(m, \mathbf{1}, p) = \mathrm{pr}(\tau, \mathbf{1}, p) = \mathrm{pr}(\tau', \mathbf{1}, p') \leq \mathrm{pr}^\star(m, \mathbf{1}, p')$, which completes our proof. □

Let $M = (T, I, P)$ be a P-model where T is a rooted 4-spider, $P = \{p_1, \ldots, p_n\}$, and p' be an arbitrary prize assignment on $V(T)$. Since every vertex of T on level two is automatically a leaf, we can, by repeated use of Lemma 6.1, obtain a prize assignment with smaller max-prize with respect to any m that has its $n-k$ largest prizes on its level-two vertices, and hence has its k smallest prizes on the level-one vertices $u_1, \ldots, u_k$ of T. By further use of the same Lemma 6.1 when considering these level-one vertices of T, we can obtain a prize assignment p that has its smallest prizes on the non-leaf vertices on level one and yet with smaller max-prize, so $\mathrm{pr}^*(m, \mathbf{1}, p) \leq \mathrm{pr}^*(m, \mathbf{1}, p')$ for any m. Note that our p satisfies

$$p(\{u_1, \ldots, u_{n-k}\}) = \{p_1, \ldots, p_{n-k}\}, \quad p(\{u_{k+1}, \ldots, u_n\}) = \{p_{k+1}, \ldots, p_n\}.$$

As the level-one vertices of T can be assumed to be ordered by their prizes, we summarize in the following.

Corollary 6.1. *From any prize assignment p' we can by repeated use of Lemma 6.1 obtain a prize assignment p on our 4-spider T, presented as in (5), such that*

$$p(u_i) = p_i \text{ for all } i \in \{1, \ldots, k\}, \text{ and } p(u_i) = p_{\pi(i)} \text{ for all } i \in \{k+1, \ldots, n\},$$

where π is a permutation of $\{k+1, \ldots, n\}$, and with $\mathrm{pr}^*(m, \mathbf{1}, p) \leq \mathrm{pr}^*(m, \mathbf{1}, p')$ for any $m \in \mathbb{N}$.

Our next lemma provides our final tool in proving Theorem 6.2.

Lemma 6.2. *Let T be a 4-spider presented as in (5) and $m \in \mathbb{N}$. Let p be a prize assignment on $V(T)$ such that for some $i, j \in \{1, \ldots, n-k\}$ with $i < j$, we have $p(u_i) \leq p(u_j)$ and $p(u_{i+k}) \geq p(u_{j+k})$. If p' is a prize assignment where the prizes on u_{i+k} and u_{j+k} have been swapped, then $\mathrm{pr}^*(m, \mathbf{1}, p) \leq \mathrm{pr}^*(m, \mathbf{1}, p')$.*

Proof. Let $\tau \subseteq T$ be a max-prize rooted subtree on m edges with respect to p, so $\mathrm{pr}(\tau, \mathbf{1}, p) = \mathrm{pr}^*(m, \mathbf{1}, p)$. We now consider the following cases.

If either both u_{i+k} and u_{j+k} are vertices of τ, or neither of them are, then clearly $\mathrm{pr}^*(m, \mathbf{1}, p) = \mathrm{pr}(\tau, \mathbf{1}, p) = \mathrm{pr}(\tau, \mathbf{1}, p') \leq \mathrm{pr}^*(m, \mathbf{1}, p')$.

If $u_{i+k} \notin V(\tau)$ and $u_{j+k} \in V(\tau)$, then

$$\begin{aligned} \mathrm{pr}^*(m, \mathbf{1}, p) &= \mathrm{pr}(\tau, \mathbf{1}, p) \\ &\leq \mathrm{pr}(\tau, \mathbf{1}, p) - p(u_{j+k}) + p(u_{i+k}) \\ &= \mathrm{pr}(\tau, \mathbf{1}, p') \\ &\leq \mathrm{pr}^*(m, 1, p'). \end{aligned}$$

If $u_{i+k} \in V(\tau)$ and $u_{j+k} \notin V(\tau)$, then we consider two (sub-)cases. If $u_j \in V(\tau)$, then since u_j is a leaf in τ, we have that $\tau' = (\tau - u_{i+k}) \cup u_{j+k}$ is also a rooted subtree of T on m vertices and $\mathrm{pr}^*(m, \mathbf{1}, p) = \mathrm{pr}(\tau, \mathbf{1}, p) = \mathrm{pr}(\tau', \mathbf{1}, p') \leq \mathrm{pr}^*(m, \mathbf{1}, p')$. If $u_j \notin V(\tau)$, then $\tau'' = (\tau - \{u_i, u_{i+k}\}) \cup \{u_j, u_{j+k}\}$ is also a rooted subtree of T on

m vertices, and

$$\begin{aligned} \mathrm{pr}^\star(m,\mathbf{1},p) &= \mathrm{pr}(\tau,\mathbf{1},p) \\ &\leq \mathrm{pr}(\tau,\mathbf{1},p) - p(u_i) - p(u_{j+k}) + p(u_j) + p(u_{i+k}) \\ &= \mathrm{pr}(\tau'',\mathbf{1},p') \\ &\leq \mathrm{pr}^\star(m,\mathbf{1},p'), \end{aligned}$$

which completes the proof. □

Proof of Theorem 6.2. Let T be a 4-spider, p a prize assignment as given in Theorem 6.2, and $m \in \mathbb{N}$. Let p' be an arbitrary prize assignment of the vertices of T. By Corollary 6.1 we can obtain a prize assignment p'' such that

$$p''(u_i) = p_i \text{ for all } i \in \{1,\ldots,k\}, \text{ and } p''(u_i) = p_{\pi(i)} \text{ for all } i \in \{k+1,\ldots,n\},$$

where π is a permutation of $\{k+1,\ldots,n\}$, and with $\mathrm{pr}^\star(m,\mathbf{1},p'') \leq \mathrm{pr}^\star(m,\mathbf{1},p')$ for any $m \in \mathbb{N}$. By Lemma 6.2 we can obtain a prize assignment p on $V(T)$ from p'' simply by ordering the prizes on the level-two leaves in a decreasing order, thereby obtaining the very prize assignment p from Theorem 6.2 that satisfies $\mathrm{pr}^\star(m,\mathbf{1},p) \leq \mathrm{pr}^\star(m,\mathbf{1},p'')$ for any $m \in \mathbb{N}$. This proves that for any $m \in \mathbb{N}$ we have $\mathrm{pr}^\star(m,\mathbf{1},p) \leq \mathrm{pr}^\star(m,\mathbf{1},p'') \leq \mathrm{pr}^\star(m,\mathbf{1},p')$, and since p' was an arbitrary prize assignment, the proof is complete. □

As a further observation, we can describe the optimal SAs on the P-model $M = (T,I,P)$, where T is a rooted 4-spider with the vertices and edges labeled as in (5), as follows.

Observation 6.1. *Let T be a 4-spider, p a prize assignment as in Theorem 6.2, and $m \in \mathbb{N}$. Then there is a max-prize rooted subtree $\tau \subseteq T$ on m edges with respect to p, so $\mathrm{pr}(\tau,\mathbf{1},p) = \mathrm{pr}^\star(m,\mathbf{1},p)$, with the following property:*

1. *If $n \leq 2k-1$, then all the leaves of τ are leaves in T, and hence in the set $\{u_{n-k+1},\ldots,u_n\}$.*
2. *If $n = 2k$, then τ has at most one leaf on level one, in which case it can assumed to be u_k.*

Proof. Suppose τ has two leaves $u_i, u_j \in \{u_1,\ldots,u_{n-k}\}$. In this case $\tau' = (\tau - u_j) \cup u_{k+i}$ is also a rooted subtree of T on m edges and has $\mathrm{pr}(\tau',\mathbf{1},p) \geq \mathrm{pr}(\tau,\mathbf{1},p)$. Hence, we can assume τ to have at most one leaf from $\{u_1,\ldots,u_{n-k}\}$.

Suppose τ has one leaf $u_i \in \{u_1,\ldots,u_{n-k}\}$. We now consider the two cases; $k > n-k$ and $k = n-k$.

FIRST CASE: $k > n-k$ or $n \leq 2k-1$. If τ has another additional leaf $u_j \in \{u_{n-k+1},\ldots,n_k\}$, then, as above, $\tau' = (\tau - u_j) \cup u_{k+i}$ has $\mathrm{pr}(\tau',\mathbf{1},p) \geq \mathrm{pr}(\tau,\mathbf{1},p)$. Otherwise, τ has no leaves from $\{u_{n-k+1},\ldots,n_k\} \neq \emptyset$. In this case $\tau'' = (\tau - u_i) \cup u_k$ is a rooted subtree of T on m edges with $\mathrm{pr}(\tau'',\mathbf{1},p) \geq \mathrm{pr}(\tau,\mathbf{1},p)$. Hence, we can assume that τ has no leaves from $\{u_1,\ldots,u_{n-k}\}$, which proves or claim in this case.

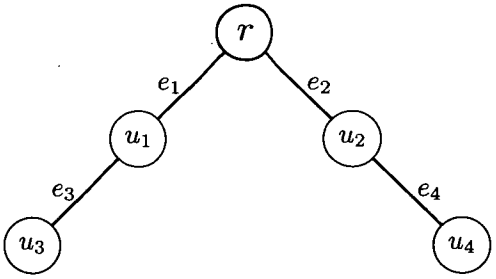
SECOND CASE: $k = n - k$ or $n = 2k$. In this case τ has the unique level-one leaf u_i. If $i < k$, then u_k has a unique child u_{2k} in τ, and so $\tau' = (\tau - u_{2k}) \cup u_{k+i}$ has the unique level-one leaf u_k and $\mathrm{pr}(\tau', \mathbf{1}, p) \geq \mathrm{pr}(\tau, \mathbf{1}, p)$. Hence, we can assume that τ has its unique level-one leaf u_k. □

Remark. Note that in the case $n \leq 2k - 1$ in the proof of Observation 6.1, all the level-one leaves of τ can be assumed to be from $\{u_{n-k+1}, \ldots, u_k\}$. If we have ℓ of them, then they can further be assumed to be $u_{k-\ell+1}, \ldots, u_k$.

7 Duality between P- and C-Models

In this section we state and use a duality between the P- and C-models, which then can be used to obtain similar results for C-models that we obtained for P-models in the previous section. In particular, we will demonstrate that if T is one of the four types of rooted trees mentioned in Theorem 5.1, then any C-model $M = (T, C, I)$ indeed has an optimal SS, as we proved was the case for the P-model. As with the P-model, we already have that any C-model $M = (T, C, I)$ (in fact, any CSM $M = (T, C, P)$), where T is a rooted path or a rooted star, does have an optimal SS.

As mentioned in Remark 5.1 right after Observation 5.3, we now explicitly examine an example of a rooted proper subtree $T_p(2)$ of $T(2)$, for which any P- or C-model $M = (T_p(2), C, P)$ has an optimal security system. For the next two examples, and just as in the convention right before Example 3.1, let $T_p(2)$ denote the rooted tree, whose underlying graph is a path, on five vertices $V(T_p(2)) = \{r, u_1, u_2, u_3, u_4\}$ and edges $E(T_p(2)) = \{(r, u_1), (r, u_2), (u_1, u_3), (u_2, u_4)\}$ rooted at its center vertex. We continue the convention of labeling the edges by the same index as their heads: $e_1 = (r, u_1)$, $e_2 = (r, u_2)$, $e_3 = (u_1, u_3)$, and $e_4 = (u_2, u_4)$, see Figure 5.



$T_p(2)$

Figure 5: The underlying graph of $T_p(2)$ is a path on five vertices.

Example 7.1.
Consider a P-model (with $c = \mathbf{1}$) on the rooted tree $T_p(2)$ where the prize values $P = \{p_1, p_2, p_3, p_4\}$ are general real positive values ordered increasingly $p_1 \leq p_2 \leq p_3 \leq p_4$. By Theorem 6.2 an optimal SS for our CSM $M = (T_p(2), I, P)$ is obtained by assigning the prizes as $p(u_1, u_2, u_3, u_4) := (p_1, p_2, p_4, p_3)$. We can explicitly obtain the max-prize subtree for each given budgets $B \in \mathbb{R}$ that yields the following:

$$\mathrm{pr}^\star(B, \mathbf{1}, p) = \begin{cases} 0 & \text{for } B < 1, \\ p_2 & \text{for } 1 \leq B < 2, \\ \max(p_1 + p_4, p_2 + p_3) & \text{for } 2 \leq B < 3, \\ p_1 + p_2 + p_4 & \text{for } 3 \leq B < 4, \\ p_1 + p_2 + p_3 + p_4 & \text{for } 4 \leq B. \end{cases}$$

Example 7.2.
Consider a C-model (with $p = \mathbf{1}$) on the rooted tree $T_p(2)$ where the penetration cost values $C = \{c_1, c_2, c_3, c_4\}$ are general real positive values ordered decreasingly $c_1 \geq c_2 \geq c_3 \geq c_4$. It is now an easy combinatorial exercise to verify directly that an optimal SS for our CSM $M = (T_p(2), C, I)$ can be obtained by assigning penetration costs as $c(u_1, u_2, u_3, u_4) := (c_1, c_2, c_4, c_3)$, in the same (index-)order as for the P-model in Example 7.1. We explicitly obtain the max-prize subtree for each given budget $B \in \mathbb{R}$ that yields the following:

$$\mathrm{pr}^\star(B, c, \mathbf{1}) = \begin{cases} 0 & \text{for } B < c_2, \\ 1 & \text{for } c_2 \leq B < \min(c_1 + c_4, c_2 + c_3), \\ 2 & \text{for } \min(c_1 + c_4, c_2 + c_3) \leq B < c_1 + c_2 + c_4, \\ 3 & \text{for } c_1 + c_2 + c_4 \leq B < c_1 + c_2 + c_3 + c_4, \\ 4 & \text{for } c_1 + c_2 + c_3 + c_4 \leq B. \end{cases}$$

Let K be a sufficiently large cost number (any real number $\geq \max(c_1, \ldots, c_4) + 1$ will do), and write each edge-cost of the form $c_i = K - c'_i$. In this way $\mathrm{pr}^\star(B, c, \mathbf{1})$ will take the following form

$$\mathrm{pr}^\star(B, c, \mathbf{1}) = \begin{cases} 0 & \text{for } B < K - c'_2, \\ 1 & \text{for } K - c'_2 \leq B < 2K - \max(c'_1 + c'_4, c'_2 + c'_3), \\ 2 & \text{for } 2K - \max(c'_1 + c'_4, c'_2 + c'_3) \leq B < 3K - (c'_1 + c'_2 + c'_4), \\ 3 & \text{for } 3K - (c'_1 + c'_2 + c'_4) \leq B < 4K - (c'_1 + c'_2 + c'_3 + c'_4), \\ 4 & \text{for } 4K - (c'_1 + c'_2 + c'_3 + c'_4) \leq B. \end{cases}$$

From the above we see the evident resemblance to the expression for $\mathrm{pr}^\star(B, \mathbf{1}, p)$ of the P-model in Example 7.1. This is a glimpse of a duality between the P-models and the C-models that we will now describe.

CONVENTION: In what follows, it will be convenient to view the cost and prize assignments c and p not as functions as in Definition 3.2, but rather as vectors $\tilde{c} = (c_1, \ldots, c_n)$ and $\tilde{p} = (p_1, \ldots, p_n)$ in the n-dimensional Euclidean space $\mathbb{R}^n$, which can be obtained by a fixed labeling of the n non-root vertices $u_1, \ldots, u_n$ and a corresponding labeling of the edges $e_1, \ldots, e_n$, with our usual convention that for each i the vertex u_i is the head of e_i, and by letting $c_i := c(e_i)$ and $p_i := p(u_i)$.

For a given $n \in \mathbb{N}$, let $\mathcal{B}(\mathbb{R}^n)$ denote the group of all bijections $\mathbb{R}^n \to \mathbb{R}^n$ with respect to compositions of maps. For $a \in \mathbb{Q}_+$ and $b \in \mathbb{Q}$ the affine map $\alpha : \mathbb{R}^n \to \mathbb{R}^n$ given by $\alpha(\tilde{x}) = a\tilde{x} + b\tilde{1}$, where $\tilde{1} = (1, \ldots, 1) \in \mathbb{R}^n$, is bijective with an inverse $\alpha^{-1}(\tilde{x}) = \frac{1}{a}\tilde{x} - \frac{b}{a}\tilde{1}$ of the same type. Further, if $\alpha'(\tilde{x}) = a'\tilde{x} + b'\tilde{1}$ is another such map, then the composition $(\alpha' \circ \alpha)(\tilde{x}) = a'a\tilde{x} + (a'b + b')\tilde{1}$ is also a bijection of this very type. Since the identity map of $\mathbb{R}^n$ has $a = 1 \in \mathbb{Q}_+$ and $b = 0 \in \mathbb{Q}$, we have the following.

Observation 7.1. *If $n \in \mathbb{N}$ then $G_n = \{\alpha \in \mathcal{B}(\mathbb{R}^n) : \alpha(\tilde{x}) = a\tilde{x} + b\tilde{1}$, for some $a \in \mathbb{Q}_+$ and $b \in \mathbb{Q}\}$ is a subgroup of $\mathcal{B}(\mathbb{R}^n)$.*

By letting G_n act on the set $\mathbb{R}^n$ in the natural way, $(\alpha, \tilde{x}) \mapsto \alpha(\tilde{x})$, then the group orbits $G_n(\tilde{x}) = \{\alpha(\tilde{x}) : \alpha \in G_n\}$ yield a partition of $\mathbb{R}^n$ into corresponding equivalence classes $\mathbb{R}^n = \bigcup_{\tilde{x} \in \mathbb{R}^n} G_n(\tilde{x})$. By intersecting with $\mathbb{Q}_+^n$ we obtain the following equivalence classes that we seek.

Definition 7.1. *For each $\tilde{x} \in \mathbb{Q}_+^n$ let $[\tilde{x}]$ denote the equivalence class of $\tilde{x}$ with respect to the partition of $\mathbb{R}^n$ into the G_n orbits: $[\tilde{x}] = G_n(\tilde{x}) \cap \mathbb{Q}_+^n$.*

We now justify the above equivalence of vectors of $\mathbb{Q}_+^n$. The following observation is obtained directly from Definition 3.2.

Observation 7.2. *Let T be a rooted tree on n labeled non-root vertices and edges, τ a rooted subtree of T, and $\alpha \in G_n$ given by $\alpha(\tilde{x}) = a\tilde{x} + b\tilde{1}$. If $\tilde{c}, \tilde{p} \in \mathbb{Q}_+^n$ are a cost and prize vector, respectively, then we have*

$$\begin{aligned} \mathrm{pr}(\tau, \tilde{c}, \alpha(\tilde{p})) &= a\mathrm{pr}(\tau, \tilde{c}, \tilde{p}) + |E(\tau)|b, \\ \mathrm{cst}(\tau, \alpha(\tilde{c}), \tilde{p}) &= a\mathrm{cst}(\tau, \tilde{c}, \tilde{p}) + |E(\tau)|b. \end{aligned}$$

If $J \subseteq \{1, \ldots, n\}$ and $\Sigma_J : \mathbb{R}^n \to \mathbb{R}$ is given by $\tilde{x} \mapsto \sum_{i \in J} x_i$, then we clearly have

$$\Sigma_J(\alpha(\tilde{x})) \leq \Sigma_J(\alpha(\tilde{y})) \Leftrightarrow \Sigma_J(\tilde{x}) \leq \Sigma_J(\tilde{y}), \tag{6}$$

and hence the following corollary.

Corollary 7.1. *Let T be a rooted tree on n labeled non-root vertices and edges, $B \in \mathbb{Q}_+$ a budget, and $\alpha \in G_n$ given by $\alpha(\tilde{x}) = a\tilde{x} + b\tilde{1}$.*

(i) If $\tilde{p} \in \mathbb{Q}_+^n$ is a prize vector, then we have

$$\mathrm{pr}^\star(B, \tilde{1}, \alpha(\tilde{p})) = a\mathrm{pr}^\star(B, \tilde{1}, \tilde{p}) + b\lfloor B \rfloor. \tag{7}$$

Further, both max prizes in (7) are attained at the same rooted subtree τ of T where $|E(\tau)| = \lfloor B \rfloor$.

(ii) If $\tilde{c} \in \mathbb{Q}_+^n$ is a cost vector, then we have

$$\mathrm{pr}^\star(aB + bm, \alpha(\tilde{c}), \tilde{1}) = m \Leftrightarrow \mathrm{pr}^\star(B, \tilde{c}, \tilde{1}) = m,$$

and further, both max prizes are attained at the same rooted subtree τ of T within the budget; that is, $|E(\tau)| = m$ and $\mathrm{cst}(\tau, \tilde{c}, \tilde{1}) \leq B$.

Remark. (i) That both max prizes are attained at the same rooted subtree τ in (i) in Corollary 7.1 simply means that

$$\mathrm{pr}(\tau, \tilde{1}, \alpha(\tilde{p})) = \mathrm{pr}^*(B, \tilde{1}, \alpha(\tilde{p})) \Leftrightarrow \mathrm{pr}(\tau, \tilde{1}, \tilde{p}) = \mathrm{pr}^*(B, \tilde{1}, \tilde{p}),$$

which is a direct consequence of Observation 7.2 and (7). (ii) Also, for a rooted subtree τ with $|E(\tau)| = m$ and $\mathrm{cst}(\tau, \tilde{c}, \tilde{1}) \leq B$, then by Observation 7.2 we also have $\mathrm{cst}(\tau, \alpha(\tilde{c}), \tilde{1}) \leq aB + bm$, and

$$\mathrm{pr}(\tau, \tilde{c}, \tilde{1}) = m = \mathrm{pr}^*(B, \tilde{c}, \tilde{1}) \Leftrightarrow \mathrm{pr}(\tau, \alpha(\tilde{c}), \tilde{1}) = m = \mathrm{pr}^*(aB + bm, \alpha(\tilde{c}), \tilde{1}).$$

We can, in fact, say a tad more than Corollary 7.1 for C-models $M = (T, C, I)$.

Definition 7.2. *Let $M = (T, C, I)$ be a C-model. For a given cost vector $\tilde{c} \in \mathbb{Q}_+^n$ let $B_m(\tilde{c})$ denote the smallest cost $B \in \mathbb{Q}_+$ with* $\mathrm{pr}^*(B, \tilde{c}, \tilde{1}) = m$.

Note that

$$\mathrm{pr}^*(B, \tilde{c}, \tilde{1}) = m \Leftrightarrow B_m(\tilde{c}) \leq B < B_{m+1}(\tilde{c}).$$

We also have the following useful lemma.

Lemma 7.1. *If $\alpha \in G_n$ is given by $\alpha(\tilde{x}) = a\tilde{x} + b\tilde{1}$, then $B_m(\alpha(\tilde{c})) = aB_m(\tilde{c}) + bm$.*

Proof. By definition of $B_m(\tilde{c})$ we have $\mathrm{pr}^*(B_m(\tilde{c}), \tilde{c}, \tilde{1}) = m$, and hence by Corollary 7.1 $\mathrm{pr}^*(aB_m(\tilde{c}) + bm, \alpha(\tilde{c}), \tilde{1}) = m$ as well. Suppose that $\mathrm{pr}^*(B', \alpha(\tilde{c}), \tilde{1}) = m$, where $B' < aB_m(\tilde{c}) + bm$. If now $B' = aB'' + bm$, then $B'' < B_m(\tilde{c})$ and we have again by Corollary 7.1 that $\mathrm{pr}^*(B'', \tilde{c}, \tilde{1}) = m$. This contradicts the definition of $B_m(\tilde{c})$. Hence, $B_m(\alpha(\tilde{c}) = aB_m(\tilde{c}) + bm$. □

Proposition 7.1. *For $m \in \{0, 1, \ldots, n\}$ and a cost vectors $\tilde{c}$ and $\tilde{c}'$ we have $B_m(\tilde{c}) \geq B_m(\tilde{c}')$ if and only if for every budget B with* $\mathrm{pr}^*(B, \tilde{c}, \tilde{1}) = m$ *we have* $\mathrm{pr}^*(B, \tilde{c}, \tilde{1}) \leq \mathrm{pr}^*(B, \tilde{c}', \tilde{1})$.

Proof. Suppose $B_m(\tilde{c}) \geq B_m(\tilde{c}')$, and let B be a budget with $\mathrm{pr}^*(B, \tilde{c}, \tilde{1}) = m$. By definition we then have $B \geq B_m(\tilde{c})$ and hence $B \geq B_m(\tilde{c}')$ and therefore $\mathrm{pr}^*(B, \tilde{c}', \tilde{1}) \geq m = \mathrm{pr}^*(B, \tilde{c}, \tilde{1})$.

Conversely, if for every budget B with $\mathrm{pr}^*(B, \tilde{c}, \tilde{1}) = m$ we have $\mathrm{pr}^*(B, \tilde{c}, \tilde{1}) \leq \mathrm{pr}^*(B, \tilde{c}', \tilde{1})$, then, in particular for $B = B_m(\tilde{c})$ we have $m = \mathrm{pr}^*(B_m(\tilde{c}), \tilde{c}, \tilde{1}) \leq \mathrm{pr}^*(B_m(\tilde{c}), \tilde{c}', \tilde{1})$, and hence, by definition, $B_m(\tilde{c}') \leq B_m(\tilde{c})$. □

CONVENTION: For a vector $\tilde{x} = (x_1, \ldots, x_n) \in \mathbb{Q}_+^n$ let $\{\tilde{x}\}$ denote its underlying multiset. So if $(T, \tilde{c}, \tilde{p})$ is an SS for a CSM $M = (T, C, P)$, then we necessarily have $C = \{\tilde{c}\}$ and $P = \{\tilde{p}\}$ as multisets. Also, we have $\{\tilde{1}\} = I$ as the multiset containing n copies of 1.

Suppose $\mathrm{pr}^*(B, \tilde{1}, \tilde{p}) \leq \mathrm{pr}^*(B, \tilde{1}, \tilde{p}')$ for all $\tilde{p}'$ with $\{\tilde{p}'\} = \{\tilde{p}\}$. Then by Corollary 7.1 we get for any $\alpha \in G_n$ with $\alpha(\tilde{x}) = a\tilde{x} + b\tilde{1}$, that

$$\mathrm{pr}^*(B, \tilde{1}, \alpha(\tilde{p})) = a\mathrm{pr}^*(B, \tilde{1}, \tilde{p}) + b\lfloor B \rfloor \leq a\mathrm{pr}^*(B, \tilde{1}, \tilde{p}') + b\lfloor B \rfloor = \mathrm{pr}^*(B, \tilde{1}, \alpha(\tilde{p}')),$$

and so we have the following.

Proposition 7.2. *The SS $(T, \tilde{1}, \tilde{p})$ is optimal for the P-model $M = (T, I, \{\tilde{p}\})$ with respect to the budget $B \in \mathbb{Q}_+$ if and only if the SS $(T, \tilde{1}, \alpha(\tilde{p}))$ is optimal for the P-model $M = (T, I, \{\alpha(\tilde{p})\})$ with respect to B.*

In a similar way, we have by Proposition 7.1 that $\mathrm{pr}^\star(B, \tilde{c}, \tilde{1}) = m \leq \mathrm{pr}^\star(B, \tilde{c}', \tilde{1})$ whenever $B_m(\tilde{c}) \leq B < B_{m+1}(\tilde{c})$ and $\{\tilde{c}'\} = \{\tilde{c}\}$ if and only if $B_m(\tilde{c}) \geq B_m(\tilde{c}')$, which by Lemma 7.1 holds if and only if

$$B_m(\alpha(\tilde{c})) = aB_m(\tilde{c}) + bm \geq aB_m(\tilde{c}') + bm = B_m(\alpha(\tilde{c}')).$$

In other words, $\mathrm{pr}^\star(B, \tilde{c}, \tilde{1}) \leq \mathrm{pr}^\star(B, \tilde{c}', \tilde{1})$ when $B_m(\tilde{c}) \leq B < B_{m+1}(\tilde{c})$ holds if and only if $\mathrm{pr}^\star(B', \alpha(\tilde{c}), \tilde{1}) \leq \mathrm{pr}^\star(B', \alpha(\tilde{c}'), \tilde{1})$ when $B_m(\alpha(\tilde{c})) \leq B' < B_{m+1}(\alpha(\tilde{c}))$. Since this holds for every $\alpha \in G_n$, which is a group with each element having an inverse, then we have the following.

Proposition 7.3. *The SS $(T, \tilde{c}, \tilde{1})$ is optimal for the C-model $M = (T, \{\tilde{c}\}, I)$ with respect to $B \in [B_m(\tilde{c}), B_{m+1}(\tilde{c}))\cap\mathbb{Q}_+$ if and only if the SS $(T, \alpha(\tilde{c}), \tilde{1})$ is optimal for the C-model $M' = (T, \{\alpha(\tilde{c})\}, I)$ with respect to $B' \in [B_m(\alpha(\tilde{c})), B_{m+1}(\alpha(\tilde{c}))\cap\mathbb{Q}_+$.*

Combining Propositions 7.2 and 7.3, we have the following summarizing corollary.

Corollary 7.2. *Let $\alpha \in G_n$.*

The SS $(T, \tilde{1}, \tilde{p})$ is optimal for the P-model $M = (T, I, \{\tilde{p}\})$ if and only if the SS $(T, \tilde{1}, \alpha(\tilde{p}))$ is optimal for the P-model $M' = (T, I, \{\alpha(\tilde{p})\})$.

The SS $(T, \tilde{c}, \tilde{1})$ is optimal for the C-model $M = (T, \{\tilde{c}\}, I)$ if and only if the SS $(T, \alpha(\tilde{c}), \tilde{1})$ is optimal for the C-model $M' = (T, \{\alpha(\tilde{p})\}, I)$.

Corollary 7.2 shows that optimality of security systems of both C- and P-models is G_n-invariant when applied to the prize and cost vector, respectively.

Recall the equivalence class $[\tilde{x}] = G_n(\tilde{x}) \cap \mathbb{Q}_+^n$ from Definition 7.1. We can now define induced equivalence classes of SS of both C- and P-models. By Corollary 7.2 the following definition is valid (that is, the terms are all well defined).

Definition 7.3. *For a C-model $M = (T, C, I)$ and a SS $(T, \tilde{c}, \tilde{1})$ of M, we let*

$$[(T, \tilde{c}, \tilde{1})] := \{(T, \tilde{x}, \tilde{1}) : \tilde{x} \in [\tilde{c}]\}.$$

We say that $[(T, \tilde{c}, \tilde{1})]$ is optimal *if one $(T, \tilde{x}, \tilde{1}) \in [(T, \tilde{c}, \tilde{1})]$ is optimal for its corresponding $M = (T, \{\tilde{x}\}, I)$, since then each element in $[(T, \tilde{c}, \tilde{1})]$ is also optimal.*

Likewise, for a P-model $M = (T, I, P)$ and a SS $(T, \tilde{1}, \tilde{p})$ of M, we let

$$[(T, \tilde{1}, \tilde{p})] := \{(T, \tilde{1}, \tilde{y}) : \tilde{y} \in [\tilde{p}]\}.$$

We say that $[(T, \tilde{1}, \tilde{p})]$ is optimal *if one $(T, \tilde{1}, \tilde{y}) \in [(T, \tilde{1}, \tilde{p})]$ is optimal for its corresponding $M = (T, I, \{\tilde{y}\})$, since then each element in $[(T, \tilde{1}, \tilde{p})]$ is also optimal.*

With the setup just presented we now can define the dual of both vector classes and SS classes for C- and P-models in the following.

Definition 7.4. *For a vector* $\tilde{x}$ *and* $[\tilde{x}] = G_n(\tilde{x}) \cap \mathbb{Q}_+^n$ *as in Definition 7.1, then* $[\tilde{x}]^* := [-\tilde{x}]$ *is the* dual vector class *of* $[\tilde{x}]$.

For a C-model $M = (T, C, I)$ *and a SS* $(T, \tilde{c}, \tilde{1})$ *of* M, *then* $[(T, \tilde{c}, \tilde{1})]^* := [(T, \tilde{1}, -\tilde{c})]$ *is the corresponding* dual P-model security system class (dual P-model SS class) *of the C-model class* $[(T, \tilde{c}, \tilde{1})]$.

Likewise, for a P-model $M = (T, I, P)$ *and a SS* $(T, \tilde{1}, \tilde{p})$ *of* M, *then the class* $[(T, \tilde{1}, \tilde{p})]^* := [(T, -\tilde{p}, \tilde{1})]$ *is the corresponding* dual C-model security system class (dual C-model SS class) *of the P-model class* $[(T, \tilde{1}, \tilde{p})]$.

Note that the double-dual yields the original class in each case: $[\tilde{x}]^{**} = [-\tilde{x}]^* = [\tilde{x}]$, and

$$[(T, \tilde{c}, \tilde{1})]^{**} = [(T, \tilde{1}, -\tilde{c})]^* = [(T, \tilde{c}, \tilde{1})], \quad [(T, \tilde{1}, \tilde{p})]^{**} = [(T, -\tilde{p}, \tilde{1})]^* = [(T, \tilde{1}, \tilde{p})].$$

For a P-model $M = (T, I, P)$ and a SS P-model class $[(T, \tilde{1}, \tilde{p})]$ we can always assume the prize vector $\tilde{p}$ is such $p_i \in [0, 1] \cap \mathbb{Q}_+$ for each i, since $\alpha(\tilde{x}) = a\tilde{x}$ is indeed an element of G_n for any $a > 0$. In this way $\tilde{c} = \tilde{1} - \tilde{p} \in ([0, 1] \cap \mathbb{Q}_+)^n$ is a legitimate cost vector, and we have $[\tilde{p}]^* = [\tilde{1} - \tilde{p}]$ and $[(T, \tilde{1}, \tilde{p})]^* = [(T, \tilde{1} - \tilde{p}, \tilde{1})]$. In what follows, we will call such a prize vector *scaled.* The following is easy to show.

Claim 7.1. *For a scaled prize vector* $\tilde{p}$ *with* $p_i \in [0, 1] \cap \mathbb{Q}_+$ *for each* i, *and a rooted subtree* τ *of* T *with* $|E(\tau)| = m$, *then* $\mathrm{pr}(\tau, \tilde{1}, \tilde{p}) + \mathrm{cst}(\tau, \tilde{1} - \tilde{p}, \tilde{1}) = m$.

Let $\tilde{p}$ be a scaled prize vector and assume B is a budget with $\mathrm{pr}^\star(B, \tilde{1} - \tilde{p}, \tilde{1}) = m$. Then there is a rooted subtree τ of T on m edges such that $\mathrm{cst}(\tau, \tilde{1} - \tilde{p}, \tilde{1}) \leq B$, and hence there is such a τ of smallest cost. Hence, we may assume τ is indeed such a rooted subtree of smallest cost. By Claim 7.1 applied to $\tilde{1} - \tilde{p}$, which is also scaled, we then have $\mathrm{pr}(\tau, \tilde{1}, \tilde{p}) = m - \mathrm{cst}(\tau, \tilde{1} - \tilde{p}, \tilde{1})$ with the smallest $\mathrm{cst}(\tau, \tilde{1} - \tilde{p}, \tilde{1})$ among rooted subtrees τ on m edges, and hence $\mathrm{pr}(\tau, \tilde{1}, \tilde{p})$ is maximum among all rooted subtrees τ on m edges, and so $\mathrm{pr}(\tau, \tilde{1}, \tilde{p}) = \mathrm{pr}^\star(m, \tilde{1}, \tilde{p})$. Hence,

$$B \geq \mathrm{cst}(\tau, \tilde{1} - \tilde{p}, \tilde{1}) = m - \mathrm{pr}(\tau, \tilde{1}, \tilde{p}) = m - \mathrm{pr}^\star(m, \tilde{1}, \tilde{p}).$$

Since $\mathrm{cst}(\tau, \tilde{1} - \tilde{p}, \tilde{1})$ is the smallest cost among all rooted subtrees on m edges, then

$$B' = \mathrm{cst}(\tau, \tilde{1} - \tilde{p}, \tilde{1}) = m - \mathrm{pr}^\star(m, \tilde{1}, \tilde{p})$$

is indeed the smallest cost with $\mathrm{pr}^\star(B', \tilde{1} - \tilde{p}, \tilde{1}) = m$. By Definition 7.2 we then have the following.

Lemma 7.2. *For* $m \in \{0, 1, \ldots, n\}$ *and a scaled (prize) vector* $\tilde{p}$, *we have*

$$B_m(\tilde{1} - \tilde{p}) = m - \mathrm{pr}^\star(m, \tilde{1}, \tilde{p}).$$

As a direct consequence of Lemma 7.2, we then have

Corollary 7.3. *For any* $m \in \{0, 1, \ldots, n\}$ *and scaled vectors* $\tilde{p}$ *and* $\tilde{p}'$, *we have*

$$B_m(\tilde{1} - \tilde{p}) \geq B_m(\tilde{1} - \tilde{p}') \Leftrightarrow \mathrm{pr}^\star(m, \tilde{1}, \tilde{p}) \leq \mathrm{pr}^\star(m, \tilde{1}, \tilde{p}').$$

We can now prove one of the main results in this section.

Theorem 7.1. *Let $M = (T, I, P)$ be a P-model, $(T, \tilde{1}, \tilde{p})$ a SS for M where $\tilde{p}$ is scaled, and $m \in \{0, 1, \ldots, n\}$. Then $\mathrm{pr}^\star(m, \tilde{1}, \tilde{p}) \leq \mathrm{pr}^\star(m, \tilde{1}, \tilde{p}')$ for any $\tilde{p}'$ with $\{\tilde{p}'\} = P$ if and only if $\mathrm{pr}^\star(B, \tilde{1} - \tilde{p}, \tilde{1}) \leq \mathrm{pr}^\star(B, \tilde{1} - \tilde{p}', \tilde{1})$ for any budget B with $\mathrm{pr}^\star(B, \tilde{1} - \tilde{p}, \tilde{1}) = m$ and for any $\tilde{p}'$ with $\{\tilde{p}'\} = P$.*

Proof. By Corollary 7.3 we have that $\mathrm{pr}^\star(m, \tilde{1}, \tilde{p}) \leq \mathrm{pr}^\star(m, \tilde{1}, \tilde{p}')$ for any $\tilde{p}'$ with $\{\tilde{p}'\} = P$ if and only if $B_m(\tilde{1} - \tilde{p}) \geq B_m(\tilde{1} - \tilde{p}')$ for any $\tilde{p}'$ with $\{\tilde{p}'\} = P$ which, by Proposition 7.1, holds if and only if $\mathrm{pr}^\star(B, \tilde{1} - \tilde{p}, \tilde{1}) \leq \mathrm{pr}^\star(B, \tilde{1} - \tilde{p}', \tilde{1})$ for all budgets B with $\mathrm{pr}^\star(B, \tilde{1} - \tilde{p}, \tilde{1}) = m$ and for all $\tilde{p}'$ with $\{\tilde{p}'\} = P$. □

Note that by Theorem 7.1 we have that $\mathrm{pr}^\star(B, \tilde{1}, \tilde{p}) \leq \mathrm{pr}^\star(B, \tilde{1}, \tilde{p}')$ for any budget B and any $\tilde{p}'$ with $\{\tilde{p}'\} = \{\tilde{p}\}$, if and only if $\mathrm{pr}^\star(B, \tilde{1} - \tilde{p}, \tilde{1}) \leq \mathrm{pr}^\star(B, \tilde{1} - \tilde{p}', \tilde{1})$ for any budget B and any $\tilde{p}'$ with $\{\tilde{p}'\} = \{\tilde{p}\}$. Hence, by Corollary 7.2 and Theorem 7.1 we therefore have the main conclusion of this section in light of Definition 7.3.

Corollary 7.4. *For a rooted tree T and a prize vector $\tilde{p} \in \mathbb{Q}^n_+$, then $[(T, \tilde{1}, \tilde{p})]$ is an optimal P-model SS class if and only if the dual C-model SS class $[(T, \tilde{1}, \tilde{p})]^* = [(T, -\tilde{p}, \tilde{1})]$ is optimal.*

In particular, if $\tilde{p}$ is scaled, then the SS $(T, \tilde{1}, \tilde{p})$ is optimal for the P-model $M = (T, I, \{\tilde{p}\})$ if and only if the SS $(T, \tilde{1} - \tilde{p}, \tilde{1})$ is optimal for the C-model $M = (T, \{\tilde{1} - \tilde{p}\}, I)$.

Consequently, by Corollary 4.1, Theorems 4.2, 5.1, 6.1 and 6.2 and Corollary 7.4, we have the following summarizing result.

Theorem 7.2. *For a rooted tree T on n non-root vertices the following are equivalent:*

1. *Any P-model $M = (T, I, P)$ has an optimal SS.*
2. *Any C-model $M = (T, C, I)$ has an optimal SS.*
3. *T is one of the following types: (i) a rooted path, (ii) a rooted star, (iii) a rooted 3-caterpillar, or (iv) a rooted 4-spider.*

Note that by (6) we have, in particular, that each $\alpha \in G_n$ preserves the order of the entries of each $\tilde{x} \in \mathbb{Q}^n_+$, so each $\tilde{x} \in [\tilde{p}]$ has the same order of its entries as $\tilde{p}$ does. But clearly, the dual operation on $[\tilde{x}]^* = [-\tilde{x}]$ is order reversing, that is, we have that $x_i \leq x_j$ for any $\tilde{x} \in [\tilde{p}]$ if and only if $y_i \geq y_j$ for any $\tilde{y} \in [-\tilde{p}] = [\tilde{p}]^*$. Since the optimal assignments of prizes from a given multiset P are given in Theorems 6.1 and 6.2, we then have by Corollary 7.4 the following theorems for C-models as well.

Theorem 7.3. *Let $M = (T, C, I)$ be a C-model where T is a rooted 3-caterpillar as in (4) and $C = \{c_1, \ldots, c_n\}$ is a multiset of possible edge-costs indexed decreasingly $c_1 \geq c_2 \geq \cdots \geq c_n$. Then the SS $(T, c, \mathbf{1})$, where $c(e_i) = c_i$ for each $i \in \{1, \ldots, n\}$ is an optimal SS for M.*

Theorem 7.4. *Let $M = (T, C, I)$ be a P-model, where T is a rooted 4-spider as in (5) and $C = \{c_1, \ldots, c_n\}$ is a multiset of possible edge-costs indexed decreasingly $c_1 \geq c_2 \geq \cdots \geq c_n$. Then the SS $(T, c, \mathbf{1})$, where $c(e_i) = c_i$ for $i \in \{1, \ldots, k\}$ and $c(e_i) = c_{n+k+1-i}$ for $i \in \{k+1, \ldots, n\}$ is an optimal SS for M.*

8 Summary and Conclusions

This paper defined a cyber-security model to explore defensive security systems. The results obtained mathematically support the intuition that it is best to place stronger defenses in the outer layers and more-valuable prizes in the deeper layers. We defined three types of SSs: improved, good, and optimal. We showed that it is not always possible to find an optimal SS for a given CSM, but demonstrated for rooted paths and stars that optimal SSs do exist. The results mathematically show that a path produces the best cyber-security, however, burying something n levels deep for large n may prevent the friendly side from accessing the "information" effectively. The results show, in general, that trees having greater depth provide more security in this setting.

We showed that any CSM is equivalent to a CSM where either all the edge penetration costs are unit priced (a P-model) or where all the vertices have a unit prize (C-model), by allowing larger underlying rooted trees. We then characterised for which trees a P-model has an optimal SSs, and we also did that for the C-models. We noted that the P- and C-models have optimal SSs for exactly the same types of rooted trees. This was then explained by obtaining a duality between the P- and C-models in the penultimate section of the paper.

We gave an $O(n \log n)$ algorithm for producing a good SS that was based on sorting. It is not clear how strong such a good SS is, as there may be many such good SSs, and some may be better than others. It would be interesting to come up with a comparison metric to rank various good SSs. We must continue to explore models of cyber-security systems to develop the foundations needed to combat the ongoing and increasing number of cyber attacks. This work is but one step in that direction.

We conclude the paper with a number of questions.

1. Can we find an efficient algorithm to develop optimal SSs in the cases where all penetration costs or all targets are from a finite set of possible values? Say, if we have two possible penetrations costs or three? Similarly for prizes?

2. In a two-player version of the model, what would be the best strategy for a defender who is allowed to reposition a prize or a portion of a prize after each move by an attacker? And, what would the complexity of this problem be?

3. Are there on-line variants of the model that are interesting to study? For example, a version where the topology of the tree changes dynamically or where only a partial description is known to the attacker.

4. Could a dynamic programming approach be used to obtain a SS that is somehow quantifiably better than a good SS or allow us to pick the "best" good SS?

5. Is there a more useful definition of neighboring configuration that could lead to an efficient algorithm for producing better SSs, for example, perhaps a definition where sibling vertices or edges can have their prizes or penetration costs swapped, respectively?

Acknowledgments

This work was supported by the Office of Naval Research. The work was also supported by Thailand Research Fund grant No. RSA5480006. – Finally, sincere thanks to the two anonymous referees for all their helpful comments and for spotting some well-hidden (and embarrassing!) typos. This greatly improved the presentation of the paper.

References

[1] El Houssaine Aghezzaf, Thomas L. Magnanti, and Laurence A. Wolsey. Optimizing Constrained Subtrees of Trees. *Mathematical Programming*, **71(2)**:113–126, Series A, (1995).

[2] Geir Agnarsson and Raymond Greenlaw. Graph Theory, Modeling, Applications, and Algorithms. *Prentice Hall*, (2007).

[3] Geir Agnarsson, Raymond Greenlaw, and Sanpawat Kantabutra. On Cyber Attacks and the Maximum-Weight Rooted-Subtree Problem *Acta Cybernetica*, **22**:591–612, (2016).

[4] Sofie Coene, Carlo Filippi, Frits Spieksma, and Elisa Stevanato. Balancing Profits and Costs on Trees. *Networks*, **61(3)**:200–11, (2013).

[5] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman and Company, New York, (1979).

[6] Go-Gulf. *Cyber Crime: Statistics and Trends*, `www.go-gulf.com/blog/cyber-crime`, retrieved March 21, (2015).

[7] Raymond Greenlaw, H. James Hoover, and Walter Larry Ruzzo. *Limits to Parallel Computation: P-Completeness Theory, Oxford University Press*, (1995).

[8] Sun-Yuan Hsieh and Ting-Yu Chou. Finding a Weight-constrained Maximum-density Subtree in a Tree. *Algorithms and Computation, Lecture Notes in Computer Science*, **3827**:944–953, Springer, Berlin, (2005).

[9] Robert Johnston and Clint LaFever. `Hacker.mil`, Marine Corps Red Team (PowerPoint Presentation). (2012).

[10] Hoong Chuin Lau, Trung Hieu Ngo, and Bao Nguyen Nguyen. Finding a Length-constrained Maximum-sum or Maximum-density Subtree and Its Application to Logistics. *Discrete Optimization*, **3(4)**:385–391, (2006).

[11] Fred B. Schneider. Blueprint for a Science of Cybersecurity, *The Next Wave*, **19(2)**:47–57, (2012).

[12] Hsin-Hao Su, Chin Lung Lu, and Chuan Yi Tang. An Improved Algorithm for Finding a Length-constrained Maximum-density Subtree in a Tree. *Information Processing Letters*, **109(2)**:161–164, (2008).

[13] Nelson A. Uhan. Stochastic linear programming games with concave preferences. *European Journal of Operations Research*, **243(2)**:637–646, (2015).

[14] R. Kevin Wood. Deterministic network interdiction. *Mathematical and Computer Modelling*, **17(2)**:1–18, (1993).

Received 10th June 2016

Acta Cybernetica **22** (2016) 771–789.

Adapting Dynamic Time Warping to the Speech of the Hearing Impaired*

László Czap† and Attila K. Varga†

Abstract

One service provided by our application 'Speech Assistant System' assisting the teaching of the hearing impaired to speak is the automatic assessment of words and sentences in the course of practice and feedback to the person. Individual speech sounds can only be correctly evaluated if they are compared with the appropriate reference speech sounds. This requires segmenting the speech to be examined. The methods currently known do not give sufficiently correct results for the speech of the hearing impaired, which is often so distorted and halting so that it prevents understanding. The paper presents a reference generation method suitable for segmenting distorted speech, a modification of dynamic time warping and its comparison with traditional methods. The procedure presented has been successfully used for the automatic assessment of the pronunciation of the hearing impaired.

Keywords: dynamic time warping, speech quality assessment, acoustic-phonetic features, distorted speech, speech of hard of hearing children

1 Introduction

The project 'Basic and Applied Research for the Internet-Based Speech Development of the Hearing Impaired and for the Objective Measurement of Progress' served the purpose of creating a new aid for the deaf and the hard of hearing in learning to speak, called the Speech Assistant System. The foundation of the research is represented by the 'talking head' developed at the University of Miskolc and the audio-visual transcoder developed at the University of Debrecen. The objective of the project is to create a complex system which provides the audio-visual representation of the speech process, by the visual representation of the sound images of speech on the one hand, and of the articulation on the other, set in a

*The research has been carried out within the framework of Mechatronics and Logistics Centre of Excellence operating as one of the strategic research areas of the University of Miskolc and as part of the TMOP-4.2.2.C-11/1/KONV-2012-0002 project funded by the European Union, co-financed by the European Social Fund.

†University of Miskolc, E-mail: {czap,varga.attila}@uni-miskolc.hu

DOI: 10.14232/actacyb.22.4.2016.3

training framework system. The 3D head model with its transparent face can visualise the tongue motion better than a natural speaker. In addition, the system includes a number of functions (visualisation of prosody, automatic assessment and implementation of the knowledge-based system) that facilitate individual practice not only on the computer, but also on a mobile device. The module of the technology developed performing the audio-visual transcoding is language independent, so the talking head and the automatic assessment can be adapted to other languages besides Hungarian.

Automatic assessment provides feedback to the hearing impaired, who can use the Speech Assistant System on their own. The assessment of the speech produced during practice shows not only progress achieved in utterance, but also serves as input to the knowledge-based system, which assists in designating the next word to be practiced based on teacher experience [1].

The methods developed for automatic speech quality assessment include speech segmentation in explicit or implicit form. Speech tempo changes from speaker to speaker, from articulation to articulation. These non-linear extensions and shortenings do not necessarily count as faulty pronunciation. The hearing impaired usually speak more slowly than the average speech tempo. For the assessment of the pronunciation of the individual speech sounds, the time segments of the reference pattern and of the actual pronunciation have to be matched. The reference and the actual waveforms can be made to have identical lengths by linear stretching and/or linear shrinking. This, however, does not ensure a time parallel of the individual speech sounds, for the pronunciation rhythm may differ from the reference. If certain speech sounds are pronounced longer and others are pronounced shorter, in linear time warping it will not be the speech sound segments that matched with which it should be similar, therefore the comparison will produce false results. The articulation of certain speech sounds differing in time from their ordinary articulation is particularly characteristic of the speech of the hearing impaired. Therefore, for the purpose of comparing the reference and the speech being examined, non-linear time-warping that is needed, procedures and algorithms developed in computer-based speech processing are available for this purpose. These methods work well for high-quality speech and pronunciation acceptable in everyday communication. However, they produce poor results for distorted speech sounds and unusually drawling and halting speech. The paper discusses our segmentation method suitable for low-quality speech that pairs the test and artificially generated reference shape.

2 Non-linear time warping

We can speak of an ideal time comparison if two samples are aligned along the individual speech sounds. This generally accepted method is used in computer-based speech recognition [15].

The hidden Markov model (HMM), by virtue of its characteristics, is suitable for handling the time structure in speech recognition, for the states belonging to the

speech sounds pronounced longer simply return into themselves repeatedly. Maier and others [9] [10] have successfully applied the method with adults whose larynx had been removed due to throat cancer and with children born with a cleft lip and palate. In these patients a close relation can be achieved between the subjective and automatic assessments. This was used as a basis for developing PEAKS (Program for Evaluation and Analysis of all Kinds of Speech Disorders), a recording and analysing system for the automatic or manual assessment of utterance disorders and speech impediments.

Typical, easy-to detect utterance disorders are associated with the different disorders. For the speech disorders, the researchers had training models available in sufficient quantities, so it was possible to develop the statistical model necessary for automatic assessment. However, the mispronunciations of the hearing impaired cannot be typified [7]. Our general-purpose speech recognition device based on the HTK Speech Recognition Toolkit [6], which is adapted to 3,600 words and 1,800 sentences recorded with the voices of 60 school children (12-14 year-old primary school children from three special institutions for the teaching of the deaf and hard-of-hearing) proved to be unsuitable for automatic assessment.

Dynamic time warping (Dynamic Time Warping, DTW) was used in the early era of speech recognition for the comparison of the patterns to be recognised and the reference samples. The procedure examines optimum time alignment as the search for the path with minimum length or weight in a given graph. Let us suppose that the x words to be examined consist of k pieces of segments and the data characterising the i-th ($i = 1, 2, \ldots, k$) segment are summed up in vector x_i . Next these elementary vectors are collected into a matrix in the classification algorithm. Thus the incoming word is characterised by the vector series x_1, x_2, ..., x_k . Let the vector series y_1, y_2, ..., y_r characterise in a similar way the vocabulary element y, with which the incoming word is to be compared. The objective is to produce a vector series $x_{1,2,\ldots,r}$ (length r) from the vector series x_1, x_2, ..., x_k by repeating some and omitting others, for which the 'distance'

$$D = \sum_{i=1}^{r} d(x_i, y_i) \tag{1}$$

takes its minimum. Here $d(x, y)$ is an arbitrary given distance function. In producing the vector series $x_{1,2,\ldots,r}$, secondary conditions are set, of which the following is a possible version:

- any vector x_i can only be repeated once (thus we can at most double, but not triple the number of vectors);
- if x_i is omitted, its neighbours (x_{i-1} and x_{i+1}) cannot be omitted, thus two neighbouring segments cannot be omitted;
- the order of segments cannot be reversed [4].

The characteristic vectors used as the inputs to the algorithm are provided by feature extraction of speech. The result of segmentation was examined on the basis

of feature extraction by means of the usual procedures:

- MFCC: Mel-Frequency Cepstral Coefficients [2],
- PLP: Perceptual Linear Prediction [5],
- MEL band energy: logarithmic energy [13].

In our experiment the references were provided by recorded speech samples of university students of liberal arts participating in competitions of proper pronunciation and school children with proper pronunciation in the same age group as the hearing impaired involved in the experiment. These references were regarded as standard recorded speech samples. The recorded speech samples of the hearing-impaired children were provided by the speech sound database recorded for the purpose of creating the assessment scale.

The database includes 2,421 words (some words occur several times, but with different speakers, therefore their time structures are also different), which were assessed by 13 teachers and 23 students. Every teacher assessed only the recorded speech samples of the pupils of a school different from his own so as to avoid bias resulting from recognising the speaker. The assessors could listen to a recorded speech sample several times and could make comments on the samples. The results were recorded via an Internet application. In the case of the teachers, the basis of the assessment was given by a five-grade scale set worked out by them.

Interpretation of the scale:

- *Unintelligible (1)*: articulation is completely distorted; the vowels and consonants are unrecognisable; the reproduction of the syllable number is not adequate or discernible; breathing and management of breath is faulty; tempo and rhythm are incorrect; the utterance is unmelodious, non-dynamic or too tense.
- *Difficult to understand (2)*: grave distortions, omission of speech sounds, speech sound replacement; only some of the vowels can be discerned; distortions due to insufficient breathing, e.g. too breathy or choked; characterised by irregular, disturbing tonality, rhythm and tempo.
- *Moderately intelligible (3)*: the articulation of vowels is correct, the number of syllables is appropriate; serious speech defects may occur, e.g. dyslalia (the speech impediment in which certain vowels are incompletely formed), nasality, head voice, prosodic inadequacies.
- *Easy to understand (4)*: slight speech defects; slight prosodic inadequacies.
- *Understandable at the same level as the speech of the hearing (5)*: at most 1-2 speech sound defects may occur.

The 23 students had to score the recorded speech samples on a scale of 1-5 on the basis of everyday usage. Three hundred words were chosen out of the 2,421 words for the detailed segmentation analyses. The stock of words chosen is sufficiently varied not only according to the lengths of the words, but also from the aspect of the occurrence of speech sound juncture features, which is characteristic also of the complete word database. The stock of words of the recorded speech samples was prepared by teachers of the hearing impaired, taking the active vocabulary of the individual students carefully into consideration. The 300 words were manually segmented by a speech processing expert, providing the basis for the comparison.

Comparing the result of the segmentation time warped to correctly articulated reference speech with the time data given by the expert provided values that could not be used for poor quality speech. Often completely different results were obtained for the standard reference samples originating from the various subjects articulating the given word.

The cause of the failure was attributed to the deficiencies in dynamic time warping. The application of dynamic time warping for the purpose of speech recognition was neglected because the comparison has to be performed for every conceivable vocabulary element, which is extremely time-consuming. In addition, more advanced decision-making methods compare the speech section to be recognised not with the voice of a given speaker, but compare the element to be recognised with the data of a population of speakers using statistical learning methods. Our solution integrates a statistical model into the input data, and eliminates the speaker dependence of the reference sample by a new method of reference generation.

It was supposed that the characteristics obtained for the individual speech sounds by statistical methods would provide more reliable results. A neural network was trained on the basis of the BABEL speech sound database.

The BABEL database consists of three different parts: recorded speech samples of numbers of isolated and connected words, CVC (consonant-vowel-consonant) syllables, and continuously read speech. Both the sentences read and the number series were planned so as to provide a good coverage of the speech sound combinations in the Hungarian language. Some of the speech samples in the continuous part are in whispers. Part of the database is segmented into phonemes and labelled. The database includes the voices of a total of 30 male and 30 female speakers as well as 2,000 sentences and 14,000 connected number series.

3 Dynamic time warping input data

We attempted to derive the essence of speech sounds by means of the output activity of neural networks. Neural networks were trained in acoustics-phonetics classification, then using their outputs, new neural networks were trained to differentiate within the class. In the course of training the correct outputs were given a value 1 in their own time frame, and the others were given the value 0. The goodness of classification was checked on testing patterns not included in the training and amounting to a quarter of the complete speech sound material. In the course of

testing, in order to obtain the goodness criterion for each speech sound, the sum of the activities of their own outputs was divided by the sum of the activities of the other outputs, calculated for all the testing time segments.

$$G_i = \sum_{\forall R} O_{NN} \Big/ \sum_{\forall F} O_{NN}, \tag{2}$$

where

G_i – goodness of neural network for feature of speech sound or class of speech sounds (i),

O_{NN} – neural network outputs,

$\forall R$ – correct output activity for all of its own time frames,

$\forall F$ – incorrect output activity for all the other time frames.

The neural network whose goodness factor was the maximum for all speech sounds was kept, so we had five neural networks for acoustic-phonetic classification and four neural networks for grouping within the class. For orthography transcription we will use SAMPA symbols. The classes formed by the neural networks are as follows:

- pause;
- vowels *(a, a:, E, e:, i, o, 2, u, y)*;
- semi-vowels *(m, n, J, r, l, j)*;
- fricatives *(f, s, S, h, v, z, Z)*;
- plosives *(p, t, ts, tS, t', k, b, d, d', g)*.

The speech sounds belonging to the outputs of the neural network dedicated to the classes are listed in parentheses. We tried to perform the acoustic-phonetic classification by using only a single neural network, but we got weaker results than when using neural networks dedicated to individual classes (Figure 1). For dynamic time warping, these outputs were directly used as a feature vector of the word analysed. Among the speech feature extraction methods examined (MFCC, PLP, MEL subband energy), PLP showed the highest goodness factors, thus the outputs of neural networks trained by PLP speech feature extraction were used as the inputs of dynamic time warping. Training was performed with several options. The setting providing the maximum of the goodness factor is: to the 12 PLP data and logarithmic energy of the actual 40ms frame were added the average of two frames of the preceding 80 ms section and the average of two frames of the subsequent 80 ms. The feature 3×13 describes the 40ms segment in the middle of the 200 ms interval. Training of the 5 neural networks meant for phonetic classification was performed with these parameters.

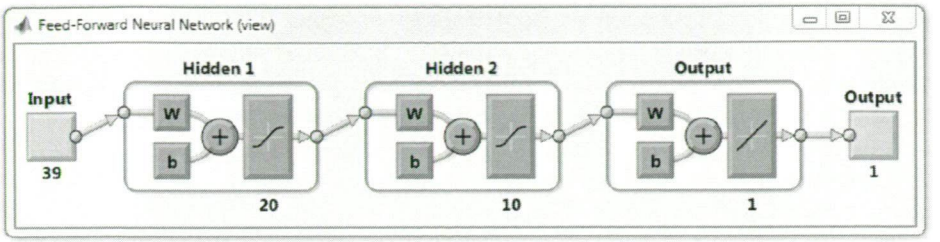


Figure 1: Model of the neural network determining the acoustic speech sound class

In addition to the 39 PLP features, neural networks trained to recognize speech sounds within the phonetic classes were also given the outputs of the five classifying neural networks as input. Figure 2 shows the structure of the neural network used for sorting vowels.

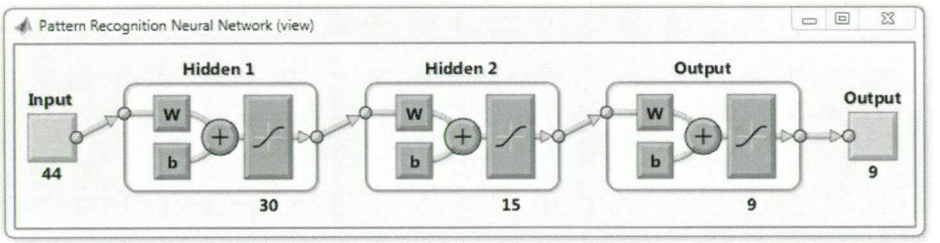


Figure 2: Neural network model of the acoustic speech sound class of vowels

Segmentation was also performed with the neural networks trained with the shorter PLP time frames; the smallest errors were obtained with the above setup. The relevant toolboxes of the program package MATLAB were used for the calculations [11].

4 Reference generation

Using a concrete recorded speech sample as reference, the failure of segmentation discussed above was attributed to individual differences. On the basis of a statistics model, a neural network trained with a great number of speakers is better at reflecting the similarities to the individual speech sounds.

In developing the reference shape, the chosen input data had to be accommodated. Since in this task the objective is not recognition of the word but the alignment of the recorded speech sample, the phonetic transcription of the word was at our disposal. For the purpose of reference generation, the output belonging to the given speech sound and output of the class including the speech sound are made active in the allocated time interval. The timing of the individual speech

sounds can be determined starting out from the average time length of the speech sounds [12]. The speech of hearing-impaired children is slower than the average speech tempo. According to our measurements, the time length of the fastest speech was one and a half times the average, therefore in reference generation one and a half times the average time lengths of speech sounds were used, thus a few speech sounds pronounced shorter also fall in the range allowed. Calculating with the above auxiliary conditions of dynamic time warping, the length of individual speech sounds may vary between $^3/_4$ of and three times the average speech sound time length after time warping. Figure 3 shows the created reference features of the word hűséges [hy:Se:gES] (meaning 'faithful'). The horizontal axis shows the time index of the frames, and the vertical axis shows (from bottom to top) the acoustic-phonetic features starting with the pause then the outputs classifying vowels, semi-vowels, fricatives and plosives, and above them the individual outputs of the speech sound classes in the above order.

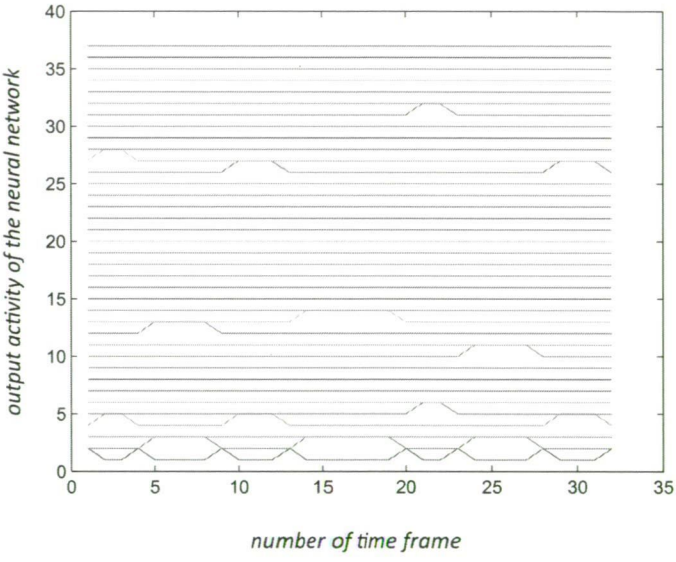


Figure 3: Features created in the reference generation of the word [hy:Se:gES]

In case of good quality speech, the outputs of the neural networks show significant activity. The word 'hűséges' can be clearly understood and is of a quality accepted in everyday communication (Figure 4).

On the other hand, the output levels decrease visibly and several outputs show activity simultaneously as Figure 5 shows the output activities belonging to the word 'valami' [vOlOmi] (meaning 'something') pronounced with a distortion making it unintelligible. The outputs of a neural network are not faultless and among the speech sound samples to be segmented there are also speech samples distorted to unintelligibility.

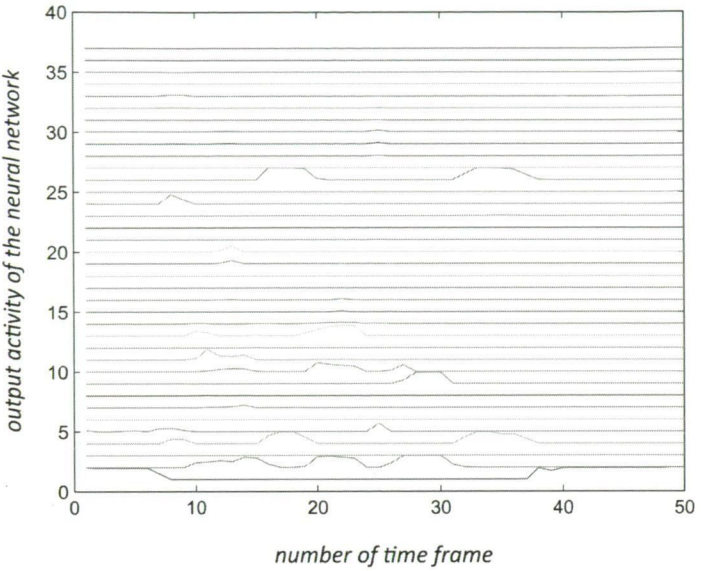


Figure 4: Significant activity of the actual outputs
in aligning the clearly understandable word [hy:Se:gES]

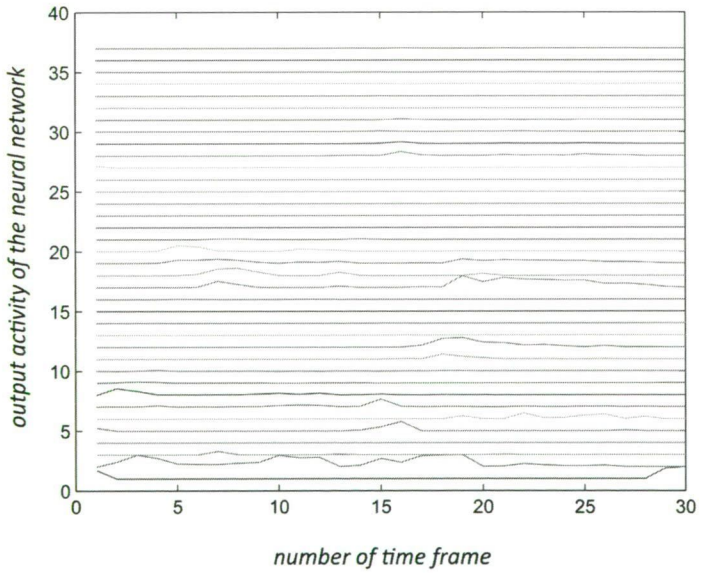


Figure 5: Weak activity of the actual outputs
in aligning the word [vOlOmi] with a distorted pronunciation

Therefore not simply the activity of the relevant output is used as input, but the activities of the outputs belonging to the same acoustic-phonetic class are summed and weighted with the similarity measurement between the speech sounds.

On the basis of the speech sound database BABEL and using PLP speech feature extraction, the average of the coefficients belonging to the total occurrence of the individual speech sounds was determined. A 40 ms segment was marked from the center of each speech sound having a stationary phase (vowels, semi-vowels and fricatives) and the last 40 ms (burst) for plosives. Then Euclidean distances were formed between the averages of Hungarian speech sounds [3]. By reversing the normalised distance, similarity measurements were formed between the individual speech sounds:

$$H(i,j) = 1 - D(i,j)/D_{max}, \tag{3}$$

where $H(i,j)$ is the similarity of the i-th and j-th speech sounds, $D(i,j)$ is the distance of the averages of the PLP coefficients of the i-th and j-th speech sounds and D_{max} is the maximum of these distances.

Summation of the similarity measurements for an acoustic-phonetic class is as follows:

$$S(i) = \sum_{j \ni O} H(i,j) * NN(j), \tag{4}$$

where O designates the speech sounds belonging to its own class and $NN(j)$ is the j-th output of the neural network.

Without transferring output activities to the similar phones, in case of misclassification or highly distorted speech the right neural network output would not get any activity, causing false pause frames in the time interval of the phone. This time shift would risk the right segmentation of neighbouring phones as well. The artificially created reference pattern and the cumulated neural network outputs of speech examined form the basis of dynamic time warping.

The following figures show the similarity measurements of speech sounds belonging to the classes of the neural network compared to each other (Figures 6–9).

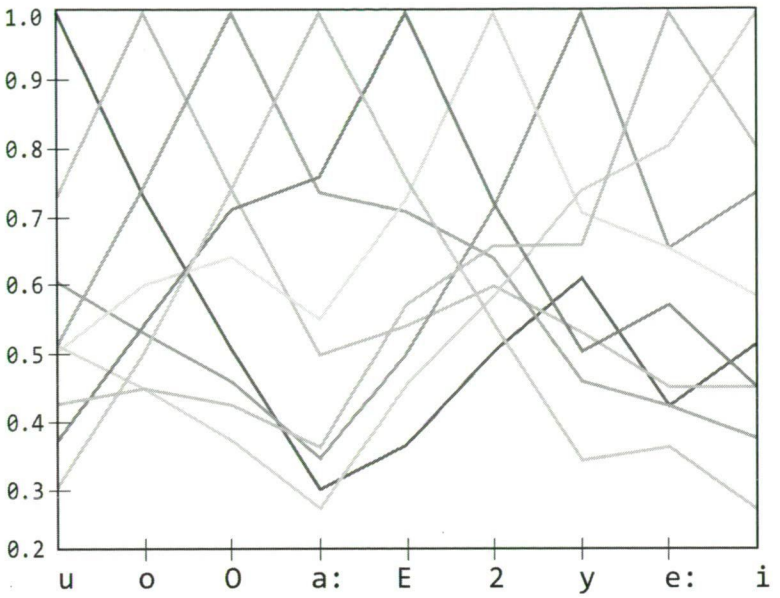


Figure 6: Similarity measurements of vowels

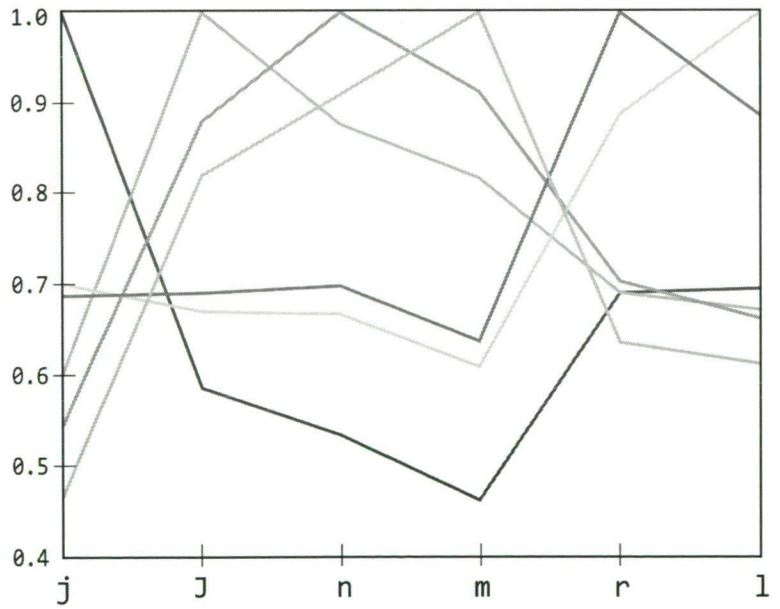


Figure 7: Similarity measurements of semi-vowels

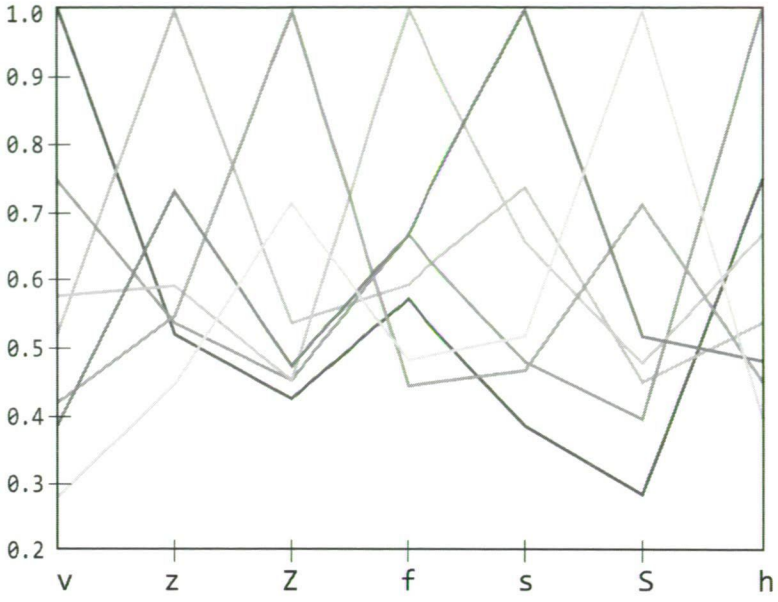


Figure 8: Similarity measurements of fricatives

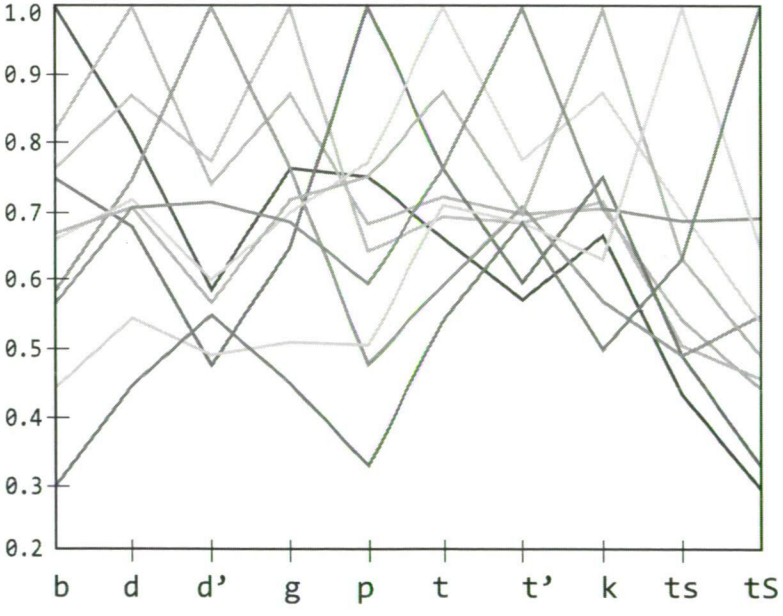


Figure 9: Similarity measurements of plosives

5 Modifying the DTW algorithm

Tested by the classical rules of dynamic time warping and using the outputs of the neural network as feature vector, time warping produced much better results than when recorded words were used as reference.

During testing, however, it was found that in the speech samples of hearing-impaired children pauses of several tenths of a second frequently occurred between speech sounds. In order to treat this problem, the secondary conditions of the dynamic time warping algorithm were modified:

- an optional pause was inserted after each speech sound in producing the reference;
- the pause can be repeated number of times.

According to the rules set out above, a time interval can be lengthened to a maximum of twice its original length. However, in the speech samples of hearing-impaired children there were often speech sounds pronounced longer than that.

Therefore:

- double reiteration of a time frame can also be allowed, thus a time interval can be lengthened to three times its original length. In the following, this will be referred to as adapted dynamic time warping (ADTW).

Figure 10 shows the segmentation results of the haltingly pronounced word 'lázmérő' [la:zme:r2:], (meaning 'clinical thermometer') and the hardly intelligible word 'valami' [vOlOmi], (meaning 'something') as examples.

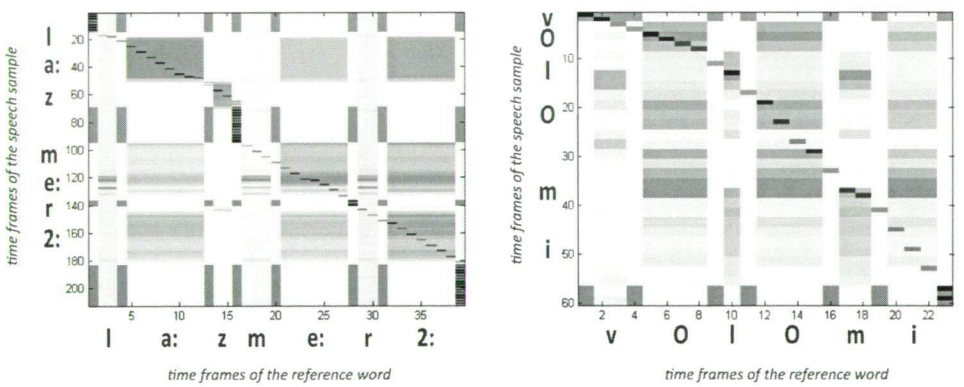


Figure 10: Dynamic time warping of
the words [la:zme:r2:] and [vOlOmi] using the ADTW method

The horizontal axis shows the reference segments and the vertical axis shows the segments of the speech sample examined. In the reference the vertical bands

of the size of one time frame indicate the pauses inserted. The shading of the points of the matrix is proportional to the similarity measurement. Darker bands mean greater similarity. The horizontal line appears in the time frames matched. On the horizontal axis of the word 'lázmérő' [la:zme:r2:], time frame 16 shows the insertion of a pause of considerable length. The word 'valami' [vOlOmi] even with an extremely distorted articulation can be successfully segmented with the procedure proposed.

The recorded speech samples can be heard at the following links:

`http://mazsola.iit.uni-miskolc.hu/~avarga/hangmintak/huseges.wav`
`http://mazsola.iit.uni-miskolc.hu/~avarga/hangmintak/lazmero.wav`
`http://mazsola.iit.uni-miskolc.hu/~avarga/hangmintak/valami.wav`

6 Evaluation

In evaluating the modified algorithm, expert segmentation was regarded as the reference. In expert segmentation, the segment boundaries were determined on the basis of a combination of the time function, the spectrogram and the speech sound played from the segment boundary (or to the segment boundary). The comparison was performed using two other segmentation procedures:

1. DTW algorithm based on acoustic-phonetic features, optimised for good-quality speech without being adapted to hearing-impaired speech samples.

   The objective of the time warping method based on acoustic-phonetic (AF) speech sound classes is to compare prosody (the combination of melody, pronunciation speed, rhythm, stress, speech sound intensity and tonality), which can be applied to several languages. It is a time warping method which aligns the two samples strictly along the speech sounds and performs scaling only within them. In this method the novelty is represented by the execution method of the computer segmentation, for which general acoustic speech sound classes were used, which determined language-independent articulation configurations [8]. Applicability to several languages followed from that. In the present case the developers supposed that the difference between the actual and the reference sample is minimal (the speaker is cooperative). Three differences were taken into account: insertion, omission and different pronunciation. The AF segmentation procedure was not adapted to poor-quality speech.

2. HMM-based speech recognition with PLP feature extraction, with the pauses between the speech sounds and their repetition of optional times included in the grammar rules and forced alignment segmentation.

Table 1: Proportions of correct segmentation for different speech feature extraction methods

Tolerance (ms)	Speech feature extraction methods		
	MFCC	**PLP**	**MEL**
<= 10	0.31	0.48	0.38
<= 20	0.61	0.72	0.60
<= 30	0.78	0.84	0.73
<= 40	0.86	0.90	0.82
<= 50	0.90	0.93	0.88
<= 60	0.93	0.94	0.91
<= 70	0.95	0.95	0.93
<= 80	0.95	0.95	0.93
<= 90	0.96	0.95	0.94
<= 100	0.96	0.96	0.94

The recordings of the 24 male and 24 female speakers of the BABEL speech sound database provided the training samples, and recordings of 6 male and 6 female speakers provided the testing samples. A 10 ms time frame was chosen and the previously used speech feature extraction procedures were examined as feature vectors from a segmentation aspect:

- MFCC: 12 coefficients and log energy give the 13 components,
- PLP: 12 coefficients and log energy,
- MEL: logarithmic band energy dividing the 125 Hz – 8 kHz frequency domain into 30 part bands on the basis of the mel-scale.

On the basis of the results (Table 1), PLP speech feature extraction was chosen here and will be referred to as HMM in the following. Since the ultimate objective of the method to be developed is automatic assessment, in speech feature extraction the centre of phones is searched for, thus the stationary phase if there is one will characterise the given speech sound [14]. Therefore a segmentation error is regarded as serious or less serious depending on its sign. If the segmenter puts the beginning of a speech sound further forward of the real limit, the error is in the incorrect direction, for the erroneous limit lies outside of the interval of the desired speech sound. Again the error is more serious if the end of the speech sound is put farther back of the real limit. The error is not so serious if the limit is placed farther back of the real beginning or further forward of the real ending within the speech sound examined (Figure 11).

Figure 11: Segmentation error by error direction

The following tables (Table 2–5) sum up the results of the segmentations performed by the different procedures concerning the 3,694 speech sounds included in the 300 words.

Table 2: Results of the acoustic-phonetic (AF) segmentation procedure

AF segmentation procedure				
Tolerance (ms)	Initial		Final	
	Incorrect	Correct	Correct	Incorrect
0	1785	62	1782	65
20	1747	100	1795	52
40	1667	180	1801	46
60	1500	347	1805	42
80	1340	507	1811	36
100	1187	660	1812	35
200	692	1155	1825	22

Table 3: Results of the HMM segmentation procedure

HMM segmentation procedure				
Tolerance (ms)	Initial		Final	
	Incorrect	Correct	Correct	Incorrect
0	1327	528	1569	286
20	620	1235	1669	186
40	296	1559	1734	121
60	190	1665	1761	94
80	142	1713	1780	75
100	122	1733	1792	63
200	66	1789	1824	31

Table 4: Results of the ADTW segmentation procedure

ADTW segmentation procedure				
Tolerance (ms)	**Initial**		**Final**	
	Incorrect	**Correct**	**Correct**	**Incorrect**
0	137	1718	1717	138
20	97	1758	1753	102
40	80	1775	1776	79
60	64	1791	1791	64
80	54	1801	1803	52
100	46	1809	1815	40
200	25	1830	1838	17

Table 5: Number of errors outside of the time interval of the speech sound from the shifts in the correct direction in the tables above

Segmentation Procedure	**Initial Shifted**	**Final Shifted**
AF segmentation procedure	15	1258
HMM segmentation procedure	135	42
ADTW segmentation procedure	39	77

The results show that 'incorrect direction' errors definitely lying outside of the time interval of the speech sound are a magnitude smaller for the proposed ADTW segmentation than for the AF procedure not adapted to poor quality speech or for the HMM procedure. 'Correct direction' errors, exceeding the time length of the speech sounds are also the fewest also with the ADTW method.

The AF segmentation considered the speech sounds shorter than their real length: placing the beginning of a speech sound is shown in the column of errors in the incorrect direction of Table 3, placing the end of a speech sound more forward is located in the field outside of the domain of Table 5. In HMM segmentation mainly the accuracy of marking the limits at the beginning of speech sounds lags behind the results of the ADTW procedure.

ADTW segmentation is utilized in automatic speech assessment. Speech samples of hearing impaired children were evaluated by the 36 assessors. The average scores served as a reference. The scores of the automatic assessment were closer to the averages than that of 28 subjects out of the 36 ones, while one teacher and seven students reached scores closer to the averages [13].

7 Summary

Adaptation of dynamic time-warping has been presented with the objective of a more efficient segmentation of the voices of hearing-impaired children. Pauses inserted between the speech sounds which can be repeated arbitrarily are able to handle the long pauses of halting speech. Time frames that can be repeated twice – that can be included a maximum of three times – make it possible to follow extremely slow speech. The acoustic-phonetic features used as inputs of the algorithm are able to create perceptible activity at the outputs of the neural networks, thus a statistical model is incorporated into the input data. The proposed method of reference generation does not require the recording of reference speech samples, thus eliminating the speaker-dependence of the reference sample.

References

[1] Czap, L. and Pintér, J. Segmentation of Poor Quality Speech. (in Hungarian). *Proceedings of XX-th International Sciencific Conference of Young Engineers*, pages 119–122, Kolozsvár, 2015.

[2] Davis, S. B., and Mermelstein, P. Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 28(4): 357–366, 1980.

[3] Deza, E. and Deza, M. *Dictionary of Distances.* Elsevier Science Publishers, Netherlands, ISBN 0444520872, 2006.

[4] Faragó, A., Fülöp, T., Gordos, G., Magyar, G., Osváth, L. and Takács, Gy. *Simple isolated word speech recognizer.* (in Hungarian) Research report, 1985.

[5] Hermansky, H. Perceptual linear predictive (PLP) analysis for speech. *Journal of the Acoustical Society of America*, 87(4): 1738–1752, 1990.

[6] Hidden Markov Model Toolkit (HTK) website. http://htk.eng.cam.ac.uk/

[7] Illésné K. M. Positive and negativ feedback of Internet-based speech development of the hearing impaired. (in Hungarian). *Alkalmazott Nyelvészeti Közlemények*, 9(1): 135–143, 2014.

[8] Kiss, G., Sztahó, D. and Vicsi, K. Language independent automatic speech segmentation into phoneme-like units on the base of acoustic distinctive features. *Proceedings of CogInfoCom2013*, Budapest, 2013.

[9] Maier, A., Haderlein, T., Eysholdt, U., Rosanowski, F., Batliner, A., Schuster, M. and Nöth, E. PEAKS – A system for the automatic evaluation of voice and speech disorders. *Speech Communication*, 51(5): 425–437, 2009.

[10] Maier, A., Hönig, F., Hacker, C., Schuster, M. and Nöth, E. Automatic evaluation of characteristic speech disorders in children with cleft lip and palate. In *Proceedings of 11th International Conference on Spoken Language Processing*, Brisbane, Australia, pages 1757–1760, 2008.

[11] MathWorks website. http://www.mathworks.com

[12] Németh, G. and Olaszy, G., editors. *Hungarian Speech.* (in Hungarian), Akadémiai Kiadó, Budapest, ISBN 978 963 05 8755 6, 2010.

[13] O'Shaughnessy, D. *Speech communication: Human and Machine.* Addison-Wesley, U.S.A., ISBN 0201165201, 1987.

[14] Pintér, J. M. *Automatic Assessment of Speech Quality.* PhD thesis (in Hungarian), University of Miskolc, Miskolc, 2015.

[15] Sakoe, H. and Chiba, S. Dynamic programming algorithm optimization for spoken word recognition. In *IEEE Transactions on Acoustics, Speech and Signal Processing*, 26 (1): 43–49, 1978.

Received 25th November 2015

Acta Cybernetica **22** (2016) 791–805.

The Holonomy Decomposition of some Circular Semi-Flower Automata

Shubh N. Singh* and Kanduru V. Krishna†

Abstract

Using holonomy decomposition, the absence of certain types of cycles in automata has been characterized. In the direction of studying the structure of automata with cycles, this paper focuses on a special class of semi-flower automata and establish the holonomy decomposition of certain circular semi-flower automata. In particular, we show that the transformation monoid of a circular semi-flower automaton with at most two bpis divides a wreath product of cyclic transformation groups with adjoined constant functions.

Keywords: transformation monoids, semi-flower automata, holonomy decomposition

1 Introduction

Usefulness of a decomposition method for any given system does not require any justification. The primary decomposition theorem due to Krohn and Rhodes has been considered as one of the fundamental results in the theory of automata and monoids [13]. Eilenberg has given a slight generalization of the primary decomposition called the holonomy decomposition [8]. Here, Eilenberg established that every finite transformation monoid divides a wreath product of its holonomy permutation-reset transformation monoids. The holonomy decomposition of an automaton is considered to be the holonomy decomposition of the transformation monoid of the automaton. The holonomy decomposition is also used to study the structural properties of certain algebraic structures [11, 12]. The holonomy decomposition method appears to be relatively efficient and has been implemented computationally [4, 5]. One can use the computer algebra package, SgpDec [7] to obtain the holonomy decomposition of a given finite transformation monoid.

In order to ascertain the structure of an automaton, the holonomy decomposition considers the monoid of automaton and looks for groups induced by the

*Department of Mathematics, Central University of South Bihar, Patna, India, E-mail: shubh@cub.ac.in

†Department of Mathematics, IIT Guwahati, Guwahati, India, E-mail: kvk@iitg.ac.in

DOI: 10.14232/actacyb.22.4.2016.4

monoid permuting some set of subsets of the state set. These groups are called the holonomy groups, which are building blocks for the components of the holonomy decomposition. Using the holonomy decomposition, Egri-Nagy and Nehaniv characterized the absence of certain types of cycles in automata [6]. In fact, they proved that an automaton is algebraically cyclic-free if and only if the holonomy groups are trivial. On the other hand, the structure of automata with cycles is much more complicated.

In the direction of studying the structure of automata with cycles, this work concentrates on a special class of semi-flower automata (SFA) [9, 15]. Using SFA, the rank and intersection problem of certain submonoids of a free monoid have been studied [10, 16, 17].

In this paper, we consider circular semi-flower automata (CSFA) classified by their bpi(s) – branch point(s) going in – and obtain the holonomy decomposition of CSFA with at most two bpis. We present some preliminary concepts and results in Section 2. The main work of the paper is presented in Section 3. Finally, Section 4 concludes the paper.

2 Preliminaries

This section has two subsections on the holonomy decomposition and automata to present necessary background materials on these topics.

2.1 The Holonomy Decomposition

In this subsection, we provide brief details on the holonomy decomposition which will be useful in this paper. For more details one may refer [2, 4, 8].

We fix our notation regarding functions. Let $f : X \to Y$ be a function from X into Y. We write an argument $x \in X$ of f on its left so that xf is the value of f at x. The *rank* of f, denoted $\text{rank}(f)$, is the cardinality of its image set Xf. The set of all functions from X into Y is denoted by Y^X. The composition of functions is designated by concatenation, with the leftmost function understood to apply first so that $xfg = (xf)g$.

A *transformation monoid* is a pair (P, M) consists of a nonempty finite set P and a submonoid M of $\mathscr{T}(P)$, where $\mathscr{T}(P)$ is the monoid of all functions on P with respect to composition of functions. Note that there is an action of submonoid M on set P. Let us denote the action of $m \in M$ on $p \in P$ as pm. If M is a subgroup of $\mathscr{T}(P)$, then (P, M) is called a *transformation group*.

A transformation monoid (P, M) *divides* a transformation monoid (Q, N), denoted $(P, M) \prec (Q, N)$, if there exists a partial surjective function $\varphi : Q \to P$ and, for every $m \in M$, an element $n \in N$ such that $(q\varphi)m = (qn)\varphi$ for each $q \in \text{Dom}(\varphi)$. The *wreath product* of two transformation monoids (P, M) and (Q, N), denoted $(P, M) \wr (Q, N)$, is the transformation monoid $(P \times Q, W)$, where $W = \{(f, n) \mid f \in M^Q \text{ and } n \in N\}$ is the monoid with operation given by

$$(f, n)(g, k) = (h, nk), \quad qh = (qf)((qn)g) \text{ for every } q \in Q,$$

and the action of $(f,n) \in W$ on an element $(p,q) \in P \times Q$ is given by

$$(p,q)(f,n) = (p(qf), qn).$$

The wreath product is an associative operation on transformation monoids.

Let (P,M) be a transformation monoid. For $p \in P$, let $\widetilde{p}$ be the constant function on P which takes the value p. The semigroup of all these constant functions on P is denoted by $\widetilde{P}$. The *closure* of (P,M) is the transformation monoid $\overline{(P,M)} = (P, M \cup \widetilde{P})$. The *skeleton* of (P,M) is $\mathscr{J} = \{Pm \mid m \in M\} \cup \bigcup_{p \in P} \{\{p\}\}$ with the subduction relation $\leq$ on $\mathscr{J}$ given by $R \leq S$ if and only if $R \subseteq Sm$ for some $m \in M$. The subduction relation is a preorder relation. Consequently, there is an equivalence relation $\sim$ on $\mathscr{J}$ given by $R \sim S$ if and only if $R \leq S$ and $S \leq R$. We write $\mathscr{J}_i$ to denote the set of all elements of $\mathscr{J}$ of cardinality i (for $i \geq 1$), i.e.,

$$\mathscr{J}_i = \{T \in \mathscr{J} \mid |T| = i\}.$$

Let (P,M) be a transformation monoid. The *height* of $T \in \mathscr{J}$ is given by the function $\eta : \mathscr{J} \to \mathbb{Z}$, which is defined by $T\eta = 0$ if $|T| = 1$, and for $|T| > 1$, $T\eta$ is the length of the longest subduction chain(s) in the skeleton starting from a non-singleton set and ending in T. The *height* of (P,M) is defined as $P\eta$. For $T \in \mathscr{J}$ with $|T| > 1$, put $K(T) = \{m \in M \mid Tm = T\}$. The *paving of* T, denoted $B(T)$, is the set of maximal subsets of T that are contained in $\mathscr{J}$, i.e.,

$$B(T) = \{R \in \mathscr{J} \mid R \subsetneq T \text{ and if } S \in \mathscr{J} \text{ with } R \subseteq S \subseteq T, \text{ then } S = R \text{ or } S = T\}.$$

The set $G(T)$ of all distinct permutations on $B(T)$ induced by elements of $K(T)$ is called the *holonomy group* of T, and $\big(B(T), G(T)\big)$ is a transformation group. We denote an element of $G(T)$ by $\check{m}$ which is induced by $m \in K(T)$. For $T, T' \in \mathscr{J}$ with $|T| > 1, |T'| > 1$, if $T \sim T'$, then $\big(B(T), G(T)\big)$ is isomorphic to $\big(B(T'), G(T')\big)$.

The *holonomy decomposition theorem* due to Eilenberg states that every finite transformation monoid divides a wreath product of its holonomy permutation-reset transformation monoids, as presented in the following:

Theorem 2.1 ([8]). *If (P,M) is a finite transformation monoid of height n, then*

$$(P,M) \prec \overline{\mathscr{H}_1} \wr \overline{\mathscr{H}_2} \wr \ldots \wr \overline{\mathscr{H}_n},$$

where, for $1 \leq i \leq n$,

$$\mathscr{H}_i = \left(\prod_{j=1}^{k_i} B(T_{ij}), \prod_{j=1}^{k_i} G(T_{ij}) \right),$$

in which k_i is the number of equivalence classes at height i and $\{T_{ij} \mid 1 \leq j \leq k_i\}$ is the set of representatives of equivalence classes at height i.

2.2 Automata

This subsection is devoted for essential preliminaries on automata and monoids. For more details one may refer [1, 9, 15].

Let A be a nonempty finite set called *alphabet* with its elements as *symbols*. The free monoid over A is denoted by A^* whose elements are called words, and ε denotes the empty word – the identity element of A^*.

By an *automaton*, we mean a quintuple $\mathcal{A} = (Q, A, \delta, q_0, F)$, where Q is a nonempty finite set called the set of *states*, A is alphabet, $q_0 \in Q$ called the *initial* state, $F \subseteq Q$ called the set of *final* states, and $\delta : Q \times A \to Q$ called the transition function. Clearly, by denoting the states as vertices and the transitions as labeled directed edges, an automaton can be represented by a digraph in which the initial state and final states shall be distinguished appropriately. A path in a digraph is an alternating finite sequence $v_0, e_1, v_1, e_2, v_2, \ldots v_{k-1}, e_k, v_k$ of vertices and labeled directed edges such that, for $1 \leq i \leq k$, the tail and the head of the edge e_i are v_{i-1} and v_i, respectively. A *path* in an automaton is a path in its digraph. For $p_i \in Q$ $(0 \leq i \leq k)$ and $a_j \in A$ $(1 \leq j \leq k)$, let

$$p_0 \xrightarrow{a_1} p_1 \xrightarrow{a_2} p_2 \xrightarrow{a_3} \cdots \xrightarrow{a_{k-1}} p_{k-1} \xrightarrow{a_k} p_k$$

be a path in $\mathcal{A}$. The word $a_1 \cdots a_k \in A^*$ is called the *label of the path*. A *null path* is a path from a state to itself labeled by the empty word ε. A path that starts and ends at the same state is called as a *cycle*, if it is not a null path.

Given an automaton $\mathcal{A}$, we can inductively extend the transition function for words by, for all $u \in A^*, a \in A$ and $q \in Q$,

$$\delta(q, \varepsilon) = q, \text{ and } \delta(q, au) = \delta(\delta(q, a), u).$$

We write qu instead of $\delta(q, u)$. There is a natural way to associate a finite monoid to $\mathcal{A}$. For each $x \in A^*$, we define a function $\delta_x : Q \to Q$ by $q\delta_x = qx$ for all $q \in Q$. The set of functions, $M(\mathcal{A}) = \{\delta_x \mid x \in A^*\}$, forms a monoid under the composition of functions. If $M(\mathcal{A})$ is a group, then $\mathcal{A}$ is called a *permutation* automaton. Note that the monoid $M(\mathcal{A})$ is generated by the functions defined by symbols. Further, for all $x, y \in A^*$, we have $\delta_{xy} = \delta_x \delta_y$ and δ_ε is the identity function on Q.

Let $\mathcal{A}$ be an automaton. A state q is called a *branch point going in*, in short *bpi*, if the number of transitions coming into q (i.e. the indegree of q – the number of edges coming into q – in the digraph of $\mathcal{A}$) is at least two. We write $BPI(\mathcal{A})$ to denote the set of all bpis of $\mathcal{A}$. A state q is *accessible* (respectively, *coaccessible*) if there is a path from the initial state to q (respectively, a path from q to a final state). An automaton $\mathcal{A}$ is called a *trim* automaton if all the states of $\mathcal{A}$ are accessible and coaccessible. An automaton $\mathcal{A}$ is called a *semi-flower automaton* (in short, SFA) if it is a trim automaton with a unique final state that is equal to the initial state such that all the cycles in $\mathcal{A}$ visit the unique initial-final state q_0.

Let $X = \{p_1, \ldots, p_r\}$ be a finite set and $Y \subseteq X$. A *Y-cycle* is a permutation f_Y on X such that f_Y induces a cyclic ordering on Y $(= \{p_{i_1}, \ldots, p_{i_s}\}$, say) and f_Y is identity on $X \setminus Y$, i.e., for $1 \leq j < s$ and $p \in X \setminus Y$,

$$p_{i_j} f_Y = p_{i_{j+1}},\ p_{i_s} f_Y = p_{i_1}, \text{ and } p f_Y = p.$$

A *circular permutation* on X is an X-cycle. It is well known that for every permutation f on X, there exists a partition $\{Y_1, \ldots, Y_t\}$ of X such that $f = f_{Y_1} f_{Y_2} \cdots f_{Y_t}$, a composition of (disjoint) Y_i-cycles.

An automaton $\mathcal{A}$ is called a *circular* automaton if there exists a symbol $a \in A$ such that δ_a is a circular permutation on Q. Circular automata have been studied in various contexts. Pin proved the Černý conjecture for circular directable automata with a prime number of states [14]. Further, Dubuc showed that the Černý conjecture is true for any circular directable automata [3].

In order to investigate the holonomy decomposition of circular semi-flower automata, we consider these automata classified by their number of bpis and complete the task for the automata with at most two bpis.

3 Main Results

We present results of the paper in three subsections. In Subsection 3.1, we obtain some properties of circular semi-flower automata (CSFA) which are useful in the work. We investigate the holonomy decomposition of CSFA with at most one bpi and two bpis in subsections 3.2 and 3.3, respectively.

In what follows, $\mathcal{A} = (Q, A, \delta, q_0, q_0)$ is an SFA such that $|Q| = n$ $(n > 1)$. Further, for $1 \leq m \leq n$, $\mathscr{C}_m$ denotes a transformation group (X, C_m) for some set $X \subseteq Q$ with $|X| = m$ and C_m is the cyclic group generated by circular permutation induced by a word on the set X.

3.1 Circular Semi-Flower Automata

In this subsection, we first ascertain that there is a unique circular permutation induced by symbols on the state set of CSFA and then we proceed to obtain certain properties pertaining to the bpis of CSFA.

Proposition 3.1. *Let $\mathcal{A}$ be an SFA and $a, b \in A$.*

(i) If δ_a is a permutation on Q, then δ_a is circular permutation on Q.

(ii) If δ_a and δ_b are permutations on Q, then $\delta_a = \delta_b$.

Proof.

(i) Write $\delta_a = f_{Q_1} \cdots f_{Q_t}$, a composition of Q_i-cycles for some partition $\{Q_1, \ldots, Q_t\}$ of Q. Let $q_0 \in Q_r$ for some $r \in \{1, \ldots, t\}$. If $Q_r = Q$, then $t = r = 1$ and so that δ_a is circular permutation on Q. Otherwise, there exists $q \in Q \setminus Q_r$ and $s \in \{1, \ldots, t\} \setminus \{r\}$ such that $q \in Q_s$. Note that the Q_s-cycle induces a cycle in $\mathcal{A}$ that does not pass through the initial-final state q_0; a contradiction.

(ii) On the contrary, let us assume that $\delta_a \neq \delta_b$. From Proposition 3.1 (i), the permutations δ_a and δ_b are circular permutations on Q. Let cyclic orderings

on Q with respect to δ_a and δ_b be as shown below.

$$\delta_a : q_0, q_{i_1}, q_{i_2}, \cdots, q_{i_{n-1}}$$
$$\delta_b : q_0, q_{j_1}, q_{j_2}, \cdots, q_{j_{n-1}}$$

Since $\delta_a \neq \delta_b$, let k be the least number such that $q_{i_k} \neq q_{j_k}$. Note that there exists $s > k$ such that $q_{i_k} = q_{j_s}$ and also there exists $r > k$ such that $q_{j_k} = q_{i_r}$. The path

$$q_{i_k} \xrightarrow{a^{r-k}} q_{i_r} = q_{j_k} \xrightarrow{b^{s-k}} q_{j_s} = q_{i_k}$$

is a cycle in $\mathcal{A}$ labeled by the word $a^{r-k}b^{s-k}$ that does not pass through the initial-final state q_0; a contradiction.

□

Corollary 3.1. *There is a unique circular permutation induced by symbols on the state set of a CSFA.*

Proposition 3.2. *Let $\mathcal{A}$ be an SFA; then*

$$BPI(\mathcal{A}) = \varnothing \iff |A| = 1.$$

Proof. In an n-state automaton,

the total indegree of all states = the total number of transitions = $n|A|$.

Since $\mathcal{A}$ is accessible, indegree of each state is at least one. Consequently,

$$BPI(\mathcal{A}) = \varnothing \iff \text{the total indegree of all states } = n \iff |A| = 1.$$

□

In what follows, $\mathcal{B} = (Q, A, \delta, q_0, q_0)$ stands for an CSFA with $|Q| = n$ $(n > 1)$. If the number of bpis in $\mathcal{B}$ is less than the number of states in $\mathcal{B}$, then there is a unique symbol induces a permutation on Q. For the rest of the paper we fix the following regarding $\mathcal{B}$. Assume that the symbol $a \in A$ induces a circular permutation δ_a on Q. Accordingly,

$$\delta_a : q_0, q_1, \cdots, q_{n-1}$$

is the cyclic ordering on Q with respect to δ_a.

Proposition 3.3. *If $\mathcal{B}$ has at least one bpi, then its initial-final state q_0 is a bpi.*

Proof. Since $\mathcal{B}$ has at least one bpi, by Proposition 3.2, we have $|A| \geq 2$. We claim that $q_{n-1}\delta_b = q_0$ for all $b \in A \setminus \{a\}$ and so that q_0 is a bpi.

On the contrary, let us assume that $q_{n-1}\delta_c \neq q_0$ for some $c \in A \setminus \{a\}$. Then $q_{n-1}\delta_c = q_i$ for some i (with $1 \leq i < n$). Note that $q_i\delta_{a^{n-i-1}c} = q_i$. Therefore there is a cycle in $\mathcal{B}$ from q_i to q_i labeled by the word $a^{n-i-1}c$ that does not pass through the initial-final state q_0; a contradiction. Hence $q_{n-1}\delta_b = q_0$ for all $b \in A \setminus \{a\}$. □

Proposition 3.4. *For $1 \leq m < n$, if $|BPI(\mathcal{B})| = m$, then any non-permutation function in $M(\mathcal{B})$ has rank at most m.*

Proof. Since $|BPI(\mathcal{B})| = m \geq 1$, by Proposition 3.2, we have $|A| \geq 2$. Note that the permutation δ_a contributes one to the indegree of each state of $\mathcal{B}$.

For $x \in A^*$, let δ_x be a non-permutation function in $M(\mathcal{B})$. The nonempty word x contains at least one symbol of $A \setminus \{a\}$. We claim that $|Q\delta_b| \leq m$ for all $b \in A \setminus \{a\}$ and so that the rank of δ_x is at most m. On the contrary, let us assume that $|Q\delta_c| > m$ for some $c \in A \setminus \{a\}$. This implies that $|BPI(\mathcal{B})| > m$; a contradiction. Hence $|Q\delta_b| \leq m$ for all $b \in A \setminus \{a\}$. □

In view of Proposition 3.3, we have the following corollary of Proposition 3.4.

Corollary 3.2. *If $\mathcal{B}$ has a unique bpi, then $Q\delta_b = \{q_0\}$ for all $b \in A \setminus \{a\}$.*

3.2 Circular Semi-Flower Automata with at most one bpi

In this subsection, we obtain the holonomy decomposition of an SFA which is permutation automaton or has no bpis or circular automaton with a unique bpi. We first prove the following result.

Proposition 3.5. *Let $\mathcal{A}$ be an SFA. If $\mathcal{A}$ is permutation automaton or has no bpis, then $M(\mathcal{A})$ is a cyclic group.*

Proof.

Case ($\mathcal{A}$ is permutation): The monoid $M(\mathcal{A})$ is a group. All elements in $M(\mathcal{A})$ induced by words are permutation functions on Q. Note that $M(\mathcal{A})$ is generated by the functions induced by symbols. Also $\mathcal{A}$ is an SFA, all the permutations on Q induced by symbols are equal (cf. Proposition 3.1). Therefore $M(\mathcal{A})$ is a cyclic group.

Case ($\mathcal{A}$ has no bpis): Here $|A| = 1$ (cf. Proposition 3.2). Clearly the function induced by the symbol is a circular permutation on Q, and so $\mathcal{A}$ is permutation SFA. Therefore, by the previous case, the monoid $M(\mathcal{A})$ is a cyclic group.

□

Theorem 3.1. *Let $\mathcal{A}$ be an SFA. If $\mathcal{A}$ is permutation automaton or has no bpis or circular automaton with a unique bpi, then $(Q, M(\mathcal{A})) \prec \overline{\mathscr{C}_n}$.*

Proof.

Case ($\mathcal{A}$ is permutation or has no bpis): Here $M(\mathcal{A})$ is a group (cf. Proposition 3.5). Therefore $|Q\delta_x| = n$ for all $\delta_x \in M(\mathcal{A})$.

Case ($\mathcal{A}$ is circular with a unique bpi): Since $\mathcal{A}$ has a unique bpi, we have $Q\delta_b = \{q_0\}$ for all $b \in A \setminus \{a\}$ (cf. Corollary 3.2). This implies that $\delta_b = \delta_c$ for all $b, c \in A \setminus \{a\}$, and so that $M(\mathcal{A})$ is generated by the set $\{\delta_a, \delta_b\}$ of functions induced by the symbols a and b. For $\delta_x \in M(\mathcal{A})$, by Proposition 3.4, we have either $|Q\delta_x| = n$ or $|Q\delta_x| = 1$.

In all the cases, the skeleton of the transformation monoid $(Q, M(\mathcal{A}))$ is $\mathscr{J} = \{Q\} \cup \mathscr{J}_1$. Clearly $B(Q) = \mathscr{J}_1$ and so that $|B(Q)| = n$. Note that $K(Q) = \{\delta_{a^i} \mid 1 \leq i \leq n\}$. The holonomy group $G(Q)$ is

$$G(Q) = \{\check{\delta}_{a^i} \mid 1 \leq i \leq n\}.$$

For $1 \leq i \leq n$, since $\delta_{a^i} = (\delta_a)^i$, we have $\check{\delta}_{a^n} = (\check{\delta}_a)^n = \check{\delta}_\varepsilon$. The holonomy group $G(Q)$ is cyclic group of order n generated by $\check{\delta}_a$. Thus, in each case, the holonomy decomposition of $(Q, M(\mathcal{A}))$ is

$$(Q, M(\mathcal{A})) \prec \overline{\mathscr{C}_n}.$$

□

3.3 Circular Semi-Flower Automata with two bpis

In this subsection, we investigate the holonomy decomposition of CSFA with two bpis. Here $\mathcal{B} = (Q, A, \delta, q_0, q_0)$ denotes a CSFA with two bpis. Note that, by Proposition 3.2, we have $|A| \geq 2$. If $|Q| = 2$, then the holonomy decomposition of $\mathcal{B}$ follows directly from Theorem 3.1. Therefore, let us assume that $|Q| > 2$. By Proposition 3.3, the initial-final state q_0 of $\mathcal{B}$ is always a bpi. Let q_m, where $1 \leq m < n$, be the other bpi of $\mathcal{B}$ so that $BPI(\mathcal{B}) = \{q_0, q_m\}$. Note that there is only one symbol $a \in A$ which induces the permutation on Q.

Lemma 3.1. *Let $\mathcal{B} = (Q, A, \delta, q_0, q_0)$ be a CSFA with two bpis.*

(i) *For a symbol $b \in A$, if rank$(\delta_b) = 2$, then $Q\delta_b = BPI(\mathcal{B})$.*

(ii) *There exists a symbol $b \in A \setminus \{a\}$ such that $Q\delta_b = BPI(\mathcal{B})$.*

Proof. We note that δ_a contributes one to the indegree of each state of $\mathcal{B}$. Since $BPI(\mathcal{B}) = \{q_0, q_m\}$, we have $Q\delta_b \subseteq \{q_0, q_m\}$ for all $b \in A \setminus \{a\}$ (cf. Proposition 3.4).

(i) Straightforward from the above statement.

(ii) By Lemma 3.1(i), it is sufficient to prove that rank$(\delta_b) = 2$ for some $b \in A \setminus \{a\}$. On the contrary, let us assume that rank$(\delta_b) \neq 2$ for all $b \in A \setminus \{a\}$. Then rank$(\delta_b) = 1$ for all $b \in A \setminus \{a\}$ (cf. Proposition 3.4). This implies that either $Q\delta_b = \{q_0\}$ or $Q\delta_b = \{q_m\}$ for all $b \in A \setminus \{a\}$. If $Q\delta_b = \{q_m\}$ for some $b \in A \setminus \{a\}$, then there is a loop at q_m; which is not possible. Consequently $Q\delta_b = \{q_0\}$ for all $b \in A \setminus \{a\}$, and so that $BPI(\mathcal{B}) = \{q_0\}$; a contradiction. Hence rank$(\delta_b) = 2$ for some $b \in A \setminus \{a\}$.

□

The following lemma provides the skeleton of the transformation monoid $(Q, M(\mathcal{B}))$.

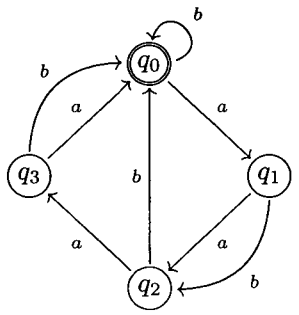


Figure 1: CSFA $\mathcal{B}_1$ with two bpis

Lemma 3.2. *Let $\mathcal{B}$ be a CSFA with $BPI(\mathcal{B}) = \{q_0, q_m\}$. Then the skeleton of transformation monoid $(Q, M(\mathcal{B}))$ is given by*

$$\mathscr{J} = \{Q\} \cup \mathscr{J}_2 \cup \mathscr{J}_1,$$

where $\mathscr{J}_2 = \big\{\{q_0, q_m\}\delta_{a^i} \mid 1 \le i \le n\big\}$.

Proof. In view of Proposition 3.4, other than Q and singletons, the skeleton $\mathscr{J}$ can have some sets of size two. Therefore it is sufficient to determine $\mathscr{J}_2$.

By Lemma 3.1(ii), there exists a symbol $b \in A \setminus \{a\}$ such that $Q\delta_b = \{q_0, q_m\}$. Therefore, for $1 \le i \le n$, the image set $Q\delta_{ba^i} = \{q_0, q_m\}\delta_{a^i} \in \mathscr{J}_2$, and so that

$$\big\{\{q_0, q_m\}\delta_{a^i} \mid 1 \le i \le n\big\} \subseteq \mathscr{J}_2.$$

Let us assume that $Q\delta_w \in \mathscr{J}_2$ for some nonempty word $w \in A^*$. Then w is of the form

$$w = a^{i_1}b_1a^{i_2}b_2 \cdots a^{i_k}b_ka^{i_{k+1}},$$

for $i_j \ge 0$ $(1 \le j \le k+1)$ and $b_t \in A$ $(1 \le t \le k)$ such that the rank of each function δ_{b_t} is two (cf. Proposition 3.4). Write $w = a^{i_1}b_1ub_ka^{i_{k+1}}$, where $u = a^{i_2}b_2 \cdots a^{i_k}$. Since $\mathrm{rank}(\delta_{b_1ub_k}) = \mathrm{rank}(\delta_{b_k}) = 2$, we have

$$Q\delta_{b_1ub_k} = Q\delta_{b_k} = \{q_0, q_m\},$$

by Lemma 3.1(i). Therefore $Q\delta_w = Q\delta_{a^{i_1}b_1ub_ka^{i_{k+1}}} = \{q_0, q_m\}\delta_{a^{i_{k+1}}}$, and consequently

$$\mathscr{J}_2 = \big\{\{q_0, q_m\}\delta_{a^i} \mid 1 \le i \le n\big\}.$$

□

Remark 3.1. As shown in Example 3.1, the cardinality of $\mathscr{J}_2$ is not necessarily n.

Example 3.1. The 4-state automaton $\mathcal{B}_1$ given in the Figure 1 is CSFA with $BPI(\mathcal{B}_1) = \{q_0, q_2\}$. We observe that

$$\{q_0, q_2\}\delta_a = \{q_1, q_3\}, \quad \{q_0, q_2\}\delta_{a^2} = \{q_0, q_2\}, \quad \text{and so that } |\mathscr{J}_2| = 2.$$

Lemma 3.3. *Let $\mathcal{B}$ be a CSFA with $BPI(\mathcal{B}) = \{q_0, q_m\}$. Then there is a nonempty word $x \in A^*$ such that $q_0\delta_x = q_m$ and $q_m\delta_x = q_0$.*

Proof. If there exists a symbol $b \in A \setminus \{a\}$ such that $q_0\delta_b \neq q_0$, then clearly the word $x = b$ will serve the purpose. Otherwise we have $q_0\delta_b = q_0$ for all $b \in A \setminus \{a\}$. However, by Lemma 3.1(ii), there exists a symbol $c \in A \setminus \{a\}$ such that $Q\delta_c = \{q_0, q_m\}$.

Note that the permutation δ_a induces the cyclic ordering $q_0, q_1, \ldots, q_{n-1}$ of the state set Q. Since $q_0\delta_c = q_0$ and the state q_m is the other bpi of $\mathcal{B}$, there exists a state q_i (with $1 \leq i < m$) such that $q_i\delta_c = q_m$. Let t (with $1 \leq t < m$) be the least integer such that $q_t\delta_c = q_m$. Choose the word $x = a^t c$ and observe that $q_0\delta_x = q_m$. We claim that $q_m\delta_x = q_0$.

On the contrary, let us assume that $q_m\delta_x \neq q_0$. Since $BPI(\mathcal{B}) = \{q_0, q_m\}$, we have $q_m\delta_x = q_m$ and so that there is a cycle in $\mathcal{B}$ from q_m to q_m labeled by the word x. Since $\mathcal{B}$ is SFA, the cycle must pass through q_0. Since $q_0\delta_c = q_0$, there exist t_1 and t_2 $(1 \leq t_1, t_2 < t)$ with $t_1 + t_2 = t$ such that

$$q_m\delta_{a^{t_1}} = q_0 \text{ and } q_0\delta_{a^{t_2}c} = q_m.$$

Note that $q_0\delta_{a^{t_2}c} = q_{t_2}\delta_c = q_m$. This contradicts the choice of t, as $t_2 < t$. Thus we have $q_m\delta_x = q_0$. □

Theorem 3.2. *If $\mathcal{B}$ is an n-state CSFA with $BPI(\mathcal{B}) = \{q_0, q_m\}$, then*

$$(Q, M(\mathcal{B})) \prec \overline{\mathscr{C}_2} \wr \overline{\mathscr{C}_r},$$

where r (with $1 < r \leq n$) is the smallest integer such that $\{q_0, q_m\}\delta_{a^r} = \{q_0, q_m\}$.

Further, if n is an odd number, then

$$(Q, M(\mathcal{B})) \prec \overline{\mathscr{C}_2} \wr \overline{\mathscr{C}_n}.$$

Proof. From Lemma 3.2, the skeleton of $(Q, M(\mathcal{B}))$ is given by

$$\mathscr{J} = \{Q\} \cup \mathscr{J}_2 \cup \mathscr{J}_1$$

in which all the elements of $\mathscr{J}_2$ are equivalent to each other.

For $1 \leq i \leq n$, note that δ_{a^i} permutes the elements of Q. Also, for $x \in A^*$, if $\delta_x \neq \delta_{a^i}$ for any i (with $1 \leq i \leq n$), then δ_x is not a permutation on Q (cf. Proposition 3.1). Consequently we have $K(Q) = \{\delta_{a^i} \mid 1 \leq i \leq n\}$. Since the elements of $\mathscr{J}_2$ are maximal in Q, we have $B(Q) = \mathscr{J}_2$. Let r (with $1 < r \leq n$) be the smallest integer such that $\{q_0, q_m\}\delta_{a^r} = \{q_0, q_m\}$ so that $|B(Q)| = r$. The holonomy group of Q is

$$G(Q) = \{\check{\delta}_{a^i} \mid 1 \leq i \leq r\}.$$

For $1 \leq i \leq r$, since $\delta_{a^i} = (\delta_a)^i$, we have $\check{\delta}_{a^r} = (\check{\delta}_a)^r = \check{\delta}_\varepsilon$. The holonomy group $G(Q)$ is cyclic group of order r generated by $\check{\delta}_a$, and so that $(B(Q), G(Q)) = \mathscr{C}_r$.

Let $P = \{q_0, q_m\}$ be representative in $\mathscr{J}_2$. Clearly $B(P) = \{\{q_0\}, \{q_m\}\}$. By Lemma 3.3, there exists a nonempty word $x \in A^*$ such that $q_0\delta_x = q_m$ and $q_m\delta_x = q_0$. This implies that $K(P) = \{\delta_x, \delta_\varepsilon\}$. Therefore the holonomy group $G(P)$ is cyclic group of order two generated by $\check{\delta}_x$, and so that $(B(P), G(P)) = \mathscr{C}_2$. Thus the holonomy decomposition of $(Q, M(\mathcal{B}))$ is

$$(Q, M(\mathcal{B})) \prec \overline{\mathscr{C}_2} \wr \overline{\mathscr{C}_r}.$$

If n is an odd number, we claim that $r = n$. On the contrary, let us assume that $r < n$. Since $\{q_0, q_m\}\delta_{a^r} = \{q_0, q_m\}$ and δ_a is circular permutation on Q. It follows that $q_0\delta_{a^r} = q_m$ and $q_m\delta_{a^r} = q_0$. This implies that $q_0\delta_{a^{2r}} = q_0$ and $q_m\delta_{a^{2r}} = q_m$ with $1 < 2r < 2n$. Therefore $2r = n$; which is a contradiction. Thus the holonomy decomposition of $(Q, M(\mathcal{B}))$ is

$$(Q, M(\mathcal{B})) \prec \overline{\mathscr{C}_2} \wr \overline{\mathscr{C}_n}.$$

□

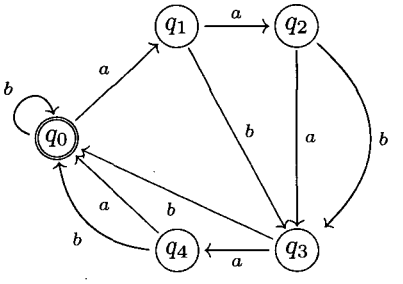


Figure 2: CSFA $\mathcal{B}_2$ with two bpis

Example 3.2. The 4-state automaton $\mathcal{B}_1$ given in the Figure 1 is CSFA with $BPI(\mathcal{B}_1) = \{q_0, q_2\}$. Using [18], we find the skeleton of $(Q, M(\mathcal{B}_1))$ as

$$\mathscr{J} = \{Q\} \cup \mathscr{J}_2 \cup \mathscr{J}_1,$$

where $\mathscr{J}_2 = \{\{q_0, q_2\}\delta_{a^i} \mid 1 \leq i \leq 4\}$. Clearly $B(Q) = \mathscr{J}_2$. The smallest integer r (with $1 < r \leq 4$) such that $\{q_0, q_2\}\delta_{a^r} = \{q_0, q_2\}$ is two, and therefore $|B(Q)| = 2$. The holonomy group $G(Q)$ is cyclic group of order two generated by $\check{\delta}_a$, and so that $(B(Q), G(Q)) = \mathscr{C}_2$.

We observe that the elements of $\mathscr{J}_2$ are equivalent to each other. Let $P = \{q_0, q_2\}$ be representative in $\mathscr{J}_2$. Clearly $B(P) = \{\{q_0\}, \{q_2\}\} \subseteq \mathscr{J}_1$. The holonomy group $G(P)$ is cyclic group of order two generated by $\check{\delta}_{ab}$, and so that $(B(P), G(P)) = \mathscr{C}_2$. Thus the holonomy decomposition of $(Q, M(\mathcal{B}_1))$ is

$$(Q, M(\mathcal{B}_1)) \prec \overline{\mathscr{C}_2} \wr \overline{\mathscr{C}_2}.$$

If the cardinality of the state set is an odd number, the following example illustrates Theorem 3.2.

Example 3.3. The 5-state automaton $\mathcal{B}_2$ given in the Figure 2 is CSFA with $BPI(\mathcal{B}_2) = \{q_0, q_3\}$. Using [18], we find the skeleton of $(Q, M(\mathcal{B}_2))$ as

$$\mathscr{J} = \{Q\} \cup \mathscr{J}_2 \cup \mathscr{J}_1,$$

where $\mathscr{J}_2 = \big\{\{q_0, q_3\}\delta_{a^i} \mid 1 \leq i \leq 5\big\}$. Clearly $B(Q) = \mathscr{J}_2$. The smallest integer r (with $1 < r \leq 5$) such that $\{q_0, q_3\}\delta_{a^r} = \{q_0, q_3\}$ is five, and therefore $|B(Q)| = 5$. The holonomy group $G(Q)$ is cyclic group of order five generated by $\check{\delta}_a$, and so that $(B(Q), G(Q)) = \mathscr{C}_5$.

We observe that the elements of $\mathscr{J}_2$ are equivalent to each other. Let $P = \{q_0, q_3\}$ be representative in $\mathscr{J}_2$. Clearly $B(P) = \big\{\{q_0\}, \{q_3\}\big\} \subseteq \mathscr{J}_1$. The holonomy group $G(P)$ is cyclic group of order two generated by $\check{\delta}_{ab}$, and so that $(B(P), G(P)) = \mathscr{C}_2$. Thus the holonomy decomposition of $(Q, M(\mathcal{B}_2))$ is

$$(Q, M(\mathcal{B}_2)) \prec \overline{\mathscr{C}_2} \wr \overline{\mathscr{C}_5}.$$

We conclude the paper by looking at two examples that exhibit that the study on the holonomy decomposition of CSFA with more than two bpis is much more complicated.

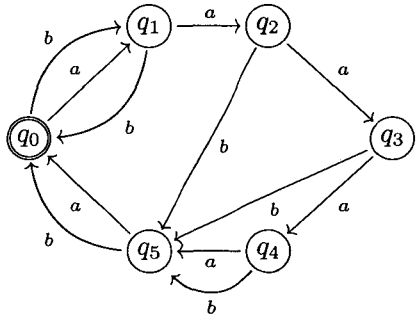


Figure 3: CSFA $\mathcal{B}_3$ with three bpis

Example 3.4. The 6-state automaton $\mathcal{B}_3$ given in the Figure 3 is CSFA with $BPI(\mathcal{B}_3) = \{q_0, q_1, q_5\}$. Using [18], we find the skeleton of $(Q, M(\mathcal{B}_3))$ as

$$\mathscr{J} = \{Q\} \cup \mathscr{J}_3 \cup \mathscr{J}_2 \cup \mathscr{J}_1,$$

where $\mathscr{J}_3 = \big\{\{q_0, q_1, q_5\}\delta_{a^i} \mid 1 \leq i \leq 6\big\}$, and $\mathscr{J}_2 = \big\{\{q_0, q_1, q_5\}\delta_{ba^i} \mid 1 \leq i \leq 6\big\}$. Clearly $B(Q) = \mathscr{J}_3$ and $|\mathscr{J}_3| = 6$. The holonomy group $G(Q)$ is cyclic group of order six generated by $\check{\delta}_a$, and so that $(B(Q), G(Q)) = \mathscr{C}_6$.

We observe that the elements of $\mathscr{J}_3$ are equivalent to each other. Let $P = \{q_0, q_1, q_5\}$ be representative in $\mathscr{J}_3$. Clearly $B(P) = \big\{\{q_0, q_1\}, \{q_0, q_5\}\big\} \subseteq \mathscr{J}_2$.

The holonomy group $G(P)$ is cyclic group of order two generated by $\check{\delta}_{ab}$, and so that $(B(P), G(P)) = \mathscr{C}_2$.

We further observe that all six elements of $\mathscr{J}_2$ are equivalent to each other. Let $T = \{q_0, q_1\}$ be representative in $\mathscr{J}_2$. Clearly $B(T) = \big\{\{q_0\}, \{q_1\}\big\} \subseteq \mathscr{J}_1$. The holonomy group $G(T)$ is cyclic group of order two generated by $\check{\delta}_b$, and so that $(B(T), G(T)) = \mathscr{C}_2$. Thus the holonomy decomposition of $(Q, M(\mathcal{B}_3))$ is

$$(Q, M(\mathcal{B}_3)) \prec \overline{\mathscr{C}_2} \wr \overline{\mathscr{C}_2} \wr \overline{\mathscr{C}_6}.$$

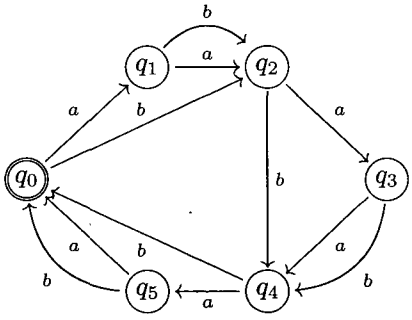


Figure 4: CSFA $\mathcal{B}_4$ with three bpis

Example 3.5. The 6-state automaton $\mathcal{B}_4$ given in the Figure 4 is CSFA with $BPI(\mathcal{B}_4) = \{q_0, q_2, q_4\}$. Using [18], we find the skeleton of $(Q, M(\mathcal{B}_4))$ as

$$\mathscr{J} = \{Q\} \cup \mathscr{J}_3 \cup \mathscr{J}_1,$$

where $\mathscr{J}_3 = \big\{\{q_0, q_2, q_4\}, \{q_1, q_3, q_5\}\big\}$. Clearly $B(Q) = \mathscr{J}_3$. The holonomy group $G(Q)$ is cyclic group of order two generated by $\check{\delta}_a$, and so that $(B(Q), G(Q)) = \mathscr{C}_2$.

We observe that the elements of $\mathscr{J}_3$ are equivalent to each other. Let $P = \{q_0, q_2, q_4\}$ be representative in $\mathscr{J}_3$. Clearly $B(P) = \big\{\{q_0\}, \{q_2\}, \{q_4\}\big\} \subseteq \mathscr{J}_1$. The holonomy group $G(P)$ is cyclic group of order three generated by $\check{\delta}_b$, and so that $(B(P), G(P)) = \mathscr{C}_3$. Thus the holonomy decomposition of $(Q, M(\mathcal{B}_4))$ is

$$(Q, M(\mathcal{B}_4)) \prec \overline{\mathscr{C}_3} \wr \overline{\mathscr{C}_2}.$$

4 Conclusion

In this paper we have initiated the investigations on the holonomy decomposition of circular semi-flower automata (CSFA) classified by their number of bpis. In fact, we have ascertained the holonomy decomposition of CSFA with at most two bpis. Our experiments for the holonomy decomposition of CSFA with more than two bpis over numerous examples exhibit that their structure is much more complicated.

However, we feel that the approach adopted in the paper may be useful to target the holonomy decomposition of CSFA having arbitrary number of bpis. In general, one can look for the holonomy decomposition of semi-flower automata.

Acknowledgements

The authors are very much thankful to anonymous referees for their valuable comments which improved the manuscript.

References

[1] Berstel, J. and Perrin, D. *Theory of codes*, volume 117 of *Pure and Applied Mathematics*. Academic Press Inc., 1985.

[2] Dömösi, P. and Nehaniv, C. L. *Algebraic theory of automata networks: An introduction*, volume 11 of *SIAM Monographs on Discrete Mathematics and Applications*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2005.

[3] Dubuc, L. Sur les automates circulaires et la conjecture de Černý. *RAIRO Inform. Théor. Appl.*, 32(1-3):21–34, 1998.

[4] Egri-Nagy, A. *Algebraic Hierarchical Decomposition of Finite State Automata–A Computational Approach*. PhD thesis, University of Hertfordshire, England, 2005.

[5] Egri-Nagy, A. and Nehaniv, C. L. Algebraic hierarchical decomposition of finite state automata: Comparison of implementations for Krohn-Rhodes theory. In *CIAA*, pages 315–316, 2004.

[6] Egri-Nagy, A. and Nehaniv, C. L. Cycle structure in automata and the holonomy decomposition. *Acta Cybernet.*, 17(2):199–211, 2005.

[7] Egri-Nagy, A. and Nehaniv, C. L. *SgpDec – software package for hierarchical coordinatization of groups and semigroups, implemented in the GAP computer algebra system*. https://github.com/gap-packages/sgpdec, 2010.

[8] Eilenberg, S. *Automata, languages, and machines. Vol. B*. Academic Press [Harcourt Brace Jovanovich Publishers], New York, 1976.

[9] Giambruno, L. *Automata-theoretic Methods in Free Monoids and Free Groups*. PhD thesis, Universit degli Studi di Palermo, Palermo, Italy, 2007.

[10] Giambruno, L. and Restivo, A. An automata-theoretic approach to the study of the intersection of two submonoids of a free monoid. *Theor. Inform. Appl.*, 42(3):503–524, 2008.

[11] Holcombe, M. Holonomy decompositions of near-rings. *Proc. Edinburgh Math. Soc. (2)*, 23(1):43–47, 1980.

[12] Krishna, K. V. and Chatterjee, N. Holonomy decomposition of seminearrings. *Southeast Asian Bull. Math.*, 31(6):1113–1122, 2007.

[13] Krohn, K. and Rhodes, J. Algebraic theory of machines. I. Prime decomposition theorem for finite semigroups and machines. *Trans. Amer. Math. Soc.*, 116:450–464, 1965.

[14] Pin, J.-E. Sur un cas particulier de la conjecture de Černý. In *Automata, languages and programming (Fifth Internat. Colloq., Udine, 1978)*, volume 62 of *Lecture Notes in Comput. Sci.*, pages 345–352. Springer, Berlin, 1978.

[15] Singh, S. N. *Semi-Flower Automata.* PhD thesis, IIT Guwahati, India, 2012.

[16] Singh, S. N. and Krishna, K. V. The rank and Hanna Neumann property of some submonoids of a free monoid. *Ann. Math. Inform.*, 40:113–123, 2012.

[17] Singh, S. N. and Krishna, K. V. A sufficient condition for the Hanna Neumann property of submonoids of a free monoid. *Semigroup Forum*, 86(3):537–554, 2013.

[18] The GAP Group. *GAP–Groups, Algorithms, and Programming, Version 4.8.4.* http://www.gap-system.org, 2016.

Received 10th November 2015

Acta Cybernetica **22** (2016) 807–822.

Bounds on the Stability Number of a Graph via the Inverse Theta Function

Miklós Ujvári*

Abstract

In the paper we consider degree, spectral, and semidefinite bounds on the stability number of a graph. The bounds are obtained via reformulations and variants of the inverse theta function, a notion recently introduced by the author in a previous work.

Keywords: *stability number, inverse theta number*

1 Introduction

In this paper we provide several new descriptions and variants of the inverse theta function, a notion recently introduced by the author (see [10]). We also present some applications in the stable set problem, bounds on the cardinality of a maximum stable set in a graph.

We start the paper with describing sandwich theorems on the inverse theta number and its predecessor, the theta number (see [4]). First we fix some notation. Let $n \in \mathcal{N}$, and let $G = (V(G), E(G))$ be an undirected graph, with vertex set $V(G) = \{1, \ldots, n\}$, and with edge set $E(G) \subseteq \{\{i,j\} : i \neq j\}$. Let $A(G)$ be the 0-1 adjacency matrix of the graph G, that is let

$$A(G) := (a_{ij}) \in \{0,1\}^{n \times n}, \text{ where } a_{ij} := \begin{cases} 0, & \text{if } \{i,j\} \notin E(G), \\ 1, & \text{if } \{i,j\} \in E(G). \end{cases}$$

The complementary graph $\overline{G}$ is the graph with adjacency matrix

$$A(\overline{G}) := J - I - A(G),$$

where I is the identity matrix, and J denotes the matrix with all elements equal to one. The disjoint union of the graphs G_1 and G_2 is the graph $G_1 + G_2$ with adjacency matrix

$$A(G_1 + G_2) := \begin{pmatrix} A(G_1) & 0 \\ 0 & A(G_2) \end{pmatrix}.$$

*H-2600 Vác, Szent János utca 1., Hungary. E-mail: ujvarim@cs.elte.hu

DOI: 10.14232/actacyb.22.4.2016.5

Let $(\delta_1, \ldots, \delta_n)$ be the sum of the row vectors of the adjacency matrix $A(G)$. The elements of this vector are the degrees of the vertices of the graph G. We define similarly the values $\overline{\delta}_1, \ldots, \overline{\delta}_n$ in the complementary graph $\overline{G}$ instead of G. Let Δ_G (resp. μ_G) be the maximum (resp. the arithmetic mean) of the degrees in the graph G. Note that

$$\mu_{\overline{G}} = n - 1 - \mu_G, \; \mu_{G_1+G_2} = \frac{n_1\mu_{G_1} + n_2\mu_{G_2}}{n_1 + n_2}. \tag{1}$$

By Rayleigh's theorem (see [7]) for a symmetric matrix $M = M^T \in \mathcal{R}^{n\times n}$ the minimum and maximum eigenvalue, λ_M resp. Λ_M, can be expressed as

$$\lambda_M = \min_{||u||=1} u^T M u, \; \Lambda_M = \max_{||u||=1} u^T M u.$$

By the Perron-Frobenius theorem (see [6]) for an elementwise nonnegative symmetric matrix $M = M^T \in \mathcal{R}_+^{n\times n}$ the maximum is attained for a nonnegative unit (eigen)vector: we have $\Lambda_M = u^T M u$ for some $u \in \mathcal{R}_+^n$, $u^T u = 1$. Furthermore, if $M = M^T \in \mathcal{R}_+^{n\times n}$, then $-\lambda_M \leq \Lambda_M$.

The maximum (resp. minimum) eigenvalue of the adjacency matrix $A(G)$ is denoted by Λ_G (resp. λ_G). By Exercise 11.14 in [5], we have

$$\mu_G, \sqrt{\Delta_G} \leq \Lambda_G \leq \Delta_G, \sqrt{\mu_G(n-1)}. \tag{2}$$

The set of the n by n real symmetric positive semidefinite matrices will be denoted by $\mathcal{S}_+^n$, that is

$$\mathcal{S}_+^n := \left\{M \in \mathcal{R}^{n\times n} : M = M^T, \, u^T M u \geq 0 \; (u \in \mathcal{R}^n)\right\}.$$

For example, the Laplacian matrix of the graph G,

$$L(G) := D_{\delta_1,\ldots,\delta_n} - A(G) \in \mathcal{S}_+^n.$$

(Here $D_{\delta_1,\ldots,\delta_n}$ denotes the diagonal matrix with diagonal elements $\delta_1, \ldots, \delta_n$.)

It is well-known (see [7]), that the following statements are equivalent for a symmetric matrix $M = (m_{ij}) \in \mathcal{R}^{n\times n}$: a) $M \in \mathcal{S}_+^n$; b) $\lambda_M \geq 0$; c) M is Gram matrix, that is $m_{ij} = v_i^T v_j$ $(i, j = 1, \ldots, n)$ for some vectors $v_1, \ldots, v_n$. Furthermore, by Lemma 2.1 in [9], the set $\mathcal{S}_+^n$ can be described as

$$\mathcal{S}_+^n = \left\{ \left(\frac{a_i^T a_j}{(a_i a_j^T)_{11}} - 1 \right)_{i,j=1}^n \;\middle|\; \begin{array}{l} m \in \mathcal{N}, \, a_i \in \mathcal{R}^m \; (1 \leq i \leq n) \\ a_i^T a_i = 1 \; (1 \leq i \leq n) \end{array} \right\}. \tag{3}$$

The stability number, $\alpha(G)$, is the maximum cardinality of the (so-called stable) sets $S \subseteq V(G)$ such that $\{i, j\} \subseteq S$ implies $\{i, j\} \notin E(G)$. The chromatic number, $\chi(G)$, is the minimum number of stable sets covering the vertex set $V(G)$.

Let us define an *orthonormal representation* of the graph G (shortly, o.r. of G) as a system of vectors $a_1, \ldots, a_n \in \mathcal{R}^m$ for some $m \in \mathcal{N}$, satisfying

$$a_i^T a_i = 1 \; (i = 1, \ldots, n), \; a_i^T a_j = 0 \; (\{i, j\} \in E(\overline{G})).$$

In the seminal paper [4] L. Lovász proved the following result, now popularly called *sandwich theorem*, see [2]:

$$\alpha(G) \leq \vartheta(G) \leq \chi(\overline{G}), \tag{4}$$

where $\vartheta(G)$ is the *Lovász number* of the graph G, defined as

$$\vartheta(G) := \inf \left\{ \max_{1 \leq i \leq n} \frac{1}{(a_i a_i^T)_{11}} : a_1, \ldots, a_n \text{ o.r. of } G \right\}.$$

The Lovász number has several equivalent descriptions, see [4]. For example, by (3) and standard semidefinite duality theory (see e.g. [8]), it is the common optimal value of the Slater-regular primal-dual semidefinite programs

$$(TP) \qquad \min \lambda, \left\{ \begin{array}{l} x_{ii} = \lambda - 1 \ (i \in V(G)), \\ x_{ij} = -1 \ (\{i,j\} \in E(\overline{G})), \\ X = (x_{ij}) \in \mathcal{S}_+^n, \ \lambda \in \mathcal{R} \end{array} \right.$$

and

$$(TD) \qquad \max \operatorname{tr}(JY), \left\{ \begin{array}{l} \operatorname{tr}(Y) = 1, \\ y_{ij} = 0 \ (\{i,j\} \in E(G)), \\ Y = (y_{ij}) \in \mathcal{S}_+^n. \end{array} \right.$$

(Here tr stands for trace.) Reformulating the program (TD), Lovász derived the following dual description of the theta number (Theorem 5 in [4]):

$$\vartheta(G) = \max \left\{ \sum_{i=1}^{n} (b_i b_i^T)_{11} : b_1, \ldots, b_n \text{ o.r. of } \overline{G} \right\}. \tag{5}$$

Analogously, the *inverse theta number*, $\iota(G)$, satisfies the *inverse sandwich inequalities*,

$$n^2/\vartheta(\overline{G}), (\alpha(G))^2 + n - \alpha(G) \leq \iota(G) \leq n\vartheta(G), \tag{6}$$

see [10], and (19) for an extension. Here the inverse theta number, defined as

$$\iota(G) := \inf \left\{ \sum_{i=1}^{n} \frac{1}{(a_i a_i^T)_{11}} : a_1, \ldots, a_n \text{ o.r. of } G \right\},$$

equals the common attained optimal value of the primal-dual semidefinite programs

$$(TP^-) \qquad \inf \operatorname{tr}(Z) + n, \ z_{ij} = -1 \ (\{i,j\} \in E(\overline{G})), \ Z = (z_{ij}) \in \mathcal{S}_+^n,$$

$$(TD^-) \qquad \sup \operatorname{tr}(JM), \left\{ \begin{array}{l} m_{ii} = 1 \ (i = 1, \ldots, n), \\ m_{ij} = 0 \ (\{i,j\} \in E(G)), \\ M = (m_{ij}) \in \mathcal{S}_+^n. \end{array} \right.$$

Moreover, rewriting the feasible solution M of the program (TD^-) as the Gram matrix $M = (b_i^T b_j)$ for some vectors $b_1, \ldots, b_n \in \mathcal{R}^m$, we obtain the following

analogue of (5):

$$\iota(G) = \max\left\{\sum_{i,j=1}^{n} b_i^T b_j : b_1, \ldots, b_n \text{ o.r. of } \overline{G}\right\}. \tag{7}$$

The structure of the paper is as follows: In Section 2 we will describe a refinement of (7) and also several new descriptions of the inverse theta function (with well-known analogues in the theory of the theta function). Some of these results will be applied in Section 3, where we present two new lower bounds for the stability number of a graph, and examine their additivity properties. Finally, in Section 4 we study three variants of the inverse theta function, and derive further bounds in the stable set problem.

2 New descriptions of $\iota(G)$

In this section we will describe three reformulations of the inverse theta number of a graph G. The results have analogues in the theory of the theta function, which we will mention in chronological order.

Let us denote by $\mathcal{A}_G$ the following set of matrices:

$$\mathcal{A}_G := \left\{ A = (a_{ij}) \in \mathcal{R}^{n\times n} \;\middle|\; \begin{array}{l} a_{ii} = 0 \ (i = 1, \ldots, n), \\ a_{ij} = 0 \ (\{i,j\} \in E(G)), \\ a_{ij} = a_{ji} \ (\{i,j\} \in E(\overline{G})) \end{array} \right\}.$$

We will describe bounds for the minimum eigenvalue λ_A with $A \in \mathcal{A}_G$.

First, we have for $A \in \mathcal{A}_G$ the lower bounds

$$\lambda_A \geq -\Lambda_{|A|} \geq -\Lambda_{\overline{G}} \cdot \max_{i,j} |a_{ij}|, \tag{8}$$

by Rayleigh's theorem and the Perron-Frobenius theorem. (Here $|A| \in \mathcal{R}^{n\times n}$ denotes the elementwise maximum of the matrices A and $-A$.)

On the other hand, using an equivalent form of the reformulation

$$\vartheta(G) = \max\left\{ \Lambda_M \;\middle|\; \begin{array}{l} m_{ii} = 1 \ (i = 1, \ldots, n), \\ m_{ij} = 0 \ (\{i,j\} \in E(G)), \\ M = (m_{ij}) \in \mathcal{S}_+^n \end{array} \right\},$$

(see for example [2], [10]), L. Lovász proved in Theorem 6 of [4] the upper bound

$$\lambda_A \leq \frac{\Lambda_A}{1 - \vartheta(G)} \ (A \in \mathcal{A}_G). \tag{9}$$

Analogously, as a consequence of the next theorem, we have also the upper bound

$$\lambda_A \leq \frac{\operatorname{tr}(JA)}{n - \iota(G)} \ (A \in \mathcal{A}_G). \tag{10}$$

(Note that by Rayleigh's theorem $\operatorname{tr}(JA) \leq n\Lambda_A$, and by the inverse sandwich theorem $\iota(G) - n \leq n(\vartheta(G) - 1)$ so there is no obvious dominance relation between the bounds in (9) and (10).)

Theorem 2.1. *The program*

$$(P_1): \qquad \sup n + \frac{\operatorname{tr}(JA)}{-\lambda_A},\ \{A \in \mathcal{A}_G$$

has attained optimal value $\iota(G)$.

Proof. The variable transformations

$$M_A := I + \frac{1}{-\lambda_A} A,\ A_M := M - I$$

show that programs (TD^-) and (P_1) are equivalent: if A and M are feasible solutions of (P_1) and (TD^-), respectively, then M_A and A_M are feasible solutions of the other program such that between the corresponding values the inequalities

$$\operatorname{tr}(JM_A) \geq n + \frac{\operatorname{tr}(JA)}{-\lambda_A},\ n + \frac{\operatorname{tr}(JA_M)}{-\lambda_{A_M}} \geq \operatorname{tr}(JM)$$

hold. Hence, the two programs have the same (attained) optimal value. □

A different approach leads to another description of the inverse theta number.

Karger, Motwani, and Sudan proved the reformulation

$$\frac{1}{1 - \vartheta(G)} = \min \left\{ \nu \;\middle|\; \begin{array}{l} n_{ii} = 1\ (i = 1, \ldots, n), \\ n_{ij} = \nu\ (\{i,j\} \in E(\overline{G})), \\ N = (n_{ij}) \in \mathcal{S}_+^n,\ \nu \in \mathcal{R} \end{array} \right\},$$

and used a variant of this theorem in their graph colouring algorithm. (See [3] for a summary of related results.) By the inverse sandwich theorem we have the lower bound

$$\frac{1}{1 - \vartheta(G)} \geq \frac{n}{n - \iota(G)}; \tag{11}$$

we will show that this latter value can be obtained as the optimal value of a semidefinite program, too.

Let us consider the primal-dual semidefinite programs

$$(P_2): \qquad \sup -\operatorname{tr} B,\ \left\{ \begin{array}{l} b_{11} - b_{ii} = 0\ (i = 2, \ldots, n), \\ b_{ij} = 0\ (\{i,j\} \in E(G)), \\ \operatorname{tr}((J - I)B) = 1,\ B = (b_{ij}) \in \mathcal{S}_+^n, \end{array} \right.$$

$$(D_2): \qquad \inf \gamma,\ \left\{ \begin{array}{l} \operatorname{tr} C = n, \\ c_{ij} = \gamma\ (\{i,j\} \in E(\overline{G})), \\ C = (c_{ij}) \in \mathcal{S}_+^n,\ \gamma \in \mathcal{R}. \end{array} \right.$$

The programs have common attained optimal value by standard semidefinite duality theory, see for example [8].

Theorem 2.2. *The programs* (P_2) *and* (D_2) *have (common attained) optimal value* $n/(n-\iota(G))$.

Proof. Similarly as in the proof of Theorem 2.1, the variable transformations

$$M_B := \frac{n}{\operatorname{tr} B} B,\ B_M := \frac{1}{\operatorname{tr}(JM) - n} M$$

show the equivalence of programs (P_2) and $n/(n-(TD^-))$, where the latter program can be obtained from (TD^-) formally exchanging its value function $\operatorname{tr}(JM)$ for $n/(n-\operatorname{tr}(JM))$ and adding the extra constraint $\operatorname{tr}(JM) > n$. □

It is left to the reader to prove that the program

$$\inf \frac{1}{1-\Lambda_R},\ \begin{cases} \operatorname{tr} R = n, \\ r_{ij} = 1\ (\{i,j\} \in E(\overline{G})) \\ R = (r_{ij}) = (r_{ji}) \in \mathcal{R}^{n\times n} \end{cases}$$

is equivalent with both (D_2) and $n/(n-(TP^-))$.

Now, we turn to the third description of the inverse theta number.

We will use the following lemma, a slight modification of (7).

Lemma 2.1. *For any graph* G,

$$\iota(G) = \sup \left\{ \sum_{i,j=1}^{n} \hat{b}_i^T \hat{b}_j \;\middle|\; \begin{array}{l} \hat{b}_1, \ldots, \hat{b}_n \text{ o.r. of } \overline{G} \\ (\hat{b}_i)_1 = e_1^T \hat{b}_i > 0 \text{ for } i = 1, \ldots, n \end{array} \right\},$$

with e_1 *denoting the vector* $(1, 0, \ldots, 0)^T$.

Proof. Let (b_i) be an orthonormal representation of $\overline{G}$ such that

$$\iota(G) = \sum_{i,j=1}^{n} b_i^T b_j$$

(that is an optimal solution in (7)). For $0 < \varepsilon < 1$, let us define an orthonormal representation $(\hat{b}_i(\varepsilon))$ of $\overline{G}$ the following way:

$$(\hat{b}_i(\varepsilon)) := \begin{pmatrix} \sqrt{1-\varepsilon^2} \cdot O \\ \varepsilon b_1, \ldots, \varepsilon b_n \end{pmatrix},$$

where $O \in \mathcal{R}^{n\times n}$ is an orthogonal matrix satisfying $e_1^T O > 0$. Note that then $e_1^T \hat{b}_i(\varepsilon) > 0$ holds for all i. On the other hand, it can easily be verified that

$$\sum_{i,j=1}^{n} \hat{b}_i^T(\varepsilon) \hat{b}_j(\varepsilon) \to \iota(G)\ (\varepsilon \to 1).$$

Hence, we have proved

$$\iota(G) \leq \sup \left\{ \sum_{i,j=1}^{n} \hat{b}_i^T \hat{b}_j \;\middle|\; \begin{array}{l} \hat{b}_1, \ldots, \hat{b}_n \text{ o.r. of } \overline{G} \\ e_1^T \hat{b}_i > 0 \text{ for } i = 1, \ldots, n \end{array} \right\},$$

which is the nontrivial part of the lemma. □

Applying the variable transformation described in (3) to the program in Lemma 2.1, as an immediate consequence we obtain an analogue of Theorem 2.2 in [9].

Theorem 2.3. *The optimal value of the program*

$$(P_3): \qquad \sup \sum_{i,j=1}^{n} \frac{d_{ij}+1}{\sqrt{(d_{ii}+1)\cdot(d_{jj}+1)}}, \left\{ \begin{array}{l} d_{ij} = -1 \ (\{i,j\} \in E(G)), \\ D = (d_{ij}) \in \mathcal{S}_+^n \end{array} \right.$$

equals $\iota(G)$. □

We will apply Theorem 2.3 in the next section for obtaining lower bounds in the stable set problem.

3 Lower bounds on $\alpha(G)$

In this section we will describe two lower bounds on the stability number of a graph G, and examine their additivity properties.

Note that the

$$Z_1 := L(\overline{G}),\ Z_2 := \Lambda_{\overline{G}} I - A(\overline{G})$$

feasible solutions in (TP^-) give the inequalities

$$\sqrt{\iota(G)} \leq \sqrt{n(\mu_{\overline{G}}+1)}, \sqrt{n(\Lambda_{\overline{G}}+1)}. \tag{12}$$

By Exercises 11.20 and 11.14 in [5], we have

$$\chi(G) \leq \Lambda_G + 1 \leq \sqrt{\mu_G(n-1)} + 1,\ \mu_G \leq \Lambda_G.$$

On the other hand, easy calculation verifies

$$\sqrt{\mu_G(n-1)} + 1 \leq \sqrt{n(\mu_G+1)}.$$

Hence, we have besides (12) also

$$\chi(\overline{G}) \leq \sqrt{n(\mu_{\overline{G}}+1)} \leq \sqrt{n(\Lambda_{\overline{G}}+1)}. \tag{13}$$

On the dual side instead of $\sqrt{\iota(\overline{G})}, \chi(\overline{G})$ we can approximate $\iota(G)/n, \alpha(G)$. Note that

$$D_1 := L(G),\ D_2 := \Lambda_G I - A(G)$$

are feasible solutions of the program (P_3) in Theorem 2.3. This fact implies the version of the following theorem, where $\alpha(G)$ is exchanged for $\iota(G)/n$. (For analogous results with $\vartheta(G)$, see [9].)

Theorem 3.1. *For any graph G,*

a)

$$\alpha'(G) := 1 + \sum_{\{i,j\}\in E(\overline{G})} \frac{2/n}{\sqrt{(\delta_i+1)\cdot(\delta_j+1)}} \leq \alpha(G);$$

b)

$$\alpha''(G) := 1 + \frac{\mu_{\overline{G}}}{\Lambda_G + 1} \leq \alpha(G).$$

Proof. By Exercise 11.14 in [5] we have $\mu_G \leq \Lambda_G$. Using this relation it is immediate that

$$\frac{n}{\Lambda_G+1} \leq \alpha''(G) \leq \frac{n}{\mu_G+1}. \tag{14}$$

We will show that the inequalities

$$\frac{n}{\mu_G+1} \leq \alpha'(G) \leq \sum_{i=1}^{n} \frac{1}{\delta_i+1} \tag{15}$$

hold also, from which the theorem follows, as

$$\sum_{i=1}^{n} \frac{1}{\delta_i+1} \leq \alpha(G) \tag{16}$$

by the Caro-Wei theorem (see [1], or for another proof [9]).

First, using the obvious inequality

$$\frac{2}{\sqrt{\delta_i+1}\cdot\sqrt{\delta_j+1}} \leq \frac{1}{\delta_i+1} + \frac{1}{\delta_j+1}, \tag{17}$$

we obtain

$$\begin{aligned} \alpha'(G) &\leq 1 + \frac{1}{n}\cdot\sum_{i=1}^{n} \frac{1}{\delta_i+1}\cdot(n-1-\delta_i) \\ &= \sum_{i=1}^{n} \frac{1}{\delta_i+1}. \end{aligned}$$

On the other hand, we will verify the relation

$$\alpha'(G) \geq \frac{n}{\mu_G+1}. \tag{18}$$

Using the arithmetic mean-harmonic mean inequality, it is easy to show that

$$\begin{aligned} \alpha'(G) &\geq 1 + \frac{4}{n}\cdot\sum_{\{i,j\}\in E(\overline{G})} \frac{1}{\delta_i+1+\delta_j+1} \\ &\geq 1 + \frac{1}{n}(n\mu_{\overline{G}})^2 \Big/ \sum_{\{i,j\}\in E(\overline{G})} (\delta_i+1+\delta_j+1)\,. \end{aligned}$$

Hence, to prove (18), it is enough to verify that

$$n\mu_{\overline{G}}(\mu_G + 1) \geq \sum_{\{i,j\} \in E(\overline{G})} (\delta_i + 1 + \delta_j + 1)$$

holds. This inequality can be rewritten as

$$\sum_{i=1}^{n}(n - 1 - \delta_i) \cdot \sum_{i=1}^{n}(\delta_i + 1) \geq n \cdot \sum_{i=1}^{n}(\delta_i + 1)(n - 1 - \delta_i),$$

and thus is a consequence of the Cauchy-Schwarz inequality. The proof of (18) is complete, as well. □

The following theorem describes additivity properties of the bounds α', α''. (For additivity properties of $\vartheta(G)$, see Sections 18, 19 in [2].)

Theorem 3.2. *With the lower bounds $\ell = \alpha', \alpha''$ we have*
a) $\ell(G_1 + G_2) \leq \ell(G_1) + \ell(G_2)$,
b) $\ell(\overline{G_1 + G_2}) \leq \max\{\ell(\overline{G_1}), \ell(\overline{G_2})\}$,
for any graphs G_1, G_2.

Proof. Case 1: $\ell = \alpha'$. a) Rewriting the statement, we have to verify

$$\sum_{i \in V(G_1),\, j \in V(G_2)} \frac{2}{\sqrt{(\delta_i + 1)(\delta_j + 1)}} \leq \alpha'(G_1)n_2 + \alpha'(G_2)n_1,$$

that is (without loss of generality assuming $G_1 = G_2 = G$)

$$\left(\sum_{i=1}^{n} \frac{1}{\sqrt{\delta_i + 1}}\right)^2 \leq \alpha'(G)n.$$

In other words, we have to prove the inequality

$$\sum_{i=1}^{n} \frac{1}{\delta_i + 1} + \sum_{\{i,j\} \in E(G)} \frac{2}{\sqrt{(\delta_i + 1)(\delta_j + 1)}} \leq n,$$

which follows immediately applying (17).

b) is obvious, as

$$\begin{aligned} \alpha'(\overline{G_1 + G_2}) &\leq \frac{\alpha'(\overline{G_1})n_1 + \alpha'(\overline{G_2})n_2}{n_1 + n_2} \\ &\leq \max\{\alpha'(\overline{G_1}), \alpha'(\overline{G_2})\} \end{aligned}$$

hold.

Case 2: $\ell = \alpha''$. By Rayleigh's theorem the formulas

$$\begin{aligned} \Lambda_{G_1+G_2} &\geq \max\{\Lambda_{G_1}, \Lambda_{G_2}\}, \\ \Lambda_{\overline{G_1+G_2}} &\geq \max\{\Lambda_{\overline{G_1}}, \Lambda_{\overline{G_2}}\} \end{aligned}$$

hold. The statements a) and b), respectively, are straightforward consequences of these inequalities, after applying (1): For example, a) can be reduced this way to the inequality

$$\frac{n_2\mu_{G_1} + n_1\mu_{G_2}}{n_1+n_2} \leq \max\{\Lambda_{G_1}, \Lambda_{G_2}\},$$

which holds true, as $\mu_G \leq \Lambda_G$ for any graph G, by Exercise 11.14 in [5]. □

Additivity properties of a lower bound on the stability number can be applied for strengthening the bound if the given graph or its complementer is not connected. In fact, if

$$G = G_1 + G_2 \text{ (or } \overline{G} = H_1 + H_2)$$

with some graphs G_1, G_2 (H_1, H_2), then $\alpha(G)$ is equal to

$$\alpha(G_1) + \alpha(G_2) \ (\max\{\alpha(\overline{H_1}), \alpha(\overline{H_2})\}).$$

Hence, both $\ell(G)$ and the, by additivity stronger, bound

$$\ell(G_1) + \ell(G_2) \ (\max\{\ell(\overline{H_1}), \ell(\overline{H_2})\})$$

are lower bounds on $\alpha(G)$.

It is left to the reader to adapt this bound-strengthening method to upper bounds $u(G)$ on the chromatic number $\chi(G)$.

Summarizing, the so-called *weak sandwich theorems* (see [9])

$$\ell(G) \leq \alpha(G) \ \chi(G) \leq u(G)$$

involve the bounds

$$\ell(G) = \alpha'(G), \alpha''(G), \ u(G) := \sqrt{n(\mu_G+1)}, \sqrt{n(\Lambda_G+1)}$$

in inverse theta number theory. In the next section we turn to the inverse sandwich theorem and its strengthened version.

4 Upper bounds on $\alpha(G)$

In this section we introduce three variants of the inverse theta number. They constitute bounds for the stability numbers of G and $\overline{G}$.

First, let us derive a bound from the original version $\iota(G)$. Let $S \subseteq V(G)$ be a stable set with cardinality $\#S = \alpha(G)$, and let $\varepsilon > 0$. Let us define the matrix $Z = Z(\varepsilon) \in \mathcal{R}^{n \times n}$ the following way: let $Z := (z_{ij})$, where

$$z_{ij} := \begin{cases} \varepsilon(n - \#S) + 0, & \text{if } i,j \in S, \\ 1/\varepsilon + (n - \#S - 1), & \text{if } i = j \notin S, \\ 0 + (-1), & \text{if } i,j \notin S,\ i \neq j, \\ (-1) + 0 & \text{otherwise.} \end{cases}$$

It can easily be verified using Schur complements (see [6]) that $Z \in \mathcal{S}_+^n$. (This statement holds even without adding the second terms in the definition of the elements z_{ij}.) For $\varepsilon = 1/\sqrt{\#S}$ the value of Z in (TP^-) satisfies

$$\operatorname{tr}(Z) + n = \left(n - \alpha(G) + \sqrt{\alpha(G)}\right)^2.$$

As this value is at least $\iota(\overline{G})$, so we obtained

Proposition 4.1. *For any graph G, we have*

$$\iota(\overline{G}) \leq \left(n - \alpha(G) + \sqrt{\alpha(G)}\right)^2, \tag{19}$$

in other words

$$\alpha(G) \leq \frac{1}{4}\left(1 + \sqrt{1 + 4\left(n - \sqrt{\iota(\overline{G})}\right)}\right)^2 \tag{20}$$

holds. □

We remark that the upper bound in (20) is between the values

$$n + 1 - \sqrt{\iota(\overline{G})}, n + 1 - \frac{\iota(\overline{G})}{n}$$

as it can easily be verified.

Proposition 4.1 allows a strengthening: a ι, ι^+ exchange in (20), where we add the $z_{ij} \geq -1$ constraints in (TP^-). Let us denote by $\iota^+(G)$ the common attained optimal value of the Slater regular primal-dual semidefinite programs

$$(P^+): \quad \inf n + \operatorname{tr} Z^+, \begin{cases} z_{ij}^+ = -1 \ (\{i,j\} \in E(\overline{G})), \\ z_{ij}^+ \geq -1 \ (\{i,j\} \in E(G)), \\ Z^+ = (z_{ij}^+) \in \mathcal{S}_+^n, \end{cases}$$

$$(D^+): \quad \sup \operatorname{tr}(JM^+), \begin{cases} m_{ii}^+ = 1 \ (i = 1, \ldots, n), \\ m_{ij}^+ \leq 0 \ (\{i,j\} \in E(G)), \\ M^+ = (m_{ij}^+) \in \mathcal{S}_+^n. \end{cases}$$

(Standard semidefinite duality theory can be found for example in [8].)

Theorem 4.1. *For any graph G,*

$$\alpha(G) \leq \frac{1}{4}\left(1+\sqrt{1+4\left(n-\sqrt{\iota^+(\overline{G})}\right)}\right)^2 \tag{21}$$

holds. □

For an analogue in theta function theory, see the results of Szegedy and Meurdesoif concerning the variant

$$\vartheta^+(G) := \sup\left\{ \operatorname{tr}(JY^+) \;\middle|\; \begin{array}{l} \operatorname{tr} Y^+ = 1, \\ y^+_{ij} \leq 0 \ (\{i,j\} \in E(G)), \\ Y^+ = (y^+_{ij}) \in \mathcal{S}^n_+ \end{array} \right\},$$

the relations $\vartheta(G) \leq \vartheta^+(G) \leq \chi(\overline{G})$, e.g. in [3].

Now, we turn to the lower variants of $\iota(G)$. Let us consider the primal-dual semidefinite programs

$$(P'): \quad \inf n + \operatorname{tr} Z', \left\{ \begin{array}{l} z'_{ij} \leq -1 \ (\{i,j\} \in E(\overline{G})), \\ Z' = (z'_{ij}) \in \mathcal{S}^n_+, \end{array} \right.$$

$$(D'): \quad \sup \operatorname{tr}(JM'), \left\{ \begin{array}{l} m'_{ii} = 1 \ (i = 1, \ldots, n), \\ m'_{ij} = 0 \ (\{i,j\} \in E(G)), \\ M' = (m'_{ij}) \in \mathcal{S}^n_+ \cap \mathcal{R}^{n\times n}_+. \end{array} \right.$$

The programs have common attained optimal value by standard semidefinite duality theory (see for example [8]), we will denote this value by $\iota'(G)$.

Obviously, $\iota'(G) \leq n\vartheta'(G)$, where $\vartheta'(G)$ is a sharpening of the theta number, due to McEliece, Rodemich, Rumsey, and Schrijver ($\alpha(G) \leq \vartheta'(G) \leq \vartheta(G)$, see for example [3]), defined as

$$\vartheta'(G) := \sup\left\{ \operatorname{tr}(JY') \;\middle|\; \begin{array}{l} \operatorname{tr} Y' = 1, \\ y'_{ij} = 0 \ (\{i,j\} \in E(G)), \\ Y' = (y'_{ij}) \in \mathcal{S}^n_+ \cap \mathcal{R}^{n\times n}_+ \end{array} \right\}.$$

Besides the mentioned relations

$$\vartheta'(G) \geq \iota'(G)/n, \ \alpha(G), \tag{22}$$

we have also

$$\frac{1}{2}\left(1+\sqrt{4(\iota'(G)-n)+1}\right) \geq \iota'(G)/n, \ \alpha(G) \tag{23}$$

as the following theorem shows. (For analogous results with $\iota(G)$, see [10].)

Theorem 4.2. *For any graph G, we have*

$$\iota'(G) \geq \alpha(G)^2 + n - \alpha(G),$$

in other words

$$\alpha(G) \leq \frac{1}{2}\left(1+\sqrt{4(\iota'(G)-n)+1}\right)$$

holds.

Proof. Let S be a stable set in G with cardinality $\#S = \alpha(G)$. Let us define the matrix $M' := (m'_{ij}) \in \mathcal{R}^{n\times n}$ the following way: let $m'_{ij} := 1$ if $i, j \in S$ or $i = j$, and let $m'_{ij} := 0$ otherwise. Then, the matrix M' is a feasible solution of the program (D') with corresponding value

$$(\#S)^2 + n - (\#S) \leq \iota'(G).$$

Hence, the statement follows. □

The bound in Theorem 4.2 implies

$$\alpha(G) \leq \sqrt{\iota'(G)}, \tag{24}$$

and also, by $\iota'(G) \leq \iota(G)$, the relations

$$\alpha(G) \leq \frac{1}{2}\left(1 + \sqrt{4(\iota(G) - n) + 1}\right) \leq \sqrt{\iota(G)} \tag{25}$$

from [10]. It is an open problem whether any of these bounds can be less than $\vartheta(G)$ or even $\vartheta'(G)$ for some graphs.

We mention a related result, see also Theorems 3 and 6 in [4] and Proposition 2.1 in [10], where the bounds in (26) appear as lower and upper bounds for $\vartheta(\overline{G})$ and $\sqrt{\iota(\overline{G})}$, respectively.

Proposition 4.2. *For any graph G, the inequalities*

$$1 + \frac{\Lambda_G}{-\lambda_G} \leq \sqrt{n\left(1 + \frac{\mu_G}{-\lambda_G}\right)}, \quad \Lambda_G + 1 \leq \sqrt{n(\mu_G + 1)} \tag{26}$$

hold.

Proof. By (2), it suffices to prove (26) after substituting μ_G with $\Lambda_G^2/(n-1)$. Then, the first inequality follows by

$$\frac{1}{2}\left(\frac{n\Lambda_G}{n-1} - 2 + \sqrt{\left(\frac{n\Lambda_G}{n-1} - 2\right)^2 + 4(n-1)}\right) \geq \frac{\Lambda_G}{-\lambda_G}$$

(note that $-\lambda_G \geq 1$ and $\Lambda_G \leq n - 1$), the second inequality is immediate. This finishes the proof. □

Finally, we mention another variant of the inverse theta number, which leads to an interesting weak sandwich theorem.

Let us define $\iota''(G)$ as the common attained optimal value of the primal-dual semidefinite programs

$$(P'') : \quad \inf \operatorname{tr}(JM''), \begin{cases} m''_{ii} = 1 \ (i = 1, \ldots, n), \\ m''_{ij} = 0 \ (\{i, j\} \in E(G)), \\ M'' = (m''_{ij}) \in \mathcal{S}^n_+, \end{cases}$$

$$(D'') : \quad \sup n - \operatorname{tr} Z'', \begin{cases} z''_{ij} = 1 \ (\{i, j\} \in E(\overline{G})), \\ Z'' = (z''_{ij}) \in \mathcal{S}^n_+. \end{cases}$$

(See, for example, [8] for standard semidefinite duality theory.)

Theorem 4.3. *For any graph G, the inequalities*
a) $\iota''(G) \leq \alpha(\overline{G})$,
b) $\iota''(G) \leq n - \chi(\overline{G})$
hold.

Proof. a) Let us introduce the notation

$$M_S := (m_{ij}) \in \mathcal{R}^{n\times n}, \text{ where } m_{ij} := \begin{cases} 1 & \text{if } i,j \in S,\ i = j, \\ -\frac{1}{\#S-1} & \text{if } i,j \in S,\ i \neq j, \\ 0 & \text{otherwise,} \end{cases}$$

for $S \subseteq V(G)$.

Let $S_1, \ldots, S_k$ be a stable set partition of $V(G)$ such that the cardinality of the index set $\{i : \#S_i \geq 2\}$ is maximal. Then,

$$\overline{S} := \cup_{i=1}^{k} \{S_i : \#S_i = 1\}$$

is a stable set in $\overline{G}$. Furthermore, the matrix

$$\sum_{i=1}^{k} M_{S_i}$$

is feasible in (P'') with corresponding value $\#\overline{S} \leq \alpha(\overline{G})$, which completes the proof of statement a).

b) Let $\overline{S}_1, \ldots, \overline{S}_\ell$ be disjoint stable sets in $\overline{G}$ covering the vertex set $V(G)$, where $\ell := \chi(\overline{G})$. Then, there exist non-edges $\overline{e}_{pq} \in E(\overline{G})$ between $\overline{S}_p$ and $\overline{S}_q$ for each $1 \leq p < q \leq \ell$. Let us define a symmetric matrix $M \in \mathcal{R}^{n\times n}$ by writing in it: 1 on diagonal positions, $-1/(\ell - 1)$ on the positions corresponding to $\overline{e}_{pq}$, and 0 otherwise. By Gerschgorin's disc theorem (see [7]) the matrix M is positive semidefinite, a feasible solution of the program (P'') with corresponding value $n - \chi(\overline{G})$. This finishes the proof of statement b), too. □

Summarizing, in this section we obtained the

$$(\alpha(G))^2 + n - \alpha(G) \leq \iota'(G)$$
$$\iota'(G) \leq \iota(G) \leq \iota^+(G)$$
$$\iota^+(G) \leq \left(n - \alpha(\overline{G}) + \sqrt{\alpha(\overline{G})}\right)^2$$

inverse sandwich theorem as an analogue of Lovász's sandwich theorem.

In the same context we mention also the well-known

$$\chi(\overline{G}) \leq n - \nu(G) \leq \frac{n + \alpha(G)}{2} \tag{27}$$

sandwich theorem, where $\nu(G)$ denotes the matching number of G, that is the largest number of pairwise disjoint edges in $E(G)$, see Section 7 in [5]. This fact, together with the formulas

$$\alpha(G) \cdot \chi(G) \geq n,\ \chi(G) + \chi(\overline{G}) \leq n + 1 \tag{28}$$

(see Exercise 9.5 in [5]), makes upper bounds for $\alpha(G)$ particularly useful in deriving other (upper and lower) bounds for $\alpha(G), \chi(G)$, for example

$$\alpha(G) \geq 2\vartheta(G) - n,\ \frac{n}{n + 1 - \vartheta(G)} \tag{29}$$

and

$$\chi(G) \leq n + 1 - \vartheta(G),\ \frac{n + \vartheta(\overline{G})}{2} \tag{30}$$

via the sandwich theorem.

5 Conclusion

In the paper we studied the inverse theta function: results analogous to sandwich theorems and their strengthened versions from the theory of Lovász's theta number were derived, based on new descriptions of the inverse theta number. Whether the new bounds on the stability number can be tighter than already known ones remained a partly undecided question.

Acknowledgements

I thank one of the anonymous referees of the paper for calling my attention to several relevant references.

References

[1] Alon, N. and Spencer, J. *The Probabilistic Method.* 4th edition, The probabilistic lens after Chapter 6, Wiley, page 100, Hoboken NJ, 2016.

[2] Knuth, D. The sandwich theorem. *Electronic Journal of Combinatorics*, 1:1–48, 1994.

[3] Laurent, M. and Rendl, F. Semidefinite programming and integer programming. In: Aardal, K. et al., editors., *Handbook on Discrete Optimization*, Elsevier B.V., Amsterdam, pages 393–514, 2005.

[4] Lovász, L. On the Shannon capacity of a graph. *IEEE Transactions on Information Theory*, IT-25(1):1–7, 1979.

[5] Lovász, L. *Combinatorial Problems and Exercises.* Corrected reprint of the 1993 second edition, AMS Chelsea Publishing, Providence, RI, 2007.

[6] Serre, D. *Matrices, Theory and Applications.* Translated from the 2001 French original, Graduate Texts in Mathematics vol. 216, Springer-Verlag, New York, 2002.

[7] Strang, G. *Linear Algebra and its Applications.* Academic Press, New York, 1980.

[8] Ujvári, M. A note on the graph-bisection problem. *Pure Mathematics and Applications* 12(1):119–130, 2002.

[9] Ujvári, M. New descriptions of the Lovász number, and the weak sandwich theorem. *Acta Cybernetica*, 20(4):499–513, 2012.

[10] Ujvári, M. Applications of the inverse theta number in stable set problems. *Acta Cybernetica*, 21(3):481–494, 2014.

Received 26th October 2014

Acta Cybernetica **22** (2016) 823–832.

An Estimation of the Size of Non-Compact Suffix Trees

Bálint Vásárhelyi*

Abstract

A suffix tree is a data structure used mainly for pattern matching. It is known that the space complexity of simple suffix trees is quadratic in the length of the string. By a slight modification of the simple suffix trees one gets the compact suffix trees, which have linear space complexity. The motivation of this paper is the question whether the space complexity of simple suffix trees is quadratic not only in the worst case, but also in expectation.

1 Introduction

A suffix tree is a powerful data structure which is used for a large number of combinatorial problems involving strings. Suffix tree is a structure for compact storage of the suffixes of a given string. The compact suffix tree is a modified version of the suffix tree, and it can be stored in linear space of the length of the string, while the non-compact suffix tree is quadratic (see [11, 14, 18, 19]).

The notion of suffix trees was first introduced by Weiner [19], though he used the name compacted bi-tree. Grossi and Italiano mention that in the scientific literature, suffix trees have been rediscovered many times, sometimes under different names, like compacted bi-tree, prefix tree, PAT tree, position tree, repetition finder, subword tree etc. [10] .

Linear time and space algorithms for creating the compact suffix tree were given soon by Weiner [19], McCreight [14], Ukkonen [18], Chen and Sciferas [4] and others.

The statistical behaviour of suffix trees has been also studied. Most of the studies consider improved versions.

The average size of compact suffix trees was examined by Blumer, Ehrenfeucht and Haussler [3]. They proved that the average number of nodes in the compact suffix tree is asymptotically the sum of an oscillating function and a small linear function.

An important question is the height of suffix trees, which was answered by Devroye, Szpankowski and Rais [6], who proved that the expected height is logarithmic in the length of the string.

*Szegedi Tudományegyetem, TTIK, Szeged, 6720, Hungary. E-mail: mesti@math.u-szeged.hu

DOI: 10.14232/actacyb.22.4.2016.6

The application of suffix trees is very wide. We mention but only a few examples. Apostolico et al. [2] mention that these structures are used in text searching, indexing, statistics, compression. In computational biology, several algorithms are based on suffix trees. Just to refer a few of them, we mention the works of Höhl et al. [12], Adebiyi et al. [1] and Kaderali et al. [13].

Suffix trees are also used for detecting plagiarism [2], in cryptography [15, 16], in data compression [7, 8, 16] or in pattern recognition [17].

For the interested readers further details on suffix trees, their history and their applications can be found in [2], in [10] and in [11], which sources we also used for the overview of the history of suffix trees.

It is well-known that the non-compact suffix tree can be quadratic in space as we referred before. In our paper we are setting a lower bound on the average size, which is also quadratic.

2 Preliminaries

Before we turn to our results, let us define a few necessary notions.

Definition 1. *An* alphabet Σ *is a set of different characters. The* size *of an alphabet is the size of this set, which we denote by* $\sigma(\Sigma)$*, or more simply* σ*. A string* S *is over the alphabet* Σ *if each character of* S *is in* Σ*.*

Definition 2. *Let* S *be a string.* $S[i]$ *is its* ith *character, while* $S[i, j]$ *is a* substring *of* S*, from* $S[i]$ *to* $S[j]$*, if* $j \geq i$*, else* $S[i, j]$ *is the empty string. Usually* $n(S)$ *(or* n *if there is no danger of confusion) denotes the* length *of the string.*

Definition 3. *The* suffix tree *of* S *is a rooted directed tree with* n *leaves, where* n *is the length of* S*.*

Its structure is the following:

Each edge e *has a* label $\ell(e)$*, and the edges from a node* v *have different labels (thus, the suffix tree of a string is unique). If we concatenate the edge labels along a path* $\mathcal{P}$*, we get the* path label $\mathcal{L}(\mathcal{P})$*.*

We denote the path from the root to the leaf j *by* $\mathcal{P}(j)$*. The edge labels are such that* $\mathcal{L}(j) = \mathcal{L}(\mathcal{P}(j))$ *is* $S[j, n]$ *and a* \$ *sign at the end. The definition becomes more clear if we check the example on 1 and 2.*

A naive algorithm for constructing the suffix tree is the following:

Notice that in 2 a leaf always remain a leaf, as \$ (the last edge label before a leaf) is not a character in S.

Definition 4. *The* compact suffix tree *is a modified version of the suffix tree. We get it from the suffix tree by compressing its long branches.*

The structure of the compact suffix tree is basically similar to that of the suffix tree, but an edge label can be longer than one character, and each internal node (i.e. not leaf) must have at least two children. For an example see 2.

With a regard to suffix trees, we can define further notions for strings.

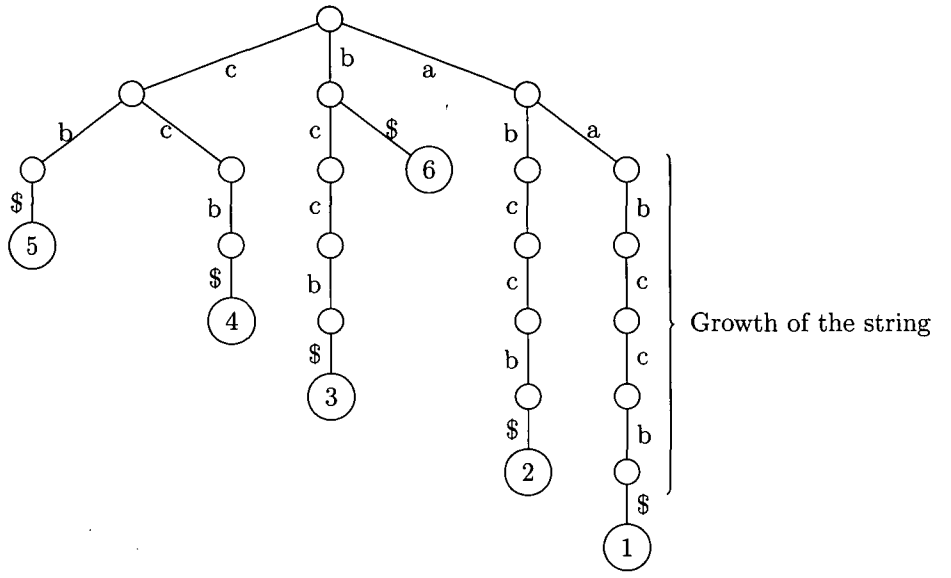


Figure 1: Suffix tree of string *aabccb*

Definition 5. *Let S be a string, and $\mathcal{T}$ be its (non-compact) suffix tree.*

A natural direction *of $\mathcal{T}$ is that all edges are directed from the root towards the leaves. If there is a directed path from u to v, then v is a* descendant *of u and u is an* ancestor *of v.*

We say that the growth of *S (denoted by $\gamma(S)$) is one less than the shortest distance of leaf 1 from an internal node v which has at least two children (including leaf 1), that is, we count the internal nodes on the path different from v. If leaf j is a descendant of v, then the common prefix of $S[j,n]$ and $S[1,n]$ is the longest among all j's.*

If we consider the string $S = aabccb$, the growth of S is 5, as it can be seen on 1.

An important notion is the following one.

Definition 6. *Let $\Omega(n,k,\sigma)$ be the number of strings of length n with growth k over an alphabet of size σ.*

Observe that the connection between the growth and the number of nodes in a suffix tree is the following:

Observation 1. *If we construct the suffix tree of S by using 2, we get that the sum of the growths of $S[n-1,n], S[n-2,n], \ldots, S[1,n]$ is a lower bound to the number of nodes in the final suffix tree. In fact, there are only two more internal nodes, the root vertex, the only node on the path to leaf n, and we have the leaves.*

In the proofs we will need the notion of period and of aperiodic strings.

Let S be a string of length n. Let $j = 1$ and T be a tree of one vertex r (the root of the suffix tree).

Step 1: Consider $X = S[j, n] + \$$. Set $i = 0$, and $v = r$.

Step 2: If there is an edge vu labelled $X[i + 1]$, then set $v = u$ and $i = i + 1$.

Step 3: Repeat Step 2 while it is possible.

Step 4: If there is no such an edge, add a path of $n - j - i + 2$ edges from v, with labels corresponding to $S[j + i, n] + \$$, consecutively on the edges. At the end of the path, number the leaf with j.

Step 5: Set $j = j + 1$, and if $j \leq n$, go to Step 1. ◇

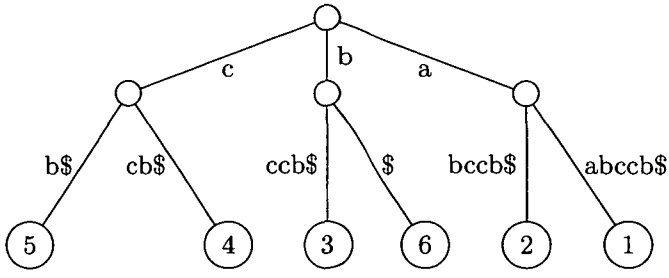


Figure 2: Compact tree of string *aabccb*

Definition 7. *Let S be a string of length n. We say that S is* periodic *with period d, if there is a $d|n$ for which $S[i] = S[i + d]$ for all $i \leq n - d$. Otherwise, S is* aperiodic.

The minimal period *of S is the smallest d with the property above.*

Definition 8. *$\mu(j, \sigma)$ is the number of j-length aperiodic strings over an alphabet of size σ.*

A few examples for the number of aperiodic strings are given in 1.

σ	$\mu(1,\sigma)$	$\mu(2,\sigma)$	$\mu(3,\sigma)$	$\mu(4,\sigma)$	$\mu(5,\sigma)$	$\mu(6,\sigma)$	$\mu(7,\sigma)$	$\mu(8,\sigma)$
2	2	6	12	30	54	126	240	504
3	3	6	24	72	240	696	2184	648
4	4	12	60	240	1020	4020	16380	65280
5	5	20	120	600	3120	15480	78120	390000

Table 1: Number of aperiodic strings for small alphabets. σ is the size of the alphabet, and $\mu(j, \sigma)$ is the number of aperiodic strings of length j

3 Main results

Our main results are formulated in the following theorems.

Theorem 2. *On an alphabet of size σ for all $n \geq 2k$, $\Omega(n, k, \sigma) \leq \phi(k, \sigma)$ for some function ϕ.*

Theorem 3. *There is a $c > 0$ and an n_0 such that for any $n > n_0$ the following is true. Let S' be a string of length $n-1$, and S be a string obtained from S' by adding a character to its beginning chosen uniformly random from the alphabet. Then the expected growth of S is at least $c \cdot n$.*

Theorem 4. *There is a $d > 0$ that for any $n > n_0$ (where n_0 is the same as in 3) the following holds. On an alphabet of size σ the simple suffix tree of a random string S of length n has at least $d \cdot n^2$ nodes in expectation.*

4 Proofs

Proof. (4)

Considering 1 we have that the expected size of the simple suffix tree of a random string S is at least

$$\mathbb{E} \sum_{m=1}^{n} (\gamma(S[n-m, n])) \geq \sum_{m=1}^{n} \mathbb{E}(\gamma(S[n-m, n])). \tag{1}$$

If $m \leq n_0$, 3 is obvious. If $m > n_0$, we can divide the sum into two parts:

$$\sum_{m=1}^{n} \mathbb{E}(\gamma(S[n-m, n])) = \sum_{m=1}^{n_0} \mathbb{E}(\gamma(S[n-m, n])) + \sum_{m=n_0+1}^{n} \mathbb{E}(\gamma(S[n-m, n])). \tag{2}$$

The first part of the sum is a constant, while the second part can be estimated with 3:

$$\sum_{m=n_0+1}^{n} \mathbb{E}(\gamma(S[n-m, n])) \geq \sum_{m=n_0+1}^{n} cn = d \cdot n^2. \tag{3}$$

This proves 4.

□

First, we show a few lemmas about the number of aperiodic strings. 1 can be found in [9] or in [5], but we give a short proof also here.

Lemma 1. *For all $j > 0$ integer and for all alphabet of size σ the number of aperiodic strings is*

$$\mu(j, \sigma) = \sigma^j - \sum_{\substack{d|j \\ d \neq j}} \mu(d, \sigma). \tag{4}$$

Proof. $\mu(1,\sigma)=\sigma$ is trivial.

There are σ^j strings of length j. Suppose that a string is periodic with minimal period d. This implies that its first d characters form an aperiodic string of length d, and there are $\mu(d,\sigma)$ such strings. This finishes the proof. □

Specially, if p is prime, then $\mu(p,\sigma)=\sigma^p-\sigma$.

Corollary 1. *If p is prime and $t\in\mathbb{N}$, then $\mu(p^t,\sigma)=\sigma^{p^t}-\sigma^{p^{t-1}}$ for all alphabet of size σ.*

Proof. We count the aperiodic strings of length p^t. There are σ^{p^t} strings. Consider the minimal period of the string, i.e. the period which is aperiodic. If we exclude all minimal periods of length k, we exclude $\mu(k,\sigma)$ strings. This yields the following equality:

$$\mu\left(p^t,\sigma\right)=\sigma^{p^t}-\sum_{1\le s<t}\mu\left(p^s,\sigma\right). \tag{5}$$

With a few transformations and using 1, we have that (5) is equal to

$$\sigma^{p^t}-\mu\left(p^{t-1},\sigma\right)-\sum_{1\le s<t-1}\mu\left(p^s,\sigma\right)=\sigma^{p^t}-\sigma^{p^{t-1}}+\sum_{1\le s<t-1}\mu\left(p^s,\sigma\right)-\sum_{1\le s<t-1}\mu\left(p^s,\sigma\right), \tag{6}$$

which is

$$\sigma^{p^t}-\sigma^{p^{t-1}}. \tag{7}$$

□

Lemma 2. *For all $j>1$ and for all alphabet of size σ , $\mu(j,\sigma)\le\sigma^j-\sigma$.*

Proof. From 1 we have $\mu(j,\sigma)=\sigma^j-\sum_{\substack{d\mid j\\ d\neq j}}\mu(d,\sigma)$. Considering $\mu(d,\sigma)\ge 0$ and $\mu(1,\sigma)=\sigma$, we get the claim of the lemma. □

Lemma 3. *For all $j\ge 1$, and for all alphabet of size σ*

$$\mu(j,\sigma)\ge\sigma(\sigma-1)^{j-1}. \tag{8}$$

Proof. We prove by induction. For $j=1$ the claim is obvious, as $\mu(1,\sigma)=\sigma$.

Suppose we know the claim for $j-1$. Consider $\sigma(\sigma-1)^{j-2}$ aperiodic strings of length $j-1$. Now, for any of these strings there is at most one character by appending that to the end of the string we receive a periodic string of length j. Therefore we can append at least $\sigma-1$ characters to get an aperiodic string, which gives the desired result.

□

Observation 5. *Observe that if the growth of S is k, then there is a j such that $S[1,n-k]=S[j+1,j+n-k]$. For example, if the string is abcdefabcdab ($n=12$), one can check that the growth is 8 (the new branch in the suffix tree which ends in leaf 1 starts after abcd), and with $j=6$ we have $S[1,4]=S[7,10]=abcd$.*

The reverse of this observation is that if there is a $j < n$ such that $S[1, n-k] = S[j+1, j+n-k]$, then the growth is at most k, as $S[j+1, n]$ and $S[1, n]$ shares a common prefix of length $n-k$, thus, the paths to the leaves $j+1$ and n share $n-k$ internal nodes, and at most k new internal nodes are created.

Proof. (2) For proving the theorem we count the number of strings with growth k for $n \geq 2k$.

First, we fix j, and then count the number of possible strings where the growth occurs such that $S[1, n-k] = S[j+1, j+n-k]$ for that fixed j. Note that by this way, we only have an upper bound for this number, as we might found an ℓ such that $S[1, n-k+1] = S[\ell+1, \ell+n-k+1]$.

We know that $j \leq k$, otherwise $S[j+1, j+n-k]$ does not exist.

If $j = k$, then we know $S[1, n-k] = S[k+1, n]$.

$S[1, k]$ must be aperiodic. Suppose the opposite and let $S[1, k] = p \ldots p$, where p is the minimal period, and its length is d. Then $S[k+1, n] = p \ldots p$. Obviously, in this case $S[1, n-d] = S[d+1, n]$, which by 5 means that the growth would be at most d. See also 3.

Therefore this case gives us at most $\mu(k)$ strings of growth k.

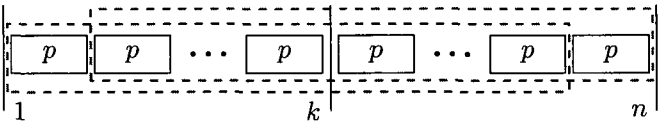


Figure 3: Proof of 2, case $j = k$

If $j < k$, then we have $S[1, n-k] = S[j+1, j+n-k]$.

First, we note that $S[1, j]$ must be aperiodic. Suppose the opposite and let $S[1, j] = p \ldots p$, where p is the minimal period, and its length is d. Then

$$S[j+1, 2j] = S[2j+1, 3j] = \ldots = p \ldots p, \tag{9}$$

which means that

$$S\left[1, \left\lfloor \frac{k}{j} \right\rfloor \cdot j\right] = S\left[j+1, j+\left\lfloor \frac{k}{j} \right\rfloor \cdot j\right] = p \ldots p. \tag{10}$$

This implies that $S[1, j+n-k] = p \ldots pp'$, where p' is a prefix of p. However, $S[1, j+n-k-d] = S[d, j+n-k]$ is true, and using 5, we have that $\gamma(S) \leq n - (j+n-k) + d = k - j + d < k$, which is a contradiction.

Further, $S[j+n-k+1]$ must not be the same as $S[k+1]$, which means that this character can be chosen $\sigma - 1$ ways.

Therefore this case gives us at most $\mu(j)(\sigma-1)\sigma^{k-j-1}$ strings of growth k for each j.

By summing up for each j, we have

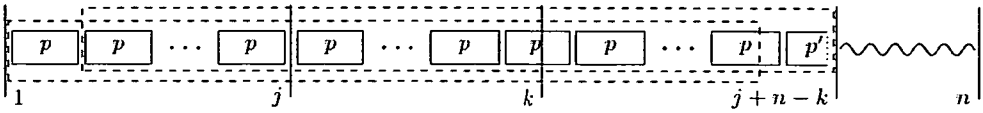


Figure 4: Proof of 2, case $j < k$

$$\phi(k,\sigma) = \sum_{j=1}^{k-1} \mu(j,\sigma)(\sigma-1)\sigma^{k-j-1} + \mu(k,\sigma) \tag{11}$$

This completes the proof. □

Proof. (3)

According to 2, $\mu(j,\sigma) \leq \sigma^j - \sigma$ (if $j > 1$).

In the proof of 2 at (11) we saw for $k \geq 1$ and $n \geq 2k-1$ that

$$\phi(k,\sigma) = \mu(k,\sigma) + \sum_{j=1}^{k-1} \mu(j,\sigma)(\sigma-1)\sigma^{k-j-1}. \tag{12}$$

We can bound the right hand side of (12) from above as it follows:

$$\mu(k,\sigma) + \sum_{j=1}^{k-1} \mu(j,\sigma)(\sigma-1)\sigma^{k-j-1} = \mu(k,\sigma) + \mu(1,\sigma)(\sigma-1)\sigma^{k-2} + \sum_{j=2}^{k-1} \mu(j,\sigma)(\sigma-1)\sigma^{k-j-1}, \tag{13}$$

which is by 2 at most

$$\sigma^k - \sigma + \sigma(\sigma-1)\sigma^{k-2} + \sum_{j=2}^{k-1} (\sigma^j - \sigma)(\sigma-1)\sigma^{k-j-1} \leq \sigma^k + \sigma^k + \sum_{j=2}^{k-1} \sigma^j \sigma \sigma^{k-j-1} \leq k\sigma^k. \tag{14}$$

Thus, $\phi(k,\sigma) \leq k\sigma^k$, which means

$$\sum_{k=1}^{m} \phi(k,\sigma) \leq \sum_{k=1}^{m} k\sigma^k \leq (m+1)\sigma^{m+1}. \tag{15}$$

The left hand side of 15 is an upper bound for the strings of growth at most m. Let $m = \lfloor \frac{n}{2} \rfloor$.

As $\sigma^n \gg \frac{n}{2}\sigma^{\frac{n}{2}}$, this implies that in most cases the suffix tree of S has at least $\frac{n}{2}$ more nodes than the suffix tree of $S[1, n-1]$.

Thus, a lower bound on the expectation of the growth of S is

$$\mathbb{E}(\gamma(S)) \geq \frac{1}{\sigma^n}\left(\frac{n}{2}\sigma^{\frac{n}{2}} + \left(\sigma^n - \frac{n}{2}\sigma^{\frac{n}{2}}\right)\left(\frac{n}{2}+1\right)\right), \tag{16}$$

which is

$$\frac{1}{\sigma^n}\left(\frac{n+2}{2}\sigma^n + \left(\frac{n}{2} - \frac{n(n+2)}{4}\right)\sigma^{\frac{n}{2}}\right) = cn, \tag{17}$$

with some c, if n is large enough.

□

With this, we have finished the proof and gave a quadratic lower bound on the average size of suffix trees.

References

[1] E.F. Adebiyi, T. Jiang, and M. Kaufmann. An efficient algorithm for finding short approximate non-tandem repeats. *Bioinformatics*, 17:5S–12S, 2001.

[2] A. Apostolico, M. Crochemore, M. Farach-Colton, Z. Galil, and S. Muthukrishnan. 40 years of suffix trees. *Communications of the ACM*, 59:66–73, 2016.

[3] A. Blumer, A. Ehrenfeucht, and D. Haussler. Average sizes of suffix trees and DAWGs. *Discrete Applied Mathematics*, 24:37–45, 1989.

[4] M.T. Chen and J. Sciferas. Efficient and elegant subword tree construction. In *Combinatorial algorithms on words*, pages 97–107. Springer-Verlag, 1985.

[5] J.D. Cook. Counting primitve bit strings. `http://www.johndcook.com/blog/2014/12/23/counting-primitive-bit-strings/`, 2014. [Online; accessed 02-May-2016].

[6] L. Devroye, W. Szpankowski, and B. Rais. A note on the height of suffix trees. *SIAM Journal on Computing*, 21:48–53, 1993.

[7] E. R. Fiala and D. H. Greene. Data compression with finite windows. *Communications of the ACM*, 32:490–505, 1989.

[8] C. Fraser, A. Wendt, and E.W. Myers. Analyzing and compressing assembly code. In *Proceedings SIGPLAN Symposium on Compiler Construction*, pages 117–121, 1984.

[9] E.N. Gilbert and J. Riordan. Symmetry types of periodic sequences. *Illinois Journal of Mathematics*, 5:657–665, 1961.

[10] R. Grossi and G.F. Italiano. Suffix trees and their applications in string algorithms. In *Proceedings of the 1st South American Workshop on String Processing*, pages 57–76, 1993.

[11] D. Gusfield. *Algorithms on Strings, Trees and Sequences.* Cambridge University Press, 1997.

[12] M. Höhl, S. Kurtz, and E. Ohlebusch. Efficient multiple genome alignment. *Bioinformatics*, 18:312S–320S, 2002.

[13] L. Kaderali and A. Schliep. Selecting signature oligonucleotides to identify organisms using DNA arrays. *Bioinformatics*, 18:1340–1348, 2002.

[14] E. M. McCreight. A space-economical suffix tree construction algorithm. *Journal of the ACM*, 23:262–272, 1976.

[15] L. O'Connor and T. Snider. Suffix trees and string complexity. In *Advances in Cryptology: Proceedings of EUROCRYPT, LNCS 658*, pages 138–152. Springer-Verlag, 1992.

[16] M. Rodeh. A fast test for unique decipherability based on suffix trees,. *IEEE Transactions on Information Theory*, 28(4):648–651, 1982.

[17] S.L. Tanimoto. A method for detecting structure in polygons. *Pattern Recognition*, 13:389–494, 1981.

[18] E. Ukkonen. On-line construction of suffix trees. *Algorithmica*, 14:249–260, 1995.

[19] P. Weiner. Linear pattern matching algorithms. In *Proceedings of the 14th IEEE Symposium on Switching and Automata Theory*, pages 1–11, 1973.

Received 13th July 2015

ACTA CYBERNETICA

Information for authors. Acta Cybernetica publishes only original papers in the field of Computer Science. Manuscripts must be written in good English. Contributions are accepted for review with the understanding that the same work has not been published elsewhere. Papers previously published in conference proceedings, digests, preprints are eligible for consideration provided that the author informs the Editor at the time of submission and that the papers have undergone substantial revision. If authors have used their own previously published material as a basis for a new submission, they are required to cite the previous work(s) and very clearly indicate how the new submission offers substantively novel or different contributions beyond those of the previously published work(s). Each submission is peer-reviewed by at least two referees. The length of the review process depends on many factors such as the availability of an Editor and the time it takes to locate qualified reviewers. Usually, a review process takes 6 months to be completed. There are no page charges. An electronic version of the puplished paper is provided for the authors in PDF format.

Manuscript Formatting Requirements. All submissions must include a title page with the following elements:

- title of the paper
- author name(s) and affiliation
- name, address and email of the corresponding author
- An abstract clearly stating the nature and significance of the paper. Abstracts must not include mathematical expressions or bibliographic references.

References should appear in a separate bibliography at the end of the paper, with items in alphabetical order referred to by numerals in square brackets. Please prepare your submission as one single PostScript or PDF file including all elements of the manuscript (title page, main text, illustrations, bibliography, etc.). Manuscripts must be submitted by email as a single attachment to either the most competent Editor, the Managing Editor, or the Editor-in-Chief. In addition, your email has to contain the information appearing on the title page as plain ASCII text. When your paper is accepted for publication, you will be asked to send the complete electronic version of your manuscript to the Managing Editor. For technical reasons we can only accept files in LaTeX format.

Subscription Information. Acta Cybernetica is published by the Institute of Informatics, University of Szeged, Hungary. Each volume consists of four issues, two issues are published in a calendar year. Subscription rates for one issue are as follows: 5000 Ft within Hungary, €40 outside Hungary. Special rates for distributors and bulk orders are available upon request from the publisher. Printed issues are delivered by surface mail in Europe, and by air mail to overseas countries. Claims for missing issues are accepted within six months from the publication date. Please address all requests to:

Acta Cybernetica, Institute of Informatics, University of Szeged
P.O. Box 652, H-6701 Szeged, Hungary
Tel: +36 62 546 396, Fax: +36 62 546 397, Email: acta@inf.u-szeged.hu

Web access. The above information along with the contents of past issues are available at the Acta Cybernetica homepage https://www.inf.u-szeged.hu/en/kutatas/acta-cybernetica .

CONTENTS

Elvira Antal and Tibor Csendes: Nonlinear Symbolic Transformations for Simplifying Optimization Problems 715
Geir Agnarsson, Raymond Greenlaw, and Sanpawat Kantabutra: The Structure of Rooted Weighted Trees Modeling Layered Cyber-security Systems 735
László Czap and Attila K. Varga: Adapting Dynamic Time Warping to the Speech of the Hearing Impaired 771
Shubh N. Singh and Kanduru V. Krishna: The Holonomy Decomposition of some Circular Semi-Flower Automata 791
Miklós Ujvári: Bounds on the Stability Number of a Graph via the Inverse Theta Function 807
Bálint Vásárhelyi: An Estimation of the Size of Non-Compact Suffix Trees . 823

ISSN 0324—721 X

Felelős szerkesztő és kiadó: Csirik János