Contents

ANNALES MATHEMATICAE ET INFORMATICAE 44. (2015)

ANNALES MATHEMATICAE ET INFORMATICAE

TOMUS 44. (2015)

HUNGARIA, EGER

ANNALES MATHEMATICAE ET INFORMATICAE

International journal for mathematics and computer science

Referred by
Zentralblatt für Mathematik
and
Mathematical Reviews

The journal of the Institute of Mathematics and Informatics of Eszterházy Károly College is open for scientific publications in mathematics and computer science, where the field of number theory, group theory, constructive and computer aided geometry as well as theoretical and practical aspects of programming languages receive particular emphasis. Methodological papers are also welcome. Papers submitted to the journal should be written in English. Only new and unpublished material can be accepted.

Authors are kindly asked to write the final form of their manuscript in LaTeX. If you have any problems or questions, please write an e-mail to the managing editor Miklós Hoffmann: `hofi@ektf.hu`

The volumes are available at `http://ami.ektf.hu`

ANNALES MATHEMATICAE ET INFORMATICAE

VOLUME 44. (2015)

INSTITUTE OF MATHEMATICS AND INFORMATICS
ESZTERHÁZY KÁROLY COLLEGE
HUNGARY, EGER

Selected papers of the 9th International Conference on Applied Informatics

HU ISSN 1787-5021 (Print)
HU ISSN 1787-6117 (Online)

A kiadásért felelős az
Eszterházy Károly Főiskola rektora
Megjelent az EKF Líceum Kiadó gondozásában
Kiadóvezető: Czeglédi László
Műszaki szerkesztő: Tómács Tibor
Megjelent: 2015. június Példányszám: 30

Készítette az
Eszterházy Károly Főiskola nyomdája
Felelős vezető: Kérészy László

Annales Mathematicae et Informaticae
44 (2015) pp. 5–13
http://ami.ektf.hu

Representation transformations of ordered lists*

Tibor Ásványi

Eötvös Loránd University, Faculty of Informatics
Budapest, Hungary
asvanyi@inf.elte.hu

Submitted September 15, 2014 — Accepted March 5, 2015

Abstract

Search and update operations of dictionaries have been well studied, due to their practical significance. There are many different representations of them, and some applications prefer this, the others that representation. A main point is the size of the dictionary: for a small one a sorted array can be the best representation, while for a bigger one an AVL tree or a red-black tree might be the optimal choice (depending on the necessary operations and their frequencies), and for an extra large one we may prefer a B+-tree, for example.

Consequently it can be desirable to transform such a collection of data from one representation into another, efficiently. There is a common feature of the data structures mentioned: they can be considered strictly ordered lists. Thus in this paper we start a new topic of interest: *How to transform a strictly ordered list form one representation into another, efficiently?* What about the time and space complexities of such transformations?

Keywords: strictly increasing list, representation-transformation, data structure (DS), linear, array, binary tree (BT), balanced, search tree

MSC: 68P05, 68P10, 68P20, 68Q25

1. Introduction

In this paper we consider strictly increasing lists. They can be represented in several different ways. For example, with a *linear data structure (LDS)* (e.g. array,

*Supported by Eötvös Loránd University, Faculty of Informatics.

linked list, sequential file), with a *binary search tree (BST)* (e.g. unbalanced BST, AVL tree, red-black tree), with a *B-tree*, *B+-tree*, etc. [1, 2, 3].

Their common features are that they can be traversed increasingly in $\Theta(n)$ time: the linear traversal of a LDS has linear operational complexity; similarly, the inorder traversal of a tree needs $\Theta(n)$ time. And the search-and-update operations can run in $O(n)$ time. [1, 2]

In this paper we use three asymptotic computational complexity measures (each time we consider the worst case by default): $O(g(n))$ (upper bound), $\Omega(g(n))$ (lower bound), and $\Theta(g(n)) = O(g(n)) \cap \Omega(g(n))$ [2].

Sorted arrays support only the *search* with $O(\log(n))$ operational complexity, but for the *balanced search tree*[1] representations each of the *search, insert, delete* operations have this complexity. On linked lists, and sequential files we cannot perform any search-and-update operation in $O(\log(n))$ time. Thus we concentrate on the *sorted array*, and *balanced search tree* representations of such lists. The *search, insert, delete* operations have been well studied. Sometimes we have to *transform these lists* from one representation into another. Consequently we pay attention to these representation-transformations. We ask, how to transform a strictly ordered list L from one representation into another, *efficiently*?

Undeniably, the operational complexity of such a transformation is $\Omega(n)$: each item must be processed. In some cases, it is also $O(n)$: Undoubtedly, a linear representation of L can be produced in $O(n)$ time, because the input representation of L can be traversed also in $O(n)$ time. Thus, regardless of the input representation of L, a linear representation of L can be generated with $\Theta(n)$ atomic operations. Besides, a balanced search tree representation of L can be generated in $O(n \log(n))$ time, because a single insert needs $O(\log(n))$ atomic operations. Nevertheless this method does not use the information that the input is *sorted*.

Consequently this is our question: Given an input representation of L, when and how can we produce a balanced search tree representation of it, with an operational complexity $\Theta(n)$, or at least better than $\Theta(n \log(n))$? We give a partial answer to this question. We invent three algorithms. With operational complexity $\Theta(n)$, we transform (1) a *strictly increasing array* into an *AVL* tree; (2) a *strictly increasing array* into a *red-black* tree; (3) an *AVL* tree into a *red-black* tree.

2. Main results

In order to expound these algorithms (a) we define *size-balanced* BSTs, and an algorithm transforming a *strictly increasing array* into such a size-balanced BST; (b) we prove that a size-balanced BST is almost complete, and so (c) it is an *AVL* tree; (d) we colour the almost complete BSTs as *red-black* trees; (e) we find a special property of *AVL* trees, and invent an algorithm colouring them as *red-black* trees.

(a-c) are needed for transforming a *strictly increasing array* into an *AVL* tree (Section 2.1). (a,b,d) result in the transformation of a *strictly increasing array* into

[1] AVL tree, red-black tree, SBB-tree, rank-balanced tree, B-tree, B+-tree, etc.

a *red-black* tree (Section 2.2). The theorems and algorithm of (e) in Subsection 2.3 form the high point of this section.

2.1. Strictly increasing array to AVL tree

First we enumerate the necessary notions. By trees we mean rooted ordered trees [2]. Remember that NIL is the empty tree. The leaves of a nonempty tree have no child. The non-leaves are the internal nodes.

If $t \neq NIL$ is a binary tree (BT), left(t) is its left and right(t) is its right subtree.

If t is a BT, $s(t)$ is its size, i.e. $s(t) = 0$, if $t = NIL$; $s(t) = 1 + s(\text{left}(t)) + s(\text{right}(t))$, otherwise. $h(t)$ is its height, i.e. $h(t) = -1$, if $t = NIL$; $h(t) = 1 + \max(h(\text{left}(t)), h(\text{right}(t)))$, otherwise.

If r is the root node of a BT $t \neq NIL$, left(r) = left(t), right(r) = right(t), and root$(t) = r$. Provided that t is a BT, $n \in t$, *iff* $t \neq NIL \wedge (n = \text{root}(t) \vee n \in \text{left}(t) \vee n \in \text{right}(t))$.

$d_t(n)$ is the depth of node n in BT t. If $t \neq NIL$, $d_t(\text{root}(t)) = 0$. If n is a node of a BT t and left$(n) \neq NIL$, $d_t(\text{root}(\text{left}(n))) = d_t(n) + 1$. If right$(n) \neq NIL$, $d_t(\text{root}(\text{right}(n))) = d_t(n) + 1$. Node n is *strictly binary* ($SB(n)$), *iff* left$(n) \neq NIL \wedge$ right$(n) \neq NIL$.

Clearly, $h(t) = \max\{d_t(n) \mid n \in t\}$, if $t \neq NIL$. A BT t is complete, *iff* $(\forall n \in t)(d_t(n) < h(t) \rightarrow SB(n))$.

Notice that for any leaf n of a complete BT t, $d(n) = h(t)$; and $s(t) = 2^{h(t)+1} - 1$. A BT t is *almost complete* ($AC(t)$), *iff* $(\forall n \in t)(d_t(n) < h(t) - 1 \rightarrow SB(n))$.

Notice that a BT is AC, *iff* compared to the appropriate complete BT, nodes may be missing only from its lowest level: Figure 1 shows such a tree. Clearly, for a leaf n of an AC BT t, $d_t(n) \in \{h(t), h(t) - 1\}$. The nodes of t at depth $h(t) - 1$ may have one or two children, or may be leaves. $s(t) \in [2^{h(t)}, 2^{h(t)+1} - 1]$.

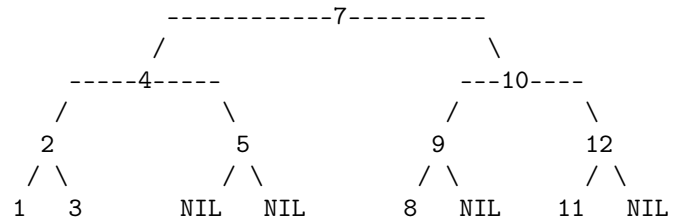


Figure 1: almost complete BST: the places of the missing nodes are shown by NILs

A node n of a BT is height-balanced, *iff* $|h(\text{right}(n)) - h(\text{left}(n))| \leq 1$. A BT t is height-balanced, *iff* $(\forall n \in t)$, n is height-balanced. An AVL tree is a height-balanced BST.

A node n of a BT is size-balanced, *iff* $|s(\text{right}(n)) - s(\text{left}(n))| \leq 1$. A BT t is size-balanced, *iff* $(\forall n \in t)$, n is size-balanced.

We transform a strictly increasing array into an equivalent size-balanced BST in linear time:

We take the middle item of a nonempty array. This will label the root node of the tree. Next we transform the left and right sub-arrays into the appropriate subtrees, recursively. An empty array is transformed into an empty tree.

The resulting size-balanced BST is also an AVL tree, as it follows from the next two theorems.

Theorem 2.1. *A size-balanced binary tree is also almost complete (AC).*

Proof. We use mathematical induction with respect to the height $h(t)$ of the size-balanced tree t. If $h(t) = -1$, then t is empty, and $AC(t)$. Let us suppose that we have this property for trees with $h(t) \leq h$. Let $h(t) = h + 1$. Then $t \neq NIL \wedge h(\text{left}(t)) \leq h \wedge h(\text{right}(t)) \leq h$. It follows by induction that $\text{left}(t)$ and $\text{right}(t)$ are almost complete. Also $|\,\text{size}(\text{left}(t)) - \text{size}(\text{right}(t))| \leq 1$. (Furthermore, remember that a complete binary tree of height h has the size $2^{h+1} - 1$, and an almost complete binary tree with size in $[2^{h+1}, 2^{h+2} - 1]$ has height $h+1$.) Now we enumerate the possible cases about the subtrees of t, and prove that $AC(t)$ in each case. If the two (almost complete) subtrees have the same size, their heights are also equal, and $AC(t)$. If the smaller subtree is complete, then the bigger one has an extra leaf at its extra level, and $AC(t)$. If the smaller subtree is not complete, then the bigger one has the same height, and $AC(t)$. □

Theorem 2.2. *An almost complete binary tree is also height-balanced.*

Proof. We can suppose $t \neq NIL$. First, if $AC(t)$, the leaves of t have depth $h(t)$ or $h(t) - 1$. Thus $h(\text{left}(t)), h(\text{right}(t)) \in \{h(t) - 1, h(t) - 2\}$. Consequently, $|h(\text{right}(t)) - h(\text{left}(t))| \leq 1$. As a result, $\text{root}(t)$ is balanced. Next, let us suppose that $lr(t) \in \{\text{left}(t), \text{right}(t)\}$. Now, if $AC(t)$, then $(\forall n \in lr(t))(d_t(n) < h(t) - 1 \rightarrow SB(n))$. Therefore $(\forall n \in lr(t))(d_{lr(t)}(n) < h(t) - 2 \rightarrow SB(n))$. We also have $h(lr(t)) \leq h(t) - 1$. For these reasons $(\forall n \in lr(t))(d_{lr(t)}(n) < h(lr(t)) - 1 \rightarrow SB(n))$. As a result, $AC(lr(t))$. Thus each (direct or indirect) subtree of t is AC, and if a subtree is nonempty, its root node is balanced. Finally, each node of t is balanced. □

Corollary 2.3. *A size-balanced BST is also an AVL tree.*

Proof. A size-balanced BST is almost complete, thus height-balanced. □

Consequently, the algorithm we defined above transforms a strictly increasing array into an equivalent AVL tree. It takes $\Theta(n)$ time, because each item of the array is processed once. Besides the $\Theta(n)$ size of its output, it needs $\Theta(\log(n))$ working memory: this is the height of the recursion. Provided that we need the heights of the nonempty subtrees in their root nodes (as it is usual with AVL trees), we can return the height of a subtree when we return from the appropriate recursive call, and compute the height of a subtree with a given root node from the heights of the two subtrees of that node.

2.2. Strictly increasing array to red-black tree

We have an algorithm transforming a strictly increasing array into an almost complete BST. We also have the height of the tree. Here we need an additional flag showing whether the tree is complete or not. Clearly, a nonempty BT is complete, *iff* its too subtrees are also complete, and their heights are the same. Thus the computation of this flag is also easily merged into the algorithm above.

Next, if we prove that an almost complete BST (with its height and flag) can be coloured in linear time, as a red-black tree, then we have also the algorithm transforming a strictly increasing array into such a tree.

Definition 2.4. A red-black tree is a BST with red and black nodes: The root node is black. We regard NILs as pointers to black, external leaves. For each node, all simple paths from the node to descendant NIL-leaves contain the same number of black nodes. If a node is red, then both its children are black. [2] (See Figure 2.)

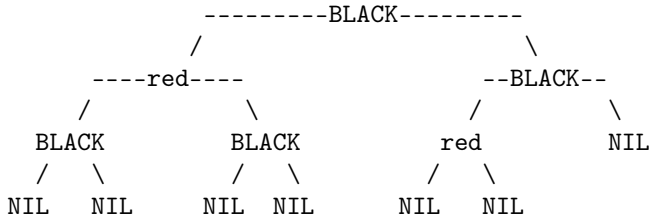


Figure 2: Red-black tree

Based on this definition, the algorithm of *colouring* is simple: consider the complete levels of an almost complete BST, and paint the nodes black. If the tree is not complete, the nodes at the lowest, partially filled level remain, and we paint them red. (See Figure 3.) Unquestionably this procedure needs $\Theta(n)$ time and $\Theta(\log(n))$ working memory. The algorithm computing its input (the almost complete tree, its height, and flag) has the same measures. Consequently, the whole transformation has these time and space requirements.

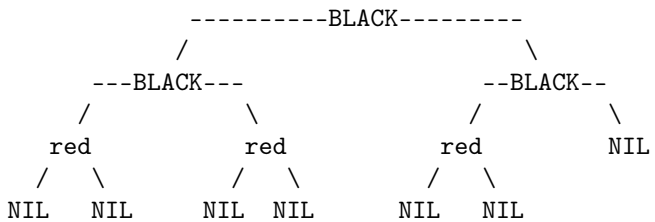


Figure 3: Almost complete tree painted as red-black tree

2.3. AVL tree to red-black tree

We colour an AVL tree t as a red-black tree. We use a postorder and a preorder traversal. As a result, our procedure needs $\Theta(n)$ time and $\Theta(\log(n))$ (proportional to the height of t) working memory [1].

Definition 2.5. Minimal height of a binary tree t: $m(t) = -1$, if $t = NIL$; $1 + \min(m(\text{left}(t)), m(\text{right}(t)))$, otherwise.

Theorem 2.6. *If t is a height-balanced tree then $m(t) \leq h(t) \leq 2m(t) + 1$.*

Proof. It comes with mathematical induction with respect to $m(t)$. If $m(t) = -1 \Rightarrow t = NIL \Rightarrow h(t) = -1 \Rightarrow m(t) \leq h(t) \leq 2m(t) + 1$. Let us suppose that we have this property for trees with $m(t) = k$. Let $m(t) = k + 1$. We can suppose that $m(\text{left}(t)) = k$. By induction: $k \leq h(\text{left}(t)) \leq 2k+1$. The tree t is height-balanced. Therefore $h(\text{right}(t)) \leq 2k + 2$. Thus $k + 1 \leq 1 + \max(h(\text{left}(t)), h(\text{right}(t))) = h(t) \leq 2k + 3 = 2(k + 1) + 1$. As a result: $m(t) \leq h(t) \leq 2m(t) + 1$. (Notice that $m(t) \leq h(t)$ for any binary tree.) □

In the *colouring* algorithm, first we calculate $m(t)$. Based on the definition, this can be done with a postorder traversal of t. In a typical AVL tree, for each non-NIL subtree s of t, the $h(s)$ attributes are already present. (If not, the computation of the $h(s)$ values can be easily merged into the postorder traversal.)

Next, with a preorder traversal of t, we *colour* t. (See Figure 4). We paint $m(t) + 1$ nodes black on each simple path from the root to a NIL-leaf. (The NIL-leaves are also considered black, but we do not paint them.)

```
PreCondition of the first call:
I0: t is AVL tree and b = m(t)+1 and
    h(s) is calculated for each subtree s of t

procedure colour( t : BinTree; b : integer )
    /* I1: b>=0 and b-1 =< m(t) and h(t) =< 2*b */
    if( t \= NIL ) {
    /* Note: paint b nodes black on each branch of t */
        if( h(t) < 2*b ) {
            colour(t) := black
            b := b-1 }
        else /* I2: 0 =< b =< m(t) and h(t) = 2*b */
            colour(t) := red
        colour(left(t),b)
        colour(right(t),b) }
end of procedure colour
```

Figure 4: Colouring an AVL tree `t` as a red-black tree

Now we are going to prove the correctness of the *colouring* algorithm in Figure 4.

Terminology: In the rest of this section we use $I0$ (i.e. the PreCondition), invariants $I1$, $I2$, and other logical statements. Let us suppose that $Ij, Ik \in \{I0, I1, I2\}$; P, Q are arbitrary statements. When we say that *Ij with P induces Ik with Q*, we mean: If *Ij and P* are true when the program is at the place of Ij, then *Ik and Q* will hold when the run of the program next time arrives at the place of Ik.

Lemma 2.7. *$I0$ induces $I1$ with $h(t) < 2b$.*

Proof. Based on the definition of $m(t)$, $m(t) \geq -1$. Consequently $b = m(t) + 1$ implies $b \geq 0 \wedge b - 1 \leq m(t)$. Theorem 2.6 implies $h(t) \leq 2m(t) + 1$. Considering $b = m(t)+1$ we receive $h(t) \leq 2m(t)+1 = 2(b-1)+1 = 2b-1$. Thus $h(t) < 2b$. □

Lemma 2.8. *$I1$ with $t \neq NIL \wedge h(t) < 2b$ induces $I1$ in both recursive calls.*

Proof. Let $s(t)$ be the left or right subtree parameterizing the appropriate recursive call. Thus we need to prove $I1^{b \leftarrow b-1, t \leftarrow s(t)}$ i.e. that the following three conditions hold:

(1) $b - 1 \geq 0$: We know that $h(t) \geq 0$ (since $t \neq NIL$) and $h(t) < 2b$. Consequently, $b > 0$, and therefore $b - 1 \geq 0$.

(2) $b - 2 \leq m(s(t))$: From $I1$, $b - 1 \leq m(t)$. From the definition of $m(t)$, $m(t) \leq 1 + m(s(t))$. As a result, $b - 1 \leq 1 + m(s(t))$, i.e. $b - 2 \leq m(s(t))$.

(3) $h(s(t)) \leq 2(b - 1)$: $h(t) < 2b$, i.e. $h(t) \leq 2b - 1$; $h(s(t)) \leq h(t) - 1$; thus $h(s(t)) \leq 2b - 2$. □

Lemma 2.9. *$I1$ with $t \neq NIL \wedge h(t) \geq 2b$ induces $I2$.*

Proof. $h(t) \leq 2b$ and $h(t) \geq 2b$ implies $h(t) = 2b$. $0 \leq b$ remains true. Considering Theorem 2.6 we have $2b \leq 2m(t) + 1$; thus $b \leq m(t) + 1/2$ i.e. $b \leq m(t)$. □

Lemma 2.10. *$I2$ induces $I1$ with $h(t) < 2b$ in both recursive calls.*

Proof. Let $s(t)$ be the left or right subtree parameterizing the appropriate recursive call. Thus we have to prove $(I1 \wedge h(t) < 2b)^{t \leftarrow s(t)}$ i.e. $b \geq 0 \wedge b - 1 \leq m(s(t)) \wedge h(s(t)) < 2b$. $b \geq 0$ remains true. From $b \leq m(t)$ and $m(t) \leq 1 + m(s(t))$ we have $b - 1 \leq m(s(t))$. $h(t) = 2b$ implies $h(s(t)) < 2b$. □

Theorem 2.11. *Provided that the precondition $I0$ holds, procedure colour paints the nodes of tree t so that t becomes a red-black tree.*

Proof. Lemmas 2.7, 2.8, 2.9, and 2.10 imply that $I1$ and $I2$ are invariants of the program. $I1$ means that when we arrive at an external leaf, i.e. $t = NIL$, $0 \leq b \leq m(t) + 1 = -1 + 1$, as a result $b = 0$. In the program b is decreased (by 1), exactly when a node is painted black. Because b is decreased to zero on each branch of any subtree while we go to a NIL-leaf, we have the same number of black nodes on these paths. Lemma 2.7 implies that the root node of the tree is painted black. Lemma 2.10 makes sure that both children of a red node will also be black. These have the effect of receiving a red-black tree. □

Let a *crb-tree* be a *BST* which can be coloured as a *red-black tree*. Then an AVL tree is also a crb-tree. This also follows from both of the following results.

(1) Bayer proved that the class of *SBB-trees* properly contains the AVL trees [4], and we know from 4.7.2 in [3] that the SBB-trees and the red-black trees are structurally equivalent.

(2) Rank-balanced trees are a relaxation of AVL trees, and form a proper subclass of crb-trees [5].

Our achievements and these results are unrelated. In this subsection our contributions are the notion of the *minimal height of an AVL tree*, theorems 2.6 and 2.11, and our *efficient colouring algorithm* proved.

3. Conclusions

This was our question: How to transform a *strictly increasing list* L from one representation into another, *efficiently*?

Summarizing this paper, we already know that given an input representation of L, we can produce another representation of it in $\Theta(n)$ time, if this other representation is a linear data structure, an *AVL* or *red-black* tree. In some cases we have direct transformations, in other cases we need a temporary array.

In three cases we invented the necessary algorithms, theorems, lemmas, and proofs. The first two, (*sorted array* $\rightarrow$ *balanced BST*) programs create new trees; but the second half of the second algorithm, and the third (*AVL* tree $\rightarrow$ *red-black* tree) procedure do not make structural changes on the actual tree, just paint its nodes *black*, and *red*. Each of the three programs needs $\Theta(\log(n))$ working memory.

Our algorithms and theorems imply three relations among four classes of BSTs: *size-balanced BSTs* $\subset$ *almost complete BSTs* $\subset$ *AVL trees* $\subset$ *crb-trees*. Actually, each of the first three classes is a proper subclass of the next one. For example[2] BST (((1)2(3))4(5)) is almost complete, but not size-balanced; AVL tree ((((1)2)3(4))5(6(7))) is not almost complete; red-black tree ((1*b*)2*b*((3*b*)4*r*(5*b*(6*r*)))) is not height-balanced.

Open questions: If L is transformed into another type of balanced search trees (*not* into an *AVL* or *red-black* tree); for example, into a B-tree, we know that the operational complexity of the transformation is $\Omega(n)$, and $O(n\log(n))$. Here we still need more sharp results. Maybe, from a strictly increasing list, each kind of balanced search trees can be generated in $\Theta(n)$ time? Are there some cases, when the time complexity is more than $\Theta(n)$, but less than $\Theta(n\log(n))$?

If the input representation of L is a search tree (or a linked list or a sequential file), and the output is an *AVL* or *red-black* tree, we can make the transformation in $\Theta(n)$ time, but – with the exception of the *AVL tree* $\rightarrow$ *red-black tree* program

[2] Using the notation *(left-subtree root right-subtree)* where the empty subtrees are omitted.

– we actually need a temporary array, thus $\Theta(n)$ working space. We ask: In which cases can we reduce the memory needed?

References

[1] Weiss, Mark Allen, Data Structures and Algorithm Analysis, *Addison-Wesley*, 1995, 1997, 2007, 2012, 2013.

[2] Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C., Introduction to Algorithms, *The MIT Press*, 2009. (Ebook: `http://bit.ly/IntToAlgPDFFree`)

[3] Wirth, N., Algorithms and Data Structures, *Prentice-Hall Inc.*, 1976, 1985, 2004. (Ebook: `http://www.ethoberon.ethz.ch/WirthPubl/AD.pdf`)

[4] Bayer R., Symmetric Binary B-Trees: Data Structure and Maintenance Algorithms, *Acta Informatica 1*, 290–306 (1972), *Springer-Verlag*, 1972.

[5] Haeupler B., Sen S., Tarjan R.E., Rank-Balanced Trees, *Algorithms and Data Structures: 11th International Symposium, WADS 2009, Banff, Canada, August 2009*, pp 351–362, *Springer-Verlag, LNCS 5664*, 2009.

Annales Mathematicae et Informaticae
44 (2015) pp. 15–22
http://ami.ektf.hu

Power spectrum estimation of spherical random fields based on covariances*

Ágnes Baran, György Terdik

Faculty of Informatics, University of Debrecen
baran.agnes@inf.unideb.hu
terdik.gyorgy@inf.unideb.hu

Submitted September 15, 2014 — Accepted November 20, 2014

Abstract

A Gaussian isotropic stochastic field on a 2D-sphere is characterized by either its covariance function or its angular spectrum. The object of this paper is the estimation of the spectrum in two steps. First we estimate the covariance function, secondly we approximate the series expansion of the covariance function with respect of Legendre polynomials. Simulations show that this method is fast and precise.

Keywords: Angular correlation, angular spectrum, isotropic fields on sphere, estimation of correlation

MSC: 60G60, 62M30

1. Introduction

There are several physical phenomena which can be described with the help of a spherical random processes. A typical example of random data measured on the surface of a sphere is the cosmic microwave background radiation (CMB). Similar random fields arise in medical imaging, in analysis of gravitational and geomagnetic data etc.. These fields are characterized by a series expansion with respect to the spherical harmonics. Under assumption of Gaussianity both the covariance function and the angular power spectrum describe completely the probability structure of

*The publication was supported by the TÁMOP-4.2.2.C-11/1/KONV-2012-0001 project. The project has been supported by the European Union, co-financed by the European Social Fund.

an isotropic stochastic field. The estimated spectrum can be used to check the underlying physical theory, while the possible non-Gaussianity can be investigated by estimating the higher order angular spectra.

1.1. Notations

Let $\mathbb{S}_2$ denote the surface of the unit sphere in $\mathbb{R}^3$, and $X(L)$ be a random field on $\mathbb{S}_2$, where the location $L=(\vartheta,\varphi)$, and $\vartheta\in[0,\pi]$ is the co-latitude, while $\varphi\in[0,2\pi]$ is the longitude. If the spatial process $X(L)$ is mean square continuous, then it has a series expansion in terms of spherical harmonics Y_ℓ^m. Spherical harmonics are defied by the Legendre polynomials

$$P_\ell(x)=\frac{1}{2^\ell\ell!}\frac{d^\ell}{dx^\ell}(x^2-1)^\ell,\quad x\in[-1,1],$$

$(\ell=0,1,2,\dots)$ and the associated Legendre functions

$$P_\ell^m=(-1)^m(1-x^2)^{m/2}\frac{d^m}{dx^m}P_\ell(x),$$

of degree ℓ and order m, where $\ell=0,1,2,\dots,$ and $m=-\ell,\dots,\ell$. Now the complex valued spherical harmonics of degree ℓ and order m ($\ell=0,1,2,\dots,$ and $m=-\ell,\dots,\ell$) are given by

$$Y_\ell^m(\vartheta,\varphi)=\lambda_\ell^m(cos\vartheta)e^{im\varphi},$$

where

$$\lambda_\ell^m(x)=\sqrt{\frac{2\ell+1}{4\pi}\frac{(\ell-m)!}{(\ell+m)!}}P_\ell^m(x),\quad \text{if } m\geq 0,$$

and

$$\lambda_\ell^m(x)=(-1)^m\lambda_\ell^{|m|}(x),\quad \text{if } m<0,$$

that implies

$$Y_\ell^{-m}(\vartheta,\varphi)=(-1)^m\overline{Y_\ell^m}(\vartheta,\varphi).$$

Using these notations the spherical harmonics expansion of the random field $X(L)\in L^2(\mathbb{S}_2)$ is

$$X(L)=\sum_{\ell=0}^{\infty}\sum_{m=-\ell}^{\ell}Z_\ell^mY_\ell^m(L),$$

where the coefficients

$$Z_\ell^m,\quad \ell=0,1,\dots,\quad m=-\ell,\dots,\ell$$

are complex valued centered random variables, while putting $EZ_0^0=\mu$ implies that $EX(L)=\mu$ and the coefficients are given by

$$Z_\ell^m=\int_{\mathbb{S}^2}X(L)\overline{Y_\ell^m}(L)dL, \tag{1.1}$$

and

$$Z_\ell^m = (-1)^m \overline{Z_\ell^{-m}}.$$

2. Spectrum

Definition. The random field $X(L)$ is called strongly isotropic if all finite dimensional distributions of $\{X(L), L \in \mathbb{S}_2\}$ are invariant under the rotation g for every $g \in SO(3)$, where $SO(3)$ denotes the special orthogonal group of rotations defined on $\mathbb{S}_2$.

If the spatial process X is strongly isotropic, then

$$E(Z_{\ell_1}^{m_1} \overline{Z_{\ell_2}^{m_2}}) = f_{\ell_1} \delta_{\ell_1 \ell_2} \delta_{m_1 m_2}$$

for $\ell_1, \ell_2 \in \mathbb{N}$, and $m_i = -\ell_i, \ldots, \ell_i$, where $\delta_{\ell k} = 1$ if $\ell = k$ and zero otherwise, while

$$E(Z_0^0 \overline{Z_\ell^m}) = \left(f_0 + E(Z_0^0)^2\right) \delta_{0\ell} \delta_{0m}, \quad f_0 = Var\left(Z_0^0\right).$$

$f_\ell = Var\left(Z_\ell^m\right)$, $\ell = 0, 1, 2, \ldots$, are nonnegative real numbers, and $(f_\ell, \ell \in \mathbb{N}_0)$ is called the *angular power spectrum* of the random field X. Note $E(X) = \mu$, hence $C_2(L_1, L_2) = E(X(L_1) - \mu)(X(L_2) - \mu)$ is the covariance function of the isotropic field $X(L)$. Due to the isotropy the covariance $C_2(L_1, L_2)$ depends on the angular distance γ of the locations L_1 and L_2 only (where $\cos\gamma = L_1 \cdot L_2$). That means

$$C_2(L_1, L_2) = C_2(g_{L_2 L_1} L_1, N) =: C(\cos\gamma),$$

where $g_{L_2 L_1}$ is the rotation which takes L_2 into the north pole N and L_1 into the plane xOz.

It is straightforward (see [5]) that

$$C(\cos\gamma) = \sum_0^\infty f_\ell \frac{2\ell + 1}{4\pi} P_\ell(\cos\gamma). \tag{2.1}$$

For the practical computation of the spectrum f_k the orthogonality of the Legendre polynomials can be used: with $t = \cos(\gamma)$ from (2.1) follows

$$\int_{-1}^{1} C(t) P_\ell(t) dt = f_\ell \frac{2\ell + 1}{4\pi} \int_{-1}^{1} [P_\ell(t)]^2 dt = f_\ell \cdot \frac{1}{2\pi},$$

that is

$$f_\ell = 2\pi \int_{-1}^{1} C(t) P_\ell(t) dt, \tag{2.2}$$

for $\ell = 0, 1, 2, \ldots$.

Example (Lapalce-Beltrami model on $\mathbb{S}_2$)**.** Consider the homogeneous isotropic field X on $\mathbb{R}^3$ according to the equation

$$\left(\triangle - c^2\right) X = \partial W,$$

where $\triangle = \frac{\partial^2}{\partial x_1^2} + \frac{\partial^2}{\partial x_2^2} + \frac{\partial^2}{\partial x_3^2}$, denotes the Laplace operator on $\mathbb{R}^3$. Its spectrum, see ([6]), is

$$S(\lambda) = \frac{2}{(2\pi)^2} \frac{\lambda^2}{\left(\lambda^2 + c^2\right)^2}, \quad \lambda^2 = \|(\lambda_1, \lambda_2, \lambda_3)\|^2,$$

with covariance of Matérn Class

$$\mathcal{C}(r) = \frac{1}{(2\pi)^{3/2}} \frac{(cr)^{1/2} K_{1/2}(cr)}{2c},$$

where $K_{1/2}$ is the modified Bessel (Hankel) function, see [1].
Now according to the Lapalce-Beltrami operator, which is the restriction of $\triangle$ onto the unit sphere $\mathbb{S}_2$,

$$\triangle_B = \frac{1}{\sin\vartheta} \frac{\partial}{\partial\vartheta} \left(\sin\vartheta \frac{\partial}{\partial\vartheta}\right) + \frac{1}{\sin^2\vartheta} \frac{\partial^2}{\partial\varphi^2},$$

we consider the stochastic model

$$\left(\triangle_B - c^2\right) X_B = \partial W_B,$$

on sphere. The covariance function $\mathcal{C}_0$ of X_B is the restriction of the covariance function $\mathcal{C}$ of X on sphere and $\mathcal{C}_0(\cos\gamma) = \mathcal{C}(2\sin(\gamma/2))$, i.e.

$$\mathcal{C}_0(\cos\gamma) = \frac{1}{(2\pi)^{3/2}} \sqrt{\frac{\sin(\gamma/2)}{2c}} K_{1/2}(2c\sin(\gamma/2)).$$

We apply the Poisson formula when $\Phi(d\lambda) = S(\lambda)\, d\lambda$, and we obtain the spectrum for X_B

$$\begin{aligned} f_\ell &= 2\pi^2 \int_0^\infty J_{\ell+1/2}^2(\lambda) \frac{1}{\lambda} \frac{2}{(2\pi)^2} \frac{\lambda^2}{\left(\lambda^2 + c^2\right)^2} d\lambda \\ &= \int_0^\infty J_{\ell+1/2}^2(\lambda) \frac{\lambda}{\left(\lambda^2 + c^2\right)^2} d\lambda. \end{aligned}$$

3. HEALPix

The most widely used pixelisation of the sphere for sampling and analyzing CMB data is the HEALPix (Hierarchical, Equal Area and isoLatitude Pixelization), see

[2]. Actually the CMB data are given on the surface of a unit ball at the discrete points defined by HEALPix. Here in the base resolution partitioning the surface of the sphere is divided into 12 quadrilateral pixels of same area, and in each further resolution the pixels are subdivided into 4 equal area pixels. Denoting by N_{side} the resolution parameter, the total number of pixels equals $12N_{side}^2$, and the pixel centers are located on $4N_{side} - 1$ isolatitude rings. Unfortunately, the pixelisation is not rotational invariant, the pixel centers can be rotated into each other in the case of some rotations around the north-south axes only.

4. Computational results

Let us suppose that we are given an observation of an isotropic field on the sphere, more precisely for each HEALPix pixel L we have a value $X(L)$. The estimator of the spectrum of the field can be based either on (1.1) or on (2.1). It means that we can approximate the integral (1.1), then for each fixed ℓ we estimate f_ℓ as the variance of approximated Z_ℓ^m, $m = -\ell, \dots, \ell$. In this case one can not expect good result for small ℓ, since the estimator of the variance f_ℓ based on $2\ell + 1$ values. The alternative method is based on the estimation of covariance function first then use the expansion (2.1) according to the Legendre polynomials for estimating f_ℓ. The advantage of this later one is that there are many distances between pixels in which the estimation of the covariance is possible.

For further improvement of this computations we are going to apply some sampling theorems concerning on spherical harmonics and Legendre polynomials. We show this method through simulations.

In our simulations we consider random fields not only with zero mean but with $f_0 = 0$ as well. The reason is that we have only one realization and when we center the observation the sample mean contains a value of Z_0^0 hence f_0 can not be identified.

To the numerical approximation of the integral (2.2) denote by $t_1, t_2, \dots, t_n$ the nodes of the quadrature ($-1 \le t_i \le 1$), and for a given i let (L_{1j}^i, L_{2j}^i), $j = 1, \dots, N$, be pairs of pixels which have angular distance t_i. Considering the samples

$$X_1, X_2, \dots, X_N, \quad \text{where} \quad X_j = X(L_{1j}^i),$$

and

$$Y_1, Y_2, \dots, Y_N, \quad \text{where} \quad Y_j = X(L_{2j}^i),$$

we use the empirical covariance

$$\widehat{C}_i = \frac{1}{N} \sum_{j=1}^{N} X_j Y_j$$

to estimate the value $C(t_i)$, $i = 1, \dots, n$.

In the program we used only pixels located in the equatorial area (i.e. pixel centers with co-latitude $-\frac{1}{3} \le \cos\vartheta \le \frac{1}{3}$). E.g. in the case of $N_{side} = 16$ these

pixels determine nearly 9000 different values for t. In the equatorial zone each ring contains the same number of pixels ($4N_{side}$), moreover the pixel centers are equidistant located. In order to calculate the possible values of $t = \cos\gamma$ we considered the first pixel center on each ring in the north equatorial belt together with the pixel centers located on and below the actual ring. More precisely it is suffices to consider on each ring only the half of the pixels. After that for a given t to collect the pixel-pairs having distance t we can use the rotation symmetry. Depending on the location of the original pixel-pair (L_1, L_2), which was used to compute t, there exist $4N_{side}$, $8N_{side}$ or $16N_{side}$ pairs having the given distance. If both of the pixels lie on the equator, or $\theta_1 = \pi - \theta_2$ and $\varphi_1 = \varphi_2$ (where $L_1 = (\theta_1, \varphi_1)$, $L_2 = (\theta_2, \varphi_2)$), that is the locations are symmetric to the equator, then the number of pairs is equal to $4N_{side}$. In the case of $\theta_1 = \theta_2 \neq \frac{\pi}{2}$ and in the case of $\theta_1 \neq \pi - \theta_2$ and $\varphi_1 = \varphi_2$, moreover if $\theta_1 = \pi - \theta_2$ and $\varphi_1 \neq \varphi_2$, there are $8N_{side}$ pairs. In all other cases there exist $16N_{side}$ pairs corresponding to the given distance.

By the numerical calculation of (2.2) using the Gaussian quadrature instead of the built-in Matlab function *trapz* enables a more efficient calculation, since these method requires much less evaluations of empirical covariances, however, this could be subject of further investigations.

Test example 1. (See Figure 1.) As a first example we considered the spatial process

$$X(L) = \sum_{\ell=1}^{100} \sqrt{f_\ell} \sum_{m=-\ell}^{\ell} Z_\ell^m Y_\ell^m(L), \tag{4.1}$$

where $Z_\ell^m \sim \mathcal{N}(0,2)$ are i.i.d. random numbers and

$$f_\ell = \frac{1}{(\ell(\ell+1)+4)^2}, \quad \ell = 1, 2, \dots$$

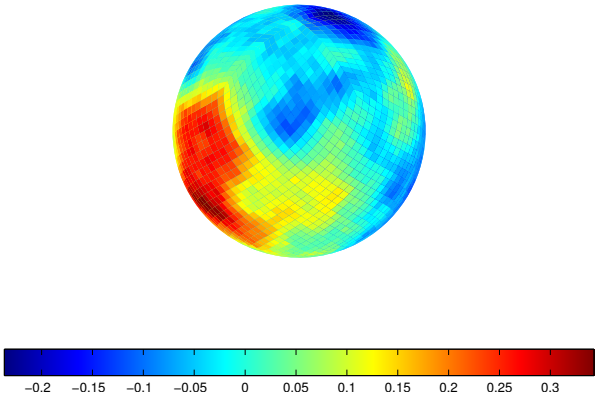


Figure 1: A random field described in Test example 1

By the discretization of the sphere we used $N_{side} = 16$ as resolution parameter, which results 3072 pixels located on 63 isolatitude rings.

The estimated and theoretical correlations can be seen on Figure 2 such that $f_0 = f_1 = 0$.

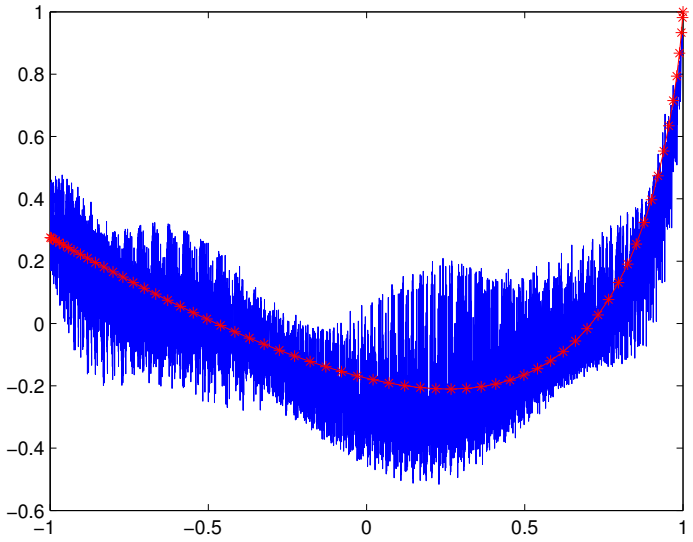


Figure 2: Estimated correlation in Test example 1

Let us denote by $\hat{f}_\ell$, $\ell = 1, \ldots, 100$ the estimated spectrum, then we obtained

$$\sum_{\ell=1}^{100} (f_\ell - \hat{f}_\ell)^2 \approx 1.93 \cdot 10^{-4}$$

and

$$\max_{1 \le \ell \le 100} |f_\ell - \hat{f}_\ell| = 4.2 \cdot 10^{-3}.$$

Test example 2. In the second example we investigated the field defined by the sum (4.1) taking

$$f_\ell = \frac{4\pi}{2\ell + 1} 0.8^\ell, \quad \ell = 1, 2, \ldots$$

The covariance is estimated from the generated field, and the theoretical correlation

$$C(\gamma) = \frac{1}{\sqrt{1 - 1.6 \cos\gamma + 0.8^2}} - 1;$$

are shown on Figure 3.

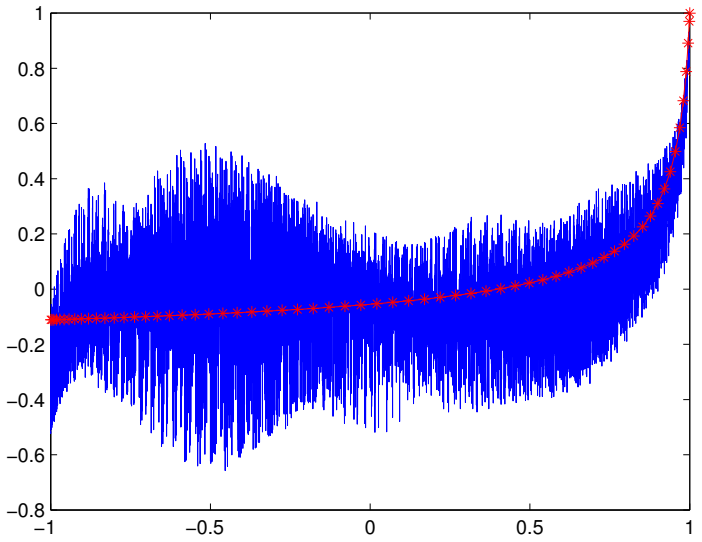


Figure 3: Estimated correlation in Test example 2

References

[1] Abramowitz, M., Stegun, I.A., Handbook of mathematical functions with formulas, graphs and mathematical tables, *Dover Publications Inc.*, New York, (1992)

[2] Górski, K.M., Hivon, E., Banday, A. J., Wandelt, B.D., Hansen, F.K., Reinecke, M., Bartelmann, M., HEALPix: A framework for high-resolution discretization and fast analysis of data distributed on the sphere, *The Astrophysical Journal*, Vol. 622 (2005), 759–771.

[3] Lang, A., Schwab, C., Isotropic Gaussian Random Fields on the Sphere, Regularity, Fast Simulation, and Stochastic Partial differential Equations, *arXiv:*1305.1170v1

[4] Marinucci, D., Peccati, G., Random fields on the Sphere, *Cambridge University Press*, Cambridge, (2011)

[5] Terdik, Gy., Angular Spectra for non-Gaussian Isotropic Fields, *arXiv:*1302.4049v2, to appear Brazilian Journal of Probability and Statistics, `http://imstat.org/bjps/papers/BJPS249.pdf`

[6] Yadrenko, M. I., Spectral theory of random fields, *Optimization Software Inc. Publications Division*, New York, (1983)

Annales Mathematicae et Informaticae
44 (2015) pp. 23–33
http://ami.ektf.hu

Component visualization methods for large legacy software in C/C++

Máté Cserép[a], Dániel Krupp[b]

[a]Eötvös Loránd University
mcserep@caesar.elte.hu

[b]Ericsson Hungary
daniel.krupp@ericsson.com

Submitted August 28, 2014 — Accepted February 24, 2015

Abstract

Software development in C and C++ is widely used in the various industries including Information Technology, Telecommunication and Transportation since the 80-ies. Over this four decade, companies have built up a huge software legacy. In many cases these programs, implementing complex features (such as OS kernels, databases) become inherently complicated and consist of millions lines of code. During the many years long development, not only the size of the software increases, but a large number (i.e. hundreds) of programmers get involved. Mainly due to these two factors the maintenance of software becomes more and more time consuming and costly.

To attack the above mentioned complexity issue, companies apply various source code cross-referencers to help in the navigation and visualization of the legacy code. In this article we present a visualization methodology that helps programmers to understand the functional dependencies of artifacts in the C++ code in the form similar to UML component diagrams. Our novel graph representation reveals relations between binaries, C/C++ implementation files and headers. Our technique is non-intrusive. It does not require any modification of the source code or any additional documentation markup. It solely relies on the compiler generated Abstract Syntax Tree and the build information to analyze the legacy software.

Keywords: code comprehension, software maintenance, static analysis, component visualization, graph representation, functional dependency

MSC: 68N99

1. Introduction

One of the main task of code comprehension software tools is to provide navigation and visualization views for the reusable elements of the source code, because humans are better at deducing information from graphical images [2, 7]. We can identify reusable software elements in C/C++ language on different abstraction levels of modularity. At a finer granularity, functions provide reusable implementation of a specific behavior, while on a higher scale (in C++) classes defines the next level, where a programmer can collect related functions and data that belong to the same subject-matter. At the file level, header files group related functions, variables, type declarations and classes (in C++ only) into a semantic unit.

State of the art software comprehension and documentation tools implement various visualization methods for all of these modularization layers. For example, on the function level call graph diagrams can show the relations between the caller and the called functions [9], while on the class level, one can visualize the containment, inheritance and usage relations by e.g. UML diagrams. On the file level, header inclusion diagrams help the developers in code comprehension [8].

However, our observations showed that the state of the art file level diagrams are not expressive enough to reveal some important dependency relationships among implementation and header files. In this paper, we describe a new visualization methodology that exposes the relations between implemented and used header files and the source file dependency chains of C/C++ software.

This paper is structured as follows. Section 2 consist of a brief overview of the state of the art literature with special focus on static software analysis. In Section 3 we describe the shortfalls of the visualization methods in the current software comprehension tools, then in Section 4 we present our novel views that can help C and C++ programmers in understanding legacy source code. Section 5 demonstrates our results by showing examples on real open-source projects, and finally in Section 6 we conclude the paper and set the directions for future work.

2. Background

Researchers have proposed several software visualization techniques and various taxonomies have been published over the past years. They address one or more of three main aspects (static, dynamic, and evolutional) of a software. The visualization of the static attributes focuses on displaying the software at a snapshot state, dealing only with the information that is valid for all possible executions of the software, assisting the comprehension of the architecture of the program. Conversely, the visualization of the dynamic aspects shows information about a particular execution of the software, therefore helps to understand the behavior of the program. Finally, the visualization of the evolution – of the static aspects – of a software handles the notion of time, visualizing the alternations of these attributes through the lifetime of the software development. For a comprehensive summary of the current state of the art see the work of Caserta et al.[3]

The static analysis of a software can be executed on different levels of granularity based on the level of abstraction. Above a basic source code level, a middle – package, class or method – level, and an even higher architecture level exists. In each category a concrete visualization technique can focus on various different aspects. A summary of categorization is shown on Table 1, classifying some of the most known and applied, as well as a few interesting new visualization techniques.

Kind	Level	Focus	Techniques
Time T Visualization	Line	Line properties	Seesoft
	Class	Functioning, Metrics	Class BluePrint
	Architecture	Organization	Treemap
		Relationship	Dependency Structure Matrix
			UML Diagrams
			Node-link Diagrams
			3D Clustered Graphs
Visualizing Evolution			

Table 1: Categorization of visualization tools

This article focuses on assisting the code comprehension through visualizing the relationships between architectural components of a software. The relevant category not only contains various prevalent and continuously improved visualizing techniques like the *UML diagrams* [4], but also recently researched, experimental diagrams like the three dimensional clustered graphs [1]. This technique aims to visualize large software in an integral unit, by generating graphs in a 3D space and grouping remote vertices and classes into clusters. The visibility of the inner content of a cluster depends dynamically on the viewpoint and focus of the user who can traverse the whole graph.

Our novel solution uses the classical node-link diagram in two dimensional space for visualization, which was formerly used at lower abstraction levels primarily.

3. Problems of visualization

Modularity on the file level of a software implementation in C/C++ is expressed by separating interfaces and definition to header and implementation (source) files. Interfaces typically contain macro and type definitions, function and member declarations, or constant definitions; while implementation files usually contain the definition of the functions declared in the headers. This separation allows the programmers to define reusable components in the form of – static or dynamic – libraries. Using this technique, the user of a library does not need to have information about the implementation details in order to use its provided services.

Separation of these concerns is enforced by the C/C++ preprocessor, compiler and linker infrastructure. When a library is to be used, its header file should be

included (through the `#include` preprocessor directive) by the client implementation or the header files. Implementation files should almost never[1] be included in a project where the specification and implementation layers are properly separated. Unfortunately naming conventions of the header and implementation files in C/C++ are not constrained (like calss and file-naming in Java). Thus, based on a name of a file, it is not possible to find out the location, where the methods of a class are declared or implemented. Furthermore, the implementation of the class members declared in a header file can be scattered through many implementation files that makes the analysis even more difficult.

When a programmer would like to comprehend the architecture of a software, the used and provided (implemented) interface of a library component or the implementers of a specific interface should be possible to be fetched.

Problem 3.1. As an example let us analyze the commonly presented header inclusion graph of a fileset in Figure 1. We assume that `lib.h` is an interface of a software library and that there are many users of this component, thus several files includes this header. If the programmer would like to comprehend where the functions declared in the header are implemented, the header inclusion graph is not helpful, since it does not unveil which C/C++ files are only using, and which are implementing the `lib.h` interface.

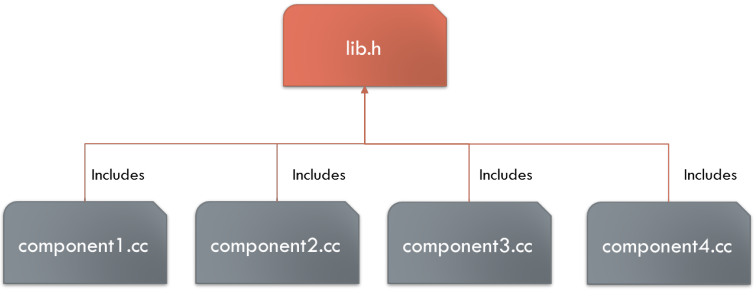


Figure 1: Implementation decision problem between component(s) and an interface.

As a solution we propose a so-called *Interface diagram* that is similar to the well-known header inclusion graph, but refines the include relation into *uses* and *provides* relationships. For this purpose we defined that a C/C++ file provides a header file when it contains its implementation, while it only uses it if the mentioned file refers to at least one symbol in the header, but does not implement any of them. A proper and precisely defined description of this diagram is given in Section 4.2.

[1] A few exceptions may exist, i.e. in some rare cases of template usage.

4. Definition of relationships and diagrams

In this section first we introduce the commonly used basic terms of relationships defined between the C/C++ source files and the binary objects (see Figure 2), then present our more complex relationship definitions to describe the connections between the various kind of files in a software project at a higher abstraction level.

4.1. Preliminaries

Definition 4.1 (Relations between source files)**.** At the level of the *abstract syntax tree* [6], the main artifacts of a C/C++ source code are the user defined symbols[2], which can be declared, defined or referred/used by either the source files (`.c/.cc`) or the header files (`.h/.hh`). A C/C++ symbol might have multiple declarations and references, but can be defined only once in a semantically correct source code. To enforce the separation of the specification and implementation layer, header files should mainly consist of declarations, whose definitions are in the appropriate source files.[3] From our perspective only those C/C++ symbols are important, which are declared in a header file and are defined or referred by a source file.

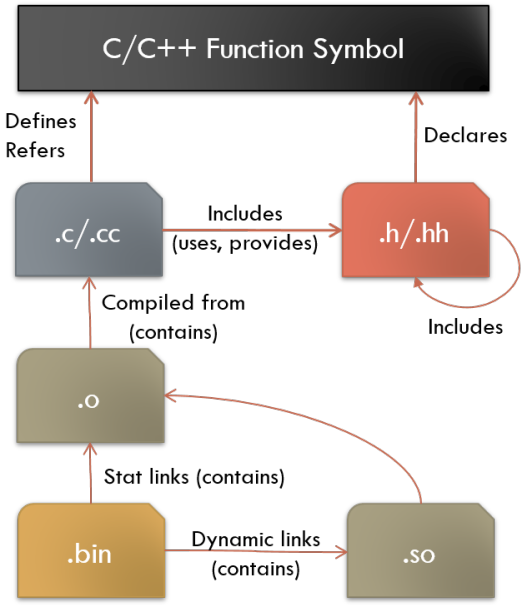


Figure 2: Relations between compilation artifacts.

[2]From our viewpoint only the function (and macro) symbols and their declaration, definition and usage are significant, although a similar classification for other symbol types can be established without difficulties.

[3]In some cases, headers may contain definition and source file may also consist forward declarations as an exception.

Definition 4.2 (Relations between binaries)**.** The source files of a project are *compiled* into object files, which are then *statically linked* into archive files (.lib/.a), shared objects or executable binaries. Shared objects are *linked dynamically* into the executables at runtime. To extract this information and visualize the relationship of binaries together with the relations declared between the C/C++ files, the analysis of the *compilation procedure* of the project is required beside the static analysis of the source code.

For the purpose of the presented visualization views in this paper the different kind of binary relationships is irrelevant, therefore they will be collectively referred as the *contains* relation henceforward (see Figure 2).

4.2. Extended classification

The basic include relationship among the implementation and header C/C++ files have already been introduced in Section 4.1, however in order to solve the problem raised in Section 3, the definitions of the proposed uses and provides relations have to be separated.

Definition 4.3 (Provides relationship from implementation c to header h)**.** We say that in a fileset a implementation file c *provides* the interface specified by the header file h, when c directly includes h and a common symbol s exists, for which h contains the declaration, while c consists the definition of it.

Definition 4.4 (Uses relationship from implementation c to header h)**.** Similarly to the previous provides relationship definition, we state that in a fileset an implementation file c *uses* the interface specified by the header file h, when c directly includes, but does not provides h and a common symbol s exists, which c refers and h contains the declaration of it.

Definition 4.5 (**Interface graph** (diagram) of implementation file c)**.** Let us define a graph with the set of nodes N and set of edges E. Let P be the set of header files which are *provided* by c and U the set of header files used by c, and B the set of binary files which *contain* c. We define that N consists of c, the elements of P, U, and B. E consists the corresponding edges to represent the relationships between the nodes in N.

Figure 2 shows the illustration for the above mentioned definitions. Based on the idea of the *Interface diagram* defined in Definition 4.5, which shows the immediate *provides*, *uses* and *contains* relations of the examined file, we defined the following more complex file-based views.

The nodes of these diagrams are the files themselves and the edges represent the relationships between them. A labeled, directed edge is drawn between two nodes only if the corresponding files are in either *provides*, *uses* or *contains* relationship. The label of the edges are the type of their relationship and they have the same direction as the relation they represent.

Definition 4.6 (**Used components graph** (diagram) of source c)**.** Let us define a graph with the set of nodes N and set of edges E, and let S be the set of implementation files which *provides* an interface directly or indirectly *used by* c. We define that N consists of c, the elements of S and the files along the path from c to the elements of S. Binaries containing any implementation file in S are also included in N. E consists the corresponding edges to represent the relationships between the nodes in N.

Intuitively we can say if source t is a used component of c, then c is using some functionality defined in t.

Definition 4.7 (**User components graph** (diagram) of source c)**.** Let us define the graph with the set of nodes N and set of edges E, and similarly to the previous definition, let S be the set of implementation files which directly or indirectly *uses* the interface(s) *provided by* c. We define that N consists of c, the elements of S and the files along the path from c to the elements of S. Binaries containing any implementation file in S are also included in N. E consists the corresponding edges to represent the relationships between the nodes in N.

Intuitively we can say if source t is a user component of c, then c is providing some functionality used by t.

5. Experimental results

In order to implement the views defined in Section 4, we created a diagram visualizing tool was created as part of a larger code comprehension supporting project – named *CodeCompass*. The software is developed in cooperation at Eötvös Loránd University and Ericcson Hungary. The tool provides an interactive graph layout interface, where the users are capable of requesting more information about the nodes representing files and can also easily navigate between them, switching the perspective of the view they are analyzing.

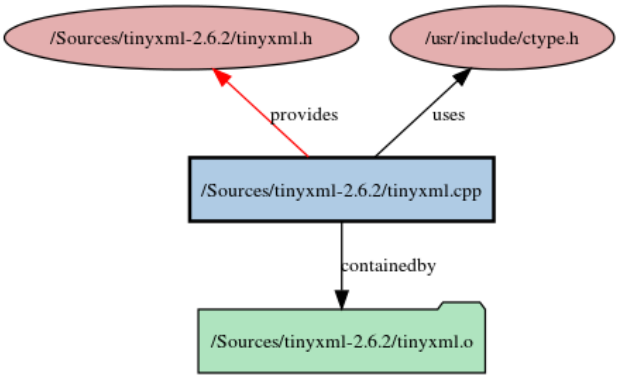


Figure 3: Interface diagram of `tinyxml.cpp`.

For demonstration purposes in this paper, the open-source *TinyXML* parser project[10] was selected. In this section altogether three examples for the use of our tool is shown and information retrievable from them is examined.

Example 5.1. Figure 3 displays an *Interface diagram*, showing the immediate relations of a selected file with other files in the software. As the image shows, the C++ implementation file in the middle (`tinyxml.cpp`) includes two header files, but the special connection of implementation (provides) is distinguished from the mere uses relation. This diagram not only presents the connections between C++ source and header files, but also displays in which object file the focused implementation file was compiled into through the compilation process of the project.

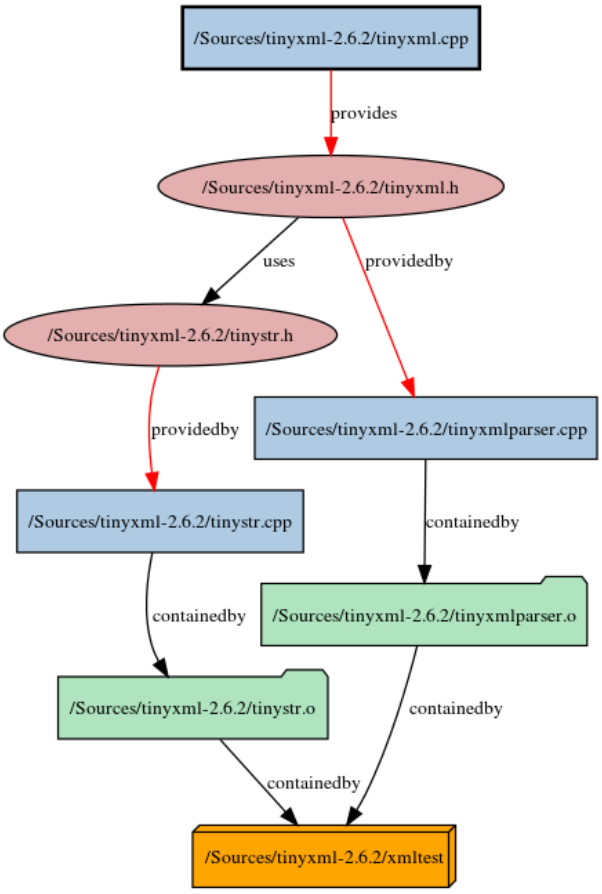


Figure 4: Used components by `tinyxml.cpp`.

Example 5.2. Figure 4 presents the *Used components diagram* of the implementation file `tinyxml.cpp` at the top. The goal of this visualization is to determine

which other files and compilation units the selected file depends on. As it is depicted in the figure, the interface specification for the `tinyxml.cpp` implementation file is located in the `tinyxml.h` header. This header file on the one part is provided by the `tinyxmlparser.cpp`, and on the other hand uses the `tinystr.h`. The latter header file is provided by the `tinystr.cpp` source. Hence the implication can be stated that the original `tinyxml.cpp` indirectly uses and depends on the `tinyxmlparser.cpp` and the `tinystr.cpp` file.

Example 5.3. Parallel to Figure 4, the following example deduces the compilation artifacts depending on the same selected source `tinyxml.cpp`. The *User components diagram* displays (see Figure 5) that this implementation file implements an interface contained by the `tinyxml.h` header. This header is used or provided by three sources (`tinyxmlparser.cpp`, `xmltest.cpp` and `tinyxmlerror.cpp`), therefore they are the users of `tinyxml.cpp`.

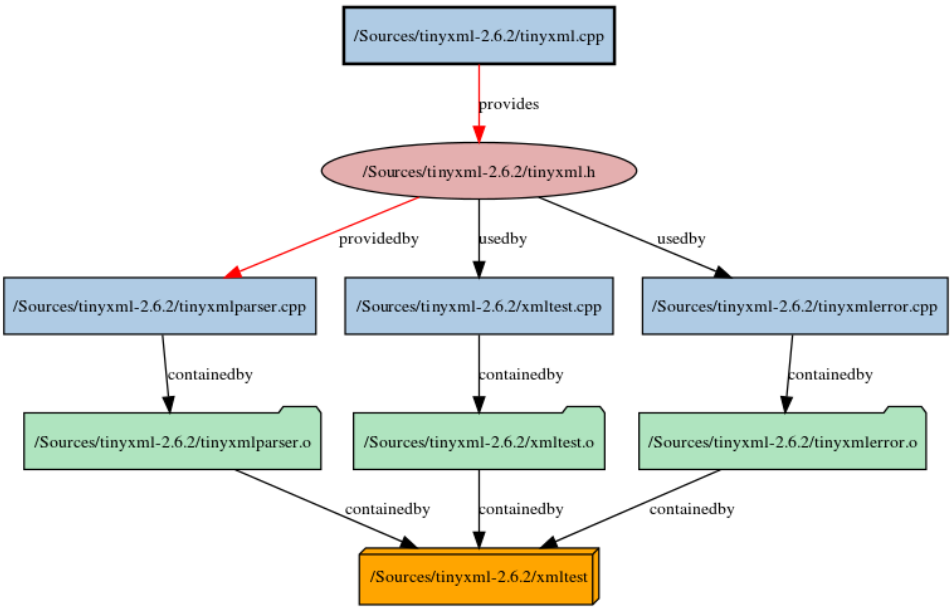


Figure 5: User components of `tinyxml.cpp`.

6. Conclusions and future work

In large legacy software projects a huge codebase can easily be built up because of the extended development time, while fluctuation among programmers can also often be a significant problem. Code comprehension support addresses these questions through assisting – both experienced and newcomer – developers with visualization views to better understand the source code. In this paper we discussed

what kind of file-level views are missing from the current code comprehension tools, regarding the relationships between different type of compilation artifacts. We defined our novel graph view as a solution to this problem, and demonstrated the practical use of our technique through examples on an open-source C++ project. The new visualization techniques were found helpful and applicable for legacy software in supporting code comprehension.

Above the file-level granularity, a higher level of modularity can also be defined, considering that related files can form the interface of a reusable binary component and are often grouped together physically (i.e. contained in a directory) or virtually (e.g. using packages or namespaces). Future development will generalize and expand the file-based dependency relationship definitions introduced in this paper to be applicable for modules containing multiple files.

Further work will also include the examination of how the information retrieved by our definition rules can be used in the field of architecture compliance checking. Software systems often impose constraints upon the architectural design and implementation of a system, for example on how components are logically grouped, layered and how they may interact with each other. In order to keep the maintainability of a software system through a long development time with a large programmer team, it bears extreme importance that the design and implementation are compliant to its intended software architecture. Due to the complexity of large software systems, guaranteeing the compliance by manual checking is almost impossible, hence automated support is required, which is still not a completely solved issue nowadays [5].

References

[1] Balzer, M., Deussen, O., Level-of-detail visualization of clustered graph layouts, *Proceedings of the 6th International Asia-Pacific Symposium on Visualization*, (2007), 33–140.

[2] Biederman, I., Recognition-by-components: a theory of human image understanding, *Psychological review*, Vol. 94 (1987), 115–147.

[3] Caserta, P., Zendra, O., Visualization of the static aspects of software: a survey, *IEEE Transactions on Visualization and Computer Graphics*, Vol. 17 (2011), 913–933.

[4] Gutwenger, C., Jünger, M., Klein, K., Kupke, J., Leipert, S., Mutzel, P., A new approach for visualizing UML class diagrams, *Proceedings of the 2003 ACM Symposium on Software Visualization*, (2003), 179–188.

[5] Pruijt, L., Koppe, C., Brinkkemper, S.,, On the accuracy of architecture compliance checking support: Accuracy of dependency analysis and violation reporting, *IEEE 21st International Conference on Program Comprehension*, (2013), 172–181.

[6] Salomaa, A., Formal Languages, *Academic Press Professional, Inc.*, (1987).

[7] Spence, I., Visual psychophysics of simple graphical elements, *Journal of Experimental Psychology: Human Perception and Performance*, Vol. 16 (1990), 683–692.

[8] Doxygen Tool: `http://www.stack.nl/~dimitri/doxygen/`.

[9] Understand Source Code Analytics & Metrics, `http://www.scitools.com/`.

[10] TinyXML parser, `http://www.grinninglizard.com/tinyxml/`.

Annales Mathematicae et Informaticae
44 (2015) pp. 35–46
http://ami.ektf.hu

Solving computational problems in real algebra/geometry

James H. Davenport*

University of Bath (U.K.)
J.H.Davenport@bath.ac.uk

Submitted September 16, 2014 — Accepted January 26, 2015

Abstract

We summarise some computational advances in the theory of real algebra/geometry over the last 15 years, and list some areas for future work.

Keywords: Real algebraic geometry, cylindrical algebraic decomposition

1. Introduction

Real Algebra and Geometry have many computational applications. In theory they solve many problems of robot motion planning [34], though practice is not so kind [38]. Understanding the (real) geometry of branch cuts [16] is crucial to issues of complex function simplification [3]. A very powerful technique in computation real geometry is *Cylindrical Algebraic Decomposition*, introduced in [12] to solve problems of Quantifier Elimination.

Notation 1.1. We assume that the initial problem posed has m polynomials, of degree (in each variable separately) at most d, in n variables.

*Thanks to Russell Bradford, Matthew England, Nicolai Vorobjov, David Wilson (Bath); Chris Brown (USNA); Scott McCallum (Macquarie); Marc Moreno Maza (UWO) and Changbo Chen (CIGIT), and to the referees. This talk grew out of an invitation to speak at ICAI 2014 in Eger, and the author is grateful to the organisers. The underlying work was supported by EPSRC grant EP/J003247/1.

2. Quantifier elimination

A key technique we are going to use is *Quantifier Elimination*: throughout, $Q_i \in \{\exists, \forall\}$. Given a statement

$$\Phi := Q_{k+1}x_{k+1} \dots Q_n x_n \phi(x_1, \dots, x_n),$$

where ϕ is in some (quantifier-free, generally Boolean-valued) language $\mathcal{L}$, the Quantifier Elimination problem is that of producing an equivalent

$$\Psi := \psi(x_1, \dots, x_k) : \qquad \psi \in \mathcal{L}.$$

In particular, $k = 0$ is a decision problem: is Φ true?

The Quantifier Elimination problem is critically dependent on the language $\mathcal{L}$ and the range of the variables x_i. For example

$$\forall n : n > 1 \Rightarrow \exists p_1 \exists p_2 \left(p_1 \in \mathcal{P} \wedge p_2 \in \mathcal{P} \wedge 2n = p_1 + p_2\right)$$
$$[\text{where } m \in \mathcal{P} \equiv m > 1 \wedge \forall p \forall q \left(m = pq \Rightarrow p = 1 \vee q = 1\right)]$$

is a statement of Goldbach's conjecture in the language of the natural numbers with, naïvely, seven quantifiers (five will do if we use the same quantifiers for the two instances of $\mathcal{P}$).

In fact, quantifier elimination is impossible over the natural numbers [29]. From this it follows that it is impossible over the real numbers if we allow unrestricted[1] trigonometric transcendental functions in $\mathcal{L}$, since $n \in \mathbf{Z}$ is equivalent to $\sin(n\pi) = 0$. The function sin satisfies a second-order (or coupled pair of first-order) differential equation(s), and there are positive results provided we restrict ourselves to Pfaffian functions, i.e. solutions of triangular systems of first-order partial differential equations with polynomial coefficients. Exploring this is beyond the scope of this paper: see [24].

However, quantifier elimination is possible for semi-algebraic (polynomials and inequalities) $\mathcal{L}$ over $\mathbf{R}$ [35]. Note that we need to allow inequalities: the quantifier-free form of $\exists y : y^2 = x$ is $x \geq 0$, and $\exists y : xy = 1$ has the quantifier-free form $x \neq 0$. Formally we define the language of *real closed fields*, $\mathcal{L}_{\mathrm{RCF}}$, to include the natural numbers, $+, -, \times, =, >$ and the Boolean operators. Then $\exists y : y^2 = x$ eliminates the quantifier to $(x > 0) \vee (x = 0)$ and $\exists y : xy = 1$ eliminates the quantifier to $(x > 0) \vee (0 > x)$.

It is worth noting that in practice we nearly always treat $\neq$ as a first-class citizen, and indeed this is necessary when proceeding via regular chains (Section 3.3).

3. Cylindrical algebraic decomposition

Unfortunately, the complexity of Tarski's method is indescribable (in the sense that no tower of exponentials can describe it) and we had to wait for [12] for a remotely

[1] Note that the undecidability comes from the fact that the function $\sin : \mathbf{R} \to \mathbf{R}$ has infinitely many zeros. Restricted versions are a different matter: see [24, (h) p. 214].

plausible method.

3.1. Collins' method

Collins proceeds via a *cylindrical algebraic decomposition*, which is (almost) what it says: a decomposition of $\mathbf{R}^n$ into cells $C_{\mathbf{i}}$ indexed by n-tuples of natural numbers (so $\mathbf{R}^n = \bigcup_{\mathbf{i}} C_{\mathbf{i}}$ and $\mathbf{i} \neq \mathbf{j} \Rightarrow C_{\mathbf{i}} \cap C_{\mathbf{j}} = \emptyset$), which is (semi-)algebraic in the sense that every $C_{\mathbf{i}}$ is defined by a finite set of equalities and inequalities of polynomials in the x_i and which is *cylindrical*, meaning that, for all $k < n$, if π_k is the projection operator onto the *first* k coordinates, then, for all $\mathbf{i}, \mathbf{j}$, $\pi_k(C_{\mathbf{i}})$ and $\pi_k(C_{\mathbf{j}})$ are either equal or disjoint. Collins constructs a cylindrical algebraic decomposition which is *sign-invariant* for the polynomials in ϕ, i.e. on each cell, every polynomial is identically zero, or everywhere positive, or everywhere negative.

The construction and use of such a decomposition is roughly[2] described below.

1 Let $\mathcal{S}_n$ be the polynomials in ϕ (m polynomials, degree $\leq d$, n variables).

2 Compute $\mathcal{S}_{n-1}$ ($\Theta(d^3m^2)$ polynomials, degree $\Theta(2d^2)$, $n-1$ variables), such that, over a cylindrical algebraic decomposition of $\mathbf{R}^{n-1}$ sign-invariant for the polynomials in $\mathcal{S}_{n-1}$, the polynomials in $\mathcal{S}_n$ are collectively delineable, meaning each branch of each of them is defined by a continuous algebraic function of $x_1, \ldots, x_{n-1}$, and the branches of all polynomials are either equal or disjoint;

3 and $\mathcal{S}_{n-2}$ ($\Theta((2d^2)^3(d^3m^2)^2)$ polynomials, degree $\Theta(2(2d^2)^2)$, $n-2$ variables) satisfying a similar condition;

$\vdots$ continue

n and $\mathcal{S}_1$ ($\leq (2d)^{3^n} m^{2^{n-1}}$ polynomials, degree $\leq \frac{1}{2}(2d)^{2^{n-1}}$, 1 variable) satisfying a similar condition.

$n+1$ Isolate the N_1 roots of S_1, decomposing $\mathbf{R}^1$ into N_1 zero-dimensional points and $N_1 + 1$ one-dimensional regions. Pick a sample point in each one-dimensional region.

$n+2$ Over each root, or at the sample points for each interval between roots, isolate roots of S_2, and pick a sample point between each adjacent pair of roots.

$\vdots$ continue

$2n$ Over each cell in the decomposition of $\mathbf{R}^{n-1}$, isolate roots of $\mathcal{S}_n$, pick a sample point between each adjacent pair of roots, and hence make our decomposition of $\mathbf{R}^n$.

[2]For example, we ignore the growth in coefficient sizes. This can be (and is in [12]) tracked in detail, but doesn't affect the general argument.

So S_n has invariant signs on each region of $\mathbf{R}^n$, and $\phi(x_1, \ldots, x_n)$ has invariant truth on each region. Therefore all questions about ϕ reduce to the values of ϕ at the sample points.

$2n+1$ Evaluate the truth of Φ on each region R of $(x_1, \ldots, x_k)$-space, by looking at the values of ϕ at the sample points lying above the sample point of R, and combining them according to the quantifiers in Φ.

$2n+2$ The quantifier-free form Ψ of Φ is then the disjunction of the definitions of those regions of $(x_1, \ldots, x_k)$-space for which Φ is true.

The time complexity ends up being bounded [12, Theorem 16] by

$$O\left(m^{2^{n+6}}(2d)^{2^{2n+8}}\right).$$

While running time is one measure of complexity, it depends on the various sub-algorithms, and in practice depends on a lot of implementation details. A more refined analysis [15] of the complexity of step $n+1$ (and its knock-on effects on the subsequent steps), for example, reduces the complexity bound[3] (not the actual time) to $O\left(m^{2^{n+\not{6}4}}(2d)^{2^{2n+\not{8}6}}\right)$. Hence practitioners in the field of cylindrical algebraic decomposition tend to concentrate on the number of cells in the final decomposition. This has several advantages [6].

- It can be directly compared across systems, irrespective of hardware or software details.
- Most applications do significant amounts of post-processing on the cells, so the complexity of the post-processing is dependent on the number of cells (and on the complexity of the descriptions of the cells and their sample points, which a simple count doesn't capture directly: however experience shows that if algorithm A generates more cells than algorithm B on a given problem, algorithm A's descriptions are at least as complex).
- For a given problem and software/hardware system, the number of cells and the processing time tend to be closely correlated: a point first made explicitly in [18].
- The known *lower* bounds on complexity [17, 4] are in fact lower bounds on the number of cells.

The number of cells produced by Collins' method is bounded, by an analysis similar to [6], by

$$O\left(m^{2^n}(2d)^{2\cdot 3^n}\right).$$

[3] This may seem like a trivial improvement. In fact, the new bound is the fourth root of the old one, an improvement that would be viewed as massive in other contexts.

3.2. McCallum's improvements

Definition 3.1. The order of f at a point x is the least k such that at least one partial derivative of f of order k does not vanish at x.

McCallum [30] introduced a new projection operator for producing $\mathcal{S}_{i-1}$ from $\mathcal{S}_i$. In fact, he constructs a stronger cylindrical algebraic decomposition, *order-invariant* for the polynomials in ϕ, i.e. on each cell, every polynomial is identically zero of constant order, or everywhere positive, or everywhere negative. Clearly every order-invariant decomposition is sign-invariant, but the converse is not true. This approach has three major features.

pro Despite the fact that we are constructing a richer final object, the new projection sets are much smaller, and our analysis [6], based on the key result [30, Lemma 6.1.1] shows that the number of cells is bounded by

$$2^{n2^{n-1}} m^{2^n-1} d^{n2^{n-1}},$$

where the key improvement[4] is in the exponent of d, being of the form $n2^n$ rather than 3^n.

con The projection might not always work. There is a technical condition, known as *well-oriented* in [30], which is only discovered in phases $n+2, \ldots, 2n-1$, when a polynomial in $\mathcal{S}_k$ turns out to vanish identically on a cell of nonzero dimension. In these circumstances we can either revert to using an improvement [26] of the full Collins method (with its attendant costs), or, as suggested in [30] but to the best of the author's knowledge never implemented, whenever, in the *projection* phases, an $\mathcal{S}_i$ contains a polynomial that *might* nullify, add its partial derivatives with respect to each variable to the set $\mathcal{S}_i$. Again to the best of the author's knowledge, the complexity of this has never been analysed.

In theory, well-orientedness ought to occur "with probability 1". However, humans don't pose random problems, and the experience of the author and his Bath colleagues is that well-orientedness can frequently fail to occur, especially when solving problems coming from simplification, as in [3].

odd Step $2n+2$ may run into difficulties, a problem first pointed out in [8]. The roots of $\mathcal{S}_1$ isolated in step $n+1$ are of the form "the (unique) root α of $p(x)$ lying in (β, γ)", where $\beta, \gamma \in \mathbf{Q}$ (in practice $\in \mathbf{Z}[1/2]$). This statement is in our language $\mathcal{L}_{\mathrm{RCF}}$ ($p(x) = 0 \wedge x > \beta \wedge \gamma > x$). However, the branches of $\mathcal{S}_2$ (and other S_i) are in the form "that branch of $p(x_1, x_2)$ such that $p(\alpha, x_2)$ lies in (β, γ)", where $\beta, \gamma \in \mathbf{Q}$, and this is not in $\mathcal{L}_{\mathrm{RCF}}$. We could equally

[4] An improved analysis of [30, Lemma 6.1.1] in fact gives

$$2^{2^{n+1}-2n} (md)^{2^n-1}, \tag{3.1}$$

reducing the exponent of d further.

describe it as "the third real branch of $p(x_1, x_2)$ when $x_1 \in (\alpha_1, \alpha_2)$", but again this statement is not in $\mathcal{L}_{\mathrm{RCF}}$. Now by Thom's Lemma [14], we can describe this branch in terms of the signs of p *and its derivatives*, but, whereas these derivatives are in the Collins projection, they are not in the McCallum projection. However, when it comes to step $2n+2$, we can just add these, so the additional cost is negligible.

3.3. Regular chains methods

The production of *Cylindrical Algebraic Decompositions by Regular Chains* was first introduced in [11], and an improved version (essentially of the first step) was presented in [9]. Unlike the previous methods, they go via complex space, essentially as:

1. construct a cylindrical decomposition of $\mathbf{C}^n$ which is *zero/nonzero-invariant* for the polynomials in ϕ;

2. refine this to a cylindrical algebraic decomposition of $\mathbf{R}^n$, which will therefore be sign-invariant for the polynomials in ϕ;

3. for the same reasons as those described under **odd** in the previous section, if necessary add extra derivatives to be able to express the quantifier-free result in $\mathcal{L}_{\mathrm{RCF}}$ [10].

Not much is known about the theoretical complexity of regular chain computation in general, but this method does seem to be[5] at least competitive with, and often better than, our implementation [22] of [30] using the same Maple technology and libraries[6].

4. Equational constraints

It is often the case that ϕ contains equations: can we make use of these? [31] (based on [13]) was the first to show how. He defined an *equational constraint* $h = 0$ as an equality logically implied by ϕ, i.e. $\phi(x_1 \ldots, x_n) \Rightarrow h(x_1, \ldots, x_n) = 0$. For the sake of simplicity, we assume that ϕ actually has the form $(h = 0) \wedge \hat{\phi}$, and that $\hat{\phi}$ involves $m-1$ polynomials g_i (therefore m in all). We will not ask for an order-invariant decomposition, but rather an *equationally sign-invariant* decomposition: one where the signs of f and the g_i are constant on every cell **where** $f = 0$ — it is quite possible that cells where $f \neq 0$ have a mixture of behaviours of the g_i. We also need h to have main variable x_n. Compared with equation (3.1), this generates [6] at most

$$2^{2^{n+1}-2n}(m+1)^{2^{n-1}-1}d^{2^n-1}$$

[5] Compare the columns RC-Inc-CAD and PL-CAD in [19, Table 1].

[6] `www.regularchains.org`.

cells: replacing m^2 by $m+1$ in a single projection, and hence reducing the double exponent in the overall complexity.

This process is generalised in [5] to consider ϕ of the form

$$\underbrace{(f_1 = 0 \wedge g_{1,1} > 0 \wedge \cdots)}_{\phi_1} \vee \underbrace{(f_2 = 0 \wedge \cdots)}_{\phi_2} \vee \cdots \vee \underbrace{(f_k = 0 \wedge g_{k,1} > 0 \wedge \cdots)}_{\phi_k}. \quad (4.1)$$

This *does* have an equational constraint, viz. $\prod_i f_i = 0$, but this is of degree kd if the f_i have degree d. What [5] produces is rather a *Truth Table Invariant* decomposition (TTICAD), in which the truth of each ϕ_i is invariant on each cell. Practically this shows further savings, and the complexity is analysed in [6]. It uses a strict subset of the projection sets $\mathcal{S}_i$ generated with the equation constraint $\prod_i f_i = 0$ (for example not having $\text{res}_{x_n}(f_2, g_{1,1})$), so is clearly an improvement.

This method is further generalised in [6] to the case where not every ϕ_i in (4.1) has an equality. The savings are less spectacular than when every ϕ_i has an equality, but much more spectacular than the McCallum projection. The complexity [6] depends on the shape of ϕ.

These methods have also been applied to the Regular Chains approach to constructing cylindrical algebraic decompositions [2], again with good experimental results, but no complexity analysis. However, a curious feature comes up here. The method of [9] is incremental: one starts with the trivial decomposition, invariant for $\emptyset$, and adds polynomials. [2] build on this, but prune the cylindrical decomposition being created according to the Boolean structure of ϕ. This means that the same ϕ, but processed in a different order, can produce different cylindrical algebraic decompositions, and different (though of necessity logically equivalent) quantifier-free forms. This is analysed in [19].

5. So I have a problem in $\mathcal{L}_{\text{RCF}}$: what do I do?

A first remark is that this is a common phenomenon: according to [1], 47% of the problems in the Tokyo University mathematics entrance examination *can be posed* in $\mathcal{L}_{\text{RCF}}$. Most of these, however, *are actually posed* in terms of elementary geometry, and translating these into forms involving the fewest number of variables (note how all the complexity formulae are doubly-exponential in n, as are the lower bounds [17, 4]) is non-trivial: an obvious translation might have twelve variables, whereas we "really only need" three.

There are also many choices of the precise way the problem is formulated: see [7] in general for projection-based methods, [19] for Regular Chains methods, and [38] for a specific example in robot motion planning.

Even after doing this, in fact, you probably have a problem that can be expressed in $\mathcal{L}_{\text{RCF}}$ in many ways. If our goal is quantifier elimination, then step $2n+1$ implies that the variables must be projected (in terms of the cylindricity property, even if we are using Regular Chains methods) in the order implied by the quantifiers. But this still leaves much unspecified, e.g. $x_1, \ldots, x_k$ can be projected

in any order, and we can apply $\forall x \forall y \equiv \forall y \forall x$ (and its $\exists$ equivalent). Other applications, such as robot motion planning [34] and simplification [3] impose no constraints on the variable order. Hence an $\mathcal{L}_{\mathrm{RCF}}$-problem with n variables may have up to $n!$ possible instantiations as a cylindrical algebraic decomposition problem. This problem was first studied systematically in [18], who produced and evaluated various heuristics to pick the best order. Having observed that no one heuristic is best for all problems, [25] used machine learning on a large set of problems (using McCallum-style methods) to see if machine learning could predict which heuristic to use in a given setting, and found that this did indeed do better than any fixed choice of heuristic. [21] explored the choice of variable ordering for Regular Chains methods. The theoretical importance of variable ordering is shown by [4, Theorem 7] who produce a family of systems that has doubly-exponentially many cells in one ordering, and a constant number (independent of n) in another. Conversely [4, Theorem 8] there are problems that are doubly exponential for all orders.

If a problem involves equalities, then, as well as the simplifications in Section 4, we can also use the equalities to perform algebraic simplification of each other, and of the appropriate inequalities. This was first noticed in the case of branch cuts by Phisanbut [33, 32], where a simple cut like $\Re(z) < 0 \wedge \Im(z) = 0$ becomes $f_{\Re}(z) < 0 \wedge f_{\Im}(z) = 0$, but $f_{\Re}(z)$ is only relevant when $f_{\Im}(z) = 0$. This was generalised in [36], who observed that it is *often* helpful, but not always. They evaluated several heuristics, including those from [18], to determine which formulation of a given problem to solve.

In the case of Projection/Lifting methods, whether Collins or McCallum, it is nearly always the case that the lifting step $(n+1, \ldots, 2n)$ are by far the most expensive. Within these, it is the lifting of cells of non-full dimension that is the most expensive component, since in general this will involve algebraic number manipulations. Hence [37] suggest using a variant of these steps that *only* lifts the full-dimensional cells, picking the ordering (or other formulation choices) that minimises this, and then recovering the lower-dimensional cells. Further improved lifting techniques in the presence of equational constraints are described in [20].

6. Conclusion

6.1. Overview

In the fifteen years since [31], there has been a great deal of work on algorithms for cylindrical algebraic decomposition, both for quantifier elimination and other problems. Notably the Regular Chains method [11, 9] has appeared as a competitor to the traditional Projection/Lifting approach of [12] and his school. Nevertheless, there are at least as many open questions as there were before, and some are given below.

6.2. Open questions

1. Produce some complexity results for the Regular Chains method (Section 3.3). It follows from the work of that section, and the fact that quantifier elimination is doubly exponential [17, 4] that Regular Chain computation (of the kind used in these computations) must be, but this has not been explored systematically.

2. Formalise the "we only need" issue of writing $\mathcal{L}_{\mathrm{RCF}}$ formulae expressing problems with as few variables as possible. A lot of this seems to be a variation on "without loss of generality", for example a naïve algebraicisation of

   > Given a triangle ABC

   might be

   > Given a triangle ABC where A is at (a_x, a_y), B is at (b_x, b_y) and C is at (c_x, c_y)

   but a mathematician using coordinates is more likely to write

   > Given a triangle ABC where, without loss of generality, A is at $(0, 0)$, B is at $(0, 1)$ and C is at (c_x, c_y)

   thus reducing the number of free variables by four, and more if we then need to choose a point on AB.

3. Extend the machine learning approach of [25] to a wider class of problems, and also a wider set of choices, not just variable ordering but also preconditioning [36] and formulation [19].

4. The method of [37], which seems to be the most accurate predictor of the "best formulation", is only applicable to Projection/Lifting methods. Can the fundamental idea be applied to Regular Chains methods as well?

5. The lower bounds for quantifier elimination are based on a construction which alternates quantifiers, e.g. [4] uses $\underbrace{\exists\forall\forall}_{\text{block}} \ldots \underbrace{\exists\forall\forall}_{\text{block}}$. There are theoretical methods [23] which are doubly-exponential only in the number a of alternations of quantifiers: $(md)^{n^{O(a)}}$. To the best of the author's knowledge, these have never been implemented. Note, however, that since all the Projection/Lifting algorithms we have shown are doubly-exponential in n (just in the Projection phases), this means that cylindrical algebraic decomposition is theoretically not the best tool.

6. This point is borne out by [27, 28], who solve the *purely existential* version of quantifier elimination, more precisely

$$\exists x_1 \exists x_2 \ldots \exists x_n \phi(x_1, \ldots, x_n), \tag{6.1}$$

by a method which can be thought of as a blend of the algebra underpinning cylindrical algebraic decomposition with the methodology of modern SAT-solvers. It currently seems to be impossible to generalise beyond (6.1), even to a single alternation, but again this is a topic crying out for progress.

References

[1] N.H. Arai, T. Matsuzaki, H. Iwane, and H. Anai. Mathematics by Machine. In K. Nabeshima, editor, *Proceedings ISSAC 2014*, pages 1–8, 2014.

[2] R.J. Bradford, C. Chen, J.H. Davenport, M. England, M. Moreno Maza, and D.J. Wilson. Truth Table Invariant Cylindrical Algebraic Decomposition by Regular Chains. In *Proceedings CASC 2014*, pages 44–58, 2014.

[3] R.J. Bradford and J.H. Davenport. Towards Better Simplification of Elementary Functions. In T. Mora, editor, *Proceedings ISSAC 2002*, pages 15–22, 2002.

[4] C.W. Brown and J.H. Davenport. The Complexity of Quantifier Elimination and Cylindrical Algebraic Decomposition. In C.W. Brown, editor, *Proceedings ISSAC 2007*, pages 54–60, 2007.

[5] R.J. Bradford, J.H. Davenport, M. England, S. McCallum, and D.J. Wilson. Cylindrical Algebraic Decompositions for Boolean Combinations. In *Proceedings ISSAC 2013*, pages 125–132, 2013.

[6] R.J. Bradford, J.H. Davenport, M. England, S. McCallum, and D.J. Wilson. Truth Table Invariant Cylindrical Algebraic Decomposition. `http://arxiv.org/abs/1401.0645`, 2014.

[7] R.J. Bradford, J.H. Davenport, M. England, and D.J. Wilson. Optimising Problem Formulation for Cylindrical Algebraic Decomposition. In J. Carette *et al.*, editor, *Proceedings CICM 2013*, pages 19–34, 2013.

[8] C.W. Brown. Guaranteed Solution Formula Construction. In S. Dooley, editor, *Proceedings ISSAC '99*, pages 137–144, 1999.

[9] C. Chen and M. Moreno Maza. An Incremental Algorithm for Computing Cylindrical Algebraic Decompositions. `http://arxiv.org/abs/1210.5543`, 2012.

[10] C. Chen and M. Moreno Maza. Quantifier Elimination by Cylindrical Algebraic Decomposition Based on Regular Chains. In K. Nabeshima, editor, *Proceedings ISSAC 2014*, pages 91–98, 2014.

[11] C. Chen, M. Moreno Maza, B. Xia, and L. Yang. Computing Cylindrical Algebraic Decomposition via Triangular Decomposition. In J. May, editor, *Proceedings ISSAC 2009*, pages 95–102, 2009.

[12] G.E. Collins. Quantifier Elimination for Real Closed Fields by Cylindrical Algebraic Decomposition. In *Proceedings 2nd. GI Conference Automata Theory & Formal Languages*, pages 134–183, 1975.

[13] G.E. Collins. Quantifier elimination by cylindrical algebraic decomposition — twenty years of progess. In B.F. Caviness and J.R. Johnson, editors, *Quantifier Elimination and Cylindrical Algebraic Decomposition*, pages 8–23. Springer Verlag, Wien, 1998.

[14] M. Coste and M.-F. Roy. Thom's Lemma, the Coding of Real Algebraic Numbers and the Computation of the Topology of Semi-Algebraic Sets. *J. Symbolic Comp.*, 5:121–129, 1988.

[15] J.H. Davenport. Computer Algebra for Cylindrical Algebraic Decomposition. Technical Report TRITA-NA-8511 NADA KTH Stockholm (Reissued as Bath Computer Science Technical report 88-10), 1985.

[16] J.H. Davenport. The geometry of $\mathbf{C}^n$ is important for the algebra of elementary functions. In M. Jowsig and N. Takayama, editors, *Algebra Geometry and software systems*, pages 207–224. Springer, 2003.

[17] J.H. Davenport and J. Heintz. Real Quantifier Elimination is Doubly Exponential. *J. Symbolic Comp.*, 5:29–35, 1988.

[18] A. Dolzmann, A. Seidl, and Th. Sturm. Efficient Projection Orders for CAD. In J. Gutierrez, editor, *Proceedings ISSAC 2004*, pages 111–118, 2004.

[19] M. England, R. Bradford, C. Chen, J.H. Davenport, M.M. Maza, and D.J. Wilson. Problem formulation for truth-table invariant cylindrical algebraic decomposition by incremental triangular decomposition. In S.M. Watt *et al.*, editor, *Proceedings CICM 2014*, pages 45–60, 2014.

[20] M. England, R. Bradford, and J.H. Davenport. Improving the use of equational constraints in cylindrical algebraic decomposition. `http://arxiv.org/abs/1501.04466`, 2015.

[21] M. England, R. Bradford, J.H. Davenport, and D.J. Wilson. Choosing a Variable Ordering for Truth-Table Invariant Cylindrical Algebraic Decomposition by Incremental Triangular Decomposition. In *Proceedings ICMS 2014*, pages 450–457, 2014.

[22] M. England, D.J. Wilson, R. Bradford, and J.H. Davenport. Using the Regular Chains Library to build cylindrical algebraic decompositions by projecting and lifting. In *Proceedings ICMS 2014*, pages 458–465, 2014.

[23] N. Fitchas, A. Galligo, and J. Morgenstern. Precise sequential and parallel complexity bounds for the quantifier elimination over algebraic closed fields. *J. Pure and Applied Algebra*, 67:1–14, 1990.

[24] A. Gabrielov and N. Vorobjov. Complexity of computations with Pfaffian and Noetherian functions. In *Normal Forms, Bifurcations and Finiteness Problems in Differential Equations*, pages 211–250. Kluwer, 2004.

[25] Z. Huang, M. England, D. Wilson, J.H. Davenport, L.C. Paulson, and J. Bridge. Applying machine learning to the problem of choosing a heuristic to select the variable ordering for cylindrical algebraic decomposition. In S.M.Watt *et al.*, editor, *Proceedings CICM 2014*, pages 92–107, 2014.

[26] H. Hong. An Improvement of the Projection Operator in Cylindrical Algebraic Decomposition. In S. Watanabe and M. Nagata, editors, *Proceedings ISSAC '90*, pages 261–264, 1990.

[27] D. Jovanović and L. de Moura. Solving Non-Linear Arithmetic. In *Proceedings IJCAR 2012*, pages 339–354, 2012.

[28] D. Jovanović and L. de Moura. Solving non-linear arithmetic. *ACM Communications in Computer Algebra*, 46(3/4):104–105, 2013.

[29] Yu.V. Matiyasevich. Enumerable sets are Diophantine. *Soviet Math. Doklady 2*, 11:354–358, 1970.

[30] S. McCallum. An Improved Projection Operation for Cylindrical Algebraic Decomposition. Technical Report 548 Computer Science University Wisconsin at Madison, 1985.

[31] S. McCallum. On Projection in CAD-Based Quantifier Elimination with Equational Constraints. In S. Dooley, editor, *Proceedings ISSAC '99*, pages 145–149, 1999.

[32] N. Phisanbut, R.J. Bradford, and J.H. Davenport. Geometry of Branch Cuts. *Communications in Computer Algebra*, 44:132–135, 2010.

[33] N. Phisanbut. *Practical Simplification of Elementary Functions using Cylindrical Algebraic Decomposition.* PhD thesis, University of Bath, 2011.

[34] J.T. Schwartz and M. Sharir. On the "Piano-Movers" Problem: II. General Techniques for Computing Topological Properties of Real Algebraic Manifolds. *Adv. Appl. Math.*, 4:298–351, 1983.

[35] A. Tarski. *A Decision Method for Elementary Algebra and Geometry.* 2nd ed., Univ. Cal. Press. Reprinted in *Quantifier Elimination and Cylindrical Algebraic Decomposition* (ed. B.F. Caviness & J.R. Johnson), Springer-Verlag, Wein-New York, 1998, pp. 24–84., 1951.

[36] D.J. Wilson, R.J. Bradford, and J.H. Davenport. Speeding up Cylindrical Algebraic Decomposition by Gröbner Bases. In J. Jeuring *et al.*, editor, *Proceedings CICM 2012*, pages 279–293, 2012.

[37] D.J. Wilson, R.J. Bradford, J.H. Davenport, and M. England. Cylindrical Algebraic Sub-Decompositions. *Mathematics in Computer Science*, 8:263–288, 2014.

[38] D.J. Wilson, J.H. Davenport, M. England, and R.J. Bradford. A "Piano Movers" Problem Reformulated. In *Proceedings SYNASC 2013*, pages 53–60, 2013.

Annales Mathematicae et Informaticae
44 (2015) pp. 47–57
http://ami.ektf.hu

What does Mathematical Notation actually mean, and how can computers process it?*

James H. Davenport

University of Bath (U.K.)
J.H.Davenport@bath.ac.uk

Submitted September 14, 2014 — Accepted November 14, 2014

Abstract

Mathematical Notation is generally though of as universal and constant. This is not as true as the layman thinks, and Notation is in fact an evolving, subject-specific, collection of sub-notations, where the same symbol can mean different things in different parts of the same sentence. This paper surveys the various ways computers process, and help humans to process, the varieties of notation.

Keywords: Mathematical Notation, MathML, OpenMath

MSC: 00A35, 68A30, 68C20

1. Notation: the perception and the reality

The outsiders' perception of mathematical notation is that it is unambiguous, unchanging, precise, and world-wide (or more so[1]). One need merely Google for the phrase "mathematically precise" to see many instances of this view. And indeed there is a lot of truth in this belief: the author has seen mathematicians be unable to speak to each other, having no human language in common, but be able to communicate by writing mathematics.

This is not just a popular belief: the computing discipline of "Formal Methods", which employs tens of thousands of people in industry, as well as many academics,

*Thanks to many people: typesetters, editors, OpenMath and MathML colleagues, TEXnicians.

[1]Witness various science-fiction stories where, e.g., Pythagoras' Theorem is used as a demonstration of intelligence.

Idea	Anglo-Saxon	French	German
half-open interval	$(0,1]$	$]0,1]$	varies
single-valued function	arctan	Arctan	arctan
multi-valued function	Arctan	arctan	Arctan
$\{0,1,2,\ldots\}$	$\mathbb{N}$	$\mathbb{N}$	$\mathbb{N}\cup\{0\}$
$\{1,2,3,\ldots\}$	$\mathbb{N}\setminus\{0\}$	$\mathbb{N}\setminus\{0\}$	$\mathbb{N}$

Table 1: Cultural Notation Differences

tries to reduce computer programming to mathematics/logic, and has substantial success in doing so.

However, mathematical notation is certainly not unchanging. Few except the scholars of the history of notation would recognise in

$$1cu.m.6ce.p.11co. \text{ equale } 6n^i$$

[2, attributed to [17]] the modern $x^3 - 6x^2 + 11x = 6$.

The + sign is less than 500 years old [19] (this text also introduced − and $\surd$).

The = sign is slightly younger [18].

Recorde [18] wrote $2\overline{a+b}$: $2(a+b)$ is later, but the parenthetic notation won because it is (much![2]) easier for manual typesetting.

Calculus had from the origin, and still has, two very different notations for ordinary differentiation: $\dot{x}$ versus $\frac{\mathrm{d}x}{\mathrm{d}t}$, which have given rise to u_{xxt} versus $\frac{\partial^3 u}{\partial^2 x \partial t}$.

Relativity introduced the summation convention [7]: $\sum_{i=1}^{3} c_i x^i$ is abbreviated as $c_i x^i$ (but $c_\mu x^\mu$ is short for $\sum_{\mu=0}^{3} c_\mu x^\mu$, i.e. the range of summation depends on the alphabet from which the index is drawn).

Mathematical notation is also not quite as international as the layman believes: see Table 1. The examples there are drawn from relatively advanced mathematics, but the differences can be more basic – [13, p. 2] lists five different forms of writing 9,435,671 found in Houston schools. These issues can spread to the description of algorithms such as division, as in Figure 1. Indeed MathML [22] recognises 10 such formats, such as `stackedleftlinetop`: see `http://www.w3.org/Math/draft-spec/mathml.html#chapter3_presm.mlongdiv.ex`.

Mathematical notation is also subject-specific: while the mathematician writes i for $\sqrt{-1}$, the electrical engineer writes j, reserving i for current. A more chaotic

[2]The author, in the 1960s, used to typeset mathematics using "cold lead" technology: $2(a+b)$ involved selecting six characters from the cases of symbols, while $2\overline{a+b}$ would have involved cutting a raised piece of lead to form the overline *and* two "sleepers" – unraised pieces of lead – to sit either side of it to ensure that the overline was over the right characters. Furthermore, any change in the paragraph which moved the $\overline{a+b}$ horizontally would involve cutting new sleepers.

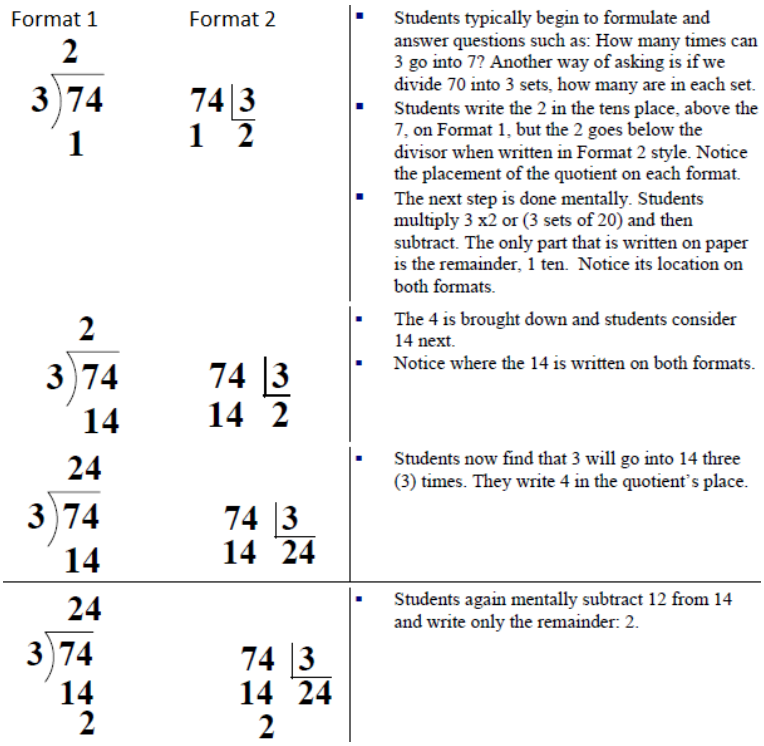


Figure 1: Division (from [13, p. 7])

example of notational clash between subjects can be seen in [5], where the algebraist uses [. . .] to indicate a polynomial ring extension, and the biochemist uses [. . .] to indicate "concentration of". Hence computations were being conducted in $\mathbb{C}[[P][S][E]]$. The fact that "reaction scheme" notation uses + to indicate combination of reagents rather than mathematical addition is a further complication for the reader.

The mathematician also knows (without, possibly, having articulated it) that notation is area-specific within mathematics. For example $(2, 4)$ might be, depending on the area, any of:

Set Theory The ordered pair "first 2, then 4";

(Geometry) The point $x = 2$, $y = 4$;

(Vectors) The 2-vector of 2 and 4;

Calculus Open interval from 2 to 4;

Group Theory The transposition that swaps 2 and 4;

Number Theory The greatest common divisor of 2 and 4;

In general, these expressions, whilst written identically, are *spoken* differently by the mathematician: the written text "we draw a line from $(2,4)$ to $(3,5)$" is spoken "we draw a line from the point $(2,4)$ to the point $(3,5)$". This can apply even within a given sentence[3]: every group theorist would read

$$\text{Since } H_i \leq G \text{ for } i \leq n \tag{1.1}$$

as "Since H sub i is a subgroup of G for i less than or equal to n" without, probably, even noticing that the two instances of $\leq$ had been pronounced very differently. This issue is a major challenge for mathematics "text-to-speech" renderers.

2. Imperfections in notation

Mathematical notation has evolved over the centuries, and some innovations were, with hindsight, less than ideal.

2.1. "Landau" Notation

This notation, apparently actually due to Bachmann [3], has two components. The first is not controversial: we use $O(f(n))$ to denote those functions that "grow no faster than $f(n)$" – formally (though rarely stated as such)

$$O(f(n)) = \{g(n) | \exists N, A : \forall n > N \ |g(n)| < Af(n)\}\,, \tag{2.1}$$

and similarly with o, Ω, ω and Θ. The second component of this notation is the use of "=" with this, as in $\log_2 n = O(\log n)$. This is not the traditional use of the = sign, as the relation is not symmetric: we can't write $O(\log n) = \log_2 n$, for example, and while we might stretch the notation to $O(n^2) = O(n^3)$, it is certainly not the case that $O(n^3) = O(n^2)$. Again, the spoken language gives a clue: the (English-speaking) mathematician would say "is" not "equals".

If we were honest with (2.1), we would write $O(\log n) \in \log_2 n$, but to the best of the author's knowledge, [12] is the only textbook to be consistently honest in this area using $\in$, though [9] does refer to $O(f)$ as a set, and is careful to use neither = nor $\in$. Being honest with (2.1) has another advantage: we can write $\Theta(f(n)) = O(f(n)) \cap \Omega(f(n))$, as [9] does.

2.2. Iterated Functions

No-one could quarrel with any of the following:

$\sin(x^2)$ square x, then apply sin

$(\sin x)^2$ apply sin to x, then square the result

$\sin(\sin(x))$ apply sin to x, then apply sin again

[3] I owe this example to Ieuan Evans of Bath.

The problem comes with $\sin^2 x$, which is generally used to mean $(\sin x)^2$, whereas, if anything, it should mean $\sin(\sin(x))$, since this is the sense in which we write $\sin^{-1}(x)$ – apply the inverse operation of sin, not $1/\sin(x)$. The author is not the first to object to this notation: "[This] is by far the most objectionable of any" [2]. The author has not encountered a definitive explanation of the origin of this notation, which was clearly common by the time of Babbage, but his experience of manual printing leads him to believe that it was economy of printing:

$$\sin^2 \frac{\theta+\phi}{2} \text{ versus } \left(\sin \frac{\theta+\phi}{2}\right)^2$$

obviates searching for the very large brackets, and building up an exponent to a non-standard height.

2.3. Continued Fractions

The "correct" notation for continued fractions, as in

$$\pi = 3 + \cfrac{1}{7 + \cfrac{1}{15+\cfrac{1}{1+\cfrac{1}{292+\ddots}}}} \tag{2.2}$$

is nearly always reduced to

$$\pi = 3 + \frac{1}{7+}\,\frac{1}{15+}\,\frac{1}{1+}\,\frac{1}{292+}\cdots, \tag{2.3}$$

which is much easier for (manual) typesetting[4], and uses less space – still a relevant consideration. Furthermore, if the individual terms of the continued fraction are complicated, as in

$$\alpha = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2+\cfrac{1}{a_3+\cfrac{1}{a_4+\ddots}}}},$$

the alternative notation is probably more readable, at least when the reader is used to it.

2.4. Conclusion

We actually see that the same printed notation can mean very different mathematical objects, and that the same mathematical object can be displayed in many different styles. This has led to a conceptual split between the computerisation of the *presentation*, how the mathematics looks, and the computerisation of the *content* (or semantics), i.e. what the mathematics means. This is formalised in the MathML standard, which has different chapters, and even different basic tokens, for the two approaches.

[4] As we (LaTeX) have written it (2.2) uses three sizes of digits, while (2.3) only uses one. Most "Hot metal" printers only had two available, so the result would not be as attractive as (2.2).

3. Computer Displays of Mathematics

Let us first look at how computers mediate the presentation of mathematical formulae.

3.1. Display of Mathematics

We can distinguish various (overlapping) periods in the computer display of mathematics.

1. Images – generally GIF or JPEG formats, though others have been used, and SVG has become more desirable [20]. The fundamental problem with an image is that it is precisely an image – all machine-processable information has been lost. In HTML, it is possible to include an `ALT`ernative representation, and this might be the LaTeX source, which at least conveys some information to a text-to-speech renderer.

2. Computer processing – as photocomposition replaced "hot metal" technology in typesetting shops, so these photocomposers became computer-controlled. Various programs, notably `troff` [16] and the associated mathematics preprocessor `eqn` [10], were developed to take advantage of this capability, and the author's PhD thesis was ported as [6] to the IBM equivalent program – YFL [8].

3. A major breakthrough came with Knuth's TeX [11]. One fundamental development here over its predecessors was the principle of boxes with width, height and depth. The requirement to know the explicit depth of a box is fundamental, as in (2.2). This has become the *de facto* gold standard for mathematical typesetting.

4. The original HTML did not support mathematics, much to its designer's regret, and MathML–Presentation 1.0 [21] soon appeared to fill this gap. However, the browsers of the period did not support the concept of 'depth' for boxes, and this can still be a problem today (Chrome's support for MathML has been intermittent, largely for this reason). A further challenge with many browsers[5] is the lack of fonts available.

5. MathJax [14] has emerged as a pragmatic solution to the vagaries of browsers, and is discussed further in [20].

3.2. Line Breaking

All systems the author knows of make a fundamental distinction between "in-line" and "display" mathematics, and the user has to state which is required, e.g. `$...$` versus `$$...$$` in TeX. TeX and its derivatives provide so support for automatic

[5] And other software: PowerPoint has often given users problems here.

breaking of lines if a displayed formula overflows the line width, and not much support for in-line formulae[6]. In the author's experience, a significant fraction of the effort in converting a paper from one format to another is in reflowing the equations, and maybe converting from display to in-line or *vice versa*.

However, the author of a web page has no control over the width within which it is displayed, and hence the browser must do *something* about linebreaking. This is also a problem for the various kinds of e-book readers, and partially accounts for the relative difficulty of handling mathematics, or technical text in general, on these devices. The MathML standard [23, §3.1.7] provides a suggested algorithm, but, as it says there:

> This algorithm takes time proportional to the number of token elements times the number of lines.

This problem, with its blend of algorithmics and aesthetics, is at least as difficult as, but less-researched than, the problem of table layout, as discussed in [15]

3.3. MathML-Presentation

While it is possible to regard MathML-Presentation as "LaTeX with pointy brackets", this view in fact does it a disservice. While $f(x)$, written as `f(x)` in TeX, could be written as

```
<mrow> <mi> f </mi> <mo> ( </mo>  <mi> x </mi>  <mo> ) </mo> </mrow>
```

it would best be represented in MathML as

```
<mrow>
  <mi> f </mi>
  <mo> &ApplyFunction; </mo>
  <mrow>
    <mo> ( </mo>
    <mi> x </mi>
    <mo> ) </mo>
  </mrow>
</mrow>
```

In this representation, the function application, and precisely what the argument is, are clearly apparent. This matters for speech rendering – "f of x", as well as semantic analysis. However, it is still presentation, and cannot solve the sort of problem seen in (1.1).

[6] TeXnically speaking, the mathematics has been converted into a list of boxes by the time it is realised that line-breaking is needed, hence rules like "break at the outermost operator" no longer make sense.

4. Computer Representation of Mathematical Content

Originally, there were two different approaches to the description of mathematical content: OpenMath and MathML-Content. We describe each, and then the convergence process.

4.1. OpenMath

The OpenMath movement grew out of the Computer Algebra community's wish to move formulae between systems. An early document is [1], which emphasises the importance of extensibility. Indeed, OpenMath is not so much an encoding as a framework for encoding, and the Standard [4] does not of itself specify how to transmit anything more complicated than integers. In fact it defines only a few basic concepts, listed here as their XML encodings.

`OMOBJ` The basic constructor, whose argument is an OpenMath objects. This exists so that OpenMath can be embedded in other documents, as formulae are in text. It's opposite is `OMFOREIGN`, indicating that we have some non-OpenMath constructs (such as Presentation MathML) embedded in an OpenMath object.

`OMS` This indicates an "OpenMath Symbol", an object to which the OpenMath process assigns a definite meaning. The arguments are the **name** of the symbol, e.g. `sin`, and the location of the "Content Dictionary" in which that definition can be found. This location can be either a simple name (`transc1` would indicate the standard Content Dictionary for basic transcendental functions) or a complete URL.

`OMA` This indicates an "OpenMath Application", where the first argument is to be considered an operator applied to the remaining arguments.

`OMBIND` This indicates an "OpenMath Binding", where the first argument is some operator to bind the variables specified in the second argument in the use of the third argument. A typical first argument would be `<OMS name="forall" cd="quant1"/>` to indicate $\forall$.

`OME` This indicates an "OpenMath Error Object" (such as "divide by zero"): the first argument is the 'name' of the error, as an `OMS`, and the rest are additional arguments depending on the error.

`OMATTR` This indicates an "OpenMath Attribution": the first argument has various attributes, such as `color` being `red`.

`OMR` This indicates an "OpenMath Reference" and allows us to build directed acyclic[7] graphs, rather than just trees.

[7] There is an explicit ban on cycles in the OpenMath standard.

Basic objects are encoded by any of `OMV` (variables), `OMI` (integers), `OMB` (byte arrays), `OMSTR` (Unicode strings) or `OMF` (IEEE floating-point numbers).

4.2. MathML-Content

This was introduced at the start of the MathML process, with a view to being "an explicit encoding of the underlying mathematical meaning of an expression, rather than any particular rendering for the expression" [23]. Equally, as have we have seen, renderings can be ambiguous, and one aim of MathML-Content is to remove this ambiguity. Consider $(F + G)x$: this could be either multiplication or function application: see Figure 2. We note that there is no need for brackets, as

```
<apply><times/>          <apply>
  <apply><plus/>           <apply><plus/>
    <ci>F</ci>               <ci>F</ci>
    <ci>G</ci>               <ci>G</ci>
  </apply>                 </apply>
  <ci>x</ci>               <ci>x</ci>
</apply>                 </apply>
```

Figure 2: Alternative MathML-Content for $(F + G)x$

`<apply>`. . . `</apply>` groups, and the meaning is explicit: in the first we have an application of `<times/>` while in the second we are applying $F + G$.

The original aim in MathML (version 1) was to handle "school" mathematics, otherwise "K–12", or Kindergarten to 12th-grade. However, this became a moving target, as constructs like `<div>` were introduced.

4.3. Convergence

The reader will have noticed that there is a strong similarity between OpenMath and MathML-Content, with `<apply>` corresponding to `<OMA>`, and `<ci>` constructs corresponding to `<OMV name=` constructs. The difference is that `<plus/>` is part of the MathML specification, whereas `<OMS name="plus" cd="arith1"/>` is just a symbol in an OpenMath content dictionary. This difference is also the source of the greater expressivity of OpenMath: MathML needed to charge to accommodate `<div/>`, whereas OpenMath just added the `veccalc1` content dictionary.

MathML version 2 therefore added the ability to use OpenMath symbols, thus buying into the expressivity of OpenMath. In MathML version 3, the authors went further, and defined MathML-Content in terms of OpenMath as follows.

> [In §4.2] a core collection of elements comprising Strict Content Markup are described. Strict Content Markup is sufficient to encode general expression trees in a semantically rigorous way. It is in one-to-one

> correspondence with OpenMath element set. OpenMath is a standard for representing formal mathematical objects and semantics through the use of extensible Content Dictionaries. [23, §4.1.1].

`<plus/>` is then defined to be a shorthand for `<OMS name="plus" cd="arith1"/>`, etc.

5. Conclusion

In terms of reproducing via computers the intricate two-dimensional layouts of mathematical notation, created (at considerable expense) by cold-metal printers, the TEX engine [11] has no equal. However, all it does is express how to lay out the symbols, and says nothing about their meaning. The invisible operator after `)` in the LATEX `(F+G)x` could be either function application or multiplication.

Although it is possible to write MathML-presentation that conveys no more information than the LATEX, well-written MathML-presentation can convey far more, as the invisible operator should be either `&ApplyFunction;` or `&InvisibleTimes;`.

However, presentation MathML can only go so far in encoding meaning, and is still unable to resolve the two uses of $\leq$ in (1.1) for example. For this, we need a representation of the semantics ,either OpenMath or MathML-Content. Fortunately, the two have converged so much that they are essentially isomorphic structures, and we can look forward to greater convergence in the future.

References

[1] J.A. Abbott, A. Díaz, and R.S. Sutor. OpenMath: A Protocol for the Exchange of Mathematical Information. *SIGSAM Bulletin 1*, 30:21–24, 1996.

[2] C. Babbage. Article "Notation". *Edinburgh Encyclopaedia*, 15:394–399, 1830.

[3] P. Bachmann. *Die analytische Zahlentheorie.* Teubner, 1894.

[4] S. Buswell, O. Caprotti, D.P. Carlisle, M.C. Dewar, M. Gaëtano, and M. Kohlhase. The OpenMath Standard 2.0. `http://www.openmath.org`, 2004.

[5] J.P. Bennett, J.H. Davenport, M.C. Dewar, D.L. Fisher, M. Grinfeld, and H.M. Sauro. Computer algebra approaches to enzyme kinetics. In Gérard Jacob and Françoise Lamnabhi-Lagarrigue, editors, *Algebraic Computing in Control*, volume 165 of *Lecture Notes in Control and Information Sciences*, pages 23–30. Springer Berlin Heidelberg, 1991.

[6] J.H. Davenport. *On the Integration of Algebraic Functions*, volume 102 of *Springer Lecture Notes in Computer Science*. Springer Berlin Heidelberg New York (Russian ed. MIR Moscow 1985), 1981.

[7] A. Einstein. Die Grundlage der allgemeinen Relativitaetstheorie (The Foundation of the General Theory of Relativity). *Annalen der Physik Fourth Ser.*, 49:284–339, 1916.

[8] A.M. Gruhn. The Yorktown Formatting Language: User Guide. Technical Report RC 6994 IBM Research, 1979.

[9] M. Hetland. *Python algorithms: Mastering Basic Algorithms in the Python Language (2nd ed.)*. Apress, 2014.

[10] B.W. Kernighan and L.L. Cherry. A System for Typesetting Mathematics. *Comm. ACM*, 18:151–157, 1975.

[11] D.E. Knuth. *The TEXbook: Computers and Typesetting Vol. A*. Addison–Wesley, 1984.

[12] A. Levitin. *Introduction to the design and analysis of algorithms*. Pearson Addison–Wesley, 2007.

[13] N.R. Lopez. Mathematical Notation Comparisons between U.S. and Latin American Countries. `https://sites.google.com/site/algorithmcollectionproject/mathematical-notation-comparisons-between-u-s-and-latin-american-countries`, 2008.

[14] MathJax Consortium. MathJax: Beautiful math in all browsers. `http://www.mathjax.org/`, 2011.

[15] Kim Marriott, Peter Moulder, and Nathan Hurst. Html automatic table layout. *ACM Trans. Web*, 7(1):4:1–4:27, March 2013.

[16] J.F. Ossanna. Nroff/Troff User's Manual. Technical Report 54 Bell Labs, 1976.

[17] Luca Pacioli. *Summa de arithmetica, geometria, proportioni et proportionalita*. Venice, 1494.

[18] R. Recorde. *The Whetstone of Witte*. J. Kyngstone, London, 1557.

[19] Stifelius [Michael Stifel]. *Arithmetica Integra*. Iohan Petreius, Norimberg, 1544.

[20] M. Schubotz and G. Wicke. Mathoid: Robust, Scalable, Fast and Accessible Math Rendering for Wikipedia. `http://arxiv.org/abs/1404.6179`, 2014.

[21] World-Wide Web Consortium. Mathematical Markup Language: First Public Draft. `http://www.w3.org/TR/WD-math-970515/`, 1997.

[22] World-Wide Web Consortium. Mathematical Markup Language (MathML) Version 3.0: W3C Recommendation 21 October 2010. `http://www.w3.org/TR/2010/REC-MathML3-20101021/`, 2010.

[23] World-Wide Web Consortium. Mathematical Markup Language (MathML) Version 3.0: second edition. `http://www.w3.org/TR/2014/REC-MathML3-20140410/`, 2014.

Annales Mathematicae et Informaticae
44 (2015) pp. 59–69
http://ami.ektf.hu

Lightweight simulation of programmable memory hierarchies*

Gergely Dévai

Eötvös Loránd University, Fac. of Informatics
deva@elte.hu

Submitted September 15, 2014 — Accepted May 13, 2015

Abstract

In most performance critical applications the bottleneck is data access. This problem is mitigated by applying memory hierarchies. There are application domains where the traditional, hard-wired cache control mechanisms are not satisfactory and the programmer is given full control where to place and when to move data. A known optimization technique on these kind of architectures is to do block-transfer of data instead of element-by-element access.

This contribution presents a lightweight library, written in and for C, to support experiments with algorithm performance on simulated programmable memory hierarchies. The library provides functions and macros to define memory layers, available block-transfer operations and data layout. The algorithm then can be run on a simple desktop computer and the library combines its real runtime with simulated memory access penalties.

Keywords: Memory hierarchy, block transfer, simulation

MSC: 68U20

1. Introduction

Memory access is the performance bottleneck of many computer architectures, because memory is an order slower than processors. This is mitigated by applying a memory hierarchy ranging from very fast but small to slow but large memory layers. If data is loaded to the faster layers by an automatic policy hidden from

*Supported by EITKIC 12-1-2012-0001.

the programmer we speak about cache layers. In some application areas, however, these policies do not perform well enough. In these cases the programmer is given control over the faster memory layers: She or he is in charge of which data to store in which layer and when to move data between the layers.

Platforms with programmable memory hierarchy usually provide special block copy operations: These can move given amount of data (typically 8, 16 or 32 bytes) from one layer to another. The cost of these operations in terms of waiting for the memory is similar to a single byte memory access. These operations make the so called bulk access optimization possible: A large array is stored in slow memory that needs to be processed elementwise. This task can be done by copying chunks of the array using block copy operations one by one to fast memory and process the chunks there. If the processing involves modification of the array then the processed chunk is copied back to its original location by a second block copy.

The goal of the created C library is to provide means of experimentation with bulk access and related optimizations without actually having a hardware platform with programmable memory hierarchy. Detailed requirements:

- The library has to provide means to define the memory hierarchy, together with the access costs.
- It has to provide means to define which object is placed in which layer.
- To be able to use the library to measure memory-related performance of an existing C program should only require minimal changes to the program.
- The library does not have to provide cycle accurate results of a given platform, instead it has to enable estimations for any platform with a memory hierarchy configured by the user of the library.

In the next section, related work is presented. Then, section 3 describes how to use our C library, followed by the implementation details in section 4. Section 5 presents measurement results of various experiments using the simulation library.

2. Related work

The theoretical background of architectures featuring storage elements with different access costs and providing block copy like instructions has been established decades ago [1]. Regarding the technical aspects of such architectures, paper [4] describes the details of efficiently using the memory hierarchies. However, this paper mainly concentrates on traditional cache systems, as opposed to programmable memory hierarchies, which is the main focus of our work.

Software controlled memory hierarchies generated a large amount of research about related optimization problems. Let us mention a few of these: [9] discusses heuristics to find good data layout, [10] concentrates on the optimization of energy consumption and also deals with allocation instructions. Paper [6] shows a dynamic memory management solution.

Complementary to the optimization aspect, there have been experiments to increase the abstraction level of programming software controlled memory hierarchies. Several extensions to existing programming languages [2, 3] or newly proposed languages [5] emerged. However, in practice, C is still the most widely used language to program these performance oriented platforms. For this reason C is selected as the base language of the simulation solution of this paper.

Simulation of program behavior on special hardware is important because it enables software evaluation without actual deployment. Another use case is the pre-evaluation of not yet existing hardware. There are simulators targeting specific hardware platforms accurately or aiming at the simulation of many different aspects of software behavior, eg. [8]. The XMSIM system [7] uses code transformations and C++ to evaluate memory hierarchies. It is, in many aspects, similar to the work presented in this paper. The main differences are that our solution uses C, and it is much simpler and lighter weight.

3. Users' perspective

This section presents the API provided by our library. Figures 1 and 2 show a possible architecture definition and a program using it. The API elements demonstrated by these examples are explained in the following subsections in detail.

```
 1 #ifndef __EXAMPLE_ARCHITECTURE_H
 2 #define __EXAMPLE_ARCHITECTURE_H
 3
 4 enum memory { scratch = 10, ram = 100 };
 5
 6 enum copy_size { _8bytes = 8,
 7                   _16bytes = 16,
 8                   _32bytes = 32
 9                 };
10
11 #endif
```

Figure 1: Example memory architecture definition (`example_architecture.h`)

3.1. Architecture definition

The first step of using the library is to describe the basic properties of the memory hierarchy to be simulated in a header file, see figure 1. Types `memory` and `copy_size` have to be defined as enumeration types, like in the following example.

```
enum memory { scratch = 10, ram = 100 };
```

```
1  #define ARCHITECTURE "example_architecture.h"
2  #include "memory.h"
3  #include <stdio.h>
4
5  void calibration_function() {
6      static int i, data[128];
7      for(i=0; i<128; ++i)
8          data[i] = i;
9  }
10
11 int main() {
12     calibrate(&calibration_function, 4*128);
13     int i;
14     char sum;
15     char _a[8];
16     #define a access(_a,scratch)
17     char _b[256];
18     #define b access(_b,ram)
19     for( i=0; i<256; ++i)
20         b[i] = i;
21     start();
22     sum = 0;
23     for( i=0; i<256; ++i)
24         sum += b[i];
25     long long elapsed = stop();
26     printf("Simple solution: %lld microseconds.\n",
27             elapsed );
28     start();
29     sum = 0;
30     for( i=0; i<256; i+=_8bytes ) {
31         block_copy(_a,scratch,&(_b[i]),ram,_8bytes);
32         sum += a[0] + a[1] + a[2] + a[3]
33              + a[4] + a[5] + a[6] + a[7];
34     }
35     elapsed = stop();
36     printf("Block copy solution: %lld microseconds.\n",
37             elapsed );
38     return 0;
39 }
```

Figure 2: Summing an array with and without block copying

```
enum copy_size { _8bytes = 8, _16bytes = 16, _32bytes = 32 };
```

Here, two memory layers are defined: scratch is a relatively fast memory, its access costs ten cycles, while accessing ram takes 100 clock cycles. The example also defines three different block copy widths: 8, 16 and 32 bytes respectively.

3.2. Importing the architecture

The next step is to implement the algorithms to be measured in some C source file(s). Let us assume that the header file `example_architecture.h` defines the simulated architecture. One can refer to it from the source files as seen in lines 1-2 of figure 2:

```
#define ARCHITECTURE "example_architecture.h"
#include "memory.h"
```

The `ARCHITECTURE` symbol defines the name of the header file describing the architecture, and `memory.h` is the header file providing the block copy operations and utility functions to measure simulated execution time.

3.3. Variable definition and access

Lines 15-18 of the example uses special notation to declare the variables that we want to store in one of the simulated memory layers.

```
char _a[256];
#define a access(_a,scratch)
```

The previous two lines define a character array of length 256 stored on the memory layer called `scratch`. The real name of this object, seen by the C compiler is `_a`. Using a macro definition, we create a synonym (`a`) for this variable and specify a memory layer (`scratch`) for it. Whenever the algorithm accesses the data of this object, the synonym has to be used. This enables the simulation library to incorporate the configured memory access cost in the execution time results. On the other hand, if the object is used without data access, for example, as an argument of the `sizeof` or the `&` operators, the real variable name (`_a`) has to be used.

3.4. Block copy operations

Block copy operations are simulated by calling the following function:

```
void* block_copy( void* destination, enum memory dest_mem,
                  void* source, enum memory src_mem,
                  enum copy_size size );
```

Destination and source are two addresses to copy to and from respectively. Parameters `dest_mem` and `src_mem` are memory layers of the destination and the source: these can be values of the memory enumeration type as given in the architecture

description. The last parameter, `size`, is one of the values of the `copy_size` enumeration type as given in the architecture description: It defines the number of bytes to copy.

Line 31 of figure 2 demonstrates a call to this function. It copies 8 bytes from array `b` in RAM to array `a` stored on the scratchpad.

3.5. Performance measurement

Before starting the first measurement cycle it is important to call the

```
double calibrate( void (*f)(), long long cycles );
```

function. The parameters are a calibration function and the number of clock cycles the function would take on the simulated hardware to execute. The `calibrate` function calculates the execution time of simulated clock cycle on the current machine. See line 12 of figure 2, where this calibration takes place.

In order to get the simulated execution time of a piece of code, it needs to be surrounded by calls to the `start()` and `stop()` functions. The latter one returns the simulated execution time in clock cycles. Figure 2 performs two such measurements, between lines 21-25 and 28-35.

4. Implementation

The C library is implemented in a single header file in terms of macros and inline functions. It first includes the header file indicated by the `ARCHITECTURE` symbol to get access to the configuration of the simulated platform.

The `calibrate` function executes the calibration function in its first parameter, measures its execution time, which is then divided by the number of clock cycles in the second parameter. The result is the amount of time that a clock cycle of the simulated machine takes on the simulator machine. This is saved for later use.

The `start` function initializes `timeval` structures from the `sys/time.h` standard header to measure execution time up to the execution of the `stop` function. This amount of time is converted to simulated machine clock cycles according to the ratio given by the calibration. This is one component of the end result.

The other component is the simulated memory access cost. Each object synonyme (see section 3.3) in the program is expanded to a call to the `access` macro. Its definition is as follows:

```
#define access(var,mem) (*(stall(mem),&var))
```

This is an expression composed by the comma (sequencing) operator of C. First, the `stall` function is called with the actual memory layer. Recall that each layer is an element in an enumeration type, and its integer value is the number of clock cycles the memory access takes on the simulated platform. The `stall` function simply accumulates these simulated stall times, which is the second component of the simulation's result. The left hand side of the comma operator defines the value

of the expression, which is the address of the accessed variable. The dereferencing operator (`*`) takes this address and returns the variable itself (`var`).

The reason for using the combination of the operators `&` and `*` instead of simply writing `(stall(mem),var)` is that the latter expression is not a left-value and would not allow the programmer to use the object synonyms on the left-hand side of assignment operations, for example.

The `block_copy` function computes the maximum stall of the destination and source memory layers and uses this maximum as the parameter to the `stall` function. The functionality of the operation is implemented by a standard `memcpy` call.

5. Experiments

The measurements were carried out on an Acer TravelMate 8572TG laptop with Intel Core i5-460M processor (2.53 GHz, 3MB L3 cache) and 4 GB DDR3 memory. The following diagrams show the simulated clock cycles translated back to microseconds according to the capabilities of this machine.

5.1. Proof-of-concept

In the first, basic experiment we have implemented the summation of a 1024 byte long unsigned character array two different ways. The array is configured to be stored in a memory layer causing 100 cycles average stall time at each access.

The first solution processes the array element-by-element. The second one copies array chunks to faster memory and processes data there. The buffer used to store the chunks has 10 cycles average stall time and the block-copy operations used are 32 bytes wide.

In this setup, the block copy solution results in more than 4 times speedup, according to our simulation. That is, block copying array chunks to faster memory and processing data there is faster than element-wise processing of the original array in slow memory.

5.2. Slowdown caused by increasing memory access penalties

This experiment examines the effect of increasing stall time to the run time of summing the elements of a 1024 element character array. The computation is done element-by-element, without block copy operations. The stall time of the memory layer storing the array was increased from 2 cycles up to 256.

The diamonds on the graph show linear slowdown for reference. The runtime results (squares) show that slowdown is smaller than linear in case of low stall times and it converges to linear slowdown as stall time increases. This is because, in the first case, the stall is not overwhelming the overall runtime of the algorithm. As stalls get longer, they become the most significant factor. This observation

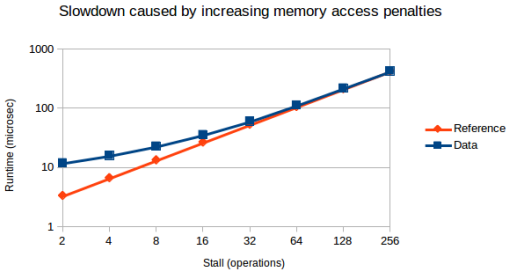


Figure 3: Slowdown caused by increasing memory access penalties

suggests the next experiment that examines the stall time versus the computation time of algorithms.

5.3. Stall – computation ratio

This experiment examines the effect of increasing the number of arithmetic instructions while keeping the stall time constant. The processed array is 3000 bytes long with unsigned character elements. Each arithmetic instruction is multiplication, and no block copying is used. Two memory layers were tested, with stall time of 2 and 256 cycles respectively.

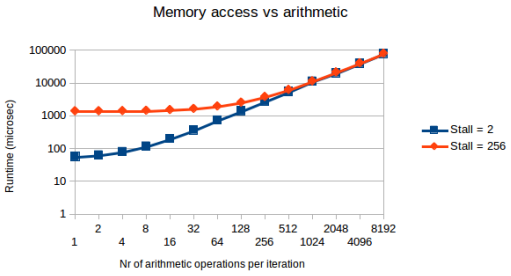
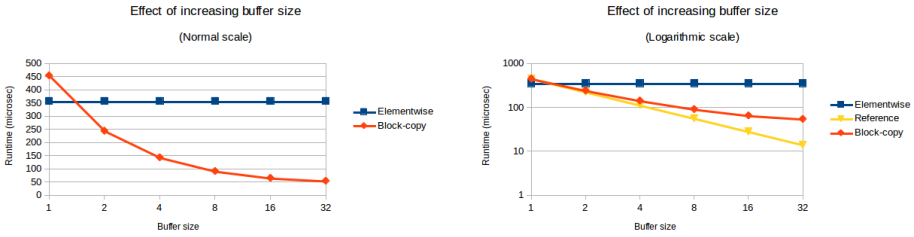


Figure 4: Memory access vs. computation

In case of small number of arithmetic instructions the stall time is the most significant factor of runtime, therefore there is considerable difference between the fast and slow memory layers. Increasing the number of arithmetic instructions causes sub-linear slowdown in this case. As more and more computation overwhelms stall time, slowdown tends to be linear and the difference between memory layers disappears.

5.4. The effect of changing block-copy size

This experiment examines the effect of increasing block-copy width. The algorithm used is scalar product working on two unsigned character arrays of 1024 elements length each. These are stored in a memory layer with 128 cycles long stall, while the buffer used is stored in memory with only 8 cycles long stall time. The block-copy width was increased from 1 byte to 32 bytes.



(a) Normal scale (b) Logarithmic scale

Figure 5: Effect of increasing buffer size

The results show that block-copying only single bytes chunks actually degrades performance compared to the element-wise solution. This is expected and shows the pure cost of the complication of the algorithm caused by block copying. Copying 2 array elements in each iteration already worth it. Figure 5b shows the speedup on logarithmic scale with the triangles depicting linear speedup for reference. Speedup is close to linear in the beginning, but using more and more powerful block-copy instructions result in sub-linear speedups, but still decreasing runtime considerably. The reason for sub-linearity is that the constant non-stall related cost of the algorithm gets more and more significant as stall time is successfully reduced by more and more effective block-copy instructions.

5.5. Array-of-structs vs. struct-of-arrays

Block-copy related optimizations often require data layout changes. A typical example is the array-of-structs to struct-of-arrays transformation. This experiment uses an algorithm that counts the RGB-coloured pixels with red component stronger than a threshold. The first data layout is array-of-structs: An array with 1024 structs, each storing the RGB values of a pixel. The array is stored in a memory layer with 100 instructions stall time. The first implementation processes the array element-wise, the second one uses 32 bytes wide block-copy operations. This is expected to result in considerable speedup, however, it copies the green and blue components of the RGB structs superfluously. In order to improve the solution further, a data layout change is needed: The second layout uses a struct of three arrays, storing the 1024 red, green and blue components respectively. In this case the red components are adjacent in memory, leading to much effective

block-copying.

The simulation data showed almost 3.5 times speedup due to the introduction of block copying on the first data layout. The layout change results in more than 1.5 times speedup between the two block-copying implementations. At this point, one could expect more than 1.5 speedup, considering that 2/3 of data copying is spared. In applications using structs with more fields this speedup could certainly be increased.

6. Summary

This paper presented a light-weight library, written in C, that allows programmers to simulate programmable memory hierarchies. The memory hierarchy, its stall times and the data layout of the program can be freely configured. Only minor changes are required in existing C code to make the simulation possible.

The paper discussed the API of the library, the implementation details and presented five different experiments carried out by using the library. On one hand, the results of the experiments are interesting because they reveal behavioral features of memory hierarchies in detail. On the other hand, the results can be understood and explained, which increases confidence in the simulation library itself.

References

[1] Alok Aggarwal, Ashok K Chandra, and Marc Snir. Hierarchical memory with block transfer. In *28th Annual Symposium on Foundations of Computer Science*, pages 204–216. IEEE, 1987.

[2] Alex Aiken, Phil Colella, David Gay, Susan Graham, Paul Hilfinger, Arvind Krishnamurthy, Ben Liblit, Carleton Miyamoto, Geoff Pike, Luigi Semenzato, et al. Titanium: A high-performance Java dialect. *Concurrency: Practice and Experience*, 10:11–13, 1998.

[3] David E Culler, Andrea Dusseau, Seth C Goldstein, Arvind Krishnamurthy, Steven Lumetta, Steve Luna, Thorsten von Eicken, and Katherine Yelick. Introduction to Split-C. 1995.

[4] Ulrich Drepper. What every programmer should know about memory. *Red Hat, Inc*, 11, 2007.

[5] Kayvon Fatahalian, Daniel Reiter Horn, Timothy J Knight, Larkhoon Leem, Mike Houston, Ji Young Park, Mattan Erez, Manman Ren, Alex Aiken, William J Dally, et al. Sequoia: programming the memory hierarchy. In *Proceedings of the 2006 ACM/IEEE conference on Supercomputing*, page 83. ACM, 2006.

[6] Mahmut Kandemir, J Ramanujam, J Irwin, Narayanan Vijaykrishnan, Ismail Kadayif, and Amisha Parikh. Dynamic management of scratch-pad memory space. In *Proceedings of the 38th annual Design Automation Conference*, pages 690–695. ACM, 2001.

[7] Theodoros Lioris, Grigoris Dimitroulakos, and Kostas Masselos. Xmsim: Extensible memory simulator for early memory hierarchy evaluation. In *VLSI 2010 Annual Symposium*, pages 199–216. Springer, 2011.

[8] Peter Magnusson and Bengt Werner. Efficient memory simulation in simics. In *Simulation Symposium, 1995., Proceedings of the 28th Annual*, pages 62–73. IEEE, 1995.

[9] Preeti Ranjan Panda, Nikil D Dutt, and Alexandru Nicolau. Efficient utilization of scratch-pad memory in embedded processor applications. In *Proceedings of the 1997 European conference on Design and Test*, page 7. IEEE Computer Society, 1997.

[10] Stefan Steinke, Lars Wehmeyer, Bo-Sik Lee, and Peter Marwedel. Assigning program and data objects to scratchpad for energy reduction. In *Design, Automation and Test in Europe Conference and Exhibition, 2002. Proceedings*, pages 409–415. IEEE, 2002.

Annales Mathematicae et Informaticae
44 (2015) pp. 71–85
http://ami.ektf.hu

A generalization of the Barabási-Albert random tree*

István Fazekas, Sándor Pecsora

Faculty of Informatics, University of Debrecen
fazekas.istvan@inf.unideb.hu
pecsora89@kmf.uz.ua

Submitted September 9, 2014 — Accepted January 20, 2015

Abstract

In this paper a random graph evolution rule is defined which can be considered as a generalization of the Barabási-Albert random tree. The evolution is a combination of the preferential attachment method and the interactions of 2 vertices. Our model is similar to the 3-interactions model studied in [2]. We describe the asymptotic behaviour of the degrees and the weights of the vertices.

Keywords: Random graph, preferential attachment, scale-free, power law

MSC: 05C80, 60G42

1. Introduction

Several real life networks are scale-free (see [4, 7]). A random graph is called scale-free, if it has a power law degree distribution, that is $P(d) \sim d^{-\gamma}$ as $d \to \infty$, where $P(d)$ is the probability that a vertex is of degree d. The well-known Barabási-Albert preferential attachment model produces a scale-free sequence of random graphs.

The Barabási-Albert model

The preferential attachment model was suggested by Barabási and Albert in [4]. See also the paper of Yule [17] for trees. The graph evolution rule given in [4] is

*The first author was supported by the TÁMOP-4.2.2.C-11/1/KONV-2012-0001 project. The project has been supported by the European Union, co-financed by the European Social Fund. The second author was supported by the Collegium Talentum and Edutus College.

the following. The starting point is a graph with a small number of vertices. At every time step a new vertex is added with m edges that link the new vertex to m different vertices already present in the graph. The preferential attachment means that the probability $p(i)$ that the new vertex will be connected to vertex i depends on the degree of that vertex, so that $p(i) = k_i / \sum_j k_j$, where k_j denotes the degree of vertex j. According to [5], the model is not defined precisely by this definition. A precise definition of the model and a rigorous proof of the scale-free property was given in [5] (see also [7, 16]). The simplest case of the model is the Barabási-Albert random tree, when $m = 1$.

In [6] a generalization of the Barabási-Albert model was introduced. In [6] besides the preferential attachment method, uniform choice of vertices are allowed, moreover, new connections can be grown between old vertices. For the recent results in the preferential attachment model see [16, 13, 10].

The 3-interactions model

In [2] the following graph evolution was introduced. We start with a single triangle. This graph contains 3 vertices and 3 edges. Each of these objects has initial weight 1. The evolution of the graph is based on the interactions of three vertices. At each step we consider three vertices and we draw all non-existing edges between them. So we obtain a triangle. The weight of this triangle and the weights all of its edges and vertices are increased by 1.

At a fixed time the evolution is the following. Independently of the past, with probability p, a new vertex is born which interacts with 2 old vertices. That is they form a triangle. The two old vertices can be chosen in two different ways. With probability r we choose an edge from the existing edges according to their weights. The two vertices of that edge will interact with the new vertex. On the other hand, with probability $1-r$, we choose 2 from the existing vertices uniformly. They will interact with the new vertex. Independently of the past, with probability $1-p$, we do not add a new vertex, but three of the old vertices interact. To select the three old vertices we have two options. With probability q we choose one out of the existing triangles according to their weights. The vertices of the triangle chosen will interact. On the other hand, with probability $1-q$, we choose from the existing vertices uniformly (that is all three vertices have the same chance).

The power law degree distribution in that model was proved in [2] and [3]. The model and the results were extended to N-interactions model in [8] and [9], if $N \geq 4$.

The goal of this paper

In this paper a random graph evolution mechanism is defined. The evolution of the graph is a combination of the preferential attachment and the interaction of 2 vertices. A vertex in our graph is characterized by its degree and its weight. The weight of a given vertex is the number of the interactions of the vertex. The asymptotic behaviour of the graph is studied. Scale-free properties both for the de-

grees and the weights are proved. The proofs are based on discrete time martingale theory.

Our model is a special case of the N-interactions model of [8] and [9]. However, our result can not be obtained as a particular case of the general results of [8] and [9] because the basic equation for the 2-interactions model is not a special case of the basic equation for the N-interactions model with $N \geq 3$. In this paper we follow the method elaborated by Backhausz and Móri in [2, 3]. We do not present detailed proofs because they are similar to the ones in [2, 3, 8, 9] .

2. The 2-interactions random graph model and the main results

In this paper we study the following version of the Barabási-Albert random tree.

At time $n = 0$ we start with two connected vertices. The initial weights of the

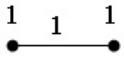


Figure 1: $n = 0$, the initial state

two vertices and the initial weight of the edge are equal to one. The weights of the non-existing edges and vertices are always considered to be 0. The evolution of the graph is based on the interactions of 2 vertices. At each step $n = 1, 2, \ldots$ we consider 2 vertices and if they are not connected, then we draw the edge between them. The weights of the two vertices and the weight of the edge connecting them are increased by 1.

The evolution of the graph is the following. On the one hand, with probability p, we add a new vertex, that will interact with 1 old vertex. On the other hand, with probability $(1 - p)$, we do not add any new vertex, but 2 old vertices interact.

(a) If we add a new vertex, then we choose 1 old vertex which will interact with the new one. To choose the old vertex we have two possibilities. With probability r we choose a vertex from the existing vertices according to the weights of the vertices. That is a vertex k with weight w_k has chance $w_k/(\sum_l w_l)$. On the other hand, with probability $1 - r$, we choose from the existing vertices uniformly, that is any vertex has the same chance.

(b) At the step when we do not add a new vertex, then 2 old vertices interact. To select the 2 old vertices we have two options. With probability q we choose one edge from the existing edges according to their weights. That is the probability that we choose an edge is proportional to its weight. Then the two vertices of that edge will interact. On the other hand, with probability $1 - q$, we choose two out of the existing vertices uniformly. That is all two vertices have the same chance.

Figure 2 shows an example for the graph evolution. At the initial step $n = 0$ we have an edge and two vertices. At step $n = 1$ we add a new vertex with initial weight 1, choose an old vertex and connect them using a new edge. The initial

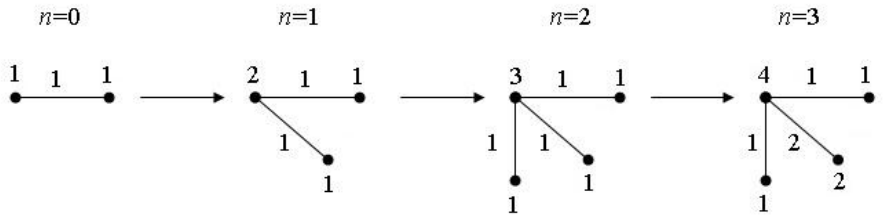


Figure 2: An example for the graph evolution

weight of the new edge is 1 and we increase the weight of the old vertex by 1. Step 2 is similar to step 1. However, at step $n = 3$, we do not add a new vertex, but 2 old vertices interact. We choose two out of the existing vertices, then increase the weights of the vertices and the weight of the edge connecting them by 1. So we can see that the weight of a given vertex is the number of the interactions of the vertex.

Our results are confined to the 2-interactions model. To describe the main results we need the following notation. Throughout the paper $0 < p < 1$, $0 \le r \le 1$, $0 \le q \le 1$ are fixed numbers. Let $X(n, d, w)$ denote the number of vertices of weight w and degree d after the nth step. Let V_n denote the number of vertices after the nth step.

Each vertex has initial weight 1 and initial degree 1. When a vertex takes part in an interaction, then its weight is increased by 1 and its degree may increase by 0 or 1. So $X(n, d, w)$ can be positive only for $1 \le w \le n + 1$ and $1 \le d \le w$.

Let

$$\begin{gathered} \alpha_1 = (1-p)\,q, \quad \alpha_2 = pr/2, \quad \alpha = \alpha_1 + \alpha_2, \\ \beta = (1-r) + 2\,(1-p)\,(1-q)/p. \end{gathered} \tag{2.1}$$

The following theorem describes the limiting behaviour of the relative frequency of vertices with a fixed weight and a fixed degree.

Theorem 2.1. *Let $0 < p < 1$, $q > 0$. Assume that at least one of the following three conditions are satisfied: $r > 0$ or $r < 1$ or $q < 1$. Then for any fixed w and d with $1 \le w$ and $1 \le d \le w$ we have*

$$X\,(n, d, w)/V_n \to x_{d,w} \tag{2.2}$$

almost surely as $n \to \infty$, where $x_{d,w}$ are fixed positive numbers. Furthermore, the numbers $x_{d,w}$ satisfy the following recurrence relation

$$x_{1,1} = 1/(\alpha + \beta + 1) > 0, \qquad x_{d,1} = 0, \ \textit{for } d \ne 1,$$

$$x_{d,w} = \frac{1}{\alpha w + \beta + 1}\left[\alpha_1\,(w-1)\,x_{d,w-1} + (\alpha_2\,(w-1) + \beta)\,x_{d-1,w-1}\right], \tag{2.3}$$

for $w \ge 2$, $1 \le d \le w$. If $1 \le d \le w$ is not satisfied, then $x_{d,w} = 0$.

The following lemma states that the numbers $x_{d,w}$, $d = 1, \ldots, w$, $w = 1, 2, \ldots$, form a (proper) two-dimensional discrete probability distribution. Moreover, its marginal distributions will be the limiting distributions of the weights and the degrees, respectively.

Lemma 2.2. *Let $p > 0$ and define $x_w = x_{1,w} + x_{2,w} + \cdots + x_{w,w}$ for $w = 1, 2, \ldots$. Then x_w, $w = 1, 2, \ldots$, are positive numbers satisfying the following recurrence relation*

$$x_1 = \frac{1}{\alpha + \beta + 1}, \quad x_w = \frac{\alpha\,(w-1) + \beta}{\alpha w + \beta + 1} x_{w-1}, \quad \text{if} \quad w > 1. \tag{2.4}$$

x_w, $w = 1, 2, \ldots$, is a discrete probability distribution. Moreover, $x_{d,w}$, $d = 1, \ldots, w$, $w = 1, 2, \ldots$, is a two-dimensional discrete probability distribution.

Next theorem shows the scale-free property of the weights of the vertices.

Theorem 2.3. *Let $X\,(n, w)$ denote the number of vertices of weight w after n steps. Assume that the conditions of Theorem 2.1 are satisfied. Then for all $w = 1, 2, \ldots$ we have*

$$X\,(n, w)/V_n \to x_w \tag{2.5}$$

almost surely, as $n \to \infty$, where x_w, $w = 1, 2, \ldots$, are positive numbers satisfying the recurrence relation (2.4)*. Moreover,*

$$x_w \sim C w^{-\left(1+\frac{1}{\alpha}\right)} \quad \text{as} \quad w \to \infty \tag{2.6}$$

with $C = \Gamma\left(1 + \frac{\beta+1}{\alpha}\right) / \left(\alpha\Gamma\left(1 + \frac{\beta}{\alpha}\right)\right)$.

Our main result is the scale-free property of the degrees.

Theorem 2.4. *Assume that the conditions $0 < p < 1$, $q > 0$, and $r > 0$ are satisfied. Let us denote by $U\,(n, d)$ the number of vertices of degree d after n steps, that is $U\,(n, d) = \sum_{w:d \le w \le n+1} X\,(n, d, w)$. Then, for any $d \ge 1$ we have*

$$\frac{U\,(n, d)}{V_n} \to u_d \tag{2.7}$$

a.s. as $n \to \infty$, where $u_d = \sum_w x_{d,w}$, $d = 1, 2, \ldots$, are positive numbers. Furthermore,

$$u_d \sim \frac{\Gamma\left(1 + \frac{\beta+1}{\alpha}\right)}{\alpha_2 \Gamma\left(1 + \frac{\beta}{\alpha}\right)} \left(\frac{\alpha}{\alpha_2}\right)^{-\left(1+\frac{1}{\alpha}\right)} d^{-\left(1+\frac{1}{\alpha}\right)} \quad \text{as} \quad d \to \infty. \tag{2.8}$$

3. Proofs and auxiliary results

The following lemma contains the basic equation of the paper. Let $\mathcal{F}_{n-1}$ denote the σ-algebra of observable events after $(n-1)$ steps. We compute the conditional expectation of $X(n, d, w)$ with respect to $\mathcal{F}_{n-1}$ for $w \ge 1$.

Lemma 3.1.

$$E(X(n,d,w)|\mathcal{F}_{n-1}) = X(n-1,d,w)\left(1-\frac{\alpha w}{n}-\beta\frac{p}{V_{n-1}}\right)+ \tag{3.1}$$
$$+X(n-1,d,w-1)(1-p)\left[q\frac{w-1}{n}+(1-q)\frac{d}{\binom{V_{n-1}}{2}}\right]+$$
$$+X(n-1,d-1,w-1)\left[p\left[r\frac{w-1}{2n}+(1-r)\frac{1}{V_{n-1}}\right]+(1-p)(1-q)\frac{V_{n-1}-d}{\binom{V_{n-1}}{2}}\right]+$$
$$+p\delta_{d,1}\delta_{w,1}.$$

Here $\delta_{a,b}$ denotes the Dirac-delta.

Proof. The probability that an old vertex of weight w takes part in the interaction at step n is

$$p\left(r\frac{w}{2n}+(1-r)\frac{1}{V_{n-1}}\right)+(1-p)\left(q\frac{w}{n}+(1-q)\frac{V_{n-1}-1}{\binom{V_{n-1}}{2}}\right)=\frac{w}{n}\alpha+\frac{p}{V_{n-1}}\beta,$$

where α and β are defined by (2.1). So the terms at the right hand side of (3.1) correspond to the following cases. The first term covers the case when neither the degree nor the weight of a vertex change. Its probability is $1-\left(\frac{\alpha w}{n}+\beta\frac{p}{V_{n-1}}\right)$. The second term covers the case when the degree does not change but the weight is increased by 1, while the third term correspond to the case when both the degree and the weight are increased by 1. A new vertex always takes part in the interaction. At each step, with probability p, a new vertex with weight 1 and with degree 1 is born. This explains term $p\delta_{d,1}\delta_{w,1}$ in (3.1). □

We shall need the following results on discrete time martingales. Let $\{Z_n,\mathcal{F}_n\}$ be a submartingale. Its Doob-Meyer decomposition is $Z_n = M_n + A_n$, where $\{M_n,\mathcal{F}_n\}$ is a martingale and $\{A_n,\mathcal{F}_n\}$ is an increasing predictable process. Here, up to an additive constant,

$$A_n = \mathbb{E}Z_1 + \sum_{i=2}^{n}(\mathbb{E}(Z_i|\mathcal{F}_{i-1}) - Z_{i-1}).$$

We see that $\{M_n^2,\mathcal{F}_n\}$ is again a submartingale. Let

$$M_n^2 = Y_n + B_n$$

be the Doob-Meyer decomposition of M_n^2. Here, up to an additive constant,

$$B_n = \sum_{i=2}^{n}\mathbb{D}^2(Z_i|\mathcal{F}_{i-1}) = \sum_{i=2}^{n}\mathbb{E}\left\{(Z_i-\mathbb{E}(Z_i|\mathcal{F}_{i-1}))^2|\mathcal{F}_{i-1}\right\}.$$

Proposition 3.1 (Propositions VII-2-3 and VII-2-4 of [12]). *Let $M_1 = 0$. On the set $\{B_\infty < \infty\}$ the martingale M_n almost surely converges to a finite limit. Moreover, $M_n = \mathrm{o}(B_n^{1/2} \log B_n)$ almost surely on the set $\{B_n \to \infty\}$.*

A consequence of the above proposition is the following.

Proposition 3.2 (Proposition 2.3 of [1]). *Let $\{Z_n, \mathcal{F}_n\}$ be a square integrable non-negative submartingale. If $B_n^{1/2} \log B_n = \mathrm{O}(A_n)$, then $Z_n \sim A_n$ as $n \to \infty$, almost surely on the set $\{A_n \to \infty\}$.*

Proof of Theorem 2.1. Applying the Marcinkiewicz strong law of large numbers to the number of vertices, we obtain

$$V_n = pn + \mathrm{o}\left(n^{1/2+\varepsilon}\right) \tag{3.2}$$

almost surely, for any $\varepsilon > 0$. Let

$$c(n,w) = \prod_{i=1}^{n} \left(1 - \frac{\alpha w}{i} - \frac{\beta p}{V_{i-1}}\right)^{-1}.$$

Then (3.2) and Taylor's expansion imply that

$$c(n,w) \sim a_w n^{\alpha w + \beta} \tag{3.3}$$

almost surely as $n \to \infty$, where a_w is a positive random variable.

Let $Z(n,d,w) = c(n,w)X(n,d,w)$. Then, by (3.1), $(Z(n,d,w), \mathcal{F}_n)$ is a non-negative submartingale. We shall apply the Doob-Meyer decompositions $Z_n = M_n + A_n$ and $M^2 = Y_n + B_n$. Then

$$\begin{aligned} A(n,d,w) = {} & EZ(1,d,w) + \\ & + \sum_{i=2}^{n} c(i,w)X(i-1,d,w-1)(1-p)\left(q\frac{w-1}{i} + (1-q)\frac{d}{\binom{V_{i-1}}{2}}\right) + \\ & + \sum_{i=2}^{n} c(i,w)X(i-1,d-1,w-1)\times \\ & \times \left[p\left(r\frac{w-1}{2i} + (1-r)\frac{1}{V_{i-1}}\right) + (1-p)(1-q)\frac{V_{i-1}-d}{\binom{V_{i-1}}{2}}\right] + \\ & + \sum_{i=2}^{n} c(i,w)p\delta_{d,1}\delta_{w,1}. \end{aligned} \tag{3.4}$$

Moreover

$$B(n,d,w) = \sum_{i=2}^{n} \mathbb{D}^2\left(Z(i,d,w) \,|\mathcal{F}_{i-1}\right) \leq$$

$$\leq \sum_{i=2}^{n} c\left(i,w\right)^2 \mathbb{E}\{\left(X\left(i,d,w\right)-X\left(i-1,d,w\right)\right)^2 |\mathcal{F}_{i-1}\} \leq$$

$$\leq 4\sum_{i=2}^{n} c\left(i,w\right)^2 = \mathrm{O}\left(n^{2(\alpha w+\beta)+1}\right). \tag{3.5}$$

We use induction on w. Let $w=1$. We see that a vertex of weight 1 took part in an interaction only when it was born. Therefore its degree must be equal to 1. By (3.4),

$$A\left(n,1,1\right) \sim p\sum_{i=2}^{n} c\left(i,1\right) \sim p\sum_{i=2}^{n} a_1 i^{\alpha+\beta} \sim pa_1 \frac{n^{\alpha+\beta+1}}{\alpha+\beta+1} \tag{3.6}$$

a.s. as $n\to\infty$. By (3.5), $B\left(n,1,1\right)=\mathrm{O}\left(n^{2(\alpha+\beta)+1}\right)$ and therefore

$$\left(B\left(n,1,1\right)\right)^{\frac{1}{2}} \log B\left(n,1,1\right) = \mathrm{O}\left(A\left(n,1,1\right)\right).$$

It follows from Proposition 3.2 that

$$Z\left(n,1,1\right) \sim A\left(n,1,1\right) \text{ a.s. on the event } \{A\left(n,1,1\right)\to\infty\} \text{ as } n\to\infty. \tag{3.7}$$

As, by (3.6), $A(n,1,1)\to\infty$ a.s., therefore using the asymptotic behaviour of V_n and $c(n,w)$, relation (3.7) implies

$$\frac{X\left(n,1,1\right)}{V_n} = \frac{Z\left(n,1,1\right)}{c\left(n,1\right)V_n} \sim \frac{A\left(n,1,1\right)}{c\left(n,1\right)V_n} \sim \frac{pa_1\frac{n^{\alpha+\beta+1}}{\alpha+\beta+1}}{a_1 n^{\alpha+\beta}pn} = \frac{1}{\alpha+\beta+1} = x_{1,1} > 0$$

almost surely. So (2.2) is valid for $w=1$.

Suppose that the statement is true for all weights less than w and for all possible degrees. It implies that $X(n,d,w-1)\sim x_{d,w-1}np$.

Then by (3.2), (3.3) and using the induction hypothesis, we have for any $w>1$

$$\begin{aligned}
A\left(n,d,w\right) \sim \sum_{i=2}^{n} \Bigg[& c\left(i,w\right) x_{d,w-1} pi\left(1-p\right) q\frac{w-1}{i} + \\
& + c\left(i,w\right) x_{d-1,w-1} pi\left(pr\frac{\left(w-1\right)}{2i} + \frac{p\left(1-r\right)}{pi} + \frac{2\left(1-p\right)\left(1-q\right)}{pi}\right)\Bigg] \sim \\
\sim \sum_{i=2}^{n} a_w i^{\alpha w+\beta} \Bigg[& x_{d,w-1} p\left(1-p\right) q\left(w-1\right) + \\
& + x_{d-1,w-1}\left(\frac{1}{2}p^2 r\left(w-1\right) + p\left(1-r\right) + 2\left(1-p\right)\left(1-q\right)\right)\Bigg] \sim \\
\sim pa_w \frac{n^{\alpha w+\beta+1}}{\alpha w+\beta+1} \Bigg[& \left(1-p\right) q\left(w-1\right) x_{d,w-1} + \\
& + \left(\frac{1}{2}pr\left(w-1\right) + \ \left(1-r\right) + \frac{2\left(1-p\right)\left(1-q\right)}{p}\right) x_{d-1,w-1}\Bigg].
\end{aligned} \tag{3.8}$$

In the above computation we deleted all terms having asymptotically smaller degree than the largest one.

Formula (3.8) implies $A(n,d,w) \sim pa_w n^{\alpha w+\beta+1} x_{d,w} \to \infty$, because $x_{d,w} > 0$, where, by (3.8),

$$x_{d,w} = \frac{1}{\alpha w+\beta+1}\left[\alpha_1 (w-1)\, x_{d,w-1} + (\alpha_2 (w-1)+\beta)\, x_{d-1,w-1}\right],$$

with α_1, α_2, α and β defined by (2.1). Therefore $(B(n,d,w))^{\frac{1}{2}} \log B(n,d,w) = \mathrm{O}(A(n,d,w))$. So, using Proposition 3.2, we have $Z(n,d,w) \sim A(n,d,w)$. Therefore

$$\frac{X(n,d,w)}{V_n} = \frac{Z(n,d,w)}{c(n,w)\,V_n} \sim \frac{A(n,d,w)}{c(n,w)\,V_n} \sim \frac{pa_w n^{\alpha w+\beta+1} x_{d,w}}{a_w n^{\alpha w+\beta} pn} = x_{d,w} \tag{3.9}$$

a.s. as $n \to \infty$. □

Proof of Lemma 2.2. If $\alpha = 0$, then the statement is obvious. Now assume $\alpha \neq 0$. As $x_{d,w}$ is defined as $x_{d,w} = 0$ for $d \notin \{1,2,\dots,w\}$, therefore $x_w = \sum_d x_{d,w}$. From the recurrence relation (2.3) we obtain

$$\begin{aligned} x_w &= \sum_{d=1}^{w} x_{d,w} = \sum_d x_{d,w} = \\ &= \frac{1}{\alpha w+\beta+1}\left[\alpha_1 (w-1)\sum_d x_{d,w-1} + (\alpha_2 (w-1)+\beta)\sum_d x_{d-1,w-1}\right] = \\ &= \frac{\alpha(w-1)+\beta}{\alpha w+\beta+1} x_{w-1}. \end{aligned}$$

Using this recursive formula for x_w, we obtain

$$\begin{aligned} x_w &= x_1 \prod_{j=2}^{w} \frac{\alpha(j-1)+\beta}{\alpha j+\beta+1} = \frac{1}{\alpha w+\beta+1} \prod_{j=1}^{w-1} \frac{\frac{\beta}{\alpha}+j}{\frac{\beta+1}{\alpha}+j} = \\ &= \frac{\Gamma\left(1+\frac{\beta+1}{\alpha}\right)}{\alpha\Gamma\left(1+\frac{\beta}{\alpha}\right)} \frac{\Gamma\left(w+\frac{\beta}{\alpha}\right)}{\Gamma\left(w+\frac{\beta+1}{\alpha}+1\right)}. \end{aligned} \tag{3.10}$$

By [15], we have the following formula:

$$\sum_{k=0}^{n} \frac{\Gamma(k+a)}{\Gamma(k+b)} = \frac{1}{a-b+1}\left[\frac{\Gamma(n+a+1)}{\Gamma(n+b)} - \frac{\Gamma(a)}{\Gamma(b-1)}\right].$$

Therefore, by some calculation, we obtain $\sum_{w=1}^{n} x_w \to 1$ as $n \to \infty$. So $\sum_{w=1}^{\infty} x_w = 1$. As $\sum_d x_{d,w} = x_w$, so $\sum_{w=1}^{\infty}\sum_{d=1}^{w} x_{d,w} = 1$ and therefore $x_{d,w}$, $d = 1,2,\dots,w$, $w = 1,2,\dots$, is a (proper) two-dimensional discrete probability distribution. □

Proof of Theorem 2.3. As

$$X(n,w) = X(n,1,w) + X(n,2,w) + \cdots + X(n,w,w),$$

Theorem 2.1 and Lemma 2.2 imply (2.5). Using (3.10), the Stirling formula gives (2.6). □

The following representation of the joint distribution of degrees and weights is useful to prove scale-free property for degrees. Let W be a random variable with distribution $\mathbb{P}(W = w) = x_w$, $w = 1, 2, \ldots$. Let $\xi_1 \equiv 1$ and $\xi_2, \xi_3, \ldots$ be independent random variables being independent of W, too. For $w \geq 2$ let ξ_w have the following distribution:

$$\mathbb{P}(\xi_w = 0) = \frac{\alpha_1(w-1)}{\alpha(w-1)+\beta}, \quad \mathbb{P}(\xi_w = 1) = \frac{\alpha_2(w-1)+\beta}{\alpha(w-1)+\beta}.$$

Let $S_w = \xi_1 + \xi_2 + \cdots + \xi_w$.

Lemma 3.2. $\mathbb{P}(S_W = d, W = w) = x_{d,w}$ *for all* $w = 1, 2, \ldots$, $d = 1, 2, \ldots, w$.

Proof. It is easy to see that the sequence $\mathbb{P}(S_W = d, W = w)$ satisfies the same recursion (2.3) as $x_{d,w}$. □

To obtain scale-free property for degrees, we need the following local limit theorem. Let $X_1, X_2, \ldots$ be independent, integer valued random variables. Let $p_{j,m} = \mathbb{P}(X_j = m)$ be the distribution, while $p_{j,m_j} = \max_m p_{j,m}$ be the maximal value of the distribution. Let $S_n = \sum_{i=1}^n X_i$ be the partial sum, $P_n(N) = \mathbb{P}(S_n = N)$ be its distribution, $M_n = \sum_{i=1}^n \mathbb{E}X_i$ be the expectation, and $B_n = \sum_{i=1}^n \mathbb{E}(X_i - \mathbb{E}X_i)^2$ be the variance of S_n.

Proposition 3.3 (Theorem 5 and its consequence in Section VII, 2 of [14])**.** *Assume that the greatest common divisor of the values*

$$\left\{ m \,:\, \frac{1}{\log n} \sum_{j=1}^{n} p_{j,m_j} p_{j,m+m_j} \to \infty \right\}$$

is equal to 1. Moreover,

$$\liminf \frac{B_n}{n} > 0, \quad \limsup \frac{1}{n} \sum_{i=1}^{n} \mathbb{E}|X_i - \mathbb{E}X_i|^3 < \infty.$$

Then

$$\sup_N \left| \sqrt{B_n} P_n(N) - \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(N - M_n)^2}{2B_n} \right) \right| = \mathrm{O}\left(\frac{1}{\sqrt{n}} \right).$$

If we apply Proposition 3.3 to the random variables ξ_k in Lemma 3.2, then we obtain the following result which will play an important role in the proof our main theorem.

Proposition 3.4. *Suppose that $\alpha_1 > 0$ and $\alpha_2 > 0$. Then*

$$x_{d,w} = x_w \frac{\alpha}{\sqrt{2\pi\alpha_1\alpha_2 w}} \left[\exp\left(-\frac{(d - \mathbb{E}S_w)^2}{2\mathbb{D}^2 S_w} \right) + \mathrm{O}\left(w^{-\frac{1}{2}} \right) \right] \quad as \quad w \to \infty, \tag{3.11}$$

where the error term $\mathrm{O}\left(w^{-\frac{1}{2}} \right)$ does not depend on d.

Proof. We follow the method of *Theorem 4.2* in [3]. Let $w > 1$. Then we have

$$\mathbb{E}\xi_w = \frac{\alpha_2 (w-1) + \beta}{\alpha (w-1) + \beta} = \frac{\alpha_2}{\alpha} + \frac{\alpha_1 \beta}{\alpha (\alpha (w-1) + \beta)},$$

hence

$$\mathbb{E}S_w = \mathbb{E}\xi_1 + \cdots + \mathbb{E}\xi_w = w\frac{\alpha_2}{\alpha} + \mathrm{O}(\log w) \tag{3.12}$$

as $w \to \infty$. By simple computation, we obtain

$$\mathbb{D}^2\xi_w = \frac{\alpha_1\alpha_2}{\alpha^2} + \mathrm{O}\left(\frac{1}{w}\right), \quad \mathbb{D}^2 S_w = \frac{\alpha_1\alpha_2}{\alpha^2} w + \mathrm{O}(\log w) \tag{3.13}$$

as $w \to \infty$.

Now, we apply Proposition 3.3 for S_w. The conditions of that proposition are satisfied, therefore we have

$$\sup_{d\in\mathbb{Z}} \left| \mathbb{D}S_w \, \mathbb{P}(S_w = d) - \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(d - \mathbb{E}S_w)^2}{2\mathbb{D}^2 S_w} \right) \right| = \mathrm{O}\left(\frac{1}{\sqrt{w}} \right). \tag{3.14}$$

Using (3.13) and (3.14), we obtain $\left| \mathbb{D}S_w - \frac{\sqrt{\alpha_1\alpha_2 w}}{\alpha} \right| \mathbb{P}(S_w = d) = \mathrm{O}\left(w^{-\frac{1}{2}} \right)$. Therefore (3.14) implies that

$$\sup_{d\in\mathbb{Z}} \left| \frac{\sqrt{\alpha_1\alpha_2 w}}{\alpha} \mathbb{P}(S_w = d) - \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(d - \mathbb{E}S_w)^2}{2\mathbb{D}^2 S_w} \right) \right| = \mathrm{O}\left(\frac{1}{\sqrt{w}} \right). \tag{3.15}$$

By the independence of W and ξ_i, we see that $x_{d,w} = \mathbb{P}(S_W = d, W = w) = \mathbb{P}(S_w = d)\, x_w$. So the result follows from (3.15). □

The well-known Hoeffding's inequality is the following.

Proposition 3.5 (Theorem 2 of [11])**.** *Let $X_1, X_2, \ldots, X_n$ be independent random variables, $a_i \le X_i \le b_i$ $(i = 1, 2, \ldots, n)$. Let $\bar{X} = (X_1 + X_2 + \cdots + X_n)/n$, $\mu = \mathbb{E}\bar{X}$. Then for any $t > 0$*

$$\mathbb{P}(\bar{X} - \mu \ge t) \le \exp\left(\frac{-2n^2 t^2}{\sum_{i=1}^n (b_i - a_i)^2} \right).$$

Proof of Theorem 2.4. Theorem 2.1 and Lemma 2.2 will imply (2.7). Hoeffding's inequality, Lemma 3.2 and Proposition 3.4 will imply (2.8).

By Theorem 2.1 and Lemma 3.2, $\dfrac{X(n,d,w)}{V_n}$ converges almost surely to the distribution $x_{d,w} = \mathbb{P}(S_W = d, W = w)$. But the cardinalities of terms in the sum $\sum_{w:d\le w\le n+1} X(n,d,w)$ are not bounded when $n\to\infty$. However, using that $x_{d,w}$, $d = 1,2,\dots,w$, $w = 1,2,\dots$ is a proper two-dimensional discrete distribution, the convergence of the marginal distributions is a consequence of the convergence of the two-dimensional distributions. So we obtain (2.7).

To obtain (2.8), we can apply the method of *Theorem 4.3* in [3]. Let

$$f = \frac{\alpha}{\alpha_2} d, \quad H = H_d = \left\{ w : f - f^{\frac{1}{2}+\varepsilon} \le w \le f + f^{\frac{1}{2}+\varepsilon} \right\},$$

$$H^- = H_d^- = \left\{ w : w < f - f^{\frac{1}{2}+\varepsilon} \right\}, \quad H^+ = H_d^+ = \left\{ w : w > f + f^{\frac{1}{2}+\varepsilon} \right\}$$

with some fixed $0 < \varepsilon < 1/6$.

Using (3.12) and Proposition 3.5, we obtain for $w \in H^-$

$$\mathbb{P}(S_w = d) \le \mathbb{P}(S_w \ge d) \le \mathbb{P}\left(S_w - \mathbb{E}S_w \ge d - \frac{\alpha_2}{\alpha} w - \mathrm{O}(\log w) \right) \le$$

$$\le \exp\left\{ -\frac{2}{w} \left(d - \frac{\alpha_2}{\alpha} w - \mathrm{O}(\log w) \right)^2 \right\} = \exp\left\{ -2 \left(\frac{\alpha_2}{\alpha} \right)^2 \frac{(f - w - \mathrm{O}(\log w))^2}{w} \right\}.$$

Now $w \in H^-$ implies that

$$(f - w - \mathrm{O}(\log w))^2 = (f-w)^2 - 2(f-w)\,\mathrm{O}(\log w) + (\mathrm{O}(\log w))^2 \ge$$
$$\ge f^{1+2\varepsilon} - \mathrm{O}(f \log f).$$

Therefore in the case when $w \in H^-$ we obtain

$$\mathbb{P}(S_w = d) \le \exp\left\{ -2 \left(\frac{\alpha_2}{\alpha} \right)^2 \frac{f^{1+2\varepsilon} - \mathrm{O}(f\log f)}{f} \right\} =$$
$$= \exp\left\{ -2 \left(\frac{\alpha_2}{\alpha} \right)^2 f^{2\varepsilon} + \mathrm{O}(\log f) \right\}.$$

This implies that

$$\mathbb{P}\left(S_W = d, W \in H^-\right) = \sum_{w\in H^-} \mathbb{P}(S_w = d, W = w) \le \sum_{w\in H^-} \mathbb{P}(S_w = d) \le$$

$$\le f \exp\left\{ -2 \left(\frac{\alpha_2}{\alpha} \right)^2 f^{2\varepsilon} + \mathrm{O}(\log f) \right\} = \mathrm{o}\left(f^{-\left(1+\frac{1}{\alpha}\right)} \right). \tag{3.16}$$

In the case when $w \in H^+$, by Hoeffding's inequality, we have

$$\mathbb{P}(S_w = d) \le \mathbb{P}(S_w \le d) \le \mathbb{P}\left(S_w - \mathbb{E}S_w \le d - \frac{\alpha_2}{\alpha} w \right) \le$$

$$\leq \exp\left\{-\frac{2}{w}\left(d-\frac{\alpha_2}{\alpha}w\right)^2\right\} = \exp\left\{-2\left(\frac{\alpha_2}{\alpha}\right)^2\frac{(f-w)^2}{w}\right\}.$$

Because $w \in H^+$ and $\frac{1}{2}+\varepsilon < 1$, we obtain $2(w-f) \geq f^{\frac{1}{2}+\varepsilon} + w - f \geq f^{\frac{1}{2}+\varepsilon} + (w-f)^{\frac{1}{2}+\varepsilon} \geq w^{\frac{1}{2}+\varepsilon}$. So

$$\mathbb{P}(S_w = d) \leq \exp\left\{-2\left(\frac{\alpha_2}{\alpha}\right)^2\frac{w^{1+2\varepsilon}}{4w}\right\} = \exp\left\{-\frac{1}{2}\left(\frac{\alpha_2}{\alpha}\right)^2 w^{2\varepsilon}\right\}.$$

Therefore

$$\mathbb{P}\left(S_W = d, W \in H^+\right) \leq \sum_{\{w\,:\,f<w\}} \exp\left\{-\frac{1}{2}\left(\frac{\alpha_2}{\alpha}\right)^2 w^{2\varepsilon}\right\} = \mathrm{o}\left(f^{-\left(1+\frac{1}{\alpha}\right)}\right). \quad (3.17)$$

Now turn to the case of $w \in H = H_d$. Consider the set

$$B = \{(d,w)\,:\, w \geq 1, d \geq 1, w \in H_d\}.$$

It is easy to see that

$$\text{if} \quad d \to \infty \quad \text{and} \quad (d,w) \in B, \quad \text{then} \quad \frac{w}{d} \to 1.$$

As $w \in H$, so we have $w = f + \mathrm{O}\left(f^{\frac{1}{2}+\varepsilon}\right)$. Then (with $\varepsilon_1 > 0$ arbitrarily small)

$$-\frac{(d-\mathbb{E}S_w)^2}{2\mathbb{D}^2 S_w} = -\frac{\left(d - w\frac{\alpha_2}{\alpha} - \mathrm{O}(\log w)\right)^2}{2\frac{\alpha_1\alpha_2}{\alpha^2}w + \mathrm{O}(\log w)} = -\frac{\alpha_2}{\alpha_1}\frac{(f-w-\mathrm{O}(\log w))^2}{2w+\mathrm{O}(\log w)} =$$

$$= -\frac{\alpha_2}{\alpha_1}\frac{(f-w)^2 + \mathrm{O}\left(f^{\frac{1}{2}+\varepsilon+\varepsilon_1}\right)}{2w+\mathrm{O}(\log w)} = -\frac{\alpha_2}{\alpha_1}\frac{(f-w)^2}{2f} + \mathrm{O}\left(f^{-\frac{1}{2}+3\varepsilon}\right) \quad (3.18)$$

as $d \to \infty$. Here the error term does not depend on w. By (3.11), (2.6) and (3.18), we obtain

$$x_{d,w} \sim$$

$$\sim C w^{-\left(1+\frac{1}{\alpha}\right)}\frac{\alpha}{\sqrt{2\pi\alpha_1\alpha_2 w}}\left[\exp\left\{-\frac{\alpha_2}{\alpha_1}\frac{(f-w)^2}{2f} + \mathrm{O}\left(f^{-\frac{1}{2}+3\varepsilon}\right)\right\} + \mathrm{O}\left(w^{-\frac{1}{2}}\right)\right] \sim$$

$$\sim C f^{-\left(1+\frac{1}{\alpha}\right)}\frac{\alpha}{\alpha_2}\frac{1}{\sqrt{2\pi\frac{\alpha_1}{\alpha_2}f}}\exp\left\{-\frac{(f-w)^2}{2\frac{\alpha_1}{\alpha_2}f}\right\}$$

as $d \to \infty$ and $w \in H$, where $C = \Gamma\left(1+\frac{\beta+1}{\alpha}\right) / \left(\alpha\Gamma\left(1+\frac{\beta}{\alpha}\right)\right)$. Therefore

$$\sum_{w\in H} x_{d,w} \sim \sum_{f-f^{\frac{1}{2}+\varepsilon}<w<f+f^{\frac{1}{2}+\varepsilon}} C f^{-\left(1+\frac{1}{\alpha}\right)}\frac{\alpha}{\alpha_2}\frac{1}{\sqrt{2\pi\frac{\alpha_1}{\alpha_2}f}}\exp\left\{-\frac{(f-w)^2}{2\frac{\alpha_1}{\alpha_2}f}\right\} =$$

$$= Cf^{-\left(1+\frac{1}{\alpha}\right)}\frac{\alpha}{\alpha_2}\sum_{-f^{\frac{1}{2}+\varepsilon}<k<f^{\frac{1}{2}+\varepsilon}}\frac{1}{\sqrt{2\pi\frac{\alpha_1}{\alpha_2}f}}\exp\left\{-\frac{k^2}{2\frac{\alpha_1}{\alpha_2}f}\right\} =$$

$$= A\sum_{-f^{\varepsilon}<\frac{k}{\sqrt{f}}<f^{\varepsilon}}\frac{1}{\sqrt{f}}\frac{1}{\sqrt{2\pi\frac{\alpha_1}{\alpha_2}}}\exp\left\{-\frac{\left(\frac{k}{\sqrt{f}}\right)^2}{2\frac{\alpha_1}{\alpha_2}}\right\}\to$$

$$\to A\int_{-\infty}^{+\infty}\frac{1}{\sqrt{2\pi\frac{\alpha_1}{\alpha_2}}}\exp\left\{-\frac{x^2}{2\frac{\alpha_1}{\alpha_2}}\right\}dx = A,$$

where

$$A = \frac{\Gamma\left(1+\frac{\beta+1}{\alpha}\right)}{\alpha_2\Gamma\left(1+\frac{\beta}{\alpha}\right)}\left(\frac{\alpha d}{\alpha_2}\right)^{-\left(1+\frac{1}{\alpha}\right)}.$$

So we obtain

$$\mathbb{P}\left(S_W = d, W\in H\right)\sim\frac{\Gamma\left(1+\frac{\beta+1}{\alpha}\right)}{\alpha_2\Gamma\left(1+\frac{\beta}{\alpha}\right)}\left(\frac{\alpha d}{\alpha_2}\right)^{-\left(1+\frac{1}{\alpha}\right)} \tag{3.19}$$

as $d\to\infty$. Finally, by (3.16), (3.17) and (3.19), we obtain

$$u_d\sim\frac{\Gamma\left(1+\frac{\beta+1}{\alpha}\right)}{\alpha_2\Gamma\left(1+\frac{\beta}{\alpha}\right)}\left(\frac{\alpha}{\alpha_2}d\right)^{-\left(1+\frac{1}{\alpha}\right)}$$

as $d\to\infty$. □

References

[1] Backhausz, Á., *Analysis of random graphs with methods of martingale theory.* PhD thesis, Eötvös Loránd University, Budapest, 2012.

[2] Backhausz, Á., Móri, T. F., A random graph model based on 3-interactions. *Ann. Univ. Sci. Budapest. Sect. Comput.* Vol. 36 (2012), 41–52.

[3] Backhausz, Á., Móri, T. F., Weights and degrees in a random graph model based on 3-interactions. *Acta Math. Hungar.* Vol. 143 (2014), no. 1, 23–43.

[4] Barabási, A.L., Albert, R., Emergence of scaling in random networks. *Science*, Vol. 286 (1999), 509–512.

[5] Bollobás, B., Riordan, O., Spencer, J., Tusnády, G., The degree sequence of a scale-free random graph process. *Random Structures Algorithms*, Vol. 18 (2001), 279–290.

[6] Cooper, C., Frieze, A., A general model of web graphs. *Random Structures Algorithms*, Vol. 22 (2003), 311–335.

[7] Durrett, R., *Random graph dynamics.* Cambridge University Press, Cambridge UK, 2007.

[8] Fazekas, I., Porvázsnyik, B., Scale-free property for degrees and weights in a preferential attachment random graph model. *Journal of Probability and Statistics,* Vol. 2013 (2013), Article ID 707960.

[9] Fazekas, I., Porvázsnyik, B., Scale-free property for degrees and weights in an N-interaction random graph model. *arXiv:1309.4258v1* [math.PR] 17 Sep. 2013.

[10] Grechnikov, E., An estimate for the number of edges between vertices of given degrees in random graphs in the Bollobás-Riordan model. *Mosc. J. Comb. Number Theory*, Vol. 1 (2011), no. 2, 40–73.

[11] Hoeffding, W., Probability inequalities for sums of bounded random variables. *J. Amer. Statist. Assoc.*, Vol. 58 (1963), 13–30.

[12] Neveu, J., *Discrete-parameter martingales.* North-Holland, Amsterdam, 1975.

[13] Ostroumova, L., Ryabchenko, A., Samosvat, E., Generalized preferential attachment: tunable power-law degree distribution and clustering coefficient. *arXiv*:1205.3015v1 [math.CO] 14 May 2012.

[14] Petrov, V. V., *Sums of Independent Random Variables.* Akademie-Verlag, Berlin, 1975.

[15] Prudnikov, A. P., Brychkov, Yu. A., Marichev, O. I., *Integrals and series.* Gordon & Breach Science Publishers, New York, 1986.

[16] van der Hofstad, R., *Random Graphs and Complex Networks.* Eindhoven University of Technology, The Netherlands, rhofstad@win.tue.nl, 2013.

[17] Yule, G. U., A Mathematical Theory of Evolution, Based on the Conclusions of Dr. J. C. Willis, F.R.S. *Phil. Transact. Royal Society London, Ser. B*, Vol. 213 (1925), 21–87.

Annales Mathematicae et Informaticae
44 (2015) pp. 87–97
http://ami.ektf.hu

Surprise event detection of the supercomputer execution queues*

Zoltán Gál[a], Tibor Tajti[b], György Terdik[a]

[a]University of Debrecen, Debrecen, Hungary
zgal@unideb.hu,terdik.gyorgy@inf.unideb.hu

[b]Eszterházy Károly College, Eger, Hungary
tajti@aries.ektf.hu

Submitted September 14, 2014 — Accepted February 10, 2015

Abstract

Huge amount of data is generated by and collected from the IoT (Internet of Things) physical and virtual devices. These sets of data series reflect in complex form the state of a given system in multidimensional space. Healthiness evaluation of a given system implies state analysis with enhanced methods. Special events can appear during the execution of jobs in a supercomputer (HPC – High Performance Computer) system. Depending on the HPC architecture hundreds or even thousands of computation nodes are working in parallel. The scheduler of the HPC front-end node manages different queues (parallel, serial, test, etc.) of the job execution. The multitude of data series captured periodically with several tens of thousands of samples creates a set of several dozen variables for each computation node. The healthiness of the whole HPC system is a temporal concept in the term of 2D or 4D multidimensional time-space domains. In this paper we propose a healthiness evaluation method for each execution queue of two different HPC system with 20 TFLOP/s and 5 TFLOP/s computation capacities, respectively. Time independent community structure is determined and controlled based on multiple similarity measures and ANN (Artificial Neural Network) based SOM (Self-Organized Map) algorithm. For each cluster of variables is

*This work was partially supported by the TÁMOP-4.2.2.C-11/1/KONV-2012-0001 (FIRST – Future Internet Research, Services and Technology) project. This project has been supported by the European Union, co-financed by the European Social Fund. This work was also partially supported by the European Union and the European Social Fund through project Supercomputer, the national virtual lab (grant no.: TAMOP-4.2.2.C-11/1/KONV-2012-0010).

determined a representing variable, including time specific and global characteristics of the own cluster. The resulting set of representing variables contains less than ten dissimilar time series. Wavelet methods are used for extreme event detection in time of each representing variable. The surprise event detection in time of the HPC execution queues is based on the simultaneity of extreme events' fingerprints.

Keywords: High Performance Computer, Sensors/actuators, IoT, Complex Event Processing, Event Stream Processing, SOM, FFT, STFT, Wavelets, Artificial Neural Networks

MSC: 91A28, 65C60, 60G35

1. Introduction

The Internet of Things (IoT) contains sensor and actuator devices connected by special network technologies providing services for different fields of the Information Society [1]. Classical network applications are extended establishing new, intelligent services based on the data generated by these devices. Conform to the current IoT conception of the European Union any usage of the IoT includes one of the following five areas: smart cities, smart energy, smart health, smart manufacturing and smart transport. The Machine-to-Machine (M2M) communication technology becomes more and more important in this aspect and creates huge amount of data every day. The sensor data has origin of different small physical or logical objects. These sensor objects are sampled periodically to capture the values or can force self-transmission of extra values asynchronously in case of special event occurrences.

The definition of special event term depends on the programmed threshold value of a given variable. The usability of the generated data depends on the type of measured variable. In some cases all the captured data should be archived (i.e. health or finance system monitoring) for offline processing possibility, but for the real time applications (i.e. process controlling) only the online filtered information has serviceable character. In any cases we are facing with the big data paradigm [2]. Efficient compressing and filtering algorithms are needed to process the data produced by the IoT. The online filtering and compressing mechanisms in IoT environment should take in consideration the energy usage, as well. The relation among the faithful of the data, the energy consumption of the sensor network and the capturing delay of the data is not trivial. Having complex aspects that need to be taken in consideration, lot of work is invested to find the optimal solution to this problem.

In this paper we are focusing on the data reduction mechanisms to decrease the amount of data captured from high number of sensor nodes with huge number of variables and significantly stored at the sink node of the sensor network. Two different architecture supercomputer systems are analysed to detect the surprise events during the execution time. In chapter 2 we present related work to the data clusterization methods and the surprise event detection based on wavelets. The measurement scenario in Massive Parallel Processing (MPP) and cluster architec-

ture High Performance Computer (HPC) systems with different job schedulers will be given in chapter 3. The mechanisms discussed previously will be applied, comparing the two HPC architectures. Chapter 4 concludes the analysis and gives possible continuation of the work.

2. Related work

One of the IoT benefit is the detection possibility of the actual events in the real world. A multitude of physical and logical sensors generate data about the analysed complex system and can be real-time analysed by the evaluation server [3]. Special events of a whole or a part of the system can happen in any interval of the execution time. Having several hundred or thousands of controllers with tens of variables each, makes necessary identification of the essential status variables. This reduction of the variables makes possible online identification of the special status events of the analysed system. Without the reduction of the number of variables huge computation capacity is reclaimed making very expensive the analysing hardware and software tool. The overall goal of the surprise event detection is to identify online the anomalies of the analysed system with minimum amount of computation capacity. The essence of three methods are presented shortly in this chapter because in the research work of surprise event detection these methods are not familiar, yet.

2.1. SOM clustering algorithm

Self-organizing map (SOM) is an Artificial Neural Network (ANN) based method to create two-dimensional picture where correlated elements are placed in physical vicinity [6]. This method allows a set of high-dimensional data to be represented on a single topographic map. The method has two phases (training, mapping) and four components (initialization, competition, cooperation, adaptation). In the training phase after a random initialization a vector quantization is executed by competition. In the mapping phase the new input vector is classified. Each element gets associated a weight vector with diameter equal to the number of elements and is placed on the map near to the elements with closest distance metric.

Two elements are considered to belong to the same cluster if their weight vectors are similar. To be more suggestive the weight vector itself is represented on two dimensional patterns with colored hexagon object for each weight. On Figure 1 the upper and the left hand patterns are similar, but they are dissimilar to the right hand side pattern. This is why $Element_1$ and $Element_2$ belong to the same cluster, but $Element_3$ does not.

2.2. DBSCAN clustering algorithm

Because of the high amount of captured data with only a small percentage of usable information, efficient clustering of the variables is proposed for special event detection of such systems [4], [5]. The clustering algorithm DBSCAN (Density-Based

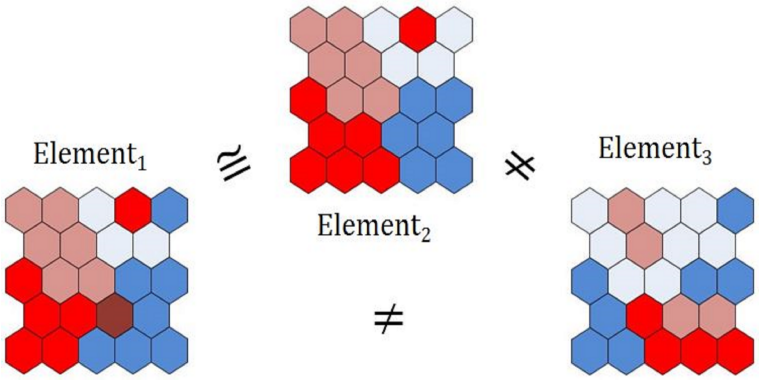


Figure 1: Cluster creation by SOM

Spatial Clustering of Applications with Noise) is the most cited one, it is efficient method to identify data clusters. It uses just two parameters: EPS, MIN_EL. EPS stands for the radius of the vicinity of a given element, and MIN_EL gives the minimum number of elements forming a cluster. If a number of MIN_EL elements are in a region of diameter EPS, then a cluster is started. If a given element does not belong to any cluster, then it is considered noise and remains alone.

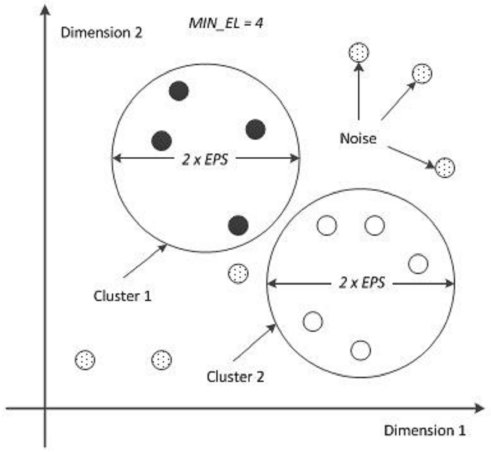


Figure 2: Cluster creation by SOM

This algorithm evaluates each element as cluster candidate and test the adherence to the existing clusters. The complexity of the algorithm for n elements is O[n log(n)] and the necessary memory capacity is O(n). The cluster elements can be shaped arbitrarily. Having notion of noise element makes this algorithm robust. As a disadvantage is that the border elements of a given cluster can take part in multiple clusters. The cluster creation depends on the distance measure

applied. Euclidian measure is used for scalar elements. For high-dimensional data other distance measures (cosine similarity, Sorensen-Dice coefficient, parametrized correlation index, Hamming distance, Jaccard index, Tanimoto similarity) can be used. For D-dimensional data the estimation of parameter MIN_EL is given by the following formula:

$$MIN\_EL \geq D + 1$$

Best value for parameter EPS is the maximum value that influence mostly the number of created clusters. If EPS is too small, then lot of noise elements will remain. If EPS is too large, then the majority of elements will belong to the same cluster. In Figure 2 three elements in the upper right corner can be included into a 2D sphere with 2xEPS diameter, but they do not form a cluster because the minimum number of elements in this case MIN_EL = 4. Enhanced variant of this algorithm is the generalized DBSCAN (GDBSCAN) where the values of the parameters are determined by the algorithm itself [5].

2.3. Special event detection by CWT

Fourier Transform (FT) is a useful tool for detecting global characteristics of a time series using spectral analysis. Identification in time of a given event is not possible by the FT or Fast Fourier Transform (FFT). Short-Time Fourier Transform (STFT) and Wavelet Transform (WT) makes possible to detect special events in time by the time-frequency representation of the analysed data series. Continuous Wavelet Transform (CWT) divides the continuous time into small waves named wavelets. A wavelet grows and decays in a limited time period. A detailed overview of the CWT can be found in [8] and [9]. Let $f(t)$ be a periodically sampled signal. The continuous wavelet coefficients $C_{a,b}$ of the signal $f(t)$ and wavelet $\psi_{a,b}(t)$ are given by:

$$C_{a,b}\left(f(t), \psi_{a,b}(t)\right) = \int_{-\infty}^{\infty} f(t)\, \psi_{a,b}(t)\, dt = \int_{-\infty}^{\infty} f(t)\, \frac{1}{\sqrt{a}} \psi\left(\frac{t-b}{a}\right) dt,$$

where a is the scaling parameter,b is the time shifting parameter and $\psi(t)$ is a mother wavelet function (i.e. Haar, Daubechies, Symlet, Coiflet, etc.). Any mother wavelet $\psi(t)$ is square integrable function with the following properties:

$$\int_{-\infty}^{\infty} \psi(t)\, dt = 0, \text{ and } \int_{-\infty}^{\infty} \psi^2(t)\, dt = 1.$$

An abrupt transition in signal produce large absolute values of the wavelet coefficients $C_{a,b}$. Wavelet coefficients $C_{a,b}$ localize the discontinuity best at small scales. The singularity of the original signal $f(t)$ only affects a small set of wavelet coefficients $C_{a,b}$.

On Figure 3, top part signal $f(t)$ can be seen being a discontinued function by shifting modification at $t_1 = 0.3$ and $t_2 = 0.7$ of the function $g(t) = 4sin(4\pi t)$.

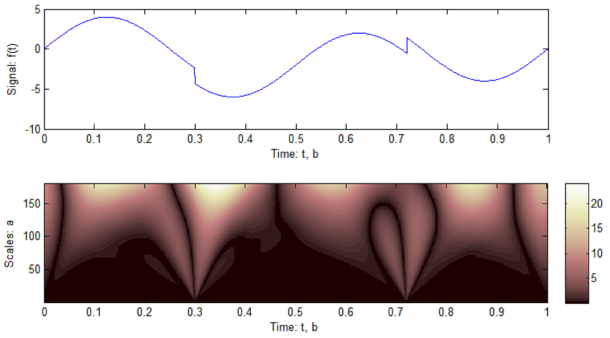


Figure 3: Special event identification by CWT

The bottom part of the figure shows the effects of this discontinuity on the CWT diagram. If periodicity exists in scaling dimension, then the original signal has fractal property [8].

3. Measurement scenario and analysis

The methods presented in the previous section were applied to analyse two different supercomputer (HPC) systems. The artificially cognitive capability of the dedicated sensor subsystem placed in HPC system measures different states of the compute nodes and transmits these values to the interpretation and processing machine [10]. There were captured state variables of the both HPC systems in production utilized by several hundred of users.

The detailed description and evaluation of the Massive Parallel Processing (MPP) architecture system is well presented in [11] and [12]. The second HPC architecture was cluster based with 32 compute nodes. The computation capacity of these two systems is 20 TFLOP/s (MPP) and 5 TFLOP/s (cluster), respectively. The MPP system was running SGE (Sun Grid Engine) job scheduler and the cluster system was scheduled by the open-source software SLURM (Simple Linux Utility for Resource Management).

SGE operates with three separated job queues but SLURM has only one queue. The number of captured variables differs only at the memory modules temperatures because the cluster HPC system has more RAM for each compute node. Each MPP CN has six RAM DIMMs, which number is sixteen at the cluster CN. The epoch time is $T = 10$ sec for both systems but the number of epochs was different because of the different continuous working times: $N_{MPP} = 55,341$; $N_{cluster} = 37,884$. The measurement time intervals were 6.4 days and 4.4 days, respectively.

Both HPC systems were running hundreds of jobs and other high number of jobs was waiting in the queue during the measurement. HPC systems have a given set of variables able to be captured by a dedicated controller based subsystem. In our case this hardware and software tool was Ganglia [11], [12]. The captured

	MPP HPC	Cluster HPC
No. of Compute Nodes (CN)	128	32
No. of CPUs/CN	2	2
No. of Cores/CPU	6	12
RAM/CN	48 GB	64 GB
Job Scheduler	SGE	SLURM
Functional Queues	SERIAL, PARALLEL, TEST	1

Table 1: Physical characteristics of the analysed HPC systems

variables are produced by to two types of sensors. Logical sensors (see Table 2.) measure capacity while physical sensors (see Table 3.) measure energy usage of the analysed systems. These variables are given by the Ganglia capturing tool depending on the HPC architecture type. Applying ANN based SOM algorithms it was found that for MPP system seven clusters can be created.

MPP and System Variable	Meaning
1. Load_one	Reported system load, averaged over one minute
2. Load_five	Reported system load, averaged over five minute
3. Proc_run	Number of running processes
4. Proc_total	Number of total processes
5. Pkts_in	Number of packets read from all non-loopback interfaces
6. Pkts_out	Number of packets written to all non-loopback interfaces
7. Bytes_in	Number of bytes read from all non-loopback interfaces
8. Bytes_out	Number of bytes written to all non-loopback interfaces
9. Mem_free	Memory free capacity
10. CPU_user	Percentage of CPU cycles spent in user mode
11. CPU_system	Percentage of CPU cycles spent in non-user mode

Table 2: Captured HPC logical variables by the Ganglia tool

Each cluster has representing variable including special characteristics of the represented group of variables. The same result was obtained with the DBSCAN algorithm, as well.

The seven variable clusters are: A(1,. . . , 4), B(5,. . . , 8), C(9), D(10), E(11), F(12,. . . , 14), G(15,. . . , 20), where the numbers are representing the index of the variables [12]. For cluster architecture having only one job queue the number of clusters is nine (see Figure 4): A(1,. . . , 3), B(4), C(5,. . . , 8), D(9), E(10), F(11), G(12), H(13,. . . , 14,29,. . . , 30), I(15,. . . , 28). This result shows that for the detection of extreme events of the HPC system the number of variables captured can be decreased by $\sim 70\%$ even the variable clusters of the two HPC architectures is different.

From the faithfulness point of view can be determined three different classes of

MPP System Variable	Cluster System Variable	Meaning
12. System_Temp	Server temperature [^{0} C]	Server temperature [^{0} C]
13. CPU1_Temp	CPU1 temperature	CPU1 temperature
14. CPU2_Temp	CPU2 temperature	CPU2 temperature
15. P1_DIMM1A_ Temp	Memory modules temperature	Memory modules temperaturel
17. P1_DIMM3A_ Temp	Memory modules temperature	Memory modules temperature
18. P2_DIMM1A_ Temp	Memory modules temperature	Memory modules temperature
20. P2_DIMM3A_ Temp	Memory modules temperature	Memory modules temperature

Table 3: Captured HPC physical variables by the Ganglia tool

variables: a) *Faithful variables* are forming strong clusters for both architectures (i.e. 1. Load_one, 2. Load_five, 3. Proc_run or 5. Pkts_in, 6. Pkts_out, 7. Bytes_in, 8. Bytes_out); b) *Migrant variables* are shifting from one variable cluster to other when the HPC architecture changes (i.e. 13. CPU1_Temp, 14. CPU2_Temp); c) *Isolating variables* remain alone independently of the HPC architecture type (i.e. 4. Proc_total, 9. Mem_free, 10. CPU_user, 12. System_Temp).

Faithful variables form clusters with relatively high number of participants. This property reduces significantly the necessary sampled variables for extreme event detection task. The class of migrant variables contains only reduced number of candidates and characterizes the difference between the two HPC architectures. The isolating variables do not depend on the HPC architecture and are strong characterizing elements of the HPC execution state. Using CWT it was found that the different class of variables has different wavelet transform.

On Figure 5 can be seen that there were three short time intervals for variable 1. Load_one at the CN5 with significant changes. The same three time intervals can be seen on Figure 6 for variable 4. Mem_free of the same node. In these moments the cluster architecture HPC had surprise event detected by the CWT. The significant changes of the two different variables are detected in this way. Coincidence of several variable changes can be detected by the CWT in the same manner.

Even different classes of variable have different CWT, extreme events are detected by every variable cluster because the fingerprint of the surprise events (vertical lines on the CWT maps for different isolating variables) matches. In our case these surprise events appeared at around three epoch numbers: $b_1 = 2.3 \times 10^6, b_2 = 3.3 \times 10^6, b_3 = 3.6 \times 10^6$.

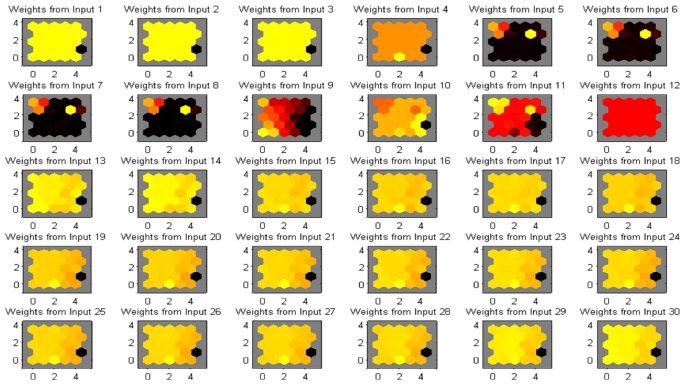


Figure 4: SOM maps for Cluster architecture HPC, CN5

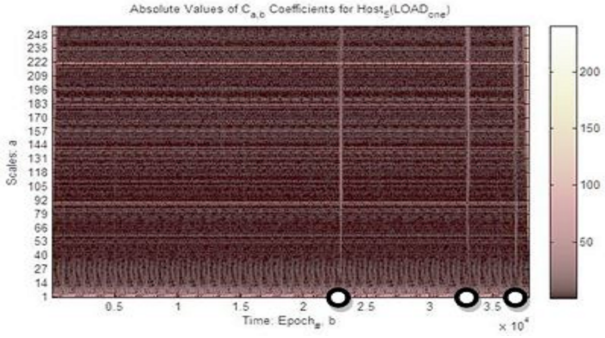


Figure 5: CWT of Cluster architecture HPC, variable $CN_{5,1.Load\_one}$

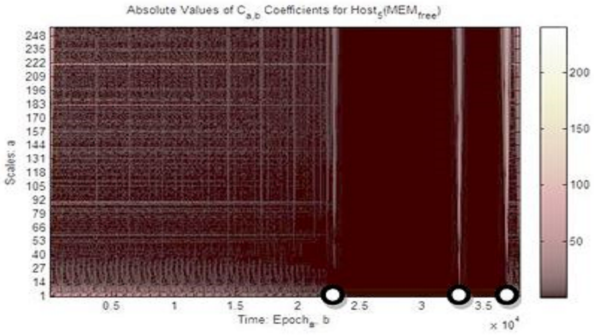


Figure 6: CWT of Cluster architecture HPC, variable $CN_{5,4:Mem\_free}$

4. Conclusions

SOM and DBSCAN are useful methods to create variable clusters of the data sampled from sensor networks. This clusterization of the variables reduces the analysed amount of data with $\sim 70\%$. By analysing variables of MPP and cluster based HPC systems three classes of variables are proposed: faithful, migrant and isolating one. Each class has special role in the minimizing method of the number of HPC state variables necessary to detect special events. Even several jobs were waiting for the execution with SGE scheduler some of the hardware resources were out of work. The occurrence of surprise events at the HPC system execution can be detected by the constant wavelet transform (CWT) of a representive element of the variable clusters. Further analysis needs to find the exact characteristics of the variable classes proposed.

References

[1] Vermesan, O., Friess, P., Internet of Things – Converging Technologies for Smart Environments and Integrated Ecosystems, *River Publishers*, 2013.

[2] Smith, J. G.,Vermesan, O., Friess, P. Furness, A., The Internet of Things 2012: New Horizons, *Halifax Publisher*, ISBN: 978-0-9553707-9-3, 2012.

[3] Jain, A. K., Murty, M. N., Flynn, P. J. , Data Clustering: A Review, *ACM Computing Surveys*, 31 (3), 264-323, 1999.

[4] Ester, M., Kriegel, H.-P., A density-based algorithm for discovering clusters in large spatial databases with noise, *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, AAAI Press. 226–231. ISBN 1-57735-004-9., 1996

[5] JIANG, D., Pei, J., Zhang, A., DHC: A Density-Based Hierarchical Clustering Method for Time Series Gene Expression Data, *BIBE*, 393-400, 2003

[6] Kohonen, T., Self-organized formation of topologically correct feature maps, *Springer-Verlag, Biological Cybernetics*, Volume 43, Issue 1 , 59-69, 1982

[7] Mallat, S., G., A theory for multiresolution signal decomposition: the wavelet representation, *IEEE Transactions on Pattern Recognition and Machine Intelligence*, 11 (7): 674–693, 1989.

[8] Percival, D., B., Walden, A. T., Cambridge Series in Statistical and Probabilistic Mathematics: Wavelet Methods for Time Series Analysis, *Cambridge University Press*, 2006.

[9] Sadek, S. A., Michaelis, A. B., Sayed, U., A statistical framework for real-time traffic accident recognition, *Journal of Signal and Information Processing*, Vol. 1, 70–81, 2010.

[10] Special Issue on Cognitive Infocommunications, *Journal of Advanced Computational Intelligence and Intelligent Informatics*, (2012) 16: 2.

[11] Terdik, Gy., Gal, Z. , Advances and practice in Internet of Things: A case study, *Proceedings of 4th IEEE International Conference on Cognitive Infocommunications*

(CogInfoCom 2013), Budapest, Hungary, December 2-5, 2013, ISBN: 978-1-4799-1544-6, 435-440.

[12] Gal, Z., Tajti, T., Complex Event Processing in Supercomputer Environment: Sensor and Neural Network Based Analysis, *Proceedings of IEEE 4th International Conference on Cognitive Infocommunications (CogInfoCom 2013)*, Budapest, Hungary, December 2-5, 2013, ISBN: 978-1-4799-1544-6, 735-740.

Annales Mathematicae et Informaticae
44 (2015) pp. 99–109
http://ami.ektf.hu

Clang matchers for verified usage of the C++ Standard Template Library*

Gábor Horváth, Norbert Pataki

Department of Programming Languages and Compilers
Eötvös Loránd University, Budapest, Hungary
xazax.hun@gmail.com, patakino@elte.hu

Submitted July 20, 2014 — Accepted March 5, 2015

Abstract

The *C++ Standard Template Library (STL)* is the exemplar of generic libraries. Professional C++ programs cannot miss the usage of this standard library because it increases quality, maintainability, understandability and efficacy of the code. However, the usage of C++ STL does not guarantee error-free code. Contrarily, incorrect application of the library may introduce new types of problems. Unfortunately, there is still a large number of properties are tested neither at compilation-time nor at run-time. It is not surprising that in implementation of C++ programs so many STL-related bugs are occurred.

We match patterns on *abstract syntax trees* (AST) with the help of predicates. The predicates can be combined and define an embedded language. We have developed a tool which finds the potential missuses of the STL as a validation of our approach. The software takes advantage of the Clang AST-Matcher technology. The tool is in-use in Ericsson. We advise new matchers that have get into the Clang code base.

Keywords: C++ STL, generic programming, Clang, AST, static analysis, code validation

MSC: 68N19 Other programming techniques

*This study/research was supported by a special contract No. 18370-8/2013/TUDPOL with the Ministry of Human Resources.

1. Introduction

The *C++ Standard Template Library* (STL) was developed by *generic programming* approach [2]. In this way containers are defined as class templates and many algorithms can be implemented as function templates [1]. Furthermore, algorithms are implemented in a container-independent way; so one can use them with different containers [22]. C++ STL is widely-used because it is a very handy, standard library that contains beneficial containers (like list, vector, map, etc.), a lot of algorithms (like sort, find, count, etc.) among other utilities.

The STL was designed to be extensible [3]. We can add new containers that can work together with the existing algorithms. On the other hand, we can extend the set of algorithms with a new one that can work together with the existing containers. Iterators bridge the gap between containers and algorithms. The expression problem is solved with this approach [23]. STL also includes adaptor types which transform standard elements of the library for a different functionality [14]. By design, STL is implemented with application of C++ templates to ensure the efficiency. A runtime model of this approach is available [20].

However, the usage of C++ STL does not guarantee bug-free or error-free code [5]. Contrarily, incorrect application of the library may introduce new types of problems [10].

One of the problems is that the error diagnostics are usually complex and very hard to figure out the root cause of a program error [24, 25]. Violating requirement of special preconditions (e.g. sorted ranges) is not checked, but results in runtime bugs [19]. A different kind of stickler is that if we have an iterator object that pointed to an element in a container, but the element is erased or the container's memory allocation has been changed, then the iterator becomes *invalid*. Further reference of invalid iterators causes undefined behaviour [4].

Another common mistake is related to algorithms which are deleting elements. The algorithms are container-independent, hence they do not know how to erase elements from a container, just relocate them to a specific part of the container, and we need to invoke a specific erase member function to remove the elements physically. Therefore the `remove` and `unique` algorithms, for example, do not actually remove any element from a container [9].

The aforementioned `unique` algorithm has uncommon precondition. Equal elements should be in consecutive groups. In general case, using `sort` algorithm is advised to be called before the invocation of `unique`. However, `unique` cannot result in an undefined behaviour but its result may be counter-intuitive.

Some of the properties are checked at compilation time. For example, the code does not compile if one uses sort algorithm with the standard list container because the list's iterators do not offer random accessibility [18]. Other properties are checked at runtime. For example, the standard vector container offers an `at` method which tests if the index is valid and it raises an exception otherwise [16].

Unfortunately, there is still a large number of properties are tested neither at compilation-time nor at run-time. Observance of these properties is in the charge

of the programmers. Lint-like tools are based on static analysis for detect some kind of missuses that can be compiled but at runtime they cause problems.

Associative containers (e.g. `multiset`) use functors exclusively to keep their elements sorted. Algorithms for sorting (e.g. `stable_sort`) and searching in ordered ranges (e.g. `lower_bound`) are typically used with functors because of efficiency. These containers and algorithms need *strict weak ordering* [17]. Containers become inconsistent, if the used functors do not meet the requirement of strict weak ordering [12].

In this paper we argue for a new approach based on static analysis. Our approach matches patterns on abstract syntax trees with predicates written in functional style. We have created a tool as a validation of the approach. This tool detects many kind of missuses of the C++ STL in the source code. The tool uses the Clang architecture [8]. Our tool is utilized by our industrial partner.

This paper is organized as follows. In section 2 we analyze how our approach works, in section 3 the overview of our system is detailed. We briefly present our tool in section 4. We present the core of some checkers as an example in section 5. We summarize other approaches that are based on static analysis in section 6. Finally, this paper concludes in section 7.

2. Pattern matching on syntax trees

Our approach is based on pattern matching on the syntax tree of the analyzed program source code. We use the syntax tree that is generated by the Clang [8] compiler. The syntax tree of a compiler contains sufficient amount of information to answer several questions regarding the source code.

However, in order to parse the source of the program we need to know the exact compiler arguments that were used to compile that application. This is necessary because the compiler arguments can modify the semantics of the source code; for example macros can be defined using compiler arguments.

To collect the compilation arguments the most efficient and portable way is to use fake compilers that are logging their parameters and forward them to a real compiler afterwards. This way the logging itself is independent of the build system that is used. The source code is parsed with the same compilation parameters that were logged.

After we retrieved the syntax tree of the analyzed program from the compiler the pattern matching process begins. Multiple patterns are matched lazily on the syntax tree with only one traversal. The source positions for the matched nodes in the syntax tree are collected.

The source positions in the collected results are filtered based on exclude lists that contains of the false positive matches. These exclude lists have to be maintained by the user of the tool. Afterwards the positions are translated into user-friendly warning messages.

One of the downsides is that, the compiler can only parse one translation unit at a time. Some useful information might reside in a separate translation unit making

it impossible to detect some class of issues. Fortunately due to the structure of STL most of the library code is available in system header files. For this reason if a translation unit is utilizing some STL features, the corresponding headers are likely to be the part of that unit. This structure mitigates the limitations of the compiler, translation unit boundaries are not likely to be a problem when analyzing STL misuse patterns.

The solutions presented in this paper are not utilizing any symbolic execution or other path sensitive data. Several patterns can be detected using only pattern matching on the AST and this pattern matching procedure is more efficient than symbolic execution.

3. Overview of the architecture

Each of the checkers that are able to detect certain class of bad smells are implemented as a predicate on the syntax tree. These predicates are loosely coupled. We focus on the extendibility, thus it is very easy to add further checkers to our tool.

Each predicate has two states representing whether it is activated in the current invocation of the tool or not and a list of matches (that was marked as false positives) that should be ignored. The predicates should not contain other states, because a new predicate object will be instantiated for each translation unit that is checked. There is no efficient way to preserve states between those invocations mainly because the user is allowed to invoke multiple instance of our checker tool on multiple translation units at the same time. If a predicate is activated it consists of a matcher and a callback. The matcher is an `ASTMatcher` object that is built by an embedded domain specific language available in the Clang ASTMatcher library. This is a first filter on the syntax tree and it can be retrieved by the `getMatcher` method. The callback can do further filtering to decide whether the match found by the ASTMatcher object is suitable to be reported to the user. The callback is the override of the `matches` method. The callback essential because while the matchers are extremely useful they can not express any possible patterns on the AST. The `getMatcher` method is never invoked on inactive checkers, the matcher expression will not be generated and the callback will never be called. The `MatcherProxy` returned by the `getMatcher` method is necessary because there are different kind of matchers for declarations, statements and types. The `MatcherProxy` is a discriminated union of fundamental matcher types from which the original type can be retrieved later.

Code 1: Predicate interface

```
struct Predicate {
  Predicate() : _active(false) {}
  virtual ~Predicate() {}

  virtual bool matches(const MatchResult &result_) = 0;
  virtual void configure(std::string conf_) = 0;
```

```
  virtual std::string getErrorMsg() = 0;
  virtual std::string getID() = 0;
  virtual std::string getBoundID() = 0;
  virtual MatcherProxy getMatcher() = 0;
  virtual std::vector<MatchPosition>& getExcludes() {
    return _excludes;
  }
  virtual bool isActive() { return _active; }
  virtual void setActive(bool active_) { _active = active_; }

protected:
  bool _active;
  std::vector<MatchPosition> _excludes;
};
```

Each predicate is configurable. This is necessary to make sure the predicates can adapt to wide variety of code bases. The ASTMatcher object and the callback function can make decisions based on configuration values. The predicates get only plain text as configuration because it is hard to predict what kind of configuration data is needed for future predicates. This gives some flexibility to the implementers. However, some basic tools are provided to parse key/value pairs.

Unfortunately static analysis tools are prone to false positive results. The users must be able to suppress the false positive warnings. The `_excludes` vector contains these locations. The exclude list is unique to all of the predicates. The exclude list is parsed after the configuration file was processed.

All of the predicates can be identified by a unique identifier. That identifier is used to determine whether a predicate must be active in an invocation, what configuration values belong to that predicate, and which excludes should be added to its exclude list. The identifier can be gathered by invoking the `getID` method.

Every predicate can have a unique error message that is displayed when a match is found. This message is emitted for every source location that is detected by the given predicate.

The predicates can match not only a single node of the syntax tree but a subtree. In case of a subtree is matched it is ambiguous which node should be used to retrieve the source location of the match. The `getBoundID` can be used to identify which node should be used as a source location to generate the warning report. For example if the subtree is a function call, there must be a way to determine what source location of the function call should be reported: the whole function call or one of its arguments.

Code 2: Registering checkers

```
Config::Config(BuildLogParser& log_parser) :
  _log_parser(log_parser) {
  ADD_MATCHER(StlBoolVectorPred);
  // ...
  ADD_MATCHER(StlPolimoContPred);
}
```

When a new checker is implemented it must be the subtype of the `Predicate` class. After all of the pure virtual methods are implemented the checker must be registered with a configuration class that handles the instantiation of the checker objects. The registration of the checker is done in the constructor of the configuration class. To avoid the necessity of modifying an unrelated file when implementing a new checker, the code that is responsible for registering checkers can be automatically generated by an external script that is inspecting the implementation files.

4. Matchers for STL usage validation

The architecture we developed proved to be useful. We implemented 14 checkers to detect STL specific errors or suspicious code snippets. We tried to focus on patterns that are most likely to appear in real world applications. These checkers are the following:

- Bool Vector checker warns about usages of vector of booleans. The reason is that `std::vector<bool>` is a template specialization that does not fulfil the requirements of `std::vector` [13].

- Container of `std::auto_ptr`s is a dangerous construct in C++. The `std::auto_ptr` is a smart pointer that manages its own resource by deleting the pointer it wraps in its destructor. The problem is that its copy constructor transfer the ownership of the pointer from the source of the copy to the target of the copy. When the user of the STL have a container that contains such pointers and use an algorithm on the container that involves copying then some pointers may reclaim their resources in an unintended way [13].

- Invoking the `std::find` and `std::count` algorithm on an associative container in STL is not efficient. The general `std::find` and `std::count` algorithms have no information about the internal representation of the containers. This is the reason why they cannot utilize that the objects in an associative container are ordered. The programmer should use the `find` and `count` methods of the associative container instead.

- One can get the wrapped iterator from a reverse iterator through the base method. However, using this method requires extra attention from the programmer, because the iterator retrieved this way is not pointing to the same object as the reverse iterator does [15].

- The functors used with STL algorithms and containers should be derived from some specific STL types that adds some typedefs to the functor to make it possible to inspect the return type and the argument types of the functor. It is very error prone to define those types multiple times. This checker warns if the types do not match.

- Allocators should not have any state in C++98/C++03 codes.
- Functors used as predicates with STL algorithms should not have any state. This is important because it is undefined by the standard how many times may the functor object be copied.
- Functors are passed by value to algorithms. The polymorphism only works through pointers and references. Developers should not use virtual methods in functor classes and it can be the indicator of an error.
- The emptiness of a container should be checked with the `empty` method instead of using the `size` method. The empty method can be implemented more efficiently for most of the containers. Moreover more containers support the `empty` method than the `size` method. In a code base using `empty` for emptiness check the container types can be swapped more easily.
- The capacity and the number of elements in `std::vector` is not the same. The capacity defines how much memory the vector uses and it can be much more than the required. To optimize the memory consumption it is a common trick to copy the contents of the vector to a temporary object and swap the vector with that temporary afterwards. Since the C++11 standard there is a much more comprehensible and better performing way to achieve the same result is using the `shrink_to_fit` method. This checker warns the user to replace the copy and swap tricks with a `shrink_to_fit` method call.
- Algorithms that remove elements from a container will not delete the elements from the container. They will only overwrite the elements that need to be removed with other elements from the end of the container. After running such algorithm at the end of the container there will be some redundant elements that needs to be erased using the `erase` method of the container.
- Copying algorithms cannot guarantee that the target container have sufficient size to store all of the elements that is in the source container. It is advised to use iterator adaptors for example the back inserter iterator adaptor to avoid buffer overflows.
- The containers in the STL are not designed to be the part of a class hierarchy. Using the containers in a polymorphic way is an error and should be avoided.
- Because of implicit type conversions a number can be assigned to a `std::string` object. In most cases this is a programming error and there is a missing explicit conversion to string.

Some of the bad smells and coding style advices that are found by our tool is not detected by other static analysis tools available at the time of writing this article. We found defects in the codebase of our industrial partner, however they did not provide us with any data on the number of defects they found.

5. Example

In this section we briefly present two checkers' implementation as an example. We focus on the technology and our approach.

Our tool analyzes if a programmer calls the `find` or `count` algorithm on a sorted container because this causes efficiency loss at runtime. The algorithm is a function template, thus it can called on the `set` or `multiset` object because the algorithm has a template parameter for the iterator. The heart of the checker takes advantage of Clang architecture. The following code snippet validates the usage of the `find` or `count` algorithm:

Code 3: Inefficient Algorithm: ASTMatcher

```
MatcherProxy StlFindCountPred::getMatcher() {
  return
    callExpr(
        callee(functionDecl(
        anyOf(hasName("std::find"),
              hasName("std::count")))),
      hasArgument(0,
        hasDescendant(expr(hasType(typedefType(hasDecl(
          matchesName(
            "std::(multiset|set).*"
            "::(const_iterator|iterator)"
           ))))))).bind("id");
}
```

This code checks if the parsed source code is a call expression, where the called function is one of enumerated standard algorithms and it is called on the sorted container. The code analyzes the type of algorithm's first argument. If the name of type matches to the arbitrary inner type `iterator` or `const_iterator` of `set` or `multiset` our tool reports warning to the user. However, this code is written in C++ but with functional approach.

The STL containers are not prepared to be used polymorphically. If a user decide to inherit from a container and cast a pointer to the derived type to a pointer to the container type it is probably the indicator of an issue.

Code 4: Polimorphic container: ASTMatcher

```
MatcherProxy StlPolimoContPred::getMatcher()
{
  DeclarationMatcher container = unless(anything());
  for(const auto& e : gContainers)
    container = anyOf(recordDecl(hasName(e)),
                      container);

  return
    implicitCastExpr(hasImplicitDestinationType(
      pointsTo(container))).bind("id");
}
```

The `gContainers` variable contains the list of the containers in the STL. The codesnippet above shows the power of creating matchers dynamically on the fly. We want to warn the user, if there is an implicit cast to a pointer to a container.

Code 5: Polimorphic container: Callback

```
bool StlPolimoContPred::matches
  (const MatchFinder::MatchResult &result_)
{
  const ImplicitCastExpr* cast =
    result_.Nodes.getDeclAs<ImplicitCastExpr>("id");

  return cast->getCastKind () == CK_DerivedToBase;
}
```

There can be several types of casts, but we are only interested in those types of implicit casts, that involves derived to base conversion. There is no matcher to ensure the cast type. For this reason we have implemented a small callback to check the type of the cast.

6. Related work

There are some approaches to discover the erroneous STL usage with static analysis. STLlint was the first application that analyses source code for detecting inaccurate application of the library [7]. STLlint first parsed C++ code (with Edison Design Group C++ Front End [6]) and then transformed it into a simpler internal representation language called "Semple". During this transformation process, STLlint replaced the implementation of STL components with simplified models that specify the interface (but not the implementation) of the components. The resulting program was then passed to the Semple static analysis engine that "executed" the program symbolically, checking that assertions (part of the component specifications) always prove true. Unfortunately, the support, maintenance and availability of this tool have been cancelled. CppCheck is a static analysis tool that also have some limited STL support [26].

C++ template metaprogramming is an emerging new paradigm that is able to add further validation to the compiler [21]. This approach effectively can be used for evaluating the semantic usage of the STL. First, it is enough to parse the source code once. Second, this approach supports the extendibility of the library which is a major feature of the generic programming paradigm. However, this approach has disadvantages, as well. In the metaprogramming realm there is no AST or something high-level approach of the source code. Metaprogram developers usually deal with template instantiations to trigger compilation failures or warnings [17].

Some of the STL-related issues are detected in a metaprogram-driven way: overcome of stateful allocators and reverse iterators are implemented in [15], proper usage copying algorithms is verified in [17]. The usage of `vector<bool>` and containers of auto pointers can be detected [13]. The semantic check of functors are detailed in [11, 12].

7. Conclusion

We have developed a static analysis tool based on Clang technologies that is easy to extend. The success of the architecture is proven by the 14 checker that we implemented. Some of the checkers are implementing new guidelines that was not published nor checked before. During the development we contributed several patches to the Clang ASTMatcher library. The tool is utilized by our industrial partner.

References

[1] Alexandrescu, A.: "Modern C++ Design", Addison-Wesley (2001)

[2] Austern, M. H.: "Generic Programming and the STL: Using and Extending the C++ Standard Template Library", Addison-Wesley (1998)

[3] Czarnecki, K., Eisenecker, U. W.:"Generative Programming: Methods, Tools and Applications", Addison-Wesley (2000)

[4] Dévai, G., Pataki, N.: *Towards verified usage of the C++ Standard Template Library*, in Proc. of the 10th Symposium on Programming Languages and Software Tools, (SPLST) 2007, pp. 360–371.

[5] Dévai, G., Pataki, N.: *A tool for formally specifying the C++ Standard Template Library*, Ann. Univ. Sci. Budapest. Comput., **31** (2009), pp. 147–166.

[6] Gibbs, T. H., Malloy, B. A., Power, J. F.: *Progression Toward Conformance for C++ Language Compilers*, Dr. Dobbs Journal **28(11)** (2003), pp. 54-60.

[7] Gregor, D., Schupp, S.: *STLlint: Lifting static checking from languages to libraries*, Software – Practice & Experience, **36(3)** (2006), pp. 225–254.

[8] Lattner, C.: *LLVM and Clang: Next Generation Compiler Technology*, The BSD Conference, 2008.

[9] Meyers, S.: "Effective STL - 50 Specific Ways to Improve Your Use of the Standard Template Library", Addison-Wesley (2001)

[10] Pataki, N.: *C++ Standard Template Library by Ranges*, in Proc. of the 8th International Conference on Applied Informatics (ICAI 2010), Volume 2, pp. 367–374.

[11] Pataki, N.: *C++ Standard Template Library by Safe Functors*, in Proc. of 8th Joint Conference on Mathematics and Computer Science, MaCS 2010, Selected Papers, pp. 363–374.

[12] Pataki, N.: *Advanced Functor Framework for C++ Standard Template Library*, Studia Universitatis Babeş-Bolyai, Informatica, **LVI(1)** (2011), pp. 99–113.

[13] Pataki, N.: *C++ Standard Template Library by template specialized containers*, Acta Universitatis Sapientiae, Informatica **3(2)** (2011), pp. 141–157.

[14] Pataki, N.: *Safe Iterator Framework for the C++ Standard Template Library*, Acta Electrotechnica et Informatica, Vol. **12(1)** (2012), pp. 17–24.

[15] Pataki, N.: *Compile-time Advances of the C++ Standard Template Library*, Annales Universitatis Scientiarum Budapestinensis de Rolando Eötvös Nominatae, Sectio Computatorica **36** (2012), Selected papers of 9th Joint Conference on Mathematics and Computer Science MaCS 2012, pp. 341–353.

[16] Pataki, N., Porkoláb, Z., Istenes, Z.: *Towards Soundness Examination of the C++ Standard Template Library*, in Proc. of Electronic Computers and Informatics, ECI 2006, pp. 186–191.

[17] Pataki, N., Porkoláb, Z.: *Extension of Iterator Traits in the C++ Standard Template Library*, in Proc. of the Federated Conference on Computer Science and Information Systems, FedCSIS 2010, pp. 911–914.

[18] Pataki, N., Szűgyi, Z., Dévai, G.: *C++ Standard Template Library in a Safer Way*, In Proc. of Workshop on Generative Technologies 2010 (WGT 2010), pp. 46–55.

[19] Pataki, N., Szűgyi, Z., Dévai, G.: *Measuring the Overhead of C++ Standard Template Library Safe Variants*, Electronic Notes in Theoret. Comput. Sci., **264(5)** (2011), pp. 71–83.

[20] Pirkelbauer, P., Parent, S., Marcus, M., Stroustrup, B.: *Runtime Concepts for the C++ Standard Template Library*, in Proc. of the 2008 ACM symposium on Applied computing, pp. 171–177.

[21] Porkoláb, Z.: *Functional Programming with C++ Template Metaprograms*, Lecture Notes in Comput. Sci., **6299** (2010), pp. 306–353.

[22] Stroustrup, B.: "The C++ Programming Language", Addison-Wesley (1999)

[23] Torgensen, M.: *The Expression Problem Revisited – Four new solutions using generics*, Lecture Notes in Comput. Sci., **3286** (2004), pp. 123–143.

[24] Zolman, L.,: *An STL message decryptor for visual C++*, C/C++ Users Journal, **19(7)** (2001), pp. 24–30.

[25] Zólyomi, I., Porkoláb, Z.: *Towards a General Template Introspection Library*, Lecture Notes in Comput. Sci., **3286** (2004), 266–282.

[26] Joshi, A., Tewari, A., Kumar, V., Bordoloi, D.: *Integrating Static Analysis Tools for Improving Operating System Security*, International Journal of Computer Science and Mobile Computing, Vol. **3(4)**, (2014), pp. 1251–1258.

Annales Mathematicae et Informaticae
44 (2015) pp. 111–120
http://ami.ektf.hu

On a secure distributed data sharing system and its implementation

Péter Kasza, Péter Ligeti, Ádám Nagy

Eötvös Loránd University
Department of Computeralgebra
pkasza@caesar.elte.hu, turul@cs.elte.hu, spigy88@inf.elte.hu

Submitted September 23, 2014 — Accepted February 10, 2015

Abstract

In this paper we propose a decentralized privacy-preserving system which is able to share sensible data in an encrypted way, that only predefined subsets of authorized entities can recover the data after getting an additional alarm message. In the paper we give a short description of the necessary cryptographic building blocks and the communication protocol. Furthermore, we present the main communication channels and the implementation of the proposed data sharing system. The proposed system achieves the desired functionalities by using secret sharing and two communication networks: an ordinary P2P network where the encrypted information is stored, and a smaller private P2P network called friend-to-friend network, which consists of the authorized parties and handles messages that are necessary to the decryption. The main part of the paper concentrates on the implementation of the system.

Keywords: private P2P network, secret sharing, symmetric cryptography

MSC: 94A62, 68P25

1. Introduction

1.1. Motivation

The number and role of smart devices show an intensive growth nowadays, they are collecting, storing and sending a large amount of sensitive data about the

owner of the device. Communication devices, like smartphones or tablets can have several built-in sensors, such as accelerometers, digital compasses, gyroscopes, GPS trackers, microphones and cameras. Dedicated self-tracking devices measuring various medical data of the user, like blood-pressure, pulse, blood-sugar level, etc. The sensible data collected by these devices are often handled (additionally stored and analyzed) by a central entity, usually a mobile or cloud service provider, raising serious privacy concerns. The main goal of this paper is to propose a privacy-preserving communication framework, wherein the sensible data is not stored by a singular trusted third party, but instead distributed to some predefined subset of users such that large coalition of users is necessary to recover the data. In the proposed solution the encrypted data packages are stored within an open P2P network and the necessary decryption key is distributed in a private network called friend-to-friend (or F2F) network. In order to decrease the communication and space consumption of the users and avoid a possible adversarial tracking, it is required that the original data can be recovered only in the presence of a special event, called an *alarm message*. Examples of such alarm messages are extremities in medical data, S.O.S. signals in case of a physical attack or stroke, etc.

1.2. Communication building blocks: P2P and F2F networks

A *peer-to-peer (P2P) network* is a type of decentralized and distributed network architecture in which the individual nodes of the network (called peers) act both as suppliers and consumers of resources, in contrast to the client-server model where client nodes request access to resources provided by central servers. In our protocol the P2P architecture is used for file sharing; the participants are able to search, upload and download messages from the P2P network.

A *friend-to-friend (or F2F) computer network* is a type of private peer-to-peer network in which the users only make direct connections with people they know. Unlike other kinds of private P2P networks, the users in a friend-to-friend network cannot find out who else is participating beyond their own circle of friends, so F2F networks can grow in size without compromising their users' anonymity. Many F2F networks support indirect anonymous or pseudonymous communication between users who do not know or trust one another. For example, a node in a friend-to-friend overlay can automatically forward a file (or a request for a file) anonymously between two friends, without telling either of them the other's name or IP address. These friends can in turn automatically forward the same file (or request) to their own friends, and so on. Historically, the first F2F system was Turtle [3], recent examples of popular implementations are RetroShare [6] and OneSwarm [5]. For the underlying P2P system we chose the BitTorrent protocol's DHT network and implemented our own friend-to-friend scheme on top of it by creating encrypted communication channels between the nodes.

1.3. Cryptographic primitives: secret sharing and symmetric cryptography

A *secret sharing scheme* is a method of distributing secret data among a set of participants so that only specified qualified subsets of participants are able to recover the secret from its parts of information called shares. In addition, if the unqualified subsets collectively yield no extra information, i.e. the joint shares are statistically independent of the secret, then the scheme is called perfect. For a given positive integer t a secret sharing scheme is called t-threshold, if every subset of participants with cardinality at least t can recover the secret.

Secret sharing was first introduced independently by Blakley [1] and Shamir [7]. In both papers the authors constructed perfect t-threshold schemes. Here we present the method of Shamir, which can be easily implemented due to its simplicity.

Example 1.1 (Shamir)**.** Let the participants indexed by the non-zero elements of a finite field $\mathbb{F}$ and let p be polynomial of degree at most $t-1$ over $\mathbb{F}$ chosen uniformly at random. The share of participant i is $p(i)$ and the secret is the the constant term of $p(x)$, i.e. $p(0)$.

In the proposed protocol we assume that at least some of our friends in the F2F network are trustful i.e. can be expected to follow the protocol. We use the pairwise secret channels between the participants for communication. The security of these end-to-end communication channels are guaranteed by *symmetric cryptographic primitives*, especially by symmetric encryption schemes. Informally, a *symmetric encryption scheme* consist of three algorithm: the *key-generation*, where the common secret key is established and sent on a secret channel; the *encryption* where the sender computes the ciphertext of a given message using the symmetric key and the *decryption*, where the receiver computes the message from the ciphertext using the symmetric key. A message (key) of an encryption scheme is *computationally secure* if any probabilistic polynomial time adversary is able to learn some information about the message (key) with negligible probability only. Note that, this is rather an informal definition, but here we just highlight the main cryptographic ideas and results: we use that most of the widely used encryption schemes are proven to be computationally secure. For example, in the case of the OneSwarm network, RSA is used: every user generates a 1024 bit public/private RSA key pair when installing the client, with the public key serving as its identity. After a key-exchange between the friends, the participants can connect to one another using secure sockets (SSLv3) bootstrapped by their RSA key pairs. Furthermore forward security can be achieved by establishing ephemeral Diffie-Hellman keys between the participants.

2. Protocol description

Within this section we present an informal description of the proposed protocol together with the desired security requirements. This paper concentrates on the communication channels and the implementation details, hence the exact protocol description and the proof of security is contained in a separate paper [4].

2.1. Parameters

The participants of the protocol are the following: Alice, the sender of the data and the alarm message, Alice's friends in the F2F network and further participants using an open P2P network. We suppose two communication channels: a F2F network consists of Alice and her friends and a P2P network. Alice is able to make a digital signature for integrity protection and the encryption for every message and key. Alice has a secret symmetric key with every friend of her in the F2F network.

2.2. Protocol description

The protocol has three main phases: the first one is *Uploading*, where the sender first generates a temporary key and uploads the encrypted message in the P2P network. Next, the sender distributes the collection of encrypted temporary keys together with the list of identifiers of other shares and the message according to a t-threshold secret sharing scheme. Finally, the shares are sent to her friends in the F2F network.

The second step is the *Downloading* phase, in which the alarm message is sent first to the friends. After getting the alarm signal, the friends distribute their encrypted shares into the P2P network and then download the remaining parts from the P2P network based on the identifiers of the shares and the message.

The last stage is the *Message recovery* step, where every friend checks the integrity and authenticity of the downloaded packages, computes the encrypted temporary key from the correct shares and decrypt the session key with the symmetric key and the original message with this decrypted key as well.

2.3. Security model

We suppose two opposite user behaviors. The first kind of participants is called *honest*, meaning that they always follow the steps of the protocol and compute/send nothing more. The other extremity is a *malicious* participant who is not supposed to send messages according to the protocol, but is not able to interrupt the communication. Because the connections in a F2F network are based on real personal relationships and trust, we will suppose that the honest friends are in majority.

Intuitively, we need that if there are "many good friends" then every friend is able to get the message when the protocol finishes (even the "bad" ones). Furthermore, it is necessary that no small subset of participants can learn any information

about session key (including non-friends) as long as Alice doesn't send the alarm signal and any subset of entities out of the F2F network learns nothing about the message. Here we collect the precise security requirements that the proposed scheme has to satisfy:

- ***Correctness:*** if there is a set of at least t honest friends, then every friend can recover the original message at the end of the protocol.
- ***Key Privacy:*** the session key used for encryption/decryption of the message is computationally secure against any coalition of participants of cardinality less than t, before getting the alarm message.
- ***Message Privacy:*** the message is computationally secure against any coalition of participants who are not friends of the sender in the F2F network.

2.4. Security analysis

Here we only present the main theorems providing the above requirements without proofs, the particular analysis can be found in [4].

Theorem 2.1. *If Alice uses a perfect t-threshold secret sharing scheme in the Uploading then the system fulfills the Correctness requirement.*

Theorem 2.2. *If Alice uses a perfect t-threshold secret sharing scheme in the Uploading and a computationally secure encryption scheme, then the system fulfills the Key Privacy requirement.*

Theorem 2.3. *If Alice use a computationally secure encryption scheme, then the system fulfills the Message Privacy requirement.*

From the implementation point of view, it is enough to use a perfect secret sharing scheme, like Shamir's scheme 1.1 and computationally secure encryption in the implementation of the used F2F network.

3. Implementation details

The created application – called *Siren* – realizes the requirements described above: the users can send and receive encrypted information (location, special message, etc.) to and from their friends and it can be restored by them if and only if the communication breaks unexpectedly or there is an alarm message. In the case of emergency (i.e. some friend sent or triggered an alarm message) the application will automatically put out the known shared pieces to the peer-to-peer network and will try to gather enough piece to restore the key for the encrypted information. The decrypted information will be shown to the user as a pop-up warning message on the device.

3.1. Structure

Each instance of the Siren application is made up of a Signaling, a Processor, a F2F network layer and a P2P network layer module. The modules are responsible for different aspects of the program. The Signaling and Processor modules implement the core protocol described above in 2.2. The F2F and P2P layers provides a simple socket based interface to the underlying friend-to-friend and peer-to-peer network.

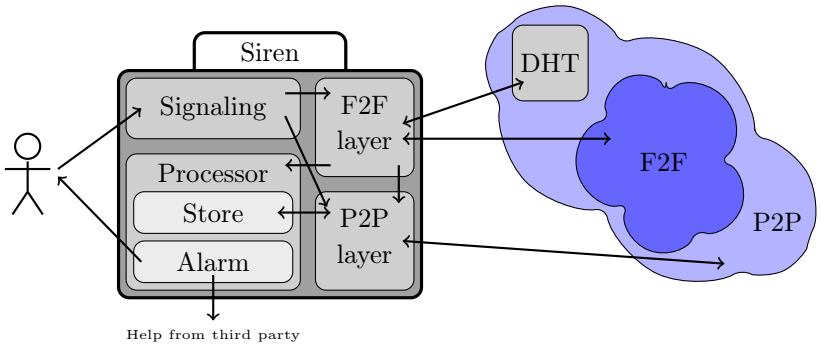


Figure 1: Modules of the Siren application

3.1.1. Signaling

The Signaling module interacts with the user through a graphical interface, makes and distributes heartbeat messages as specified in the Uploading phase or sends a panic signal if requested. After starting the module it keeps making and distributing heartbeat messages with a given frequency until the user stops the module or it becomes impossible to send any message. When the user stops the module, it will send a special closing message to the friends (not an alarm), hence they will don't start the Upload phase.

The heartbeat messages contain shared pieces of an unique key – which was used to encrypt information about the user – as well as some parameters about when and how the alarm message will be sent or should be triggered. Among others every message contains an expiration time and a deadline too. After the expiration time the receiver's Processor module will drop the message; while after the deadline an alarm will be triggered unless another newer message arrives from the sender.

3.1.2. Processor

The Processor module has two important tasks. The first task is to store the messages sent by friends from both F2F and P2P networks. It will organize and keep the messages until they are expired (the expiration time sent within the heartbeat message) and can announce them on the peer-to-peer network.

The other task of this module is to listen to alarm messages. If a friend has sent or triggered such message then it will gather the shared data from the P2P network, decrypt the information after recovering its key as described in the Message recovery phase in 2.2 and show it to the user. Because there can be more non-expired message from the same sender, the module will simultaneously try to gather all the required number of pieces for every message. Hence if there is a chance to recover a message that is not expired then it will be.

3.1.3. F2F Layer

The F2F layer abstracts away the F2F network, providing a simple socket interface where the individual nodes can be addressed by their identifiers calculated from their public keys.

To build up communication between two F2F nodes, their network addresses has to be resolved first. In our case, the node addresses are resolved used the BitTorrent DHT network. The network address for each node consists of an (IP address, port) pair. These pairs are identified by the SHA-256 hash of the node's public key. The F2F layer maintains a cache of these (id,IP,port) triples to speed up connection requests. Between two nodes, if only one node's network address changes, this node can notify the other one of its new address. The DHT only needs to be checked if the node identifier is not found in the cache or if both node's addresses change simultaneously. The nodes must however continuously keep advertising their network addresses through the DHT network, because the DHT has a tendency to "forget" information as old nodes leave and new ones enter the cloud.

Once the network address for a node is known, an encrypted channel can be established using the public key for the node. Through this channel, the nodes negotiate an ephemeral Diffie-Hellman key, which they use to encrypt their further messages. This provides better performance then using the asymmetric encryption and also achieves forward security.

Having a fully functional P2P layer, one can calculate and advertise F2F node identifier. The identifier of a F2F node is the SHA-256 hash of the node's public key. The calculation of these keys are done in two step. Firstly at the first start of the application it generates a new RSA keypair to sign messages and identify the host[1] and send/receive invites. When a friend's identifier known it can check out his address from the DHT and start the Diffie–Hellman key exchange. At the last phase of the key exchange it receives the public key of the friend with some data (like name, etc.) and calculates the common key for the communication channel.

Separating this functionality from the main application promotes modularity and encourages reuse. We hope that our implementation of the F2F layer will provide a guideline for developing similar cryptographic software.

[1]It uses this hash calculated from the public key as identifier in the distributed hash table too.

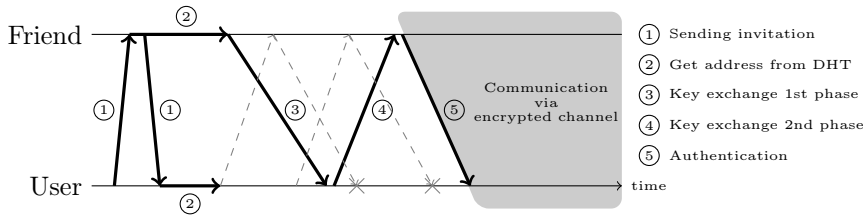


Figure 2: The process of establishing communication channel

3.1.4. P2P Layer

Bootstrapping the P2P layer. As seen in figure 1, the modules are highly interdependent, with the P2P layer being the most fundamental part of the application. The P2P layer provides a reliable address resolution mechanism for the F2F network. Because of this, the application needs to bootstrap the P2P network first to be able to use the F2F network. Bootstrapping from a handful of nodes can take some time to finish (it usually takes a few minutes), however in most cases we only need to find a few nodes only, because the addresses of the nodes are cached and stored between sessions. The cache contains a selected subset of the P2P nodes. The node selection based on their uptime and node identifier.

The nodes' uptime is usually thought of as a decreasing failure rate system (see [2]) in which the nodes with a longer uptime have a higher probability of being available for some fixed amount of time from now. If the cache is empty, we bootstrap the DHT from the following "master" nodes:

- `udp://router.bittorrent.com:6881`
- `udp://router.utorrent.com:6881`
- `udp://dht.transmissionbt.com:6881`

Communication via P2P. This implementation uses the peer-to-peer network for simple data announcing too. The encrypted data – which key was shared via the F2F network – is sent to the P2P network as well as all of the key pieces when a friend is in an emergency.

3.2. Key management

Our keypair is generated on the first run of the application and stored on the local disk for further use. The program manages a contact list of friends in the F2F network, which contains the hash of partners' public keys so that their network addresses can be resolved. The actual public keys are not stored, but they can be retrieved from an alive node after the Diffie-Hellman key exchange. One can add new friends to the contact list by sending a special invitation message which can take the form of an URL or a QR code. The invitation contains the hash of the

public key and an initial network address. Both of them are signed with the private key. This prevents forgeries and guarantees the authenticity of the invitation.

3.2.1. User interface

The application was made for Android operation systems so it can be used on the majority of mobile devices. The user interface can be separated into three parts: main, friends and settings.

- In the main part of the UI the user can start/stop the signaling module and can see some numerical information how: many friend added to his network, how many is active and how many turned on the signaling module. Also the other interfaces can be reached from here.
- The friends UI was made to handle friends so here the user can add, delete, invite or disable[2] friends. Reached this interface the user will see a list which items contain friends and their connection information like invitation states (sent/received) and that the friend is active or sending heartbeat messages.
- With the settings UI the user can set the parameters of Signaling module, the F2F and P2P networks and also can change the information of the encrypted message that the others will known in case of emergency.

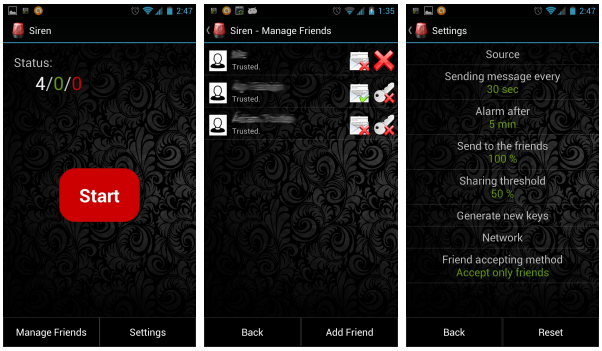


Figure 3: The three main graphical interfaces of the application

Acknowledgements. The research was carried out as part of the EITKIC_12-1-2012-0001 project, which is supported by the Hungarian Government, managed by the National Development Agency, financed by the Research and Technology Innovation Fund and was performed in cooperation with the EIT ICT Labs Budapest Associate Partner Group. (`www.ictlabs.elte.hu`). This research has been partially supported by the Lendület program of the Hungarian Academy of Sciences. The second author was partially supported by the grant OTKA PD-100712.

[2]Disable friends means they won't participant in the user's secret sharing but the application will receive messages from them and will warning the user if they are in emergency.

References

[1] Blakley, G. R., Safeguarding cryptographic keys *Proceedings of the National Computer Conference* Vol. 48 (1979) pp. 313–317.

[2] Carrda, D., Building a Reliable P2P System Out of Unreliable P2P Clients: The Case of KAD (2007) `http://www.eurecom.fr/fr/publication/2430/download/ce-carrda-071210.pdf`

[3] Isdal, T., Piatek, M., Krishnamurthy, A., Anderson, T., Privacy-preserving P2P data sharing with OneSwarm `http://www.oneswarm.org/f2f_tr.pdf`

[4] Kasza, P., Nagy, Á., Ligeti, P., Siren: secure data-sharing over P2P and F2F networks *submitted to Studia Scientiarum Mathematicarum Hungarica*

[5] Popescu, B.C., Crispo, B., Tanenbaum, A. S., Safe and Private Data Sharing with Turtle: Friends Team-Up and Beat the System *12th International Workshop on Security Protocols* (2004).

[6] RetroShare: secure communications with friends, available online at `http://retroshare.sourceforge.net/`

[7] Shamir, A., How to share a secret *Communications of the ACM* Vol. 22 (1) (1979) pp. 612–613.

Annales Mathematicae et Informaticae
44 (2015) pp. 121–130
http://ami.ektf.hu

A special localization algorithm in Wireless sensor networks for telemetry application

Gyöngyi Kocsisné Szilágyi[a], **Attila Kocsis**[b]

[a]Department of Programming Languages and Compilers
Eötvös Loránd University, Budapest
szilagyi@inf.elte.hu

[b]bI-QRS International Ltd., Budapest
atkocsis@iqrs.hu

Submitted July 18, 2014 — Accepted March 5, 2015

Abstract

Accurate positioning [1] in Wireless sensor networks [3] is an important and emerging technology for many research areas, such as health care, telemedicine [2], sports, commercial, public-safety and military applications. This paper presents an exact positioning approach for a telemetry system in ad hoc sensor network measuring patients' vital and motion parameters. The telemetry system includes two kinds of wireless data-gathering devices (sensors) neither of which knowing their own positions: 1. reference point devices, which retain fix positions, and are uniformly distributed over an area of interest, 2. all the other sensors (mobile nodes) which are allowed to change their positions during the measurements. The presented algorithm is based on measurements of distances between sensor nodes with 4 Hz sampling frequency. The aim is to compute the exact positions of the nodes in some fixed coordinate system. The measurements of distances between sensor nodes are not sufficiently accurate. The developed algorithm at first estimates the exact position of the reference points from initial measuremets based on the minimization of the localization error. During the measurements the positions of the reference points do not change. The mobile nodes localize themselves continuously with the help of location references received from the reference points using trilateration.

Keywords: Wireless Sensor Networks, Localization algorithms, Telemetry system

MSC: AMS classification numbers

1. Introduction

A wireless sensor network [3] consist of spatially distributed autonomous sensor nodes for data acquisition. Each node is able to sense the environment, perform simple computations and communicate with its other sensors or with the central unit.

Accurate localization of sensor nodes is a strong requirement in a wide area of applications such as health care, telemedicine, sports, rescue, disaster relief, commercial and military and applications [2].

This paper describes an algorithm for localization of sensor nodes using range measurements, trilateration and heuristic geometrical approach.

The paper is organized as follows. Section 2 presents the basic problem, the formulation of localization problem in wireless sensor networks, and the applied exact positioning approach for the I-QRS Telemetry system. In section 3 the developed localization algorithm is presented. In section 4 the applied heuristic and the implementation results are described. Future and Related work has been discussed in section 5 and 6, respectively.

2. The basic problem

2.1. Wireless sensor networks (WSN)

A wireless sensor network (WSN) [3] has important applications such as remote environmental monitoring, health monitoring and target tracking. The type of wireless sensor network investigated here refers to a group of sensors, or nodes, that are equipped with wireless interfaces through which they can communicate with one another to form a network. Each node is able to sense the environment, monitor physical or environmental conditions (such as temperature, sound, pressure, etc). The sensors perform simple computations, and cooperatively pass their data through the network to a main location. The design of a WSN depends significantly on the application, and can vary from a simple star network to an advanced multi-hop wireless mesh network.

2.2. Localization algorithms

The theoretical background of localization is presented briefly in this section. Please find more details on the topic in references [1, 7]. The goal of localization is to determine the physical coordinates of a group of sensor nodes. Many methods have been proposed in the literature and used in practice to localize wireless devices.

Localization algorithms can be classified into *exact* and *approximative* localization.

Exact localization is based on precise measurements of distances or angles between sensor nodes not knowing their own position and nodes with preinstalled

localization systems. These methods result high precision of position determination but need extensive calculations.

Approximative algorithms do not require extensive calculations and result in less network traffic. From network managemenet aspect the research can be classified into two categories: *centralized* and *distributed* localization.

In the *centralized* localization it is not necessary to compute each node. Sensor nodes gather environmental data and pass it to a base station. After analysis the computed positions are transported back into the network. Disadvantage of this method is high communication costs.

In the case of *distributed* localization algorithms all computations are done on the nodes. The sensor nodes communicate with each other to get their positions in a network.

In the network model, the nodes are located in distinct physical locations in some region of space. We assume below that nodes have some means by which they can measure the distance between themselves.

A wireless ad-hoc network can be modeled by a distance graph $G = (V, E, D)$, where V is the set of wireless nodes, E is the set of links, and $D(u, v)$ denotes the distance measurements between a pair of nodes u and v. An important question is whether or not a network is localizable by given its distance graph.

Let N be a network in R^2 with nodes labeled $0, 1, ..., n$, and assume G has a bilateration ordering. A sensor network with a total number of n nodes consists of k sensor nodes and b beacons (anchors) ($b << k$), where **beacons** are able to determine their own position, and the positions of **sensor** nodes are not known. Note that the distance between any two anchors is known since the positions of all of the anchors are known. The positioning error of these localization systems depends on the quality of the used localization devices.

2.3. The applied exact positioning approach

This paper presents an *exact centralized positioning approach.* The presented algorithm is based on measurements of distances between sensor nodes with 4 Hz sampling frequency. None of the nodes knows their own position, so there are not beacons, but there are two kinds of sensor nodes: **reference points** (fix) and **mobile nodes** (changing its position continuously). The **reference points** do not know their own position, and the measurements of distances between sensor nodes are not sufficiently accurate, but during the measurement their positions are fixed. The first step is to compute the exact positions of the reference nodes in some fixed coordinate system. After getting the positions of the reference points, they are fixed during the measurements, and the **mobile** nodes localize themselves continuously with the help of distance measurements received from the reference points. The aim is to compute the exact positions of the reference points and mobile nodes in some fixed coordinate system.

2.4. The I-QRS Telemetry System

The IQRS Sport Telemetry System [10] monitors the vital and motion signs of the player continuously. It records and transmits data in real-time by wearing a chest belt. The local computer receives the signals, uses automatic data-analysis and alarm algorithms, and transfers the signals through a data channel (mobile phone, broadband internet, or other similar connections) to an internet connected server, where an expert system further processes the signals. The trainers and doctors may use a web-based system to access the data. *The motion analysis and animation module of the system is based on the positioning algorithm presented in this paper.* With the help of the animation module the movement of players and their techniques can be monitored and also played back in 3D format after the trainings. In the system the number of reference points is smaller then 10, and the number of mobile nodes is bellow 100, and the size of the sport field is about 250 meter.

3. The developed exact positioning algorithm

The developed positioning algorithm is optimized for the I-QRS Telemetry system. The aim was to develop a sufficient algorithm for each step of the telemetry system's process, starting from the setting up of the device, during the whole measurements until the analysis ends. Our aim was not to develop a new optimal algorithm for large networks, but to analyze the existing techniques, combine and modify them, and develop a new heuristic according to the needs of the telemetry system for daily use. The developed algorithm consists of two main steps.

Algorithm I. : Localisation of the reference points

The developed algorithm at first estimates the exact position of the reference points from initial measurements based on the minimization of the localization error. During the measurements the positions of the reference points do not change.

Algorithm II. : Localisation of the mobile nodes

The mobile nodes localize themselves continuously with the help of location references received from the reference points using trilateration.

3.1. Algorithm I.: Localisation of the reference points

This iterated algorithm can be used to provide the locations of the reference points. In each iteration an initial set of two nodes is fixed and used to define a coordinate system, and each node uses distance estimates to each other to solve a set of circle-circle intersection problems, solved through a coordinate geometric formulation. The computed possible positions of the reference nodes form polygons.

The resulting polygons of the iterations are transformed to a common coordinate system, where the exact positions of the nodes are computed with respect to the minimization of the localisation error. The examples are presented for 8 reference point, since the I-QRS Telemetry system uses 8 reference nodes.

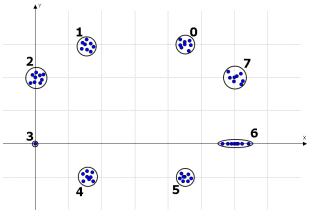


Figure 1: Octagons

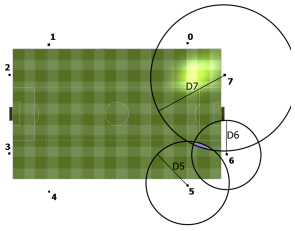


Figure 2: Trilateration

Input: *Measurements of distances between reference nodes* $(Ref_1, \ldots, Ref_n)$ *with 4 Hz sampling frequency*

Output: *The exact positions of the reference points in a fixed coordinate system*

3.1.1. Step 1. Initialization

- **collect** a few thousand distance measurements:

$$distances[Ref_i, Ref_j](0 \leq i, j \leq n)$$

- **for** all pair of the reference points (Ref_i, Ref_j) **do average** the values, and **filter** the outliers
- the **output** is a distance matrix D, where $D(Ref_i, Ref_j)$ denotes the distance between reference points Ref_i and Ref_j

3.1.2. Step 2. Computing the polygons

for all pair of the reference points (Ref_i, Ref_j) **do**

- Estimate the coordinates of the other nodes ($Ref_k : 0 \leq k \leq n, k \neq i, j$) using coordinate geometry of circles, solving the circle-circle intersection problem

- Two circles may intersect in two imaginary points (throw away), a single degenerate point (good), or two distinct points ($k1(x1, y1)$ and $k2(x2, y2)$- the choice is based on the minimization of the localization error)

$$u = d(i,k)^2 - d(j,k)^2 + x_i^2 + x_j^2 - 2x_i x_j - y_i^2 + y_j^2; v = x_j - x_i; w = y_i - y_j$$

$$k_{y1/2} = \frac{-4(uw - 2v^2 y_i) \pm \sqrt{4(uw - 2v^2 y_i)^2))}}{2 * 4(w^2 + v^2)} -$$

$$\frac{-4(uw - 2v^2 y_i) \pm \sqrt{-4 * 4(w^2 + v^2)(u^2 - 4v^2(d(i,k) - y_i^2))}}{2 * 4(w^2 + v^2)}$$

$$k_{x1/2} = \frac{x_j^2 - x_i^2 - ((k_{y1/2} - y_i)^2 - d(i,k)^2) + ((k_{y1/2} - y_i)^2 - d(j,k)^2)}{2v}$$

For every iteration the result is a polygon. The number of iteration step is $n * (n-1)/2$.

3.1.3. Step 3. Transformation of the polygons in a common coordinate system

The identification number of the reference points, and the polygons angels and side lengths are known, so they can be transformed to a common coordinate system (Ref_i is in the **origo** and the coordinate of Ref_j is $(D(Ref_i, Ref_j), 0)$) for some fixed i and j. The output is a set of polygons, and a set of computed possible coordinate for each reference point (Figure 1).

3.1.4. Step 4. Find the exact positions

The aim is to find the exact position for every reference point. The goal is to minimize the sum of squares of the errors between the real positions of the references and their computed possible positions with respect to the measured distances ($D(Ref_i, Ref_j)$) and distances computed from the possible positions ($\bar{D}(Ref_i, Ref_j)$).

$$min \sum_{i,j,i<j} (\bar{D}(Ref_i, Ref_j) - D(Ref_i, Ref_j))$$

Since the search space is too large we can calculate the average of the possible positions or apply beam search or other searching techniques to solve this problem. In the implementation we have used a special heuristic described in Section 4.

3.2. Algorithm II.: Localisation of the mobile nodes

The mobile nodes localise themselves continuously with the help of distance measurements received from the reference points using trilateration (Figure 2.). The trilateration algorithm is applied for every set of three reference nodes in Step 1, the result of this step is a set of possible mobile node positions. In Step 2 the optimal positon is computed based on the set of possible mobile nodes positions. The **input** of the algorithm is set of distance measurements between the mobile node and reference points, and the **output** is the exact position of the mobile node.

3.2.1. Step 1. Trilateration for every set of three reference nodes

Denote $P(x, y)$ the coordinate of the mobile node have to be positioned, , $r_\sharp$ the measured distances from the three actual reference nodes indexed by $i_\sharp$.

for each mobile nodes **do for** each set of three reference nodes **do**

- The trilateriation part (Figure 2): format the matrixes based on Pythagoras theorem

$$\begin{bmatrix} (x-x_{i1})^2+(y-y_{i1})^2 \\ (x-x_{i2})^2+(y-y_{i2})^2 \\ (x-x_{i3})^2+(y-y_{i3})^2 \end{bmatrix} = \begin{bmatrix} r_{i1}^2 \\ r_{i2}^2 \\ r_{i3}^2 \end{bmatrix}$$

- The least square algorithm: derive the matrix above to get the format: $(H * P(x, y) = Z)$. To do this subtract the first equations from the others. After some algebric manipulation we get the following:

$$\begin{bmatrix} 2x_1-2x_2 & 2y_1-2y_2 \\ 2x_1-2x_3 & 2y_1-2y_3 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} r_2^2-r_1^2+x_1^2-x_2^2+y_1^2-x_2^2 \\ r_3^2-r_1^2+x_1^2-x_3^2+y_1^2-y_3^2 \end{bmatrix}$$

This is a least square system $(H * P(x, y) = Z)$ which has the following solution for x and y

$$H^T * H * P = H^T * Z$$

$$(H^T * H)^{-1} * (H^T * H) * P = (H^T * H)^{-1} * H^T * Z$$

$$P = (H^T * H)^{-1} * H^T * Z$$

The output of this step is for each mobile node a set of possible position $P(x, y)$.

3.2.2. Step 2. Choose the optimal position

This step compute the optimal position for each mobile node from a set of computed positions of the mobile nodes.

1. filter the to the outliers from the set of computed positions $P(x, y)$
2. **for** each set of three reference nodes **do** compute the following
   - $err_i = |r_i^2 - ((x - x_i)^2 + (y - y_i)^2)|(i = 1, 2, 3)$

- $err = err_1 + err_2 + err_3$

3. The position having the minimal value of err is the **output**

3.3. Analysis and Optimization of the algorithm

The algorithm described above is a general solution. The exact identification of the class of network types that can be completely localized using this algorithm is a task of the future work. In the implementation a special heuristic is used during the computation of the polygons in order to reduce the integration errors. The implementation results show that the localizability depends on the quality of the distance measurements and on the shapes of the polygons formed by the reference points. Different optimization approaches can be applied to improve the accuracy of the positions.

- Iterated trilateration [4] can be used for three reference points instead of solving the circle-circle intersection problem for two reference nodes in each iteration
- The polygons can be weighted according to an error function
- Error model can be used which depends on the quality of the used localization devices (Temperature, Power, Clock skew, Moisture, Reflection, Constant error)

4. Implementation results

The localization algorithm is optimized for the I-QRS Telemetry system [10], in which the number of the reference points is smaller then 10, and the number of the mobile nodes is bellow 100. In the localisation of the reference nodes (Algorithm I.) a special heuristic is used. The polygons are created in a special order iteratively and they are weighted according to the iteration number. For all reference point triplets a triangle is created, and the triangle with the largest area is choosed in the first iteration to create the polygon (solving the appropiate circle-circle intersection problems). In the second iteration step in similar way the triangles having common side with the first triangle are chosen to create the polygons. This method is done for three iteration (Figure 4). After transforming and rotating the polygons in a common coordinate system, the final positions of the reference points are calculated with respect to the weights computed in the different iteration steps. We first evaluated the performance of the algorithms with generated precise distance measurements. The result was very good, the accuracy is lower than 0,001 meter.

The distance measurements of the I-QRS telemetry system in everyday use are often very noisy (the average accuracy is 3 meter), and on many sport fields the reflexion (the measured distance is about double of the real distance) is very high.

The test results of the algorithm show that the accuracy of the reference points positioning algorithm is about 0,3 meter, and of the mobiles nodes calculation algorithm is about 1,2 meter. To improve further the quality of the system, the calculated positions are filtered using different kinds of filtering methods.

5. Future work

An exact centralized positioning approach was presented in order to determine positions on the basis of distance measurements. An important goal of the further work is to improve the accuracy of the positioning algorithm. Two promising approaches are the application of quaternion based algorithm presented in [5], and using Klaman filters [6] based on the information of other sensors (gyroscope, accelemrometer, magnetometer). Figure 3 shows the work under development.

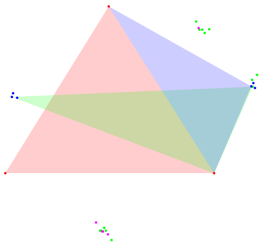


Figure 3: Three iteration

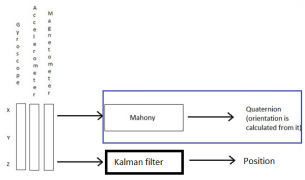


Figure 4: Future work

6. Related Work

Localization in WSN is an active area of research and so there are some existing literature surveys [1,7] on this topic. A promising class of approach for precise localization is fine-grained localization [8].

This paper presents an exact centralized positioning approach. The idea of the presented algorithm is related to simple iterated trilateration and the so called Sweeps algorithm [9]. In iterated trilateration, an initial set of three nodes is fixed

and used to define a coordinate system. At each stage of the algorithm, there is a set of localized nodes and a set of unlocalized nodes. If an unlocalized node has distance measurements to at least three localized nodes, its position will be alculated and it will be added to the set of localized nodes. Simple iterated trilateration is suboptimal in that there are many localizable networks which it cannot localize. In [9] a class of algorithms is described for fine-grained localization called Sweeps. Sweeps correctly finitely localizes all nodes in bilateration networks. Sweeps also handles angle measurements and noisy measurements. The algorithm presented in this paper is based range measurements, trilateration and heuristic geometrical approach. Some step of the developed algorithm shows similarities to the mentioned approaches above, but our algorithm covers each step of the telemetry system's process, starting from the setting up of the device, during the whole measurements until the analysis ends. We developed a new heuristic optimized to the needs of the telemetry system for daily use.

References

[1] Amitangshu, P., Localization Algorithms in Wireless Sensor Networks: Current Approaches and Future Challenges, *Network Protocols and Algorithms*, Vol. 2 (2010), No. 1., ISSN 1943–3581.

[2] Rashid, L. B., On the Definition and Evaluation of Telemedicine, *Telemedicine Journal*, Vol. 1(1) (1995), 19–30.

[3] Jennifer, Y., Biswanath, M., Dipak G., Wireless sensor network survey, *Computer Networks*, Vol. 52 (2008), Issue 12, 2292-–2330.

[4] Zheng, Y., Yunhao L., Confidence-Based Iterative Localization. Parallel Distrib. Syst., IEEE Transactions 21(5) (2010), 631–640.

[5] MAHONY, R., HAMEL, T., Pflimlin, Jean-Michel, Nonlinear Complementary Filters on the Special Orthogonal Group, Automatic Control, IEEE Transactions, Vol. 53 (2008), Issue 5, 1203-–1218.

[6] Chen H., A robust location algorithm with biased extended Kalman filtering of TDOA data for wireless sensor networks, *International Conference on Wireless Communications, Networking and Mobile Computing, Proceedings*, Vol. 2 (2005), 883—886.

[7] J., Bachrach, C., Taylor, Localization in Sensor Networks, *Handbook of Sensor Networks: Algorithms and Architectures*, (I. Stojmenovic, Ed.), (2005)

[8] J., Albowicz, A., Chen, L., Zhang, Recursive position estimation in sensor networks. *In Proceedings of the 9th International Conference on Network Protocols* (2001.), 35-–41.

[9] David, K., Goldenberg et. all, Localization in Sparse Networks using Sweeps, MobiCom '06, *Proceedings of the 12th annual international conference on Mobile Computing and Networking*, 110–121.

[10] `http://www.iqrs.hu/`

Annales Mathematicae et Informaticae
44 (2015) pp. 131–140
http://ami.ektf.hu

A local PageRank algorithm for evaluating the importance of scientific articles*

András London†‡, Tamás Németh
András Pluhár, Tibor Csendes

Institute of Informatics, University of Szeged, Hungary
london@inf.u-szeged.hu

Submitted July 15, 2014 — Accepted March 5, 2015

Abstract

We define a modified PageRank algorithm and the PR-score to measure the influence of a single article by using its local co-citation network. We also calculate the reaching probability and RP-score of a paper starting at an arbitrary article of its co-citation network for the same purpose. We highlight the advantages of our methods by applying them on the celebrated paper of Jenő Egerváry that is underrated by the standard indices.

Keywords: Scientometric, PageRank, Ranking algorithms, Co-citation networks

1. Introduction

The relevance of scientometrics – aiming at measuring the productivity and quality of scientific research – has long been widely discussed in the academic domain. Among the most popular measures used are scientific citation indices due to their easy accessibility. Several of these indices have been introduced such as h-index (or Hirsch-index) proposed by Hirsch [14], the g-index proposed by Egghe [11],

*This work was partially supported by the European Union and the European Social Fund through project FuturICT.hu (grant no.: TAMOP-4.2.2.C-11/1/KONV-2012-0013).

†The first author was supported by the *European Union* and the *State of Hungary, co-financed by the European Social Fund* in the framework of TAMOP-4.2.4.A/2-11-1-2012-0001 'National Excellence Program'.

‡Corresponding author: london@inf.u-szeged.hu

the w-index and maximum-index both proposed by Woeginger [27]. All of these indices are based on the citation records of the researchers. These indices have been extensively criticized since they are much dependent on the scientific field (e.g. number of researchers and available journals, popularity of the area, gender ratio, etc., see e.g. [1, 19, 26]). Another drawback is that the number of citations does not give a clear picture on the influence and quality of a single paper.

Several studies have been addressed this problem using network approach. Co-citation networks, in which nodes represent single articles and a directed edge represents a citation from a citing article to a cited article, describes the relation between citations of different papers have been widely studied previously[6, 15, 20]. Chen et al. [7] applied the PageRank algorithm [5] (developed by the founders of Google) for co-citation networks, later Raddichi et al. [23] defined an iterative ranking method analogous to different ranking algorithms such as PageRank, CiteRank [25] and HITS [16] in order to evaluate the influence of single articles by using co-authorship networks (where nodes represent publications and weighted edges represent the number of common authors of them). Several modifications and variants of the network models have been designed in the context of scientometrics (see e.g. [12, 21, 24, 28]).

More recently, the Eigenfactor Score and the Article Influence Score [4] have been developed to estimate the relative influence of single articles based on citation networks as well. Furthermore the underlying algorithms can also be applied to journals, authors, and institutions.

Following the network approach, our main goal is to measure the influence of a single article regardless of the specialties of the field. Based on the previous results of Csendes and Antal [9] and by applying the experimental results of Chen et al. [8] that is later mathematically proved to be efficiently applicable for many classes of graphs by Bar-Yossef and Mashiach [2], we use a local PageRank estimating method for this purpose. It is important to note that we do not want to attempt to determine the scientific value of the articles (which will be probably judged in the future).

This article is organized as follows: in Section 2 we give a brief mathematical overview of the PageRank algorithm. In Section 3, we describe how a local PageRank method can be applied to determine the scientific influence of a research paper. Finally in Section 4 we compute the local PageRank values of the articles in the co-citation graph of the famous paper of Jenő Egerváry [10] and highlight the main advantages of our approach from the scientometric point of view.

2. Methods

In this section, we give a short mathematical overview of the PageRank algorithm. We describe the main notions, definitions, and theoretical results of a local PageRank method that we used. We omit the proofs of the theorems that can be found in [2].

2.1. Overview of the PageRank method

The PageRank algorithm was originally designed to provide a good approximation of the importance of web pages. Since it works on directed graphs, it is a natural idea to use the PageRank method for ranking the articles in co-citation graphs.

Let $G = (V, E)$ be a directed graph of N nodes. Let $d^-(i)$ $(i = 1, 2, \ldots, N)$ be the number of outgoing edges from a node i and $N^+(i) = \{j \in V : j \to i \text{ exists}\}$, i.e. the set of nodes having an edge to node i. PageRank of a node $i \in V$ is defined then by the following recursion formula [5]:

$$PR(i) = \frac{\lambda}{N} + (1 - \lambda) \sum_{j \in N^+(i)} \frac{PR(j)}{d^-(j)}, \tag{2.1}$$

where $\lambda \in [0, 1]$ is a free parameter (usually set between 0.1 and 0.2).

The PageRank formula defined by equation (2.1) can be written in vector equation form, and then the PageRank vector $\mathbf{PR}$ is defined as

$$\mathbf{PR} = \frac{\lambda}{N}[I - (1 - \lambda)AD^{-1}]^{-1}\mathbb{1}, \tag{2.2}$$

where A is the adjacency matrix of G, D is a diagonal matrix such that $D_{ii} = \sum_{\ell=1}^{N} A_{i\ell}$ and $D_{ij} = 0$, if $i \neq j$, I is the $N \times N$ identity matrix and finally $\mathbb{1}$ is the N-dimensional vector having each component equals to 1.

Assuming that $\mathbb{1}\mathbf{PR} = 1$, Eq. (2.2) implies, that

$$\mathbf{PR} = [\frac{\lambda}{N}\mathbb{1}\mathbb{1}^T - (1 - \lambda)AD^{-1}]\mathbf{PR}, \tag{2.3}$$

which shows, that $\mathbf{PR}$ is the eigenvector of the matrix $\frac{\lambda}{N}\mathbb{1}\mathbb{1}^T - (1-\lambda)AD^{-1}$ due to the fact that an eigenvalue equals to 1, which is the largest eigenvalue of this matrix by a consequence of the Frobenius-Perron theorem for row-stochastic matrices [22].

More intuitively, let us consider a random walk on the nodes of the graph. Starting from a node i, a random surfer selects one of the node's outgoing edges randomly with uniform distribution, moves to the end node j of that edge, and repeat this process from j, etc. The parameter λ can be understood as a "damping" factor which guarantees that the random walk restarts in a random node of the graph, chosen uniformly random, almost surely in every $1/\lambda$-th step. This can guarantee, that the process would not stop by reaching a node with an out-degree zero. If the surfer reaches a node, the number of visits of that node increases by one. The damping factor ensures that each node receives a contribution λ/N at each step. Thus, the PageRank of a node i can be considered as the long-term fraction of time spent in node i during the random walk. The steady-state of the random walk is given by the solution of Eq. (2.3).

2.2. Local PageRank approximation

Although in many applications PageRank scores are needed to be computed for all nodes of the graph, there are situations in which one is interested in computing

PageRank scores only for a small subset of the nodes. Chen et al. [8] developed an algorithm to approximate the PageRank score of a target node of the graph with high precision. Their algorithm crawls backwards a small subgraph around the target node(s) and applies various heuristics to calculate the PageRank scores of the nodes at the boundary of this subgraph and then computes the PageRank of the target node(s) by using only the crawled subgraph. By using simulations, they showed that this algorithm gives a good approximation on average. On the other hand, they also pointed out that high in-degree nodes could make the algorithm very expensive and incorrect.

From now in this section, we use the same notions as in [2]. An algorithm is said to be an ϵ-*approximation* of the PageRank, if for a graph $G = (V, E)$, a target node $i \in V$ and a given error parameter $\epsilon > 0$, the algorithm outputs a value $PR'(i)$ satisfying

$$(1 - \epsilon)PR_G(i) \leq PR'(i) \leq (1 + \epsilon)PR_G(i). \tag{2.4}$$

For a directed path $p = (k_1, \ldots, k_t)$ from node k_1 to k_t, let $w(p) = \prod_{i=1}^{t-1} \frac{1}{d^-(k_i)}$, that is the reaching probability of k_t from k_1 in a given path, where the transition probabilities are proportional to the number of outgoing edges. Let $p_t(i, j)$ be the set of all directed path of length t from i to j. Then, the *influence* of node i on the PageRank of node j at radius t is defined as

$$I_t(i, j) = \sum_{p \in p_t(i,j)} w(p), \tag{2.5}$$

and thus, the total influence of i on j is

$$I(i, j) = \sum_{t=0}^{\infty} I_t(i, j). \tag{2.6}$$

By using the definition of influence, PageRank of node j at radius r can be defined as

$$PR_G^r(j) = \frac{\lambda}{N} \sum_{t=0}^{r} \sum_{i \in V(G)} (1 - \lambda)^t I_t(i, j). \tag{2.7}$$

It can be proved that for every node $j \in G$, $PR_G(j) = \lim_{r \to \infty} PR_G^r(j)$ holds (the proof can be found e.g. in [2]). The interesting question is that how small the radius r can be such that the PageRank approximation would even be appropriate.

In [2] it was proved, that the hardness and inappropriate nature of local approximation of PageRank on certain graphs (constructed examples) is caused by two factors: the existence of high in-degree nodes and the slow convergence of PageRank iteration algorithm. We shall see, that in our case (and in most of the co-citation graphs in scientometrics) these properties does not hold.

It was also shown, that the several variants of the approximation algorithms proposed by Chen et al. are still efficient on graphs having bounded in-degrees and admitting fast PageRank convergence.

Let us be given a $G = (V, E)$ graph, node $j \in V$ and the approximation parameter ϵ. The *point-wise influence mixing time* of j is defined as

$$T_G^{\epsilon}(j) = \min\{r \geq 0 : \frac{PR_G(j) - PR_G^r(j)}{PR_G(j)} < \epsilon\}. \tag{2.8}$$

The algorithm we use computes $PR_G^r(j)$ for a given node j (see in Section 4) and it follows from the definitions that it runs with $r = T_G^{\epsilon}(j)$ and gives an ϵ-approximation of PR. To complete the description of the theoretical background, we should see the upper bound on $T_G^{\epsilon}(j)$ (or radius r).

For graph $G = (V, E)$ with $j \in G$ and $r \geq 0$ the *crawl size* at radius r is defined as

$$C_G^r(u) = \#\{i \in G : \exists p_t(i, j) \text{ with } t \leq r\}. \tag{2.9}$$

It is immediate from the definition, that if the local PageRank algorithm runs for r iteration, its cost is $C_G^r(u)$. A trivial upper bound for the crawl size is that $C_G^r(u) < d^r$, where d is the maximum in-degree of G.

Finally, it was also proved that for any G directed graph, node $j \in G$ and $\epsilon > 0$ it holds that a radius $r = \mathcal{O}(\log(1/PR_G(u)))$ is always sufficient (while in practice a much lower radius could be enough).

2.3. Reaching Probabilities

A possible simplification of the PageRank method is to consider only the reaching probabilities of the nodes in the network. We would like to know the probability of reaching a node j starting from an arbitrary chosen node i of the network. The reaching probability, RP of node j can be defined as

$$RP(j) = \sum_{i \in N^+(j)} p_{ij} RP(i), \tag{2.10}$$

where p_{ij} is the reaching probability of node j from a neighbor node i. It is natural to assume, that each possible selection of a neighbor of node i has equal probability, thus we can write $p_{ij} = 1/d^-(i)$ in Eq. (2.10). By this choice, Eq. (2.10) is the PageRank equation without the damping factor. However, in contrast to the calculation of PageRank, we do not want to evaluate the vector RP in the steady-state. Instead, we only determine the reaching probability of a given node j, which can be calculated as

$$RP(j) = \frac{1}{N} \sum_{i \in V} I(i, j), \tag{2.11}$$

where $I(i, j)$ is as defined in (2.6). In the point of view of published articles, RP can be interpreted as the probability of a given article can be found by someone (e.g. a scientist), who starts the search at any article and goes to another randomly chosen article cited by the current one.

3. Application to scientometrics

In the last decade, co-citation networks have been investigated aiming to measure the importance of a scientific article. A co-citation network is defined as a directed graph $G = (V, E)$ of N nodes, where each node $i \in V$ refers to an article and there is a directed edge $i \to j \in E$ from node i to node j if article j is cited in article i. Our method, that aims to measure the "influence" of a scientific article, is based on the following three phases, by applying some experimental results of [8]:

1. **Subgraph building**: Starting form certain target nodes (articles), for which we are interested in measuring their scientific impact, and expanding backward by following reversely the nodes having out-going links to the target nodes. The procedure stops after a fixed number of levels. This can be done by an iterative deepening depth-first search. In this work, the graphs contain all nodes, from which the target nodes can be reached in at most three steps and we consider the *induced subgraph* of that nodes.

2. **Estimating the PR of the boundary**: We use a heuristic to estimate the individual PR: in each iteration turn, we add an extra term to the PR value of each boundary node that equals to the fraction of its in-coming edges to all edges in the subgraph.

3. **Calculating the PR and RP**: On one hand, we run the PageRank algorithm on the subgraph, in each step we use the estimated PR value of the boundary nodes adding the λ/N damping factor to each node. On the other hand, we also calculate the reaching probability, RP, of the target node(s) in the subgraph.

The idea behind the necessity of the second phase is that, although the PageRank values cannot be calculated exactly without having run the algorithm on the full graph, still the estimation heuristic we defined gives an acceptable approximation for the constructed subgraph as it has been already proven in [2], and tested by simulations [8]. We also note that the convergence of the PageRank is guaranteed by this method opposite to that one defined by Csendes and Antal for the same

Algorithm 1: Local PageRank method for a scientific article

Input : Scientific article ID A.
Output: The PR-score of the article from its local co-citation network,
1 Build the article's local co-citation network with radius r
2 Fix the PageRank values of each boundary node v as $PR(v) = |N^+(v)|/|E(G)|$
3 Calculate PR-scores of each node in the subgraph by using the PageRank algorithm
4 Return $PR(A)$.

purpose. We set the radius size $r = 3$ from the target nodes because of two reasons: the first is that the number of nodes in the fourth layer is $\mathcal{O}(N)$ and the in-degrees are bounded with a constant, thus, with respect to PageRank algorithm, it is enough to consider the number of in-coming links to the boundary nodes from this layer, and not to consider the linking structure between them to get a good approximation of the PR-scores. The second reason is that we assume, that the articles at a distance more than three (with respect to the co-citation graph) do not have much impact on the target articles in scientific sense (which may be acceptable in scientometrics).

4. Results and discussion

As it is known, Harold Kuhn developed an algorithm for solving the assignment problem [18] and he named it as the Hungarian method acknowledging the contribution of Jenő Egerváry and Dénes Kőnig [10, 17]. The paper of Egerváry received just a few citations (probably because it was written in Hungarian) while some of the citing papers received much more: for Egerváry's paper 38 citations can be found in the ISI Web of Knowledge database, while the artice of Kőnig and Kuhn received there 215 and 726, respectively. In contrast to classic scientometrics that only takes into account the direct number of citations, we shall see that the network based methods show a more realistic picture of the importance of Egerváry's paper.

We constructed a network which contains the following articles as nodes: the famous paper of Jenő Egerváry: *On combinatorial properties of matrices* (published in Hungarian, 1931), the three articles which referred in Egerváry's paper, the articles that cite Egerváry's one, all articles that cite at least one of the previous ones and all articles that cite articles on the "second level". We consider the network that is induced by these nodes as described in the first phase; it contains $N = 1155$ nodes and 1923 edges. Figure 1 shows the network, where the paper of Egerváry highlighted with big black square.

We applied the modified PageRank algorithm (with $\lambda = 0.1, 0.15, 0.2, 0.25$) described in Section 3 for this network and also calculate the reaching probabilities of the nodes. We observed that the PageRank method is robust against the choice of λ. The results (with $\lambda = 0.2$) are summarized in Table 1 for four notable publications in the co-citation network.

Publication	PR-Score	PR-rank	RP-score	RP-rank	#Cites	Cite rank
Egervári [10]	0.891	4	0.009	2	39	65
Kuhn [18]	1.189	1	0.042	1	726	1
Ford, Fulkerson [13]	0.525	8	0.004	9	39	65
Bellman [3]	0.399	11	0.003	10	18	158

Table 1: PR-score (with $\lambda = 0.2$), reaching probabilies and number of citations of the famous publications in the Egerváry co-citation graph. PR-score is multiplied by 10^2

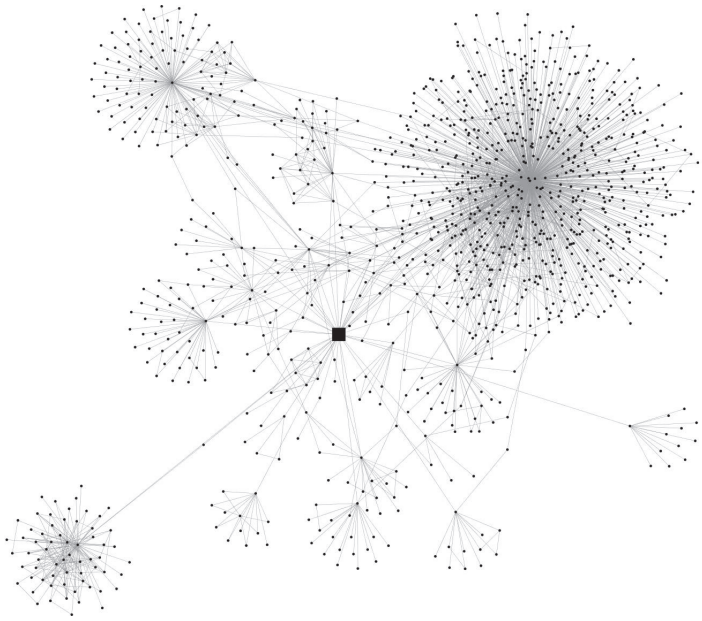


Figure 1: Local co-citation network containing the famous paper of Egerváry (highlighted with big square)

First, we observed that the choice of the damping factor λ does not influence the final ranking of the first ten publications, only small changes can be noticed in the rest of the ranking. The ranks and the relative values of the papers to each other show a more realistic picture of the importance of them. It is not surprising, that Kuhn's paper PR value is the highest by far, the 726 citations for this paper is outstanding in the field. The second and third articles in the PR rank became D. Kőnig: *Graphs and their applications for the theory of determinants and sets* (in Hungarian, 215 citations) and G. Frobenius : *Über zerlegbare Determinanten* (11 citation), respectively. Both articles were cited in Egerváry's paper which became the fourth highest ranked paper although it only received 39 citations and that it is only in the 65th place in the citation ranking. The very high position of Forbenius's paper in the ranking is definitely due the reputation it obtains from Egerváry's article. It is worth highlighting that Ford and Fulkerson's article, which received the same number of citations as that of Egerváry, was ranked lower but it is still in the top ten. This two facts also indicate the advantages of the PageRank based evaluation, since this paper was also quite important in the development of operation research. We also point out, that the similarly important paper of Bellman was ranked 11th (although it received just 18 citations) which shows a much clearer picture of its impact (in contrast to its citation rank). It is also interesting to observe, that the RP-rank of Egerváry's article is two, which means that a random searcher who checks the articles of the field finds that paper with

the second highest probability.

We hope that network-based ranking methods gain more space in scientometric since they show a more objective picture of the impact of scientific publications. It follows from the implementation of the PageRank algorithm that citations received from more important papers contribute more to the ranking of the cited paper than those coming from less important ones. Furthermore, simplicity and fast computability of this method are also advantageous. On the other hand, co-citation networks give a more detailed contextual information (compared to the number of citations) for evaluating the impact of an article.

Acknowledgment The authors are grateful to Elvira Antal for constructing the network and for her respective contribution.

References

[1] Alonso, S., Cabrerizo, F., Herrera-Viedma, E., and Herrera, F. H-index: A review focused in its variants, computation and standardization for different scientific fields. *Journal of Informetrics 3*, 4 (2009), 273–289.

[2] Bar-Yossef, Z., and Mashiach, L.-T. Local approximation of pagerank and reverse pagerank. In *Proceedings of the 17th ACM conference on Information and knowledge management* (2008), ACM, pp. 279–288.

[3] Bellman, R. Mathematical aspects of scheduling theory. *Journal of the Society for Industrial & Applied Mathematics 4*, 3 (1956), 168–205.

[4] Bergstrom, C., West, J., and Wiseman, M. The eigenfactor metrics. *The Journal of Neuroscience 28*, 45 (2008), 11433–11434.

[5] Brin, S., and Page, L. The anatomy of a large-scale hypertextual web search engine. *Computer networks and ISDN systems 30*, 1 (1998), 107–117.

[6] Chen, C. Visualising semantic spaces and author co-citation networks in digital libraries. *Information Processing & Management 35*, 3 (1999), 401–420.

[7] Chen, P., Xie, H., Maslov, S., and Redner, S. Finding scientific gems with google's pagerank algorithm. *Journal of Informetrics 1*, 1 (2007), 8–15.

[8] Chen, Y.-Y., Gan, Q., and Suel, T. Local methods for estimating pagerank values. In *Proceedings of the 13th ACM International conference on Information and knowledge management* (2004), pp. 381–389.

[9] Csendes, T., and Antal, E. Pagerank based network algorithms for weighted graphs with applications to wine tasting and scientometrics. In *Proceedings of the 8th International Conference on Applied Informatics* (2010), pp. 209–216.

[10] Egerváry, J. Mátrixok kombinatorikus tulajdonságairól (On combinatorial properties of matrices, in Hungarian)). *Matematikai és Fizikai Lapok 38* (1931), 16–28.

[11] Egghe, L. An improvement of the h-index: The g-index. *ISSI Newsletter 2*, 1 (2006), 8–9.

[12] Fiala, D., Rousselot, F., and Ježek, K. Pagerank for bibliographic networks. *Scientometrics 76*, 1 (2008), 135–158.

[13] Ford, L. R., and Fulkerson, D. Solving the transportation problem. *Management Science 3*, 1 (1956), 24–32.

[14] Hirsch, J. An index to quantify an individual's scientific research output. In *Proceedings of the National Academy of Sciences of the USA* (2005), vol. 102, pp. 16569–16572.

[15] Jeong, H., Néda, Z., and Barabási, A. Measuring preferential attachment in evolving networks. *Europhysics Letters 61*, 4 (2007), 567–572.

[16] Kleinberg, J. Authoritative sources in a hyperlinked environment. *Journal of the ACM 46*, 5 (1999), 604–632.

[17] König, D. Über graphen und ihre anwendung auf determinantentheorie und mengenlehre. *Mathematische Annalen 77*, 4 (1916), 453–465.

[18] Kuhn, H. W. The hungarian method for the assignment problem. *Naval research logistics quarterly 2*, 1-2 (1955), 83–97.

[19] Kumar, M. Evaluating scientists: Citations, Impact factor, h-index, Online page Hits and What Else? *IETE Technical Review 26*, 3 (2009), 165–168.

[20] Lehmann, S., Lautrup, B., and Jackson, A. Citation networks in high energy physics. *Physical Review E 68*, 2 (2003), 026113.

[21] Liu, X., Bollen, J., Nelson, M., and Van de Sompel, H. Co-authorship networks in the digital library research community. *Information processing & management 41*, 6 (2005), 1462–1480.

[22] Norris, J. R. *Markov chains*. Cambridge University Press, 1998.

[23] Radicchi, F., Fortunato, S., Markines, B., and Vespignani, A. Diffusion of scientific credits and the ranking of scientists. *Physical Review E 80*, 5 (2009), 056103.

[24] Su, C., Pan, Y., Zhen, Y., Ma, Z., Yuan, J., Guo, H., Yu, Z., Ma, C., and Wu, Y. Prestigerank: A new evaluation method for papers and journals. *Journal of Informetrics 5*, 1 (2011), 1–13.

[25] Walker, D., Xie, H., Yan, K., and Maslov, S. Ranking scientific publications using a model of network traffic. *Journal of Statistical Mechanics: Theory and Experiment 2007*, 06 (2007), P06010.

[26] Wendl, M. C. H-index: however ranked, citations need context. *Nature 449* (2007), 403.

[27] Woeginger, G. An axiomatic characterization of the Hirsch-index. *Mathematical Social Sciences 56*, 2 (2008), 224–232.

[28] Yan, E., and Ding, Y. Discovering author impact: A pagerank perspective. *Information Processing & Management 47*, 1 (2011), 125–134.

Annales Mathematicae et Informaticae
44 (2015) pp. 141–150
http://ami.ektf.hu

An improved Community-based Greedy algorithm for solving the influence maximization problem in social networks*

Gábor Rácz, Zoltán Pusztai, Balázs Kósa, Attila Kiss

Eötvös Loránd University
{gabee33,puzsaai,balhal,kiss}@inf.elte.hu

Submitted September 15, 2014 — Accepted March 30, 2015

Abstract

The influence maximization problem is to find a subset of vertexes that maximize the spread of information in a network. The Community-based Greedy algorithm (CGA) is one of the many that approximates the optimal solution of this problem. This algorithm divides the social network into communities, and then it takes into account for each node only its influence inside the cluster to which it belongs. Our method improves this algorithms with two modifications. We replace the clustering method of the CGA with a commonly used algorithm, namely the Louvain method, which runs by even one magnitude faster. We performed measurements to test how this replacement affects the running time and the precision of the algorithm. The results show that our variant significantly reduces the running time and the precision loss is less than five percent.

Keywords: influence spread, social network, community detection

MSC: AMS classification number(s): 91D30, 91C20, 51E23

*This work was partially supported by the European Union and the European Social Fund through project FuturICT.hu (grant no.: TAMOP-4.2.2.C-11/1/KONV-2012-0013). This work was completed with the support of the Hungarian and Vietnamese TET (grant agreement no. TET 10-1-2011-0645).

1. Introduction

Over the last few years a large variety of on-line social networks has become available. There are general purpose social networks such as Facebook[1] or VK[2] which provide medium to their users for sharing thoughts or talk about their everyday life. Other social networks have special interests such as the business-orientated LinkedIn[3] or the music-oriented Last.fm[4]. In addition to the above mentioned ones, social networks can be constructed based on email communications, phone call records, or co-authorship of scientific papers. The diversity and the volume of these networks have posed serious challenges to the scientists, however, they also offer great opportunity to understand human relationships. One interesting question among many others is to find a fixed number of vertexes through which the largest possible part of a network can be reached. This problem is mainly referred as influence maximization problem. It has a lot of practical usages, for example, in case of viral marketing the question is who should be targeted with sample products or who should be conceivably paid in a marketing campaign in order to influence as many members of the network as it is possible. In addition, if the most influential members of the network are found, it can be investigated why they are the most influential members [10].

In [1], Kempe et al. introduced two basic models, namely the *Independent Cascade Model* and the *Linear Threshold Model* for representing the diffusion of influence in networks. The influence maximization was considered as a discrete optimization problem. It was proven that the problem is NP-hard in both cases; nevertheless, it was also shown that based on submodularity of the scoring function the simple greedy algorithm assuredly approaches the optimal solution by a factor of $1 - \frac{1}{e}$. However, a serious drawback of this algorithm is that the influence of the candidate sets should be evaluated in each turn, which owing to the non-deterministic nature of the process is accomplished by using Monte Carlo simulations. As for large graphs these simulations can be very time consuming, several improvements were introduced since the greedy algorithm was published. In this paper, we focus on the *Independent Cascade Model* only.

In [5], a Cost-Effective Lazy Forward (CELF) optimization was presented that can significantly reduce the number of evaluations by exploiting the submodularity of the scoring function.CELF results a candidate set that has the same influence spread as the original greedy algorithm but is much faster (even 700 times faster [5]). Chen et al. in [2] proposed the NewGreedy algorithm that is an improvement of the original method in which at the beginning of an iteration each edge of the input graph is deleted with a certain probability. In this way the original problem can be converted into a reachability problem where the influence spread of a node set S is measured as the number of reachable nodes from S. It constructs a candidate set that has the same influence as the original

[1] `https://facebook.com`
[2] `https://vk.com`
[3] `https://www.linkedin.com`
[4] `http://www.last.fm`

greedy algorithm but it has shorter running time. In [3], Wang et al. introduced the COMMUNITY-BASED GREEDY algorithm, referred as CGA, which consists of two phases, a clustering and a dynamic programming phase. Their main idea is to divide the network into communities. The influence degree of a node in the community approximates its influence degree in the whole network. In addition, a dynamic programming method is used to select which cluster should contain the next member of the candidate set in each turn.

In this paper we present a solution for the influence maximization problem which relies on the CGA. In our solution, the clustering method of the CGA is replaced by a community detection algorithm, called LOUVAIN METHOD [4], which is a simple method and it can be computed extremely fast even in the case of large networks. However, in contrast to the original one, this method does not provide theoretical bound to the precision loss that the approximation can cause. Moreover, the dynamic programming phase is also simplified in our solution. Namely, in each turn only those nodes are re-evaluated which belong to the community that contains the previously selected member of the candidate set. We evaluated how these changes affect the running time and the precision of the algorithm in comparison with the CGA and to the NEWGREEDY algorithms. Our results show that the modified algorithm can run ten times faster than NEWGREEDY three times faster than CGA and its precision loss is less than five percent.

2. Background

A social network is modeled as an undirected graph $G = (V, E)$, where nodes represent individual persons while an edge between two nodes models some sort of relationships. The influence maximization problem is to find an S subset of V with cardinality k, where k is a fixed constant, that maximize the σ *influence function* which assigns a non-negative real value to each subset of V. Two basic diffusion models were introduced in [1] by means of which the influence function can be calculated. In both models, each node has an active or an inactive state, where the active nodes represent influenced persons who themselves can also influence others.

In the *Linear Threashold Model*, a node v has a random threshold θ_v, and v is influenced by its neighbour w according to a weight b_{vw} such that $\sum_{w \text{ neighbours of } v} b_{vw} \leq 1$. The diffusion process starts from an arbitrary set of nodes S, called seeds and the process unfolds in discrete steps: in step t, all the active nodes remain active, and any v node becames active for which the total weight of its active neighbors is at least θ_v, formally $\sum_{w \; active, w \text{ neighbours of } v} b_{vw} \geq \theta_v$.

In the *Independent Cascade Model*, the diffusion process also starts from an arbitrary set of nodes S and it unfolds in discrete steps: in the $(i+1)^{th}$ step, each node that has become active in the i^{th} step has a single attempt to influence its currently non-active neighbours. More precisely, for such a node the connected edges are taken one after the other with a fixed activation probability p. If an edge

was chosen, then the other endpoint is also get activated. The process stops if no new node has become active in a round or every node has been activated. The influence of S will be the number of activated nodes. In the rest of this paper, we focus on only the latter diffusion model.

In [1], it was shown that the influence function is submodular and monotone in the *Independent Cascade Model*. In other words, for each $S \subseteq V$ and a node v: $\sigma(S) \leq \sigma(S \cup \{v\})$. Moreover, the marginal gain of adding the same node to a growing set decreases as the set becomes larger, i.e. for each $S \subseteq H \subseteq V$ and a node v: $\sigma(S \cup \{v\}) - \sigma(S) \geq \sigma(H \cup \{v\}) - \sigma(S)$. With these properties, it can be guaranteed that the result of the greedy algorithm is less than $(1 - \frac{1}{e})$ times of the optimal solution. Formally, $\sigma(S_{greedy}) \geq (1 - \frac{1}{e})\sigma(S_{opt})$, where S_{greedy} denotes the result of the greedy algorithm, while S_{opt} the optimal solution respectively. Owing to the non-deterministic nature of the diffusion model in practice the values of σ are approximated by means of Monte Carlo simulations. For a given node v, usually 10.000 simulations are performed to approximate $\sigma(S \cup \{b_v\})$, where S denotes the set of nodes selected in the previous steps of the algorithm, therefore the algorithm is time consuming in case of large networks.

An improvement was introduced in [2], in which at the beginning of an iteration each edge of the original graph is deleted with probability $1-p$. Then, the influence of a set of nodes S can be measured by the number of reachable nodes from S. In addition, the computation of the marginal gain of a node v with respect to an $S \subseteq V$ can be seen as a reachability problem which is defined in the following way:

$$\sigma(S \cup \{v\}) - \sigma(S) = \begin{cases} 0, & \text{if } v \in R(S), \\ |R(\{v\})| & \text{otherwise,} \end{cases}$$

where $R(S)$ denotes the set of the reachable nodes from S.

In this paper, we focus on the Community-based Greedy algorithm that was introduced in [3]. Its approach is orthogonal with the improvement applied in NewGreedy, it is based on graph partition. The algorithm consists of two phases, a clustering and dynamic programming phase. In the first phase, a community detection algorithm is performed on the input graph, this algorithm has two subphases, namely a label propagation and a combination step. Initially, each node has a unique community label. Next, for each node the set of its influenced neighbours are computed using the *Independent Cascade Model*. Then the community labels are propagated iteratively in τ rounds (where τ is given in advance) through the network. The main principle of the propagation is that a node v should belong to the community that contains the majority of its influenced neighbors. Formally, $v.c^t = maxCMT(w_1.c^{t-1}, ..., w_k.c^{t1-1})$, where t denotes the tth round, $w_1, ..., w_k$ are the neighbours of v, $v.c$ denotes the community label of v, and $maxCMT$ is to compute the majority of the labels.

In the combination phase, the algorithm combines community C_l and C_m, if the combination entropy of C_l to C_m is above a given threshold. This phase helps to reduce the difference between the node's influence degree in its community and its influence degree in the whole network. The *Combination entropy* was introduced

to measure the connection between two communities and it is defined as:

$$CoEntropy(C_l, C_m) = max_{v \in C_m, u \in C_l, isLive(e_{uv})} \frac{\bar{R}_m(\{u\})}{R_m(\{v\})},$$

where $R_m(\{v\})$ is the influence degree of v in C_m, $\bar{R}_m(\{u\})$ is the influence degree of u outside C_m. $isLive(e_{uv})$ denotes that the node u and the node v are connected with a live edge. An $(u, v) \in E$ edge is a live edge, if the node v influenced the node u, namely u becomes active from inactive for at least Q/r times out of Q simulations of the previous step. (In the original paper, the r was set to 2, however, during the evaluation we experiments additional values.) The second phase of the CGA algorithm is a dynamic programing method for selecting the communities which includes the best candidates. To mine the k^{th} seed, the method chooses the community that will yield the largest increase of influence degree. Any existing algorithms can be used to calculate the influence in the chosen community. The CGA algorithm is the basis of our solution which is described in the next section.

3. LouvainGreedy

In this section, we present our solution, namely the LouvainGreedy algorithm, to solve the influence maximization problem. Our algorithm is based on the Community-based Greedy algorithm with two modifications.

First, the clustering phase was replaced by a lately introduced community detection method called Louvain method presented in [4]. The Louvain method is a hierarchical agglomerative community detection algorithm which uses modularity maximization. The modularity measures the quality of a partition; and it is defined as in the following:

$$Q = \frac{1}{2m} \sum_{i,j} [A_{ij} - \frac{k_i k_j}{2m}] \delta(c_i, c_j),$$

where A denotes the weighted adjacency matrix of the graph,

$$A_{ij} = \begin{cases} weight(e_{ij}), & \text{if } e_{ij} = (v_i, v_j) \in E \\ 0 & \text{otherwise.} \end{cases}$$

$k_i = \sum_j A_{ij}$ denotes the degree of node v_i, $m = \frac{1}{2} \sum_{ij} A_{ij}$ denotes the total weight of the edges, and c_i, c_j denotes the cluster of the node v_i and v_j respectively, δ is the Kronecker delta

$$\delta(c_i, c_j) = \begin{cases} 1, & \text{if } c_i = c_j, \\ 0 & \text{otherwise.} \end{cases}$$

The algorithm consists of a label propagation and a node merging step. Initially, each node has a unique label. Next, each node adopts the community label of its neighbors, if the overall modularity increasing with the label adoption. Namely, for

all neighbors j of a node i, the gain of modularity are evaluated mean by removing i from c_i and by placing it into c_j. Then node i is placed in the community for which the gain is maximum, but only if it is positive. This propagation step is repeated until a local maximum has been obtained. (Note, that the propagation may depend on in the order the nodes are processed.)

When a local maximum has been obtained, the nodes with the same community label are merged into one single node keeping the outgoing edges and transforming the inside edges into weighted self-loops. After the merging step, the label propagation starts again. These two steps are repeated iteratively. The process terminates when each node has a different label at the end of the label propagation step, since in that case, there are no more merge-able nodes. The process results a hierarchical decomposition of the input graph. Because of the simplicity of the algorithms, it can be computed extremely fast even in case of large graphs. Moreover, according to [8], it is one of the best modularity based community detection algorithm.

The second important modification that we made on the CGA is the replacement of the dynamic programming phase. In our solution, after a graph has been partitioned into communities, the most influential node is computed within each community using the NewGreedy algorithm. The node with the maximum influence degree is selected as the first member of the candidate set. Then, in the community that belongs to the selected node, the influence degree of the nodes are recomputed. The process is repeated until all the seeds are selected.

Note, that if in the k^{th} turn, a node v has been selected from the cluster C_v, then in the $(k+1)^{th}$ turn, the marginal gain of nodes that are not members of C_v remain unchanged. That is because we compute the influence of a node inside the cluster only. Therefore, for each u that $C_u \neq C_v$ the following holds $\sigma_{C_u}(S \cup \{u\}) = \sigma_{C_u}(S \cup \{v\} \cup \{u\})$, where S denotes the candidate set in the k^{th} turn and σ_{C_u} denotes the influence of a set inside C_u.

Algorithm 1 shows the pseudo code of our solution. Initialy, the seed set S is empty, and the Louvain mehod is called to compute the clusters or communities (line 2). Next, for each cluster (line 3-7) the subGraph submethod computes the subgraph which belongs to the cluster. A subgraph contains the nodes of a cluster and the edges among them, but the outgoing edges are not included. After the subgraphs are computed, the NewGreedy algorithm assigns the influence degree to each node within each subgraph. The node that has the maximum influence degree in the cluster is recorded by $C.max$. After this initialization, a process is repeated k times (the cardinality of the candidate set). The process (line 8-13) selects the cluster ($max\_cluster$) containing the most influential node ($max\_cluster.max$) in each step. The most influental node is added to the seed set S, and then the marginal gains of nodes in $max\_cluster$ are recomputed. The node with the maximum marginal gain within the cluster is refreshed. At the end of the process, the algorithms returns S which contains the selected seeds.

Algorithm 1 LouvainGreedy

Input: $G = (V, E, W)$, number of seeds k, activation probability p, MC count r;
Output: list of seeds S;

```
1: S ← the empty list
2: Clusters = Louvain(G)                          ▷ community detection
3: for all C ∈ Clusters do
4:     C.SG ← subGraph(G, C)
5:     NewGreedy(C.SG, p, r)   ▷ assign marginal gain to each node in cluster C
6:     C.max ← argmax_{v∈C}{v.influence}
7: end for
8: for i ← 1, k do
9:     max_cluster ← argmax_{C∈Clusters}{C.max.influence}
10:    S = S ∪ {max_cluster.max}
11:    NewGreedy(max_cluster.SG, p, r)   ▷ refresh marginal gains in cluster C
12:    max_cluster.max ← argmax_{v∈C}{v.influence}
13: end for
14: return S
```

4. Results and discussion

We compared our LouvainGreedy (LG) algorithm with the NewGreedy (NG) and the Community-based Greedy Algorithm to reveal how our modifications on CGA affect the running time and precision. Section 4.1 describes our experiments and Section 4.2 discusses the precision of the methods in details.

4.1. Experiments

In the comparison process, two real-life networks were used. The first, which is called NetPHY, is extracted from the arXiv[5] academic collaboration network by Wei Chen et al. [2]. It is constructed using the full paper list of Physics section from 1991 to 2003. Each node represents an author and an edge is added between two authors whenever they jointly wrote a paper. The numbers of nodes and edges are respectively 37 154 and 231 584. The second data set, which is referred EmailEnr[6], is derived from the Enron email network, which consists of around half million emails. Nodes represent email addresses and if an address i has sent at least one email to address j, then an undirected edge between i and j is contained in the graph. It consists of 36 692 nodes and 183 831 edges. The experiments were done on a server with 12-core 2.67 GHz Intel Xeon CPU and 24 GB memory.

All the three algorithms were re-implemented in Java 1.7. In the combination step of (CGA) we computed the live edges as follows. We performed the edge-deleting part of the NewGreedy algorithm 100 times and we recorded for each

[5] `http://arXiv.org`
[6] It is available at `http://research.microsoft.com/enus/people/weic/graphdata.zip`

edge that how many times it was not deleted in the resulted graphs. If an edge has remained intact at least 1/8 part of the simulation count, then the edge was marked as a live edge. In addition, we used the Gephi Toolkit[7] [9] implementation of the Louvain community detection algorithm in our solution.

Table 1 contains the results belonging to the NetPHY data set where the cardinality of the seed sets was 20, the activation probability was 0.02. In the greedy steps 10 000 Monte Carlo simulations were performed. The running times of the algorithms consist of a clustering and a greedy phase. The clustering phase can be performed in advance as a pre-processing step and its result is reusable afterwards. As the table shows, the main differences among the running times of the investigated algorithms are in the lengths of the greedy phases. That is because the size distributions of the resulted communities are significantly different in each clustering algorithm which affect the running time of the greedy phases as the greedy algorithms run faster on smaller graphs.

	Running time (sec)				**Influence**	
	clustering	**greedy**	**all**	**relatively**	**average**	**relatively**
LG	7	373	380	9.5%	890	97.2%
CGA	21	1203	1224	30.0%	915	99.9%
NG	–	4021	4021	100%	916	100%

Table 1: $NetPHY, k = 20, p = 0.02, MC = 10\,000$

The quality of results was tested by starting with 10, 000 random cascade diffusion processes and taking the average number of the influenced nodes at the end of the processes. It can be seen in Table 1, that our LouvainGreedy algorithm ran ten times faster than the NewGreedy and its precision loss was less then 3% of the result of the NewGreedy.

	Running time (sec)				**Influence**	
	clustering	**greedy**	**all**	**relatively**	**average**	**relatively**
LG	5	564	569	10.7%	4500	99.0%
CGA	524	4555	5079	95.1%	4535	99.7%
NG	–	5339	5339	100%	4547	100%

Table 2: $EmailEnr, k = 20, p = 0.02, MC = 10\,000$

Table 2 includes the results on EmailEnr data set with the same parameters as above. As can be seen, CGA is much slower on this data set. It is because EmailEnr network has one and a half times more edges than NetPHy. Moreover, the clustering steps of CGA results a cluster that contains approximately two-thirds of the nodes, therefore the running time of the greedy algorithm could not be decreased. However, our algorithm was ten times faster than NewGreedy with 1% loss of precision using this data set as well.

[7] `http://gephi.github.io/toolkit/`

4.2. Precision

In [3], Wang et al. proved that using the *CGA* algorithm, the influence degree of the resulted set $R(I)$ (where I is the resulted set) is $(1 - e^{-\frac{1}{1+\triangle d*\theta}})$ approximate by the influence degree of the optimal solution, denoted by $R(I^*)$, where θ is the threshold used in the combination step and $\triangle d$ is the maximal difference between the number of nodes affected by a node in the network and that in its community. That is $R(I) \geq (1 - e^{-\frac{1}{1+\triangle d*\theta}})R(I^*)$.

As can be seen, the approximation highly depends on the threshold of the combination phase, where the communities are combined based on the *combination entropy*. Therefore, we conducted experiments by applying the combination step of the CGA algorithm on the communities that are resulted by the Louvain community detection method. However, these experiments gives very similar running times and precisions as the original algorithm. This is because in the combination step many communities were merged as they combination entropy was above the threshold. The threshold was set to 0.3 as in the original paper.

However, our experiments described in the previous section show that the *LouvainGreedy* algorithm can achieve high precision without the combination step. We suppose that is because the other factor of the approximation, the $\triangle d$ that is the maximal difference between the influence degree of nodes affected by a node in the whole network and that in the community, remains low when the *Louvain method* is used. It suggests the nodes did not effect each other across the resulted communities.

As we saw in Section 3, the *Louvain method* is based on modularity maximization, which is a measure of the quality of a graph partition. Therefore, to give theoretical bound to the approximation factor of the *LouvainGreedy* algorithm, we should describe how the modularity affects the result. But it remains an open question. Although our experimental results are promising, without such a theoretical bounds, we can not be sure how precise result we have got.

5. Summary and future plans

We presented a new method for solving influence maximization problem which is based on the Community-based Greedy algorithm. Our method combines the Louvain method, a wildly used community detection algorithm, with the NewGreedy which is a greedy algorithm that approximates the optimal solution of the problem.

We compared the presented algorithms w.r.t. running time and quality of their results measured by the number of influenced nodes at the end of random cascade processes starting from the resulted seed sets. The experiments show that LouvainGreedy can run ten times faster than NewGreedy and the precision loss is less than five percent. However, our solution can not provide theoretical bound to the goodness of its result. Thus, we tested the Louvain community detection algorithm along with the combination step of the CGA, which merges communities

if their combination entropy is above a threshold. The tests showed that in the combination step a large community is formed because of the community merging. this has a significant effect on the running time as the greedy step is time consuming on large clusters.

In the future, we would like to improve the presented algorithms using parallization. The most consuming part of the presented algorithms is the performance of Monte Carlo simulations. Running these simulations in parallel can significantly reduce the computation time of the greedy steps. Currently, Apache Hadoop [6] and the Pregel [7] systems are under investigation for this purpose.

References

[1] Kempe, D., Kleinberg, J., and Tardos, É., Maximizing the spread of influence through a social network. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '03, pp. 137–146. ACM, 2003.

[2] Chen, W., Wang, Y., and Yang, S., Efficient influence maximization in social networks. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 199-208. ACM, 2009.

[3] Wang, Y., Cong, G., Song, G., and Xie, K., Community-based greedy algorithm for mining top-k influential nodes in mobile social networks. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 1039–1048. ACM, 2010.

[4] Blondel, V. D., Guillaume, J., Lambiotte, R., and Lefebvre, E., Fast unfolding of communities in large networks. In *Journal of Statistical Mechanics: Theory and Experiment*, Vol. 10 (2008): P10008.

[5] Leskovec, J., Krause, A., Guestrin, C., Faloutsos, C., VanBriesen, J., and Glance, N., Cost-effective outbreak detection in networks. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '07, pp. 420-429. ACM, 2007.

[6] White, T., *Hadoop: The Definite Guide*. O'Reailly Media, 2009.

[7] Malewicz, G., Austern, M. H., Bik, A. J., Dehnert, J. C., Horn, I., Leiser, N., and Czajkowski, G., Pregel: a system for large-scale graph processing. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*, pp. 135-146. ACM, 2010.

[8] Lancichinetti, A., and Fortunato, S., Community detection algorithms: a comparative analysis. *Physical review E*, Vol. 80(5) (2009): 056117.

[9] Bastian, M., Heymann, S. and Jacomy, M., Gephi: an open source software for exploring and manipulating networks. In *Proceedings of the third International Conference on Weblogs and Social Media*, pp. 361-362, 2009.

[10] Kósa, B., Rácz, G., Pinczel, B. and Kiss, A., Properties of the Most Influential Social Sensors. In *Proceedings of 2013 IEEE 4th International Conference on Cognitive Infocommunications (CogInfoCom)*, pp. 469-474. IEEE, 2013.

Annales Mathematicae et Informaticae
44 (2015) pp. 151–164
http://ami.ektf.hu

Survey of attacking and defending in the RFID system

Tibor Radványi, Csaba Biró, Sándor Király, Péter Szigetváry, Péter Takács

Eszterházy Károly College, Eger, Hungary
radvanyi.tibor@ektf.hu
birocs@aries.ektf.hu
ksanyi@aries.ektf.hu
takip@aries.ektf.hu
szigipet@aries.ektf.hu

Submitted October 5, 2014 — Accepted May 5, 2015

Abstract

In this article we are dealing with the connections between nowadays most dynamically improving automatic identification-related RFID technology and cryptographic algorithms. You are going to be introduced to the possibility of attacks against RFID systems and the ways to defeat them. We are also dwelling on the suitable and non-suitable cryptographic algorithms among the well known and frequently used ones. The type of the RFID tag highly influences the group of the suitable algorithms. The size of the tag's integrated memory matters a lot, as well as the fact that it uses its own intelligence or we are working with a cheap passive tag.

Keywords: RFID, data security, cryptography, Gen2, AES, DES, RSA

MSC: 68U35, 68M01, 90B18, 94A860, 68P25

1. Introduction

Nowadays widespread forms of identification systems are in use. It means such code- and communication systems which identify people and objects uniquely. The most dynamically improving state-of-the-art identification system is the RFID

(radio-frequency identification), using more channels in the electromagnetic frequency interval between 125 KHz and 2.4 GHz to carry out its functions. We can expand the basic frequency to the 2.4–5.8 GHz range. This range is called SHF. The reading distance here is above 3 m.

Paired with different types of sensors or location systems it can be used in many fields. This technology is employed in motor vehicle production, logistics, pharmaceutical and army technology, and in a lot of other areas. Modern passports, digital identifiers, and the newest ways of payment all take advantage of its identification and security opportunities. [12, 13]

2. The RFID Systems and the direction of their development

In this section we will be presenting the part of RFID focused on the transponders and readers. Because they are exposed to attacks.

The RFID has a great advantage over the bar code systems, that it does not require a direct view of the reading. When we want to read the barcode, we need to catch all products, on witch there is label or vignette. It is a very slow task and it requires a lot of human resources. Therefore it is expensive. At the same time we can read a lot of transponders. If we have due RFID reader with a well positioned special antenna, we can realize long-distance reading too. Storage capacity depends on the integrated chip's capacity. It can range from a couple of dozen bytes to a few megabytes.

The RFID system can be partitioned to isolated sections of equipments. These are the transponders, named tags, readers with transmitting and receiver aerials, middle-ware, database and the user applications. The user application can be developed to smart-phones, to tablets and to PCs. The primary targets for attacks are the tags, which store the identification and descriptive data. The readers and aerials are the other side of electro-magnetic communication. The readers can be attacked with software tools. If the attack is successful, through the software of the middleware the database can be infected. In this case the danger escalates, because all information can be lost or worse, if it falls into the wrong hands.

2.1. Passive tags

The passive tags are low-cost and can store large quantities of data. They do not have an active transmitter, so they can not radiate data independently. They use the radiated energy from the reader. They collect the energy out of the reader wave and with own aerials send back the modulated wave to the reader. The aerials must be well set and tuned, because the energy will be spread and the aerial of reader will not detect it. The usability is increased by special or hard housing in industrial applications. This method can increase the costs too. So the industrial hard tags are used identification of valuable objects . Typically these tags are used in LF

(125 KHz), HF (13.5 MHz), UHF (900 MHz) and microwave (2.4 GHz). Standards direct for their distribution. But these might vary from country to country. [1]

2.2. Active tags

Active tags have a independent transmitter with energy source. They can radiate the stored data continuously. The energy source is usually a battery that could last for some years. Sometimes the active tags contain an energy harvesting chip, which collects the energy from the background radiation. These tags use the 433 MHz, 2.4 GHz and 5.2–5.8 GHz frequency. The reading distance can be around 100 m. One or two order of magnitude can be the difference compared to the passive tags. The UHF passive tag can be read from a distance of 1–3 m, but the active tag can be read from 100 m. Their price can be very high, but it can fluctuate depending on the size of the memory, the life of battery and the kind of their wrapping. The active tags are appropriated by high-value containers, rail cars. If the tags are collected and reused at the end of the logistics process, you can save a lot in termsof cost. [1]

2.3. The RFID operation principle

Those RFID technologies , which use the LF and HF/NFC frequency, are characterized with the small distance reading. Between the tag and the aerial an inductive coupling builds up. There is a coil-antenna in the antenna of the reader and the tag. They create together an electro-magnetic field. The tag collects energy from this common EM field and with the help of it the tag can radiate back the stored data. Therefore the tag and the aerial of the reader must be near to each other. That means a couple of inches. These systems are less sensitive to interference caused by liquid and metal. So the LF and HF/NFC tags can be used better and more efficiently in these environments.

The passive UHF tags apply the propagation coupling. This coupling uses the backscatter communication method. Backscattering has important applications in astronomy, photography and medical ultrasonography. In this way the reader and the tag don't constitute a combined EM field. The tag uses the energy emitted by the reader to radiate back an altered (modulated) wave.

2.4. Fields of application

Application, where we can use RFID technology.

- Access Control Systems
- Identification of people and animals
- Identification of things
- Products, tracking vehicles

We can use it in toll systems, because it is fast, reliable and we can write back to the memory of the tag a timestamp or validating data. This technology can be well used in the following important cases: for retail sale, for the library and timing sport events. It is in the e-passport too. It is frequently used and very necessary while travelling, but as such, it is at serious risk. It can become the main target of attacks, primarily cloning attacks. The e-passport to expand requires additional means of protection or identification capabilities. This can be biometric identification, for example fingerprint, or retinal scanner.

3. Researching and improving the RFID in order to increase its efficiency; security or efficiency

Today's RFID protocols are formed in such a way that optimizing efficiency has become more highlighted than consumers' data security. We insist on using so-called cryptographic protocols to be able to save all the pieces of information in a trustworthy manner. Our aim is to broaden the tasks and only affect efficiency in a minimized way.Through this procedure communication with identification could be made more efficient and safe.

The RFID based identify-and-follow systems used in the retail trade environment are the living examples of this hidden working principle. However, there are numerous danger factors regarding this method. We can picture it the way that the personal tools, used by the consumers, contain invisible microchips. Via them smooth, discreet checks can be taken through different procedures. As during these checks data streams and exchanges are going on in the system, outsiders and users might get others' personal information. These problems are really important and require an swift solution as nowadays the protection of personal information is the top priority during the operation of a computer system. Some experiments have been done, some of them are used in practice, but sometimes the technology is unsafe even with them.

Finally, the priority of safe and clear information handling has been accepted and the importance of efficiency has been overtopped. We also agree with the thesis that the data security is our top most priority, especially in the case of systems where privacy is paramount. For example, bank services should rather be slower but safer, than faster with the possibility of less security.

4. Main attack possibilities

Algorithmic attacks: performed through the transfer channel. We separate active and passive ways of attacking. The passive method means that the attacker gets hidden messages by bugging a public channel. Contrary to the passive, during an active attack the attacker distributes on the channel himself. We can see the attacks on the figure 1. In this figure we can see some attacks against the RFID system. You can see the attacks can come from many directions. The attack can

be aimed at the transponder, the communication channel and at the software and hardware system. The destruction and detachment are danger factors too, but they cannot be taken advantage of. [6, 7]

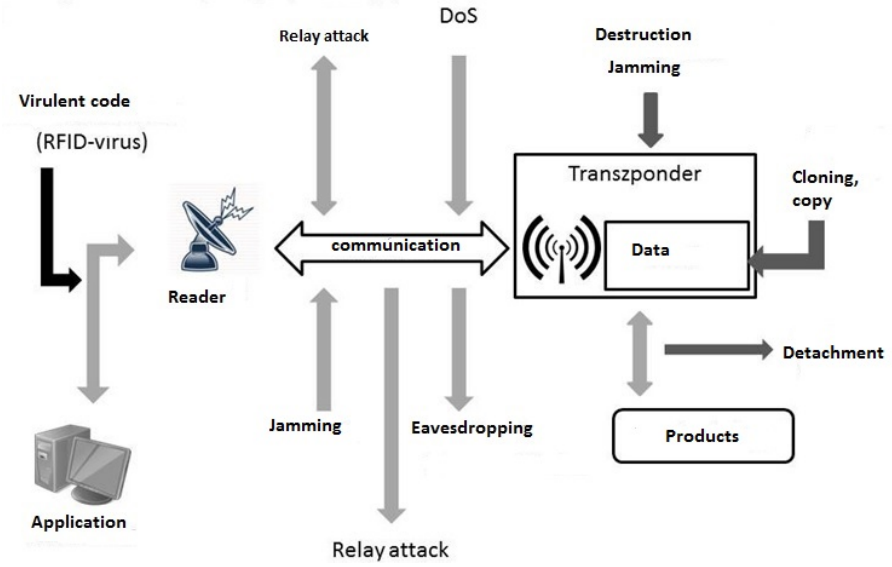


Figure 1: Attacks against the RFID system

4.1. Cloning a tag

If the RFID tag is not protected with encryption and we know its command set, function, and the memory partitioning, well, it is very easy to copy it or replace it with a copy. It causes problems if the system does not pay attention to the persistence of duplicated tags.

There are different complex solutions for storing data on tags. The simplest one is the read-only tag- it just contains an invariant unique identifier. If the read-only transponder gets close enough to the reader's field, it instantly wakes up and starts to radiate its identifier number. The attacker can easily create a clone which contains the exact same code. It is not necessary to get in physical touch with the transponder, we only need a reader which is able to read the identifier and write it to another. In more serious and complicated systems, where security is paramount we should avoid using read-only transponders and insecure data storage.

When working with more complex systems we should avoid using read-only transponders and unencrypted data storage. By default the transfer medium, in this case radio waves, transfer the data without encryption. Reading is not recorded on the transponder, so readings can be repeated without authorization.

Of course the transfer channel might be encrypted with the well-known encryption methods (DES, block encryption, RSA). Due to the small data quantity the

transponder is able to store the key is usually small, so in case someone has the required amount of information the key can be decoded.

4.2. The Malware

Malware is malicious software, and its aim is to break and interfere with computer systems. RFID malware is a malware spreading through RFID tags.

A totally different type of threat comes to life when hackers and criminals make RFID tags work unexpectedly and maliciously. Through the queries of the computer-connected or the mobile readers hackers do it for the RFID tags' unique identifier, or for the data stored on the tag, which is usually the key for the database, or evokes a real action.

Until now everybody who works on RFID technology, silently claimed that reading the RFID tag cannot modify the background software, especially in a harmful way. Unfortunately they all were wrong. During a research it was proven that the RFID software has a weak point. A tag intentionally can be infected with a virus and the virus is able to infect the background database the RFID software uses. After that the virus can easily spread to other RFID tags.

For example: The perpetrator only needs to purchase a cheap item from the supermarket, which has a tag attached. After paying for the good he goes home and cuts off or destroys the tag. Then he puts an infectious tag on the product. Goes back to the supermarket, to the cash desk, and pays for the item again. But when the tag is scanned it infects the supermarket's product database, potentially inflicting any kind of damage, like changing the prices. But it means a far higher risk at airports. The virus might help drug smugglers or terrorists hide their packages. Furthermore, the infected database and luggage can infect other airports' databases as well. As a result of the virus packages might get to entirely different destinations than they ought to.

By this time the general structure of the problem has become clear. When an unsuspicious reader scans an infected tag, there is a risk that the tag takes advantage of a vulnerability in the middleware to evoke unwanted actions, like infecting the database. [10]

Classes of RFID malware:

1. The RFID exploit is a malicious tag data, using the parts of the RFID system it gets in touch with. RFID systems are as sensitive to hacker attacks as traditional computer systems. When a reader reads an RFID tag, it expects to get some information in a given format. However, someone can write a piece of data, the format and content of which are both so unexpected that it can influence the software, or potentially, the database of the RFID reader, too.

2. The RFID worm is an RFID-based exploit, taking advantage of the network connection to gain the ability of self-replication. RFID worms multiply by using the online RFID services, but they are able to spread themselves on tags as well. RFID worms make the unsuspicious RFID servers download

and out some files and carry out the instructions in them. This file aims to compromise the RFID middleware server the way most of the internet-based malwares do. The worm-infected RFID software is able to infect other RFID tags by overwriting their pieces of data with a replication of the RFID worm code.

3. The RFID virus is such an RFID-based exploit that replicates its code by itself to other tags without network access. RFID viruses might have contents that can interfere with or modify the RFID system's work in the background. As a freshly infected enters the world it infects other RFID systems (provided they use a common software system). Then these RFID systems infect other RFID tags, which infect other RFID software systems and so on.

4.3. Attack through the RF interface

Other types of attacks against RFID systems come through the RF interface. RFID systems communicate with the help of radio systems and electromagnetic waves both over short and long distance. This way the attacker has a chance to attack the RFID system through the radio frequency interface, as there is no need to get into physical connection with the reader or the transponder. One type of this attack is well known, so in the following paragraph We would like to explain it.

4.4. Eavesdropping

All kinds of wireless communications can be eavesdropped on a device which restrictive is made up of RFID technology. The attacker does not need energy and physical contact with the reader and the tag. Consequently, the attack can be performed from longer distances. The attacker has to catch the transmitted signals before the system stores it.

Eavesdropping occurs between the reader and the transponder. The effective range of the RFID systems vary between a few centimetres (e.g.: 13,56 MHz) to more meters (e.g.: 868 MHz). A radio's antenna requires a far smaller output voltage to get usable signals, so communication can be eavesdropped from a large distance.

Finke and Kelter have appointed that the 13.56 MHz inductively charged system can be eavesdropped from 3 meters. The receiver can sense the reader's motionless signals from hundreds of meters on a few kHz range. From a greater distance the signal might be disturbed by metal objects like fences, aluminium objects, or huge buildings. [14]

What does the success of eavesdropping our devices (reader and transponder) depend on? The number of influential factors is high.

- Depends on the characteristics of the RF space. This defines the geometry, structure and output power of the antenna.
- Interfering object between the reader and transponder and the size and location of metal objects are also an important factor.

- It is influenced by the quality, structure and geometry of the attacker's device, and also depends on the power emitted by the reader.

- It is also an important factor that passive or active transponders are used in the RF communication. If the tag is passive, it uses the power generated by the reader, this way the reflected useful information participates in the communication with lower energy usage. In the case of UHF tags (868 MHz – 915 MHz) 1–3 meter. If the tag is active or semi-passive so it has its own power source this range can be increased up to 10–30 meters. In case of active tags the emitted information is easier to catch due to its energy and easier to hide in larger attack areas. The at-tack area is a space where the attacker sets his eavesdropping device until he can perform a successful attack.

The following attacks may occur during eavesdropping:

- Secret or personal data may get into unauthorized hands. In this case the attack does not effect the communication, and it is almost impossible to detect the attack. Using cryptographic protocols may help defending the data.

- The attacker modifies the eavesdropped data and the false information is transmitted to the reader. This act requires a specific device and it is really hard to perform.

- Another possibility is that the attacker does not modify the data but replaces it. This could happen when the transponder sends a lot of information to the reader, so the communication requires much more time. In these cases of eavesdropping the attacker may get detected and his data blocked. Using control data, cryptographic algorithms and combinations of protocols may help detecting the attacker.

- The "relay attack" is a much more complicated type of eavesdropping and it also requires serious technical preparation. In this case the attacker does not only gather data but also transmits it on another channel. eg.: WIFI -– longer range. In the other place the data could get processed by an-other device eg.: during a purchase. This attack is really hard to block due to the properties of contactless payment methods. For the time being combined with other identifying methods it provides a good possibility. The simplest way is using a pin code but any personal or stationary biometric identification can be used. [15, 16]

It is clear that eavesdropping can be performed really easily sometimes. It holds a lot of possibilities for the attacker and it is really hard to detect and block. In order to protect data, if we can not secure the communication channel, we should make the information difficult to process in case of an attack. Cryptographic protocols provide data protection during in-formation exchange.

5. Cryptographic protocols - can be used?

Creating security in low-cost RFID systems is very difficult. Due to limited resources, strong ciphers are difficult or sometimes impossible to implement, so extremely simple algorithms and protocols need to be designed that take into account the limitations of passive systems. [8, 9, 11]

CrySys (Laboratory of Cryptography and Systems Security) recommends the XOR, and RSA.

The basic concept of development started out of the following:

$$R \to T : x \oplus k = a$$
$$T \to R : f(x) \oplus k = b$$

$$I(h, k) = H(x \oplus f(x))$$

where the

$$H(x \oplus f(x)) \text{ entropy of the } x \oplus f(x)$$

5.1. The XOR protocol

The structure of the XOR protocol is similar to the previous example, but it uses different keys in different directions. One option to achieve this the XOR key generation, where R randomly chooses a new k (i) , does XOR encryption, and performs k (i-1) key. We get the following protocol:

$$R \to T : x \oplus k_1 = a$$
$$T \to R : f(x) \oplus k_2 = b$$

We need a secure key-updating scheme.

$$R \to T : a^{(i)} = x^{(i)} \oplus k^{(i)}, k^{(i)} \oplus k^{(i+1)}$$
$$T \to R : b^{(i)} = x^{(i)} \oplus k^{(i)}$$

where $i = 2, 3, \ldots$ a counter, which is increased in every run, $x(i)$ the i-th random numeric and $k^{(0)}$ and $k^{(1)}$ shared keys are pre-set . $k^{(1)}$, $k^{(2)}$, $\ldots$ sequence does not change randomly, but the attacker cannot follow their value.

5.2. Other possibilities for defeating attacks

RFID systems are under different types of attacks.In several of these cases we need to protect the tag itself, but in other cases we have to inhibit the tag from being read and modified. In some countries where the e-passports are in use no copy-protection is provided for the users. We can prohibit cloning or attach-connected attacks with second level identification like a PIN code or biometric identification. Communication attacks are relatively easy to defeat with cryptographic methods as the messages can be encrypted with different algorithms.

6. Authentication

6.1. Legitimation based on derived keys

The main disadvantage of the legitimation mentioned above is that every transponder uses the same k encryption key. This feature means a potential danger factor for every similar application- which uses a very large number of transponders. Taking into consideration the fact that these transponders are available for everyone we have to consider the possibility of the key getting compromised. In this case the procedure becomes totally useless. To provide it, every transponder gets a unique key, increasing their security this way.

6.2. Encrypted data connection

We expand the previous situation with a potential attacker. Here the attackers can be divided into two groups. The first –Attacker1- tries to stay in the background and get useful information from the data store by eavesdropping and other passive methods. On the other hand Attacker2 actively takes part in the communication and modifies its content for his (or for someone else's) good. The cryptographic methods provide a solution against both attackers. The data which needs to be transferred (plain text) is encrypted so the attacker will not be able to reveal its original content. Random number generation is possible in such small sizes as the chip.Taking advantage of this, it is possible to expand cheap passive tags in order to implement the above mentioned encryption algorithms. [18]

6.3. Hash based access control

In case of the cheap smart labels it is necessary to provide a simple security scheme based on simple hash functions. The implementation of the algorithm is implemented in hardware. With this method the tags operating in closed and opened state are separating small slices from the memory in order to store so called metaIDs. The algorithms are relatively simple. The hash functions need to be implemented in the transponders. The scheme is flexible so it can be extended by multiple access or write authorizations. The meta-IDs are simple to query in this way the background database provides an easy building opportunity to third-party manufacturers. Furthermore the uniqueness of the meta-IDs may provide an easy identification.

6.4. Asymmetric key checking

Finally among the encryption methods the asymmetric key checking method needs to be emphasized. In this method when a reader wants to send data to a selected transponder (the message is v) it is enough for the tag generates a r random number during the transfer of listening-sensitive data and then sends it back to the reader. From this value the reader calculates the $v \oplus r$ then sends it to the tag. The listener

who is out of the range of the backward channel just hears this $v \oplus r$ and he cannot conclude to value of the original v. [5]

Against eavesdropping we should detail and confirm the processes mentioned:

- The reader transmits a request to the tag.
- The tag identifies itself to the reader.

We try to force the defense to the communication between the tag and the reader, assuming that the inner data transmission between the reader and the background server is al-ready safe.

Of course the following protocol highly depends on the use of active or passive tags in the communication. The existing requirements are already different and the available computing capacity also shows a huge difference. Take a look at the communication scheme on figure 2.

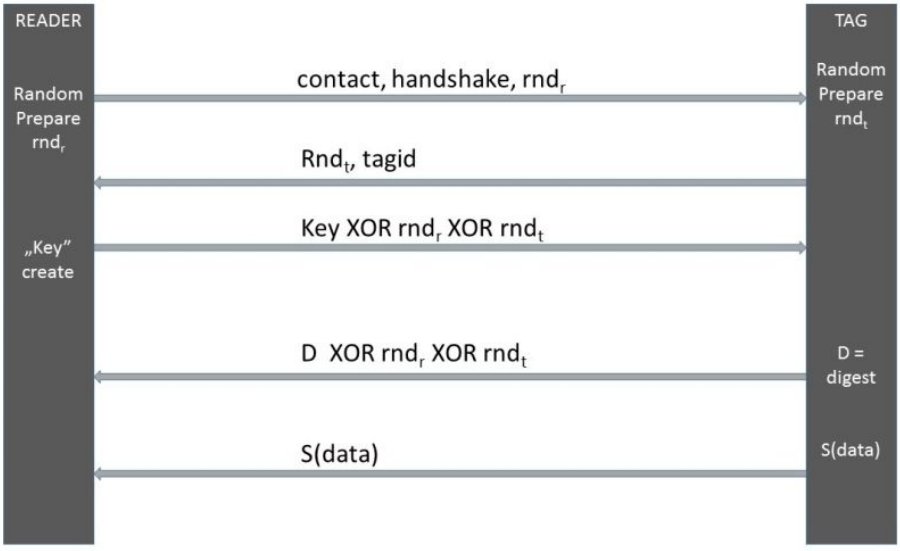


Figure 2: Reader and transponder communication

As shown we use a XOR function in the encryption which can be easily implemented on hardware level, so there is no difficulty in using it in passive tags. The XOR protocol uses different keys in different directions.

Also an S function appears which requires a little detail. First, consider the so-called P and S boxes. These are the basis of cryptographic algorithms. Their advantage is that they are easy to implement electro-technically. This way they can be integrated to the passive tag's limited set of tools. In case of active tags this is not a problem, because the tag contains intelligence, a programmable processor so the whole AES algorithm is feasible at a relatively low energy input and short time.

In case of passive tags we use the combination of P and S boxes. The P box is a function that creates an 8 bit output from and 8 bit input. A fast and simple electro-technical de-vice, and it's inverse function can be easily generated if we know the P box's assignment rule. It is responsible for mixing the 8 bit and a creating a bit permutation.

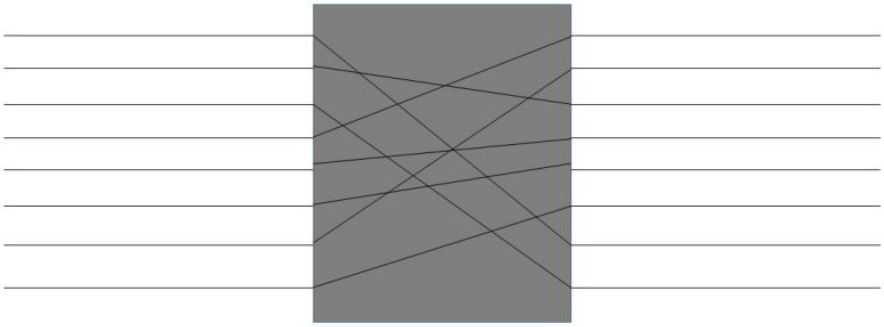


Figure 3: A possible P box

The S box is a device that implements a nonlinear function which creates 4 bit output from 6 bit input. [17] The operation of the S box is described by a table of 4 rows and 16 columns. Each S box has a different table. These tables allow us to encode the S box. Out of the incoming 6 bits the 1st and 6th gives the row indexes, while the other 4s decimal equivalent gives the column indexes. This way we get the 4 output bits based on the table cells.

	0000	**0001**	**0010**	**0011**	**0100**	**0101**	**0110**	**0111**	**1000**	**1001**	**1010**	**1011**	**1100**	**1101**	**1110**	**1111**
00	0010	1100	0100	0001	0111	1010	1011	0110	1000	0101	0011	1111	1101	0000	1110	1001
01	1110	1011	0010	1100	0100	0111	1101	0001	0101	0000	1111	1010	0011	1001	1000	0110
10	0100	0010	0001	1011	1010	1101	0111	1000	1111	1001	1100	0101	0110	0011	0000	1110
11	1011	1000	1100	0111	0001	1110	0010	1101	0110	1111	0000	1001	1010	0100	0101	0011

Figure 4: A possible S box

Shows the S function, which generates the memory content of the user $S(data)$, and transmits it to the reader. This is a complex function that contains S and P boxes. As we know the used tables of S and P boxes.

$$S^{-1}(S(data)) = data$$

Based on this we get back the data stored in the tags. The use of S and P boxes is defined by the reader's key. The reader is a specific computer which has the computing and storage capacity that is required for generating keys and

decrypting S(data). The tags are the electronic realizations of S and P boxes. For data control we create a digest from the stored data. The well-known HASH functions are suitable for solving this problem. We implement one of the HASH functions in the tags, eg.: SHA1 function. Using this method we are able to check the data after decrypting. This provides extra defense against data modifying, or data insertion attacks.

7. Conclusion

A product is exposed to many dangers during the way from the manufacturer to the costumer. From the factory it goes to a temporary storage, from there to a wholesale storage, from there to a retail distribution centre and finally the shelves of the supermarkets. This is a long process; during this the products can be lost, accidentally interchanged or stolen. EU regulations are pushing the responsibility of the manufacturer further and further so the traceability is becoming more and more important which can be implemented by RFID technology completely. During the production the way of the product is traceable, the technological orders, work phases and persons who took part in production or any other data. Use of RFID systems are changing nowadays. Many new technology are coming out, manufacturers and multinational companies are to carry these technologies to the users. Transponders come out from factories day by day cheaper and smaller; this helps their spread. Thanks to them being widespread, these systems can be found in more and more segments, so their vulnerability has to be considered scrutinously. Nowadays in Hungary reconciliation is in progress about mobile payment possibilities. Their spread might be a milestone for development. The users are not aware of the dangers of these. Most of them cannot or does not want to deal with this problem. So the manufacturers have to pay attention to the security, consider those possibilities that can cause any defect or data theft in systems surrounding users. Due to the decreasing production costs the data storage limit specific to cheap passive RFID systems is becoming irrelevant sooner or later. A move is in progress to active labels which can be used with more security and we don't have to create special algorithms in order to work on simple systems.

Acknowledgements. Tibor Radványi's research was supported by the European Union and the State of Hungary, co-financed by the European Social Fund in the framework of TÁMOP-4.2.4.A/ 2-11/1-2012-0001 "National Excellence Program".

The research of Csaba Biró, Sándor Király, Péter Szigetváry, Péter Takács was supported by the European Union and the State of Hungary, co-financed by the European Social Fund in the framework of the TAMOP-4.2.2.C-11/1/KONV-2012-0014, title: The developing possibilities of RFID/NFC technology by the conception of "Internet of Things".

References

[1] Klaus Finkezeller, RFID Handbook, *Third Edition*(2010).

[2] Ziv Kfir and Avishai Wool, Picking Virtual Pockets using Relay Attacks on Contactless Smartcard Systems, `http://eprint.iacr.org/2005/052.pdf`.

[3] Jeongkyu Yang, Jaemin Park, Hyunrok Lee, Kui Ren, Kwangjo Kim , KOMSCO, ICU, WPI Mutual Authentication Protocol for Low-cost RFID, 2005.

[4] Sebastien Canard, Iwen Coisel ,(Orange Labs RD, Caen, France) Data Synchronization in Privacy-Preserving RFID Authentication Schemes , 2008.

[5] Hee-Jin Chae, Daniel J. Yeager, Joshua R. Smith, and Kevin Fu , (University of Massachusetts) Maximalist Cryptography and Computation on the WISP UHF RFID Tag, 2007.

[6] Sindhu Karthikeyan and Mikhail Nesterenko, RFID Security without Extensive Cryptography, 2005.

[7] M. McLoone and M.J.B. Robshaw, Public Key Cryptography and RFID Tags, 2008.

[8] Markku-Juhani O. Saarinen, Daniel Engels A Do-It-All-Cipher for RFID: Design Requirements IACR Cryptology ePrint Archive 2012, 317.

[9] SAARINEN, M.-J. O., Cryptanalysis of Hummingbird-1., In FSE 2011 (2011), A. Joux, Ed., vol. 6733 of LNCS, Springer, pp. 328—341.

[10] KROVETZ, T., AND ROGAWAY, P, The software performance of authenticated-encryption modes. In FSE 2011 (2011), A. Joyx, Ed., vol. 6733 of LNCS, Springer, pp. 306—327.

[11] ENGELS, D., SAARINEN, M.-J. O., SCHWEITZER, P., AND SMITH, E. M., The Hummingbird-2 lightweight authenticated encryption algorithm. In RFID-Sec 2011 (2011), A. Juels and C. Paar, Eds., vol. 7055 of LNCS, Springer, pp. 19-–31.

[12] Biró Csaba, Radványi Tibor, Takács Péter, Szigetváry Péter, RFID rendszerek sebezhetőségének vizsgálata, MAFIOK 2013. ISBN: 978-963-358-035-6, 15–24 oldal.

[13] Radványi Tibor, Adatbiztonság az RFID alkalmazásakor, Acta Carolus Robertus 3(1) pp: 121–127, Gyöngyös, ISBN-978-963-269-201-2, 2012.

[14] Ernst Haselsteiner, Klemens Breitfuss, Security in Near Field Communication (NFC) Philips Semiconduc-torsMikronweg 1, 8101 Gratkorn, Austria.

[15] S. Yu, K. Ren, W. Lou, A privacy-preserving lightweight authentication protocol for low-cost RFID tags, in: IEEE MILCOM 2007, October 2007, pp. 1-–7.

[16] Y.-C. Lee, Y.-C. Hsieh, P.-S. You, T.-C. Chen, An improvement on RFID authentication protocol with privacy protection, in: Third International Conference on Convergence and Hybrid Information Technology – ICCIT 2008, vol. 2, November 2008, pp. 569-–573.

[17] Lauren De Meyer, Beg Bilgin, and Bart ,Extended Analysis of DES S-boxes, Proceedings of the 34rd Symposium on Information Theory in the Benelux, 30-31 May 2013, Leuven, Belgium, pp. 140–146.

[18] Vikram Belur Suresh,On-Chip True Random Number Generation, Thesis, 2012, University of Massachusetts Amherst, Department of Electrical and Computer Engineering.

Annales Mathematicae et Informaticae
44 (2015) pp. 165–176
http://ami.ektf.hu

Building of a mathematics-based RFID localization framework*

Zoltán Ruzsa[b], Zsolt Parisek[b], Roland Király[a]

Tibor Tómács[a], Tamás Szakács[a], Henrik Hajagos[a]

[a]Eszterházy Károly College, Institute of Mathematics and Informatics
kiraly.roland@ektf.hu, tomacs@ektf.hu, szakacstam@gmail.com,
hajagos.henrik@gmail.com

[b]Bay Zoltán Nonprofit Ltd. for Applied Research
ruzsaz@gmail.com, parisek@gmail.com

Submitted September 4, 2014 — Accepted March 5, 2015

Abstract

In this article, we examine problems related to RFID systems in which the antennas and the connected readers are able to find a transponder with greater precision and are able to locate their position with a greater probability.

The central problem is how to cover a relatively large area, such as an airport terminal or a railway station's waiting room, with the smallest possible number of RFID antennas.

The second important question of the research is what infrastructure and mathematical apparatus are required to decide the location of the transponder within an area under question, with the highest possibility.

1. RFID based localization and the problems of the presence sensors

Firstly, as part of localization problems and the possibilities of expanding perception, let us define what we can see with a perception sensor and what we mean by

*Their RFID research project was supported by the European Union and the State of Hungary, co-financed by the European Social Fund in the framework of TÁMOP-4.2.2.C-11/1/KONV-2012-0014.

an RFID based perception-sensor. This is important, because both the localization problem and the mathematical model are based on the examination of perception.

An RFID Presence sensor is a device capable of detecting the presence of a specific object and relaying this presence as binary information (whether it is present or not).

There are countless applications for sensors and sensor networks based on WIFI, RFID and NFC technologies in real life. Producing information regarding the position and motion of the perceived object by means of connected perception sensors (sensor network) has been the topic of a number of researchers [4, 5, 6, 7].

Several researches [8] also utilize the strength of the perceived signal to better determine the position of the object or a person. A good example of the situation underlined above is an office building where each door requires verification by an entrance card for passing.

The entrance card is, in reality, an RFID transponder, while the doors are outfitted with RFID readers.

These readers, along with their connected antennas measure whether the transponder is in their range of perception or not at preset time intervals. If the transponder is far away, the outcome is most likely negative, but positive reading are not always guaranteed even when the transponder is closer. However, the possibility of positive feedback is higher than in the previous case.

In case somebody is standing between the transponder and the antenna, his body water blocks the spread of the signal, resulting in a negative reading. Another possibility is that the RFID tag is situated on the periphery of the reception range. In this case, the outcome of the reading depends on other uncertainty figures such as humidity, temperature, etc.

Perception-based localization complicates the problem even further, as the currently used systems are not suited to handle this task.

The fact is that transponder localization is uncertain on a smaller area and the problems mentioned before soon make the use of current RFID based devices problematic.

We can safely conclude then with using only perception sensors and RFID based systems, it is relatively difficult to accurately decide the position of an object. However, we can see possibilities of expanding the perception capabilities of mentioned devices (RFID reader, antenna, antennas networks) with the implementation of intelligent algorithms and mathematics based computational models what these algorithms are based upon.

Problem (Extending of the RFID based localization)**.** *One of the central problems of our research is how to cover a relatively large area, such as an airport terminal or a railway station's waiting room, with the smallest possible number of RFID antennas, and to estimate the exact location of as many objects tagged with RFID tags as possible with the greatest likelihood. We would also like to find a solution to expand the perception process to permanently or temporarily uncovered areas also by upgrading current RFID technologies.*

As a part of the solution for the problem described above, we present the mathematical model used to expand the perception capabilities of the localization process along with the functioning of the software background, that is, the framework which provides estimations regarding the whereabouts of the transponders using the mathematics based subsystem.

2. Planning the measurement process and calibrating the test environment

In order to create and produce the mathematical model, and to prepare the controlling algorithms for the new antenna, we need to measure the characteristics of the antennas used in our test environment and the coverage of the test area by RFID readers.For the calibration of antennas and readers, we took measurements of the signal strength RSSI perceivable by the antennas within the discrete areas covered by each antenna, first by an accuracy of 50, then by 25 centimeters. Taking constructive and destructive interference into account, we created the map of coverage.

We have used similar measurements to determine the perception characteristics of the transponders what we used for localization (see in Figure 1).

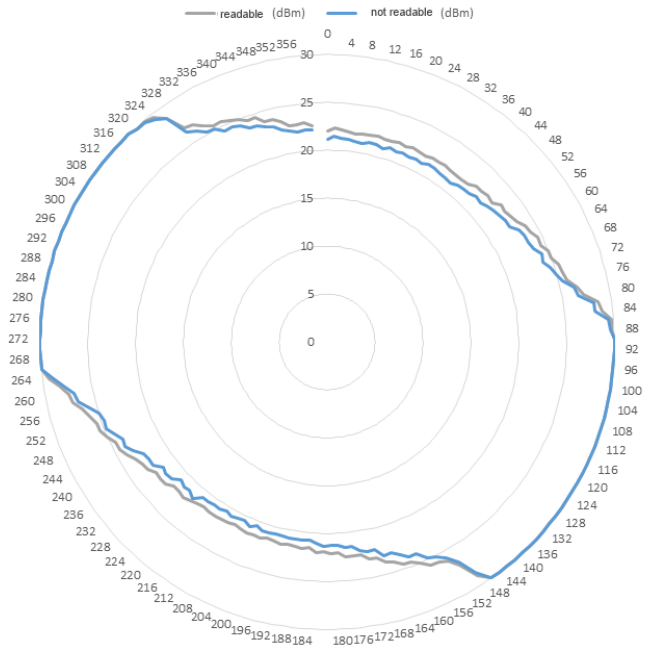


Figure 1: Characteristic of the rotated UHF transponder

Parallel to these, to construct a mathematical model, we rotated the transponder in a given position under the same conditions. For about 1000 times/position, we measured how many times the transponder was visible (sufficiently perceivable) to the reader connected to the antenna. Comparing the results to the RSSI values measured before proved that the reading ratio of the antennas influenced perception frequency. It is also clear that specific space segments show relatively unique values depending on reading frequency. We excluded the possibility of any obstruction between the antenna and the reader (for example: aqueous medium or the passing of a metal-based object).

It was under these circumstances that we had to design the mathematical apparatus responsible for predicting where a transponder (outside of visibility range) is located inside a known but yet uncovered area.

Provided that the mathematical model is working as intended, and that the system is able to estimate the most probable position of the tag, this extra information can be used to decide the real position of the transponder (in order to make the search attempts more efficient).

3. The mathematical foundations of the extension of localization

Let $(\Omega, \mathcal{F}, P)$ be a probability space where Ω is the set of factors that define the location of the tag or rather the outcome of the detection attempts made by antennas.

The Borel-measurable space part in which the tag should be located is denoted by $H \subset \mathbb{R}^k$ and let $H_1, H_2, \ldots, H_m$ be partitions of the Borel-measurable sets. These H_i sets are undivided sectors in which the presence of the tag should be detected. In practice $H \subset \mathbb{R}^2$ is a rectangle, sets of H_i are the cells of a grid.

Let $\theta_t \colon \Omega \to H$ denote a probability variable the position of an object at time moment t.

Let $X_{it} \colon \Omega \to \{0, 1\}$ probability variable ($i = 1, \ldots, n$, $t \geq 0$) be 1, if the antenna i detects the tag at time t be 0 if it does not. Assume, that $X_{1t_i} X_{2t_2}, \ldots, X_{nt_n}$ are independent of each other in case of arbitrary values $t_1, \ldots, t_n$. Let

$$p(i, j, 1) = P(X_{it} = 1 \mid \theta_t \in H_j) \quad \text{and} \quad p(i, j, 0) = 1 - p(i, j, 1),$$

where $i = 1, 2, \ldots, n$, $j = 1, 2, \ldots, m$ and $t \geq 0$. Assume that $p(i, j, 1)$ is independent from t, therefore the probability of antenna i detecting the tag is $p(i, j, 1)$ if it is located in the space part j, and it is $p(i, j, 0)$ if it does not. These values are not known, so in practice $p(i, j, 1)$ is replaced by $\hat{p}(i, j, 1)$, the relative frequency of the event $\{X_{it} = 1\}$ assuming that $\theta_t \in H_j$. Likewise $p(i, j, 0)$ is replaced by $\hat{p}(i, j, 0)$.

3.1. The case of simultaneous measurements

The most manageable, but unrealistic case is when all the antennas try to detect the tag at a time instant $t = t_0$. Let $x_1, x_2, \ldots, x_n$ be the measurement results.

Similar to the principle of maximum likelihood estimates, as the current estimated place of the object the sector H_j is accepted in which the achieved $x_1, x_2, \ldots, x_n$ measurements are the most likely to occur. Therefore the maximum point of the function

$$P(X_{1t_0} = x_1, X_{2t_0} = x_2, \ldots, X_{nt_0} = x_n \mid \theta_{t_0} \in H_j)$$

in j is required to be found. Due to the independence

$$P(X_{1t_0} = x_1, X_{2t_0} = x_2, \ldots, X_{nt_0} = x_n \mid \theta_{t_0} \in H_j) = \prod_{i=1}^{n} p(i, j, x_i).$$

Here the $p(i, j, x_i)$ values are not known, instead their $\hat{p}(i, j, x_i)$ estimation will be used. Thus, if the

$$\prod_{i=1}^{n} \hat{p}(i, j, x_i)$$

function has only one maximum point, j_0, then set H_{j_0} will be accepted as the estimated position of the object. (If the maximum value of the function is taken of the arguments of $j_1, j_2, \ldots, j_z$, then the estimation will be the union of the corresponding sets $H_{j_1}, \ldots, H_{j_z}$.)

3.2. The case of separate measurements made at various times

Take some measurements by sensor i at time moments $t_{i1}, t_{i2}, \ldots, t_{ik_i}$ close to time t_0.

As a result, the measured value recorded $X_{it_{i1}}, X_{it_{i2}}, \ldots, X_{it_{ik_i}}$ probability variables can not be considered independent therefore the results of the measurements by the antennas must be aggregated. Denote the results with $x_{it_{i1}}, x_{it_{i2}}, \ldots, x_{it_{ik_i}}$.

Assume that, there exists a $K : \mathbb{R} \to \mathbb{R}_+$ function, what

$$P\left(\bigcap_{l=1}^{k_i} \{X_{it_{il}} = x_{it_{il}}\} \;\middle|\; \theta_{t_0} \in H_j\right) = \frac{\sum_{l=1}^{k_i} K(t_0 - t_{il}) p(i, j, x_{it_{il}})}{\sum_{l=1}^{k_i} K(t_0 - t_{il})}.$$

Due to the independence

$$P\left(\bigcap_{i=1}^{n}\bigcap_{l=1}^{k_i} \{X_{it_{il}} = x_{it_{il}}\} \;\middle|\; \theta_{t_0} \in H_j\right) = \prod_{i=1}^{n} \frac{\sum_{l=1}^{k_i} K(t_0 - t_{il}) p(i, j, x_{it_{il}})}{\sum_{l=1}^{k_i} K(t_0 - t_{il})}.$$

Again, similar to the principle of maximum likelihood estimation the sector of H_j is accepted as the estimation of the location of the object at time t_0 where in case of being there the probability value of the test results is maximal. Therefore if the previously obtained

$$\prod_{i=1}^{n} \frac{\sum_{l=1}^{k_i} K(t_0 - t_{il}) p(i, j, x_{it_{il}})}{\sum_{l=1}^{k_i} K(t_0 - t_{il})}$$

function has its maximum value in j at j_0, the sector H_{j_0} will be accepted as the estimation of the object's location. (If the function has its maximum at $j_1, j_2, \dots j_z$ simultaneously at the same time, similarly to the previous cases the estimation of the object's location will be the union of sets $H_{j_1}, \dots H_{j_z}$).

In practical terms, the existence of an unification kernel K that can unify the measurements made by the same antenna at close time instants cannot be proven. What is even more unpleasant, K function above cannot even be determined by measurements. The rightness of core function of

$$K(t) = \begin{cases} \left(1 - \left(\frac{t}{\tau}\right)^2\right)^2 & \text{if } |t| < \tau, \\ 0, & \text{otherwise,} \end{cases}$$

used in the implementation can be supported only when using the calculated position proves to be sufficiently accurate. (The variable $\tau \geq 0$ appearing in the function is used to adjust the time interval from which measurements are used to determine the position of the object. This value is defined during the process of fine-tuning. In practical tests $\tau = 6$ was used.)

3.3. Probability of being in a given sector

Now we are looking for the answer to with what probability the tag was at time t_0 in a designated H_j sector. Assume that the tag could be in any sector with the same probability before the measurement, namely

$$P(\theta_t \in H_j) = P(\theta_t \in H_r) \qquad \forall j, r = 1, \dots, m.$$

The probability of

$$P\left(\theta_{t_0} \in H_j \;\middle|\; \bigcap_{i=1}^{n} \bigcap_{l=1}^{k_i} \{X_{it_{il}} = x_{it_{il}}\}\right)$$

should be determined. Based on the Bayes' theorem, this equals:

$$\frac{P\left(\bigcap_{i=1}^{n} \bigcap_{l=1}^{k_i} \{X_{it_{il}} = x_{it_{il}}\} \mid \theta_{t_0} \in H_j\right) P(\theta_{t_0} \in H_j)}{\sum_{r=1}^{m} P\left(\bigcap_{i=1}^{n} \bigcap_{l=1}^{k_i} \{X_{it_{il}} = x_{it_{il}}\} \mid \theta_{t_0} \in H_r\right) P(\theta_{t_0} \in H_r)} =$$

$$= \frac{\prod_{i=1}^{n} \frac{\sum_{l=1}^{k_i} K(t_0 - t_{il}) p(i,j,x_{it_{il}})}{\sum_{l=1}^{k_i} K(t_0 - t_{il})}}{\sum_{r=1}^{m} \prod_{i=1}^{n} \frac{\sum_{l=1}^{k_i} K(t_0 - t_{il}) p(i,r,x_{it_{il}})}{\sum_{l=1}^{k_i} K(t_0 - t_{il})}} \approx \frac{\prod_{i=1}^{n} \frac{\sum_{l=1}^{k_i} K(t_0 - t_{il}) \hat{p}(i,j,x_{it_{il}})}{\sum_{l=1}^{k_i} K(t_0 - t_{il})}}{\sum_{r=1}^{m} \prod_{i=1}^{n} \frac{\sum_{l=1}^{k_i} K(t_0 - t_{il}) \hat{p}(i,r,x_{it_{il}})}{\sum_{l=1}^{k_i} K(t_0 - t_{il})}}.$$

3.4. The implementation

For testing a virtual test environment has been created. The basic set in which the localization of the tag was carried out was considered a square divided into 10×10 grid. These squares in the grid become the sets H_j.

For the sake of simplicity the elementary grid squares are indexed by $\{0,1,\ldots,9\}$ $\times\{0,1,\ldots,9\}$ as their coordinates. Three virtual antennas were created, A_1, A_2, A_3, and probabilities $p(1,j,1)$, $p(2,j,1)$, $p(3,j,1)$ were defined being associated with their corresponding antennas ($j \in \{0,1,\ldots,9\} \times \{0,1,\ldots,9\}$). Figure 2 illustrates this.

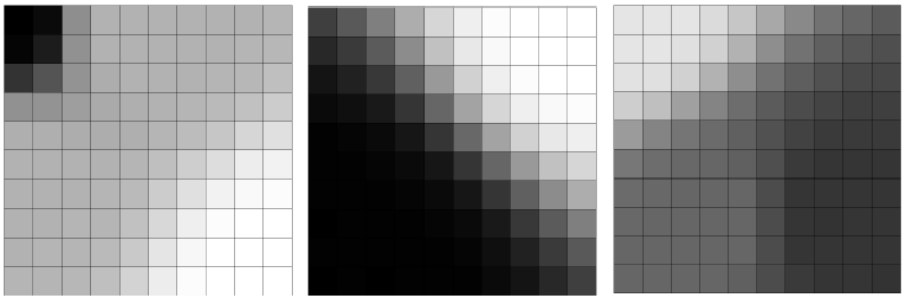


Figure 2: The detection probability of antennas A_1, A_2, A_3. The probability is 1 in the black squares and 0 in the white ones

3 testroutes are generated that represent the roaming of the tag (see Figure 3). All 3 testroutes take 30 seconds and considering that the 3 antennas try to read the tag in every second, so $t \in \{0,1,\ldots,29\}$.

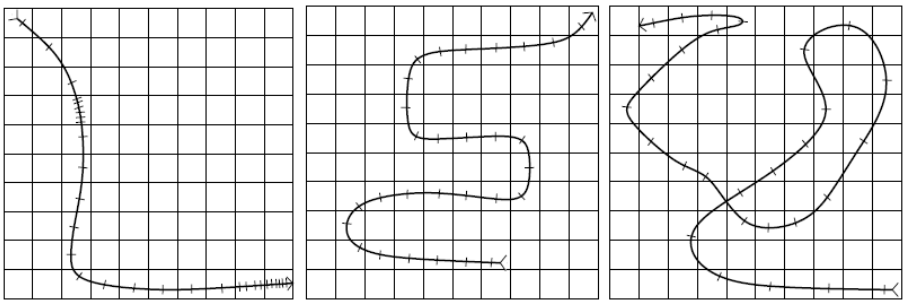


Figure 3: The three virtual test routes (test tracks). The notches indicate the position of the tag in integer minutes

After each detection attempt when the antenna A_i tried to read the tag at time t the position (x_t, y_t) of the tag during the route at time t was checked against the associated probability of $p(i,(x_t,y_t),1)$ and a random p number with uniform distribution has been chosen from the interval of $[0,1]$ and the

$$x_{it} = \begin{cases} 1, & \text{if } p < p(i,(x_t,y_t),1), \\ 0, & \text{otherwise,} \end{cases}$$

virtual series of measurement has been created. The reading results of the antennas

during the traversal path were simulated this way. For the resulting sets of measurements x_{it} ($i \in \{1,2,3\}$, $t \in \{0,1,\ldots,29\}$) the positioning algorithm outlined above was performed, and the estimated $(\hat{x_t}, \hat{y_t})$ coordinates by the algorithm were checked how accurately they approach the tag's exact coordinates of (x_t, y_t).

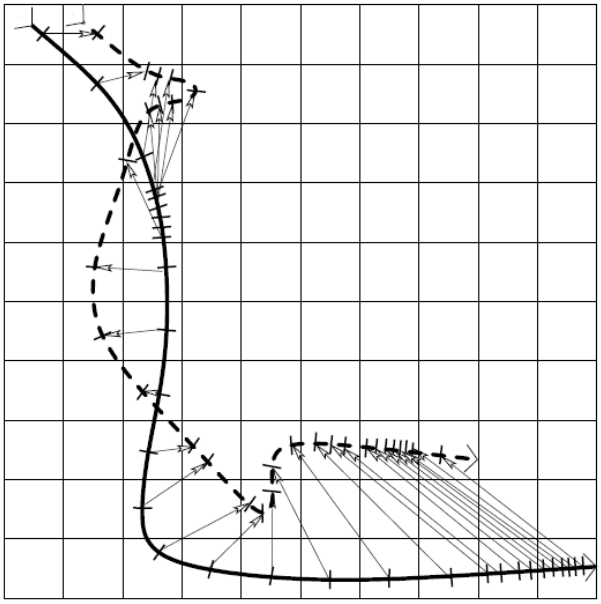


Figure 4: The route 1 (continuous curve) and a characteristics estimation (dotted line). Actual and estimated position in the same time moments are connected by arrows. Squared error for the estimate is 5.8

To quantify the results the squared error of the route estimate

$$\frac{1}{30}\sum_{t=0}^{29}\left((x_t-\hat{x_t})^2+(y_t-\hat{y_t})^2\right)$$

has been calculated. The method above was repeated 1000 times and the average of the square errors was recorded. The considered values during the route can be seen in the Table 1. As a reference the squared errors of the random guess (uniform distribution independently considered (x_t, y_t) coordinates) and the constant estimation $(5,5)$ were also determined.

The results show that the algorithm works best when the path is close to the antennas and the movement of the tag is slow (Route 1) while the detection is less accurate when the the tag is roaming in the middle of the area covered by the antennas (Route 2). Not surprisingly, the estimate of constant $(5,5)$ provides a lesser error in this case.

	route 1	route 2	route 3
squared error of algorithm	6.953	15.707	10.732
squared error of random estimate	38.947	27.893	32.351
squared error of constant estimate(5,5)	21.500	12.500	17.900

Table 1: Squared errors of the three route estimates

3.5. The localization framework

We have two large expectations, regarding the localization framework: One being for it to be able to provide an interface for the middlewares, that control the antennas, on which they can relay their detections to the system. It is expected to be able to store the information, gathered using this method, permanently.

The other being, that the system should be able to calculate the coordinates of the transponder (or maybe even a moving reader in the future), that is fixed in the system. These calculations are based on the previously stored detections.

As a solution for the aforementioned expectations, we have implemented a web-service, coded in Java programming language. Designing and implementing of the webservice was done while abiding the policies of REST (Representational State Transfer) [10]. We have chosen JBoss (Jboss AS 7.1.0.Final) [11] as our service's application-server, for it is strongly supported by REST. For the forming of the interfaces, we employed the javax.ws.rs [12] (High-level interfaces and annotations used to create RESTful service resources) package, introduced by Java EE 6, making the usage of all these independent from language and platform. This property is nearly indispensable for a system capable of integrating heterogeneous hardware, like this one.

We can upload a detection to the framework, by employing a simple HTTP-POST [13] method, assuming that the detection we would like to upload is written in a specified XML structure. The first step of processing an uploaded detection is the serialisation of the given xml, which will result in the creation of a Detection objectitem. The Detection class indicates between which transponder and reader did the detection occur, and optionally, through which RSSI value did the communication between the two happening. Among the Detection items exists a special item, which represents negative detection. The middlewares present a negative detection when the reader attempts to locate the transponders in their environment, but no answer is given to it.

The permanent storing of the Detection items is done inside a database, in our case, this is a PostgreSQL 9.3 [14] released database-item. Communication between the database and the server application is accomplished by the help of an ORM (Object/Relational Mapping) [15] tool, which makes our job a lot more easier, during the upgrading of the framework. Our choice for an ORM tool fell upon Hibernate (Hibernate ORM 4.3.1. Final Release), for the reason that JBoss AS 7 provides native support in case of using Hibernate ORM.

Further expectations of the localization framework include, for it to be able

to make calculations using different localization algorithms, simultaneously. This property of the system allows the upgrading of the localization algorithms to be quick and simple, in parallel with different project team's different expectation systems, if need be.

3.6. The software component from client side

The task of the client side control is to communicate with the RFID reader then transmit the acquired data to the server. To incorporate the functions of the reader, we have developed an interface, that is able to convert the parameterized function calls into XML structures that are compatible with the reader, deliver them, and transform the given answer into a serviceable form, then return it to the location at which the call was made.

Communication with the server is also done by an interface, that is similar to the one before, having the task of converting the given data into an XML structure that is compatible with the server then processing the answer returned to it by the server.

Along with keeping up the connection with the server, the client also serves as a controlling role a controlling function, for it is able to work with more than one antenna, that are connected to the reading mechanism, simultaneously, while also having the ability to clearly identify them too. This way, we can transmit information regarding from which antenna the data came from. On top of all this, (with the help of this), we can also monitor the "GPIO status" of the nodding antennas (what kind of status was the given antenna in last time), because the different statuses are handled with different kinds of antenna identifications. (In order to conduct indoor localization, we are using a certain type of intelligent antenna, which can change the antenna characteristics, by determining the current status of the transponders and using three integrated patch antennas. The antenna is able to locate the transponders in a given location, by turning into the appropriate direction (without mechanical movement).

The client decides exactly which reader, and which antenna should read with what kind of specifications, while during runtime by switching between these, it also governs the scanning of the given location. If necessary, it is able to modify the signal strength of the reading, the reading modes, and controls the GPIO ports of the antennas.

4. Conclusion

We can solve a lot of problems that affect or inhibit the use of RFID by extending the software in this direction and by the application of mathematical estimate, and we can give proper answer to the problem, what has been presented previously in this article.

In our opinion the system constructed by us can be used to the extension of localization processes based on RFID, and we can achieve a goal presented in the

introduction, so we can cover quite large areas by using a smaller amount of readers, and this way we can use much less resources and smaller expenses.

With the help of the above stated model we can estimate the position of those transponders that are temporarily uncovered, but they are known to be positioned within the examined area. The implemented version can decide where most probably the object to be localized is, among the temporarily or permanently uncovered discreet areas.

The arguments and the implemented algorithms presented by us help the development and realization of the IOT (Internet of Things) conception since we will be able to identify the devices and their positions as well as the routes that they have covered.

References

[1] Bánlaki, J., Hoffmann, M., Juhász, T., UHF RFID tags with user interface – the ability of individual control and a new source of information, *RFID Journal, 2014.*

[2] Intelliflex On-Demand Data Visibilty – Intelliflex Readers, `http://www.intelleflex.com/Products.asp`

[3] Intermec By Honeywell, `http://www.intermec.com/`

[4] Qiang Le and Lance M Kaplan., Design of operation parameters to resolve two targets using proximity sensors. *In Information Fusion (FUSION), 2010 13th Conference on, pages 1–8. IEEE, 2010.*

[5] Qiang Le and Lance M Kaplan., Target localization using proximity binary sensors. *In Aerospace Conference, 2010 IEEE, pages 1–8. IEEE, 2010.8*

[6] J Miguez and A Artes-Rodriguez., Monte carlo algorithms for tracking a maneuvering target using a network of mobile sensors. *In Computational Advances in Multi-Sensor Adaptive Processing, 2005 1st IEEE International Workshop on, pages 89–92. IEEE, 2005.*

[7] Ruixin Niu and P Varshney., Target location estimation in wireless sensor networks using binary data. *In Proceedings of the 38th Annual Conference on Information Sciences and Systems, Princeton, NJ, 2004.*

[8] Ruşen Öktem and Elif Aydin., An rfid based indoor tracking method for navigating visually impaired people. *Turk J Elec Eng and Comp Sci, 18(2),2010.*

[9] Mahesh Vemula, Joaquín Miguez and Antonio Artes-Rodriguez., A sequential monte carlo method for target tracking in an asynchronous *Positioning, Navigation and Communication, 2007. WPNC '07. 4th Workshop on, 22-22 March 2007, pages.: 49 - 54, E-isbn: 1-4244-0871-7, Printed ISBN: 1-4244-0871-7. Location.: Hannover*

[10] The REST protocoll - Representational State Transfer. `https://www.ics.uci.edu/~fielding/pubs/dissertation/rest/`

[11] JBoss AS 7.1 `http://jbossas.jboss.org/`

[12] Java EE 6 javax.ws.rs Package High-level interfaces and annotations used to create RESTful service resources http://docs.oracle.com/javaee/6/api/javax/ws/rs/package-summary.html

[13] Hypertext Transfer Protocol http://www.w3.org/Protocols

[14] PostgreSQL 9.3 http://www.postgresql.org/about/news/1481/

[15] Object/Relational Mapping http://hibernate.org/orm/

Annales Mathematicae et Informaticae
44 (2015) pp. 177–185
http://ami.ektf.hu

Optimization of coefficients of lists of polynomials by evolutionary algorithms

J. Rafael Sendra[a], Stephan M. Winkler[b]

[a]University of Alcalá, Department of Physic and Mathematics
Rafael.Sendra@uah.es

[b]Upper Austria University of Applied Sciences,
Heuristic and Evolutionary Algorithms Laboratory
Stephan.Winkler@fh-hagenberg.at

Submitted July 30, 2014 — Accepted December 12, 2014

Abstract

We here discuss the optimization of coefficients of lists of polynomials using evolutionary computation. The given polynomials have 5 variables, namely t, a_1, a_2, a_3, a_4, and integer coefficients. The goal is to find integer values α_i, with $i \in \{1, 2, 3, 4\}$, substituting a_i such that, after crossing out the gcd (greatest common divisor) of all coefficients of the polynomials, the resulting integers are minimized in absolute value. Evolution strategies, a special class of heuristic, evolutionary algorithms, are here used for solving this problem. In this paper we describe this approach in detail and analyze test results achieved for two benchmark problem instances; we also show a visual analysis of the fitness landscapes of these problem instances.

Keywords: Optimization of parametrizations, symbolic computation, evolutionary computation, evolution strategies.

MSC: 65K10, 68T05, 68W30

1. Problem statement

In this section, trying to avoid as much as possible mathematical technicalities, we describe the problem, and we explain its interest in the field of mathematics.

The problem statement. We are given a list with infinitely many (at least 3) non-constant polynomials ($L = [p_1; p_2; \ldots; p_n]$). These polynomials have 5 variables, namely t, a_1, a_2, a_3, a_4, and integer coefficients. The problem consists in finding integer values α_1, α_2, α_3, and α_4 for a_1, a_2, a_3, and a_4 such that:

1. $\alpha_1\alpha_4 - \alpha_2\alpha_3 \neq 0$
2. We substitute $a_1 = \alpha_1$, $a_2 = \alpha_2$, $a_3 = \alpha_3$, $a_4 = \alpha_4$, in L. This yields $L'(\alpha)$, a list of polynomials with one variable, namely t, and integer coefficients. We cross out the greatest common divisor (gcd) of all non-zero coefficients of the polynomials in $L'(\alpha)$ to get a new list $L''(\alpha)$.

The goal is to find that substitution $a_i = \alpha_i$, $i \in \{1, 2, 3, 4\}$, so that the maximum of the absolute values of all the coefficients of all polynomials in $L''(\alpha)$ is minimum.

An illustrating example. We consider the list with three polynomials $L = [p_1; p_2; p_3]$ where

$$\begin{aligned}
p_1(t) &= 13923t^2{a_1}^2 + 5474t^2a_1a_3 - 1904t^2{a_3}^2 + 27846ta_1a_2 + 5474ta_1a_4 \\
&\quad + 5474ta_2a_3 - 3808ta_3a_4 + 13923{a_2}^2 + 5474a_2a_4 - 1904{a_4}^2 \\
p_2(t) &= 7564t^2{a_1}^2 - 10298t^2a_1a_3 - 990t^2{a_3}^2 + 15128ta_1a_2 - 10298ta_1a_4 \\
&\quad - 10298ta_2a_3 - 1980ta_3a_4 + 7564{a_2}^2 - 10298a_2a_4 - 990{a_4}^2 \\
p_3(t) &= 15845t^2{a_1}^2 - 106t^2a_1a_3 + 2146t^2{a_3}^2 + 31690ta_1a_2 - 106ta_1a_4 \\
&\quad - 106ta_2a_3 + 4292ta_3a_4 + 15845{a_2}^2 - 106a_2a_4 + 2146{a_4}^2
\end{aligned}$$

If we substitute $a_1 = 1, a_2 = 1, a_3 = 1, a_4 = 1$ we get

$$L'(\alpha) = [17493t^2 + 34986t + 17493; -3724t^2 - 7448t - 3724; 17885t^2 + 35770t + 17885]$$

Since $\gcd(17493, 34986, 17493, -3724, -7448, -3724, 17885, 35770, 17885) = 49$, one gets $L''(\alpha) = 1/49L'(\alpha)$, that is

$$L''(\alpha) = [357t^2 + 714t + 357; -76t^2 - 152t - 76; 365t^2 + 730t + 365]$$

and the maximum, in absolute value, is 730. However, if we take $a_1 = 45, a_2 = 11, a_3 = 31, a_4 = -122$ the new list is

$$L'(\alpha) = [34000561t^2 - 34000561; 68001122t; 34000561t^2 + 34000561].$$

The corresponding gcd is now 34000561. Therefore

$$L''(\alpha) = \frac{1}{34000561}L'(\alpha) = [t^2 - 1; 2t; t^2 + 1]$$

whose maximum, in absolute value, is 2.

The mathematical origin of the problem. This optimization question, we are dealing with, comes from a central problem in the field of the symbolic computation of algebraic curves (see [6] for further details), appears in many computational aspects of the practical applications of curves and is, to our knowledge, not solved. Let us first motivate the problem: In many practical applications, such as in computed aided geometric design, in physics, etc., one deals with parametric representations of a curve. For instance, if we have to compute a line integral along an arc of the curve of equation $y^3 = x^2$, we might use the parametric representation $x = t^3, y = t^2$ of the curve. In general a rational parametrization of a curve, say for simplicity planar, is a nonconstant pair

$$\left(\frac{p_1(t)}{q(t)}, \frac{p_2(t)}{q(t)}\right)$$

where p_1, p_2, q are polynomials in the variable t. The difficulty here is the following: If we replace t by a polynomial or by a rational function, then we get another parametrization of the same object; for instance, in the example above, $(1000t^3, 100t^2)$ and $((t^2+1)^3, (t^2+1)^2)$ are also parametrizations of $y^3 = x^2$. Thus, we have infinitely many possibilities, but some parametrizations are more complicated and increase the computational time when using them. The question is how to choose the simplest parametrization. Achieving an optimal degree in the polynomials is solved by means of symbolic deterministic algorithms (see [6]). However, the question of determining a parametrization with the smallest (in absolute value) integer coefficients is open. Here, in this paper, we show how to approach the problem by means of evolutionary algorithms.

In order to translate the original parametrization problem into the the problem stated above, we use Lüroth's theorem that establishes how all parametrizations, with optimal degree, are related. More precisely, if $\mathcal{P}(t) = (\frac{P_1(t)}{Q(t)}, \frac{P_2(t)}{Q(t)})$ is a parametrization with optimal degree and integer coefficients, then all the other parametrizations with optimal degree and integer coefficients are of the form

$$\mathcal{P}\left(\frac{a_1t+a_2}{a_3t+a_4}\right) = \left(\frac{P_1\left(\frac{a_1t+a_2}{a_3t+a_4}\right)}{Q\left(\frac{a_1t+a_2}{a_3t+a_4}\right)}, \frac{P_2\left(\frac{a_1t+a_2}{a_3t+a_4}\right)}{Q\left(\frac{a_1t+a_2}{a_3t+a_4}\right)}\right),$$

where a_1, a_2, a_3, a_4 are integers such that $a_1a_4 - a_2a_3 \neq 0$. Simplifying this expression, we get the three polynomials in the variables t, a_1, a_2, a_3, a_4.

Revisiting the illustrating problem. We are given the parametrization

$$\mathcal{P}(t) = \left(\frac{13923\,t^2 + 5474\,t - 1904}{15845\,t^2 - 106\,t + 2146}, \frac{7564\,t^2 - 10298\,t - 990}{15845\,t^2 - 106\,t + 2146}\right)$$

Performing the formal substitution $t = \frac{a_1t+a_2}{a_3t+a_4}$, simplifying expressions and collecting numerators and denominators in a list we get the list $L = [p_1; p_2; p_3]$ of the three

polynomials shown above. Now, after taking $a_1 = 45, a_2 = 11, a_3 = 31, a_4 = -122$, we get the parametrization

$$\mathcal{P}\left(\frac{45t+11}{31t-122}\right) = \left(\frac{t^2-1}{t^2+1}, \frac{2t}{t^2+1}\right).$$

The parametrization in this example corresponds to the unit circle $x^2 + y^2 = 1$.

2. Parameter optimization by evolutionary algorithms

Evolution strategies (ES; [4], [5]), beside genetic algorithms (GA; [2], [1]) the second major representative of evolutionary computation, are here used for optimizing $\alpha_1, \ldots, \alpha_4$. ES are population based, i.e., each optimization process works with a population of potential solution candidates that are initially created randomly and then iteratively optimized. In each generation, new solution candidates are generated by randomly selecting parent individuals and forming new individuals applying mutation and (optionally) crossover operators; λ children are produced by μ parent individuals.

By offspring selection, the best children are chosen and become the parents of the next generation. Typically, parent selection in ES is performed randomly with no regard to fitness; survival in ESs simply saves the μ best individuals, which is only based on the relative ordering of their fitness values. Basically, there are two selection strategies for ESs:

- The (μ, λ)-strategy ("comma selection"): μ parents produce λ children; the best μ children are selected and form the next generation's parents.

- The $(\mu+\lambda)$-strategy ("plus selection"): μ parents produce λ offspring; parents and children form a pool of potential new parents, and the best μ individuals are selected from this pool to become the next generation's parents.

Thus, the main driving forces of optimization in ESs are offspring selection and mutation. For the optimization of vectors of real values, mutation is usually implemented as additive Gaussian perturbation with zero mean or multiplicative Gaussian perturbation with mean 1.0. Mutation strength control [4] is based on the quotient of the number of the successful mutants (i.e., those that are better than their parents): If this quotient is greater than $1/5$, then the mutation variance is to be increased; if the quotient is less than $1/5$, the mutation variance should be reduced.

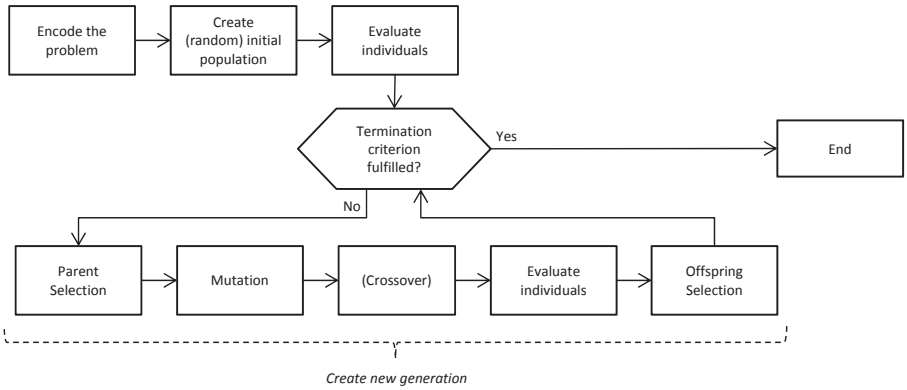


Figure 1: The main workflow of an evolution strategy

3. Test series

3.1. Problem instances

We have used the following two test instances:

- The example (*Ex1*) introduced in Section 1
- The second example (*Ex2*) is defined as $L2 = [p1; p2; p3]$ with

$$\begin{aligned} p_1 &= 1685t^2{a_1}^2 + 2252t^2a_1a_3 + 769t^2{a_3}^2 + 3370ta_1a_2 + 2252ta_1a_4 \\ &\quad + 2252ta_2a_3 + 1538ta_3a_4 + 1685{a_2}^2 + 2252a_2a_4 + 769{a_4}^2 \\ p_2 &= -627t^2a1^2 - 1148t^2a_1a_3 - 481t^2{a_3}^2 - 1254ta_1a_2 - 1148ta_1a_4 \\ &\quad - 1148ta_2a_3 - 962ta_3a_4 - 627{a_2}^2 - 1148a_2a_4 - 481{a_4}^2 \\ p_3 &= 2467t^2{a_1}^2 + 3235t^2a_1a_3 + 1069t^2{a_3}^2 + 4934ta_1a_2 + 3235ta_1a_4 \\ &\quad + 3235ta_2a_3 + 2138ta_3a_4 + 2467{a_2}^2 + 3235a_2a_4 + 1069{a_4}^2 \end{aligned}$$

  For this example, taking $\alpha_1 = -25, \alpha_2 = 12, \alpha_3 = 34, \alpha_4 = -23$ (note that $\alpha_1\alpha_4 - \alpha_2\alpha_3 = 167 \neq 0$) we get

$$L' = [27889t^2 + 27889; 27889t^2 - 27889; 27889t^2 + 27889t + 27889].$$

  The gcd is 27899 and $L'' = [t^2 + 1; t^2 - 1; t^2 + t + 1]$ and the maximum of the absolute values is 1, which is clearly optimal.

3.2. Algorithm configurations

The following 10 algorithm variants have been used for solving the problems defined in the previous section:

	Population size (μ)	Number of children (λ)	Selection mechanism
Settings 1	100	1,000	comma
Settings 2	100	10,000	comma
Settings 3	100	1,000	plus
Settings 4	100	10,000	plus
Settings 5	1,000	10,000	comma
Settings 6	1,000	100,000	comma
Settings 7	1,000	10,000	plus
Settings 8	1,000	100,000	plus
Settings 9	10,000	100,000	comma
Settings 10	10,000	100,000	plus

Table 1: Algorithm parameter settings used for solving the here discussed coefficients optimization problem.

The range of values for initial solution candidates was set to ± 200. For creating offspring we have used multiplicative mutation: The average value of the multiplication factors μ was set to 1.0, the standard deviation σ was initially set to 1.0 and according to the 1/5 success rule updated after each generation (with multiplicative factor / divisor 0.9).

3.3. Results

We have executed ES test series using all parameter configurations defined previously; each algorithm configuration was executed 5 times independently, and for guaranteeing a fair comparison of results the maximum number of evaluations used as termination criterion was set to 1,000,000. Thus, the number of generations executed was not equal for all test configurations.

The results achieved in these test series are summarized in Table 2.

We see that the success rate for small populations is very low, when using bigger populations (with size 1,000 or 10,000) the results are significantly better; when using populations of size 1,000, then significantly better results are achieved using higher selection pressure, i.e. selecting the 1,000 best out of 100,000 offspring each generation.

Problem *Ex1* seems to be harder for the algorithm than *Ex2*. For *Ex1* the algorithm was able to find the optimal solution at least once using settings 8 and 10; for *Ex2* the algorithm was successful in finding the optimum in 4 or 5 out of 5 runs using the settings 6, 8, 9, and 10.

	Problem instance *Ex1*	Problem instance *Ex2*
Settings 1	4807.4	483.6
Settings 2	1270.6	402.2
Settings 3	2136.6	671.0
Settings 4	438.2	3.8
Settings 5	854.4	110.6
Settings 6	160.0	1.0
Settings 7	120.4	21.2
Settings 8	58.2	1.2
Settings 9	230.8	1.4
Settings 10	35.4	1.6

Table 2: Test results. For each algorithm configuration we give the average result qualities achieved for problem instances *Ex1* and *Ex2*.

Fitness Landscape analysis [3] methods can be used for estimating an optimization problem's hardness. As we see in Figures 2 and 3, the fitness landscape of the here used problem instances *Ex1* and *Ex2* are very rugged, which makes it very hard for optimization algorithms to find optimal solutions.

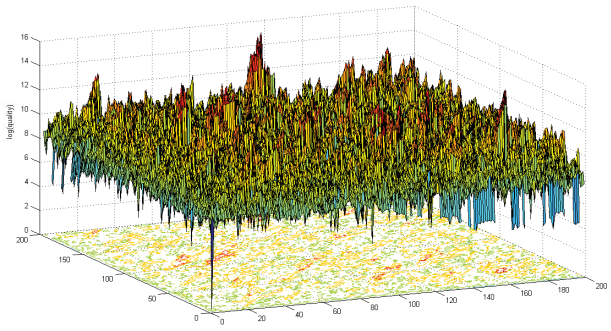


Figure 2: Fitness landscape analysis for example *Ex1*. We have created 40,000 solution candidates for *Ex1* that are arranged on the x-y-plane; the optimal solution discussed in Section 1 ($a_1 = 45, a_2 = 11, a_3 = 31, a_4 = -122$) is positioned at (1,1), and at all other cells are assigned solution candidates that are produced by mutating one of their neighbors (using $\sigma = 1.0$). On the z-axis we draw the fitness of the so created solution candidates for *Ex1*. We see high fluctuations of the fitness values which indicates that fitness values of neighboring solutions vary significantly.

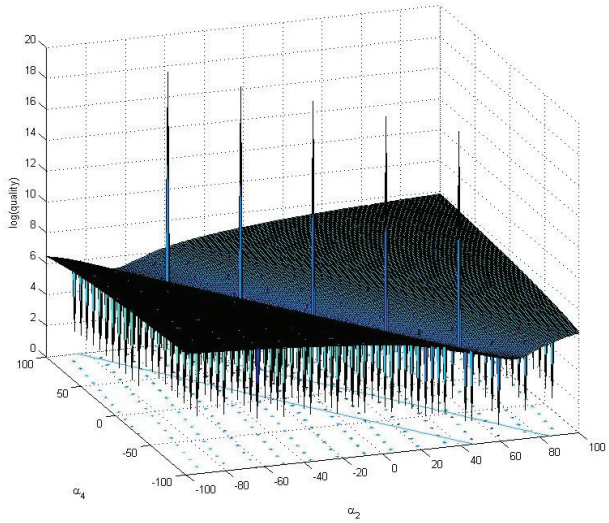


Figure 3: Fitness landscape analysis for example *Ex2*. All possible solution candidates for the here used problem with α_1 and α_3 set optimally ($\alpha_1 = -25$, $\alpha_3 = -34$) are created, their fitness is drawn on the z-axis. We see that even when setting two of four parameters optimally, the resulting fitness landscape is very rugged.

4. Conclusion, outlook

Future work will concentrate on the improvement of mutation and selection operators for this problem class in order to solve problem instances involving significantly bigger coefficients. Additionally, we are working on strategies to decrease the search space. We are also working on the integration of the here discussed class of problems in HeuristicLab [7], a framework for heuristic and evolutionary algorithms that is developed by members of the Heuristic and Evolutionary Algorithms Laboratory (HEAL).

Acknowledgements. The authors thank Franz Winkler at the Research Institute for Symbolic Computation, Johannes Kepler University Linz, for his advice. R. Sendra is partially supported by the Spanish Ministerio de Economía y Competitividad under the project MTM2011-25816-C02-01 and is a member of the Research Group ASYNACS (Ref. CCEE2011/R34). The authors also thanks members of the Heuristic and Evolutionary Algorithms Laboratory as well as of the Bioinformatics Research Group, University of Applied Sciences Upper Austria, for their comments.

References

[1] M. Affenzeller, S. M. Winkler, S. Wagner, and A. Beham. *Genetic Algorithms and Genetic Programming - Modern Concepts and Practical Applications*. Chapman & Hall / CRC, 2009.

[2] J. H. Holland. *Adaption in Natural and Artifical Systems*. University of Michigan Press, 1975.

[3] E. Pitzer. *Applied Fitness Landscape Analysis*. PhD thesis, Institute for Formal Models and Verification, Johannes Kepler University Linz, 2013.

[4] I. Rechenberg. *Evolutionsstrategie*. Friedrich Frommann Verlag, 1973.

[5] H.-P. Schwefel. *Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie*. Birkhäuser Verlag, Basel, Switzerland, 1994.

[6] J. R. Sendra, F. Winkler, and S. Perez-Díaz. *Rational Algebraic Curves: A Computer Algebra Approach*. Algorithms and Computation in Mathematics. Volume 22. Springer-Verlag Heidelberg, 2007.

[7] S. Wagner, G. Kronberger, A. Beham, M. Kommenda, A. Scheibenpflug, E. Pitzer, S. Vonolfen, M. Kofler, S. Winkler, V. Dorfer, and M. Affenzeller. *Advanced Methods and Applications in Computational Intelligence*, volume 6 of *Topics in Intelligent Engineering and Informatics*, chapter Architecture and Design of the HeuristicLab Optimization Environment, pages 197–261. Springer, 2014.

Annales Mathematicae et Informaticae
44 (2015) pp. 187–196
http://ami.ektf.hu

Real time human activity monitoring*

József Sütő, Stefan Oniga, Attila Buchman

University of Debrecen
Faculty of Informatics
suto.jozsef@inf.unideb.hu

Submitted August 28, 2014 — Accepted January 24, 2015

Abstract

Human activity monitoring is one of those research areas whose importance and popularity have notably increased in recent years. The popularity of this topic increased in the previous years. Most of the used movement analysis techniques in the area are based on the measurement of the acceleration change of different parts of the body. This is done by attaching one or more little devices with an accelerometer to the body of the observed patient. Usually, the role of the body-attached devices are only data acquisition, the processing of the acquired data happens offline. This article presents a new solution for this task which combines digital time-frequency signal processing with a parallel programming approach.

Keywords: movement analysis, Raspberry Pi, accelerometer, signal processing, parallel programming

MSC: 92C50

1. Introduction

Accelerometer-based activity monitoring devices are becoming more popular. By activity monitoring, we can obtain information about the health and mental status of the observed person. Many articles deal with the possibilities of the information which comes from the accelerometers [1-6]. Those studies demonstrate that movement classification is possible by accelerometers. Sometimes the sensors are attached to the fix points of the body: chest, hip, etc. In this case, the center of

*The publication was supported by the TÁMOP-4.2.2.C-11/1/KONV-2012-0001 project. The project has been supported by the European Union, co-financed by the European Social Fund.

the observation is the whole body and one sensor is enough [1, 2]. In other cases, when we examine walking patterns or postures, the sensors are attached to different parts of the body: head, trunk, foot, tibia [3, 4, 5, 6].

Activity analysis is a very extended research area. It includes the monitoring of patients who suffer from fall, back pain, overweight, physiological tremors, mental disorders and other kinds of diseases (Parkinson disease, osteoarthritis) [2, 4, 6]. Another research field of this topic is movement pattern recognition. It includes the main activities of daily living: sitting, standing, walking, jogging, running and climbing stairs. Beyond the main activities, the importance of the unexpected events such as different kinds of fall rises at elderly patients [7, 8, 9, 10]. The movement analysis can help support elderly people in their everyday life. The number of ageing population is rapidly growing thus the demand of assisted living systems will gradually increase [1].

The aim of this study is to present a method of activity classification, and in addition, to create a real time system which monitors the daily activities of an observed patient. The observed patient should wear the device during the day. Therefore, the device have to be portable and small. One of the most important criteria of the device is the cost. Since the system consists of only one data collector device, the cost of the system is low. Obviously, more data collector devices and sensors require higher cost. Another criteria is independence. This means that hospital environment is not necessary to the observation. The system facilitates self-care and enhances the independence of the patients against the public health systems.

The data collector device is a Raspberry Pi (RPi) with an ADXL345 (3-axis) accelerometer and a Roving RN-171 WIFI module. The device was fixed into a thin plate which provides stable position on the chest. There is a Linux operation system on the RPi, therefore the developer can easily create high level programs. Figure 1 shows the constructed device.

The disadvantage of most analysis techniques is the offline mode [1-6]. In this case, the analysis take place after the data acquisition on a computer by some well known software (Matlab, Labview, etc.). In contrast, our method works in real time. This is the first difference between previous techniques and our method. The same device performs the data acquisition and data evaluation simultaneously, the two tasks run in parallel on the device.

In our system an individual pattern recognition technique was applied for this problem which works better than the correlation or artificial neural network based pattern recognition techniques [11]. This simple technique runs an ideal pattern through the time varying signal and calculates the shifted and summarised square error (SSE). Every activity has an own rhythm which describes a periodical pattern. If an algorithm can recognise the patterns, then it can define the current movement.

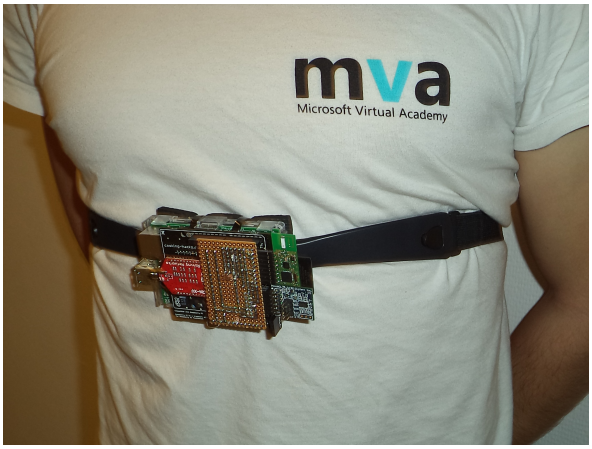


Figure 1: The applied data collector device

2. Parallel operation

On the RPi a C++ program performs the data acquisition and data evaluation. In the C++ code the *POSIX thread* or *Pthread* library allows parallel programming [12]. Pthread is a set of C programming types and procedure calls. The data acquisition and the analysis are independent tasks which can be executed concurrently. The analyser program uses two threads. The main thread collects the measured acceleration components and an auxiliary thread performs the data analysis. Figure 2 illustrates the parallelized data collection and evaluation.

While the device is active, the data acquisition function runs continuously and stores the collected data in buffers. Currently, the data acquisition frequency is 100 Hz and the buffer size is 2^8. Consequently, the analyser method splits the continuous signal to short parts (about 2.5 seconds long) and tries to decide the current activity. When the buffers are full, the auxiliary thread starts and gets a *void** structure which includes the buffers. Pthread permits to pass only one argument to the new thread, therefore every argument have to be embedded into a structure. The auxiliary thread will call the digital signal processing (DSP) and pattern recognition functions. Furthermore, the auxiliary thread is responsible for the storage and notification.

If the data analysis finishes, the decision about the movement will be stored in a file and sent to a server in IP packet. A strict rule, that the auxiliary thread have to be faster than the data acquisition process. Thereby, the program can avoid thread collision.

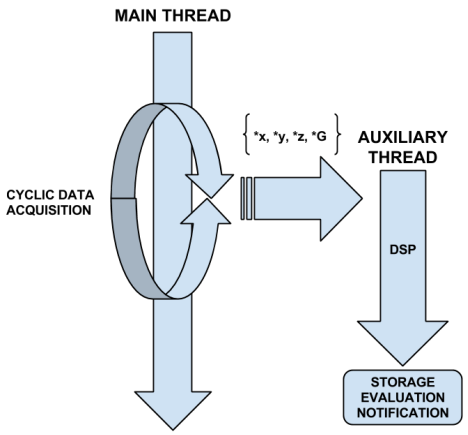


Figure 2: The structure of parallel operation. The **x, *y, *z* are buffers which contain the *x, y, z* components of the accelerometer and **G* buffer includes the normalised acceleration magnitude.

3. Description of the method

The activity analysis is based on time-frequency signal processing. First, the algorithm categorises the acquired signal in the frequency domain. In the next step, according to the frequency category, it will search the possible patterns which belong to the assigned category. The detected pattern will identify the movement type. Figure 3 illustrates the flowchart of the applied algorithm.

On figure 3 there is an *unknown activity* state. It means that, if the examined signal contains an incomprehensible sequence, the algorithm will not give decision.

During the acquisition process, the program calculates the normalised acceleration magnitude ($G[i]$) from the collected parameters (x, y, z).

$$G[i] = \frac{\sqrt{x[i]^2 + y[i]^2 + z[i]^2}}{1g}$$

where $1g$ depends on the resolution of the accelerometer. The G characterises the change in the movement, thus it will be the key in the analysis [1, 7]. Firstly, the frequency coefficients of the G signal will be calculated. Before the fast Fourier transformation (FFT), the signal was *diluted* and *windowed* with a Blackman-Nuttall window in order to minimize the *leakage* and separate the closely spaced frequencies [15, 16]. The length of the signal influences the frequency resolution. Consequently, if the frequency resolution is higher, then the frequency categorisation is easier. The real signal size is 2^8 which will be diluted to 2^9 with zeros. After the dilution the extended signal will be multiplied with the window function.

In order to the auxiliary thread can analyse the G signal faster than the data acquisition (less than 2.5 seconds), we created an optimized FFT algorithm to

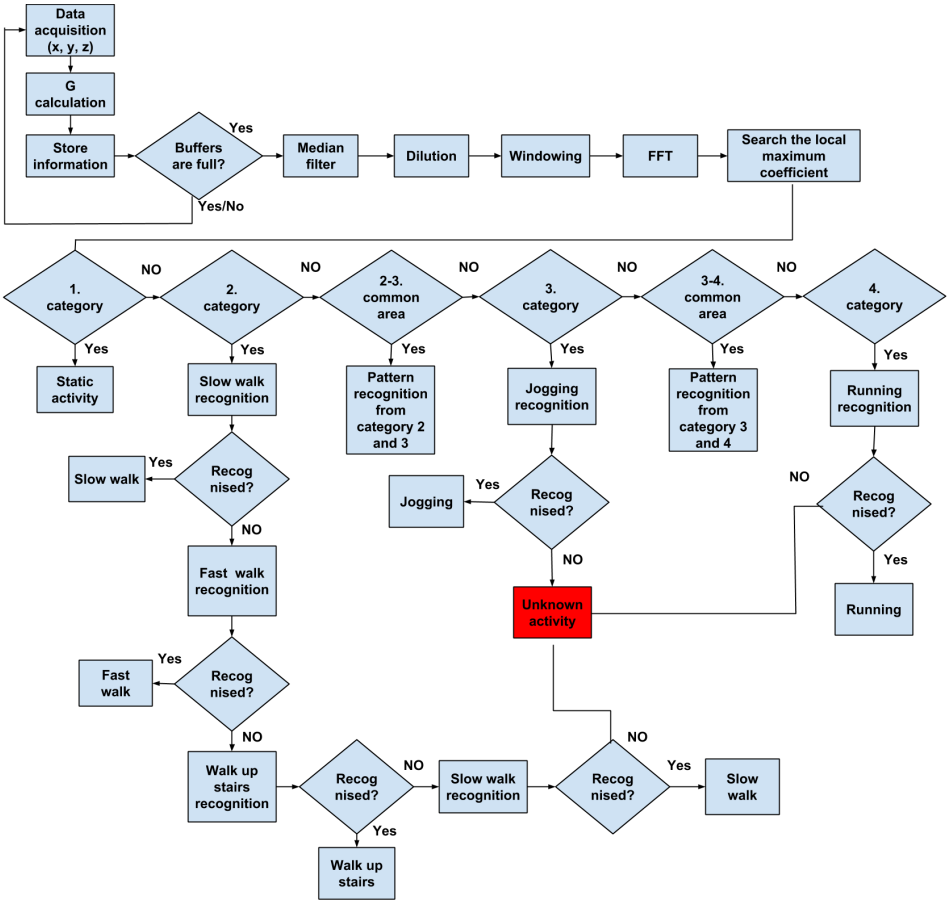


Figure 3: The flowchart of the algorithm

calculate the frequency coefficients [13]. Since, the signal length is fix and the FFT runs periodically in the program therefore worth to store the *"twiddle-factors"* into memory as pre-defined constants. Generally the radix-2 FFT can be written as,

$$X(k) = E(k) + W_N^k O(k) \qquad k = 0, 1, \ldots, \frac{N}{2} - 1$$

$$X(k) = E(k - \frac{N}{2}) - W_N^k O(k - \frac{N}{2}) \qquad k = \frac{N}{2}, \ldots, N - 1.$$

In the equation $E(k)$ and $O(k)$ contain the frequency coefficients to the even and odd elements and W_N^k for $k = 0, ..., N-1$ is the N'th root of unity [18]. If we take advantage of the relations between W_N^k factors (3.1), we will save memory because it is enough to store one-fourth of factors.

$$W_{N_{Im}}^{k+\frac{N}{4}} = -W_{N_{Re}}^k$$

$$W_{N_{Re}}^{k+\frac{N}{4}} = W_{N_{Im}}^{k}$$
$$W_{N}^{k+\frac{N}{2}} = -W_{N}^{k} \tag{3.1}$$

In the above formulas Re and Im refer to the real and imaginary parts of a complex number. The Euler's formula (3.2) allows the decomposition of the W_N^k factors into real and imaginary parts thus the real and imaginary parts of the factors will be stored separately in the program.

$$W_N^k = \cos\left(k\frac{2\pi}{N}\right) - i\sin\left(k\frac{2\pi}{N}\right) \tag{3.2}$$

After the frequency coefficients are available, the algorithm searches the maximum value inside a specified frequency interval. The frequency categories bound an interval between 0.8 Hz and 3.8 Hz. The maximum value defines the activity category. Table 1 contains the applied frequency categories and an approximate frequency interval. Actually, the data is stored in binary form thus the frequency intervals are based on the indexes (bins) of the frequency vector.

Categories	Min frequency	Max frequency
1.	0.0 Hz	0.8 Hz
2.	0.81 Hz	2.25 Hz
2. - 3.	2.26 Hz	2.5 Hz
3.	2.51 Hz	2.65 Hz
3. - 4.	2.66 Hz	2.8 Hz
4.	2.81 Hz	3.8 Hz

Table 1: Frequency categories

According to the category, the appropriate pattern detection functions will be used on the G signal. Therefore, the number of operations greatly decreases. Unfortunately, between some categories there is a narrow overlapping. In that case, if the maximum coefficient is in the common area of two categories, then the algorithm will search each patters which belong to the two adjacent categories. Figure 4 illustrates the frequency spectrum of some main activities.

The maximum frequency coefficient should higher than an appropriate amplitude limit. In this case, a reliable limit is between 14.5 and 16.5. On the above figure the blue lines indicate the frequency interval and the amplitude limit. If the maximum is lower than the limit, then the movement is static. According to the position the algorithm concludes the frequency category.

3.1. Pattern recognition

The pattern recognition starts with a median filtering with a seven samples length window to reduce the noise on the G signal before the recognition [6, 7]. In the

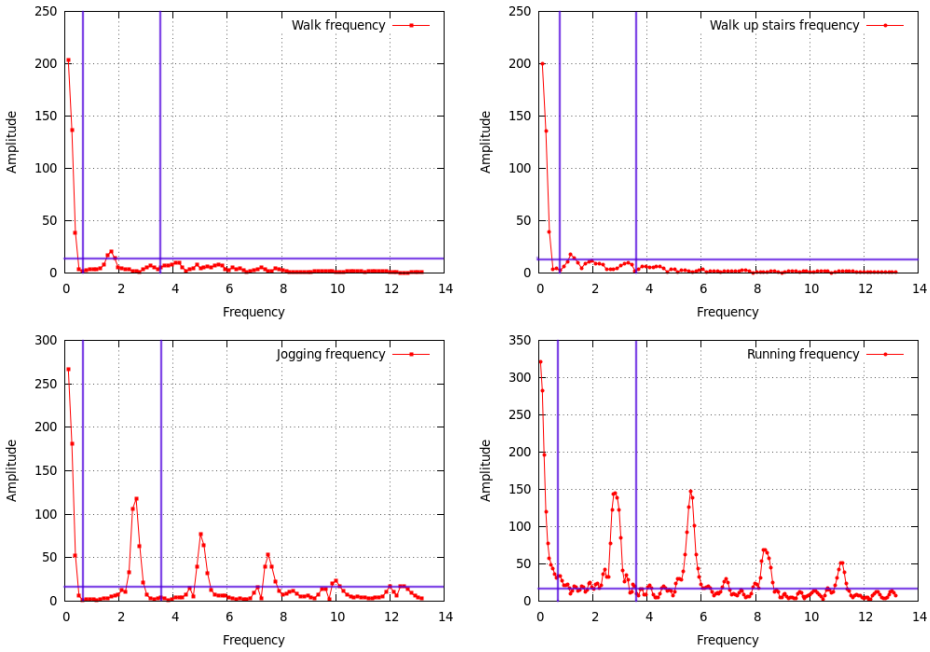


Figure 4: Frequency spectrum of some main activities

literature, one of the most common pattern recognition technique is the correlation [17].

$$(f \cdot g)[n] := \sum_{m=0}^{N-1} f[m]g[m+n]$$

where f and g are real vectors. However, in some cases the correlation does not provide acceptable result. Consequently, we developed a simple and individual pattern recognition method. The method is a combination of the square error and the correlation. It can be characterized as a shifted and summarised square error (SSE).

$$SSE(f,g)[n] := \sum_{m=0}^{N-1} (f[m] - g[m+n])^2$$

In both cases, an ideal pattern (or kernel) with a special shape will pass through the examined signal. In a correlated signal the peak(s) are higher when the similarity between the pattern and the examined signal is large. However, in the SSE the low parts indicate the high similarity. The difference between the correlation and the SSE can be visualised with a simple example. The example compares two similar activities: walking and the climbing of stairs. In the example the climbing stairs pattern was used on the two activity sequences. In ideal case, if the sought

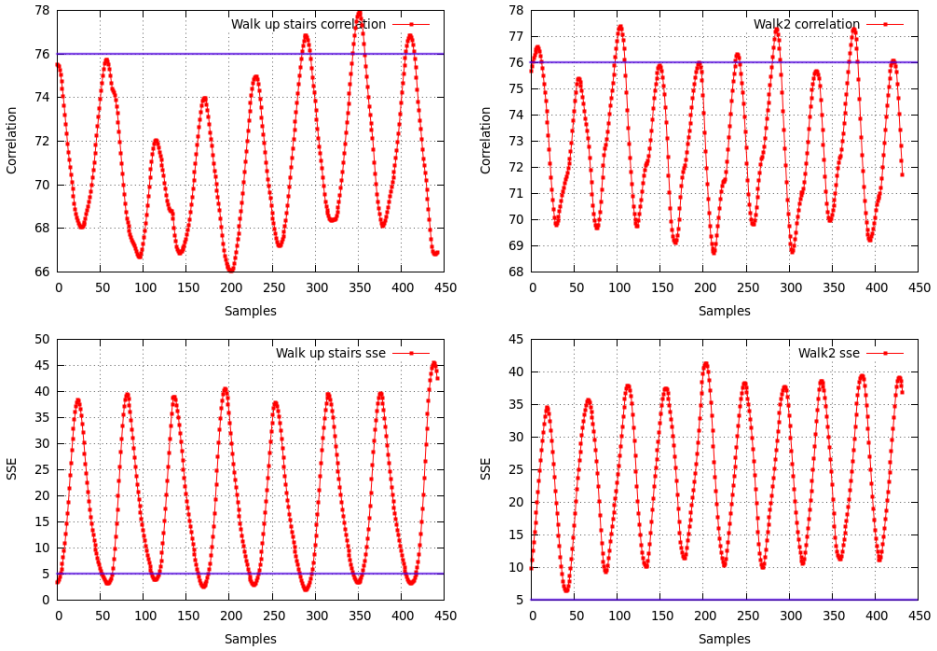


Figure 5: Comparison of the two pattern recognition techniques

pattern does not appear in the signal, all resulted points have to be less or higher as a well defined limit, according to the applied technique. Figure 5 presents the result where the blue lines are imaginary limits to the two pattern recognition techniques. As the example shows, the correlation can not separate the activities with the used kernel. For instance, if we use the 76 (blue line) as limit, the same number of points will be higher than the limit in both cases. In contrast, the SSE separates the activities well. On figure 5 a lot of points belong under 5 of the climbing stairs sequence while in the walk sequence every point is higher than 5.

4. Conclusion

The presented technique is well applicable for activity recognition. If an activity describes an individual and periodic acceleration change then the presented recognition algorithm will find the patterns. As figure 3 shows, if we know the frequency interval and the described pattern of an activity, we can easily past the new activity into the method. Obviously, an unexpected event such as a fall is similarly recognizable because a fall (regardless of the direction) has a suddenly ascending and then decaying acceleration fluctuation.

The ability to evaluate the movement types provides an exceptional source of knowledge to doctors to diagnose patients. The movement analysis is a useful aid to

detect potential causes of gait and lifestyles abnormalities [14]. As was mentioned at the beginning of the article, human movement is researched in relation to a lot of diseases, such us reduced mobility disorders (stroke, obesity), sclerosis and Parkinson's disease [2, 4, 14]. To sum up, human movement analysis provides much information about the health condition of the observed patient.

Today, as the importance of research in the area of Future Internet is increasing, applications of the so called Internet of Things (IoT) are becoming more and more popular. The IoT is a network of different types of objects (people, sensors, devices, etc.) which can communicate with each other via the Internet [19, 20, 21, 22]. The presented solution to the activity recognition problem belongs to this branch of research which is expected to be determinative in the following years.

References

[1] Smith, T.F., Waterman, M.S., Elderly activities recognition and classification for applications in assisted living, *Expert Systems with Applications*, Vol. 40 (2013), 1662-1674.

[2] Godfrey, A., Bourke, K.A., Ólaighin, M.G., Ven de van, P., Nelson, J., Activity classification using a single chest mounted tri-axial accelerometer, *Medical Engineering & Physics*, Vol. 33 (2011), 1127-1135.

[3] Lyons, M.G, Culhane, M.K., Hilton, D., Grace, A.P., Lyons, D., A description of an accelerometer-based mobility monitoring technique, *Medical Engineering & Physics*, Vol. 27 (2005), 497-504.

[4] Kavanagh, J.J, Menz, B.H., Accelerometry: A technique for quantifying movement patterns during walking, *Gait & Posture*, Vol. 28 (2008), 1-15.

[5] Forester, F., Smeja, M., Fahrenberg, J., Detection of posture and motion by accelerometry: a validation study in ambulatory monitoring, *Computers in Human Behavior*, Vol. 15 (1999), 571-583.

[6] Lugade, V., Fortune, E., Morrow, M., Kaufman, K., Validity of using tri-axial accelerometers to measure human movement—Part I: Posture and movement detection, *Medical Engineering & Physics*, Vol. 36 (2014), 169-176.

[7] Kangas, M., Konttila, A., Lindgren, P., Winblad, I., Jamsa, T., Comparison of low-complexity fall detection algorithms for body attached accelerometers, *Gait & Posture*, Vol. 28 (2008), 285-291.

[8] Chern-Sheng Lin, Hung Chun Hsu, Yun-Long Lay, Chuang-Chien Chiu, Chi-Shih Chao, Wearable device for real-time monitoring of human falls, *Measurement*, Vol. 40 (2007), 831-840.

[9] Bourke, K.A., O'Brien, V.J., Lyons, M.G., Evaluation of a threshold-based tri-axial accelerometer fall detection algorithm, *Gait & Posture*, Vol. 26 (2007), 194-199.

[10] Ge Wu, Distinguishing fall activities from normal activities by velocity characteristics, *Journal of Biomechanics*, Vol. 33 (2000), 1497-1500.

[11] Suto, J., Oniga, S., Testing artificial neural network for hand gesture recognition, *Creative Mathematics and Informatics*, Vol. 22 (2013), 223-228.

[12] Buttlar, D., Farrell, J., Nichols, B., PThreads Programming. O'Reilly Media, USA (1996).

[13] Suto, J., Oniga, S., Hegyesi Gy., A simple fast Fourier transformation algorithm to microcontrollers and mini computers, *IEEE 18th International Conference on Intelligent Engineering Systems*, (2014), 61-65.

[14] Godfrey, A., Conway, R., Meagher, D., Ólaighin, G., Direct measurement of human movement by accelerometry, *Medical Engineering & Physics*, Vol. 30 (2008), 1364-1386.

[15] Smith, W.S., The Scientist and Engineer's Guide to Digital Signal Processing. California Technical Publisher, USA (1999).

[16] Lyons, G.L., Understanding Digital Signal Processing. Prentice Hall PTR, USA (2001).

[17] Kammler, D.W., A First Course in Fourier Analysis. Cambridge University Press, UK (2008).

[18] Mertins, A., Signal Analysis: Wavelets, Filter Banks, Time-Frequency Transforms and Applications. John Wiley & Sons, England (1999).

[19] Lung, C., Oniga, S., Buchman, A., Tisan, A., Wireless data acquisition system for IoT applications, *Carpathian Journal of Electronic and Computer Engineering*, Vol. 6 (2013), 64-67.

[20] Suto, J., Oniga, S., Orha, I., Microcontroller based health monitoring system, *IEEE 19th International Symposium for Design and Technology in Electronic Packaging (SIITME)*, (2013), 227-230.

[21] Terdik, Gy., Gal, Z., Advances and practice in Internet of Things: A case study, *IEEE 4th International Conference on Cognitive Infocommunications*, (2013), 435-440.

[22] Berczes, T., Sztrik, J., Orosz, P., Moyal, P., Limnios, N., Georgiadis, S., Tool supported modeling of sensor communications networks by using finite-source priority retrial queues, *Carpathian Journal of Electronic and Computer Engineering*, Vol. 5 (2012), 13-18.

Annales Mathematicae et Informaticae
44 (2015) pp. 197–210
http://ami.ektf.hu

Boundaries of membrane in P systems relying on multiset approximation spaces in language R

Péter Takács[a], Zoltán Ernő Csajbók[a], Tamás Mihálydeák[b]

[a]Department of Health Informatics, Faculty of Health, University of Debrecen
{takacs.peter,csajbok.zoltan}@foh.unideb.hu

[b]Department of Computer Science, Faculty of Informatics, University of Debrecen
mihalydeak.tamas@inf.unideb.hu

Submitted September 11, 2014 — Accepted May 13, 2015

Abstract

Membrane computing is an area within computer science which aims to develop a new computational model through the study of the characteristics of biological cells. It is a distributed and parallel computing model. Communication between regions through membranes, as well as membrane system and its environment, plays an important role in the process. Combination of P system with multiset approximation space leads to the abstract concept of 'to be close enough to a membrane'. The designated goal is to perform calculations in this two-fold system by the help of language R. Some packages can perform calculations with multisets in R (such as 'sets' package), but they are more closely linked to fuzzy systems. In this paper a new program library in language R is initiated which had been created to encourage some fundamental calculations in membrane systems combined with multiset approximation spaces. Data structures and functions are illustrated by examples.

Keywords: multiset approximation spaces, membrane computing, R language

MSC: 68U20

1. Introduction

The classical set theory does not enable multiply occurrences of the same objects. However, the multiset theory provides the opportunity to do so [2, 14]. Membrane computing also works with multisets [8, 9, 10, 11]. In [4, 5], the authors developed an abstract concept of the 'to be close enough to a membrane'. They used a generalization of classical Pawlakian rough set theory for multisets which is called the Pawlakian multiset approximation space (MAS). One part of the work is to carry out related calculations in MAS quickly and accurately.

There are several packages in language R (e.g., package 'sets' [15]), which allows multiset calculations. However, by the use of these applications, it is difficult to perform MAS calculations. The goal of this paper is to present R functions which facilitate in a quick and easy manner to perform all the most important calculations in membrane system combined with MAS. In Section 2, some initial relations and operations for multisets with illustrative R examples are described. Then, in Section 3 and 4, R functions for Pawlakian multiset approximation spaces are introduced in order to apply them to membrane computing.[1]

2. Multiset functions in R

2.1. Multisets

Let U be a finite nonempty set called the universe. A *multiset* (or *mset*) M over U is a mapping $M : U \to \mathbb{N} \cup \{\infty\}$ ($\mathbb{N}$ is the set of natural numbers). For instance, if $a \in U$ and M is a multiset with three occurrences of a, then $M(a) = 3$. This fact is often referred to as a^3. In general, if more than one element are repeated in a multiset, it is usually expressed in power form. For example, $M = a^3b^2$ is a multiset with three a's and two b's. M is *empty multiset*, denoted by $\emptyset$, if $M(a) = 0$ $(a \in U)$.

Let $\mathcal{MS}(U)$ denote the set of all multisets over U. $M \in \mathcal{MS}(U)$ is *finite*, if $M(a) < \infty$ $(a \in U)$. A *macroset* $\mathcal{M}$ is a set of finite multisets [3]. In the framework the following two fundamental macrosets are used:

- $\mathcal{MS}^n(U)$ $(n \in \mathbb{N})$ is the set of all multisets M such that $M(a) \leq n$ $(a \in U)$;
- $\mathcal{MS}^{<\infty}(U) = \bigcup_{n=0}^{\infty} \mathcal{MS}^n(U)$.

Let us note that $\mathcal{MS}^0(U) = \emptyset$ and $\mathcal{MS}^n(U) \subsetneqq \mathcal{MS}^{n+1}(U)$ $(n = 0, 1, 2, \dots)$.

Some basic R functions over $\mathcal{MS}^{<\infty}(U)$ have been developed to calculate the multiset relations and operations. Let M, M_1, $M_2 \in \mathcal{MS}^{<\infty}(U)$.

Set-theoretical relations for multisets implemented in R are the following:

- *Multiplicity relation*: $a \in M$ $(a \in U)$ if $M(a) \geq 1$.

[1] There is no room here to describe R functions in code level. We send it to everyone who is interested in.

- *Equality relation* is: $M_1 = M_2$ if $M_1(a) = M_2(a)$ $(a \in U)$.
- *Inclusion relation*: $M_1 \sqsubseteq M_2$ if $M_1(a) \leq M_2(a)$ $(a \in U)$.

Set-theoretical operations for multisets implemented in R are the following:

- *Set-type union*: $(M_1 \sqcup M_2)(a) = \mathsf{max}\{M_1(a), M_2(a)\}$ $(a \in U)$.
- *Intersection*: $(M_1 \sqcap M_2)(a) = \mathsf{min}\{M_1(a), M_2(a)\}$ $(a \in U)$.
- *Multiset addition*: $(M_1 \oplus M_2)(a) = M_1(a) + M_2(a)$ $(a \in U)$.
- *n-times addition* $(n \in \mathbb{N})$: it is given by the following inductive definition:
  1. $\oplus_0 M = \emptyset$
  2. $\oplus_1 M = M$
  3. $\oplus_n M = \oplus_{n-1} M \oplus M$ $(n > 1)$.
- *Multiset subtraction*: $(M_1 \ominus M_2)(a) = \mathsf{max}\{M_1(a) - M_2(a), 0\}$ $(a \in U)$.

By the help of n-times addition, a new multiset relation can be defined:

- *n-times inclusion relation* $(n \in \mathbb{N})$: Let $M_1 \neq \emptyset$. $M_1 \sqsubseteq^n M_2$ if $\oplus_n M_1 \sqsubseteq M_2$ but $\oplus_{n+1} M_1 \not\sqsubseteq M_2$.

2.2. Implementation of multiset relations

Throughout our implementation, it is assumed that the universe U is finite, fixed and its elements are totally ordered.

Implemented R functions are demonstrated by the help of a running example. To this end, first, let $U = \{a, b, c, d, e\}$ with the natural English alphabet ordering. It will be given as the fixed universe with the following command:

```
> U <- c("a","b","c","d","e").
```

Remark 2.1. Here and later on, '>' denotes the R prompt. `c()` is the concatenation function in R. Therefore, the universe U in R can be viewed as the string "*abcde*".

Each multiset is described in Parikh vector representation form. This means that the elements which are not actually included in the multiset are indicated by zero exponent. For instance, let us take the multiset $M = ce^5$. Its R representation in Parikh vector form is $a^0b^0c^1d^0e^5$, whereas its R realization is:

```
> M <- c(0,0,1,0,5).
```

Remark 2.2. M in R can be viewed as the string "00105".

Turning to the implementation of multiset relations, the first function is a technical one. It is a verification function which is called for every relation and operation.

mcheck(mS, SU)
Parameters: multiset **mS**; universe **SU**.

Description: This function checks the number of elements of multiset **mS**. If the cardinality of distinct elements in **mS** is equal to the cardinality of **SU**, the function returns 1, otherwise it returns 0.

The next three functions realize the set-theoretical relations for multisets.

min(mS, o, SU) – Multiplicity relation
Parameters: multiset **mS**, object $\mathbf{o} \in \mathbf{SU}$; universe **SU**.
Description: This function checks the membership of **o** in **mS**. It returns 1 if $\mathbf{mS}(\mathbf{o}) \geq 1$, otherwise it returns 0.

Example 2.3. Multiplicity relation

`> M <- c(1,2,3,0,0)`	$M = ab^2c^3$
`> min(M,"a",U)`	
`[1] 1`	$a \in M$
`> min(M,"e",U)`	
`[1] 0`	$e \notin M$

Remark 2.4. Result of R command, if any, is located in its underlying row beginning with '[1]' sign. '|' is a selector line which separates the mathematical formulae (the second column) from their implementations in R (the first column).

mequal(mS1, mS2, SU) – Equality relation
Parameters: multisets **mS1**, **mS2**; universe **SU**.
Description: This function checks the equality relation of two multisets **mS1**, **mS2**. It returns 1 if **mS1** and **mS2** are equal, otherwise it returns 0.

Example 2.5. Equality relation

`> M1 <- c(0,0,1,0,5)`	$M_1 = ce^5$
`> M2 <- c(3,2,0,1,0)`	$M_2 = a^3b^2d$
`> mequal(M1,M1,U)`	
`[1] 1`	$M_1 = ce^5 = ce^5 = M_1$
`> mequal(M1,M2,U)`	
`[1] 0`	$M_1 = ce^5 \neq a^3b^2d = M_2$

mpartof(mS1, mS2, SU) – Inclusion relation
Parameters: multisets **mS1**, **mS2**; universe **SU**.
Description: This function checks whether **mS1** is included in **mS2** or not. It returns 1 if the multiset **mS1** is part of the multiset **mS2**, otherwise it returns 0.

Example 2.6. Inclusion relation

`> M3 <- c(0,0,2,1,5)`	$M_3 = c^2de^5$
`> mpartof(M1,M3,U)`	
`[1] 1`	$M_1 = ce^5 \sqsubseteq c^2de^5 = M_3$
`> mpartof(M1,M2,U)`	
`[1] 0`	$M_1 = ce^5 \not\sqsubseteq a^3b^2d = M_2$

2.3. Implementation of basic multiset operations

In demonstration examples, these multisets will be used in the following:

```
> W1 <- c(0,0,1,0,5)
> W2 <- c(3,2,0,1,0)
> W3 <- c(1,2,3,4,0)
> W4 <- c(3,1,0,0,0)
> W5 <- c(1,1,2,3,0)
```

$W_1 = ce^5$
$W_2 = a^3b^2d$
$W_3 = ab^2c^3d^4$
$W_4 = a^3b$
$W_5 = abc^2d^3$

munion(mS1, mS2, SU) – Set-type union
Parameters: multisets **mS1**, **mS2**; universe **SU**.
Description: This function computes the set-type union of multisets **mS1**, **mS2**.

Example 2.7. Set-type union

```
> munion(W4,W5,U)
[1] 3 1 2 3 0
```

$W_4 \sqcup W_5 = a^3bc^2d^3$

mintersec(mS1, mS2, SU) – Intersection
Parameters: multisets **mS1**, **mS2**; universe **SU**.
Description: This function computes the intersection of multisets **mS1**, **mS2**.

Example 2.8. Intersection

```
> mintersec(W2,W3,U)
[1] 1 2 0 1 0
```

$W_2 \sqcap W_3 = ab^2d$

madd(mS1, mS2, SU) – Multiset addition
Parameters: multisets **mS1**, **mS2**; universe **SU**.
Description: This function computes the multiset addition of multisets **mS1**, **mS2**.

Example 2.9. Multiset addition

```
> madd(W4,W5,U)
[1] 4 2 2 3 0
```

$W_4 \oplus W_5 = a^4b^2c^2d^3$

mnadd(mS, n, SU) – n-times addition
Parameters: multiset **mS**, $\mathbf{n} \in \mathbb{N}$; universe **SU**.
Description: This function computes n-times addition of multiset **mS**.

Example 2.10. n-times addition

```
> mnadd(W1,0,U)
[1] 0 0 0 0 0
> mnadd(W1,1,U)
[1] 0 0 1 0 5
> mnadd(W1,3,U)
[1] 0 0 3 0 15
```

$\oplus_0 W_1 = \emptyset$
$\oplus_1 W_1 = ce^5$
$\oplus_3 W_1 = c^3e^{15}$

mdiff(mS1, mS2, SU) – Multiset subtraction
Parameters: multisets **mS1**, **mS2**; universe **SU**.
Description: This function computes the multiset subtraction of multisets **mS1**, **mS2**.

Example 2.11. Multiset subtraction

```
> mdiff(W3,W2,U)
[1] 0 0 3 3 0
```

$W_3 \ominus W_2 = c^3d^3$

mnpartof(mS1, mS2, SU) n-times inclusion relation
Parameters: multisets **mS1**$(\neq \emptyset)$, **mS2**; universe **SU**.
Description: This function determines how many times **mS1** is included in **mS2**. It returns $n(\in \mathbb{N})$ if $\oplus_n$**mS1** $\sqsubseteq$ **mS2** but $\oplus_{n+1}$**mS1** $\not\sqsubseteq$ **mS2**.

Example 2.12. n-times inclusion relation

```
> M  <- c(0,0,0,0,0)
> M1 <- c(0,0,1,3,0)
> M2 <- c(0,0,1,3,1)
> M3 <- c(1,0,3,11,1)
> mnpartof(M3,M,U)
[1] 0
> mnpartof(M1,M2,U)
[1] 1
> mnpartof(M1,M3,U)
[1] 3
```

$M = \emptyset$
$M_1 = cd^3$
$M_2 = cd^3e$
$M_3 = ac^3d^{11}e$

$M_3 \sqsubseteq^0 M = \emptyset$

$M_1 \sqsubseteq^1 M_2$ (i.e., $M_1 \sqsubseteq M_2$)

$M_1 \sqsubseteq^3 M_3$

3. Calculation in Pawlakian multiset approximation spaces

3.1. Pawlakian multiset approximation spaces

To define the abstract notion of boundaries in membrane systems, rough set theory (RST) should be a plausible opportunity [12, 13]. However, RST works within the traditional set theory, while regions in membrane systems are represented by multisets. Thus, to be able to apply the notions of RST, first, we have to generalize them for multisets.

Such a generalized multiset approximation space has four basic components:

- *Domain*: a set of multisets whose members are approximated.
- *Base system*: a set of some distinguished multisets (called *base multisets*) of the domain as the basis of approximations. Members of the base system are primary tools of the approximation process. Definable sets describe how they are combined, whereas approximation primitives give an account of how they are utilized in this process.

- *Definable multisets*: a set of multisets. They are
  - derived from base multisets (all base multisets are definable);
  - candidates for possible approximations and boundaries of the members of the domain.
- *Approximation primitives*: they determine lower/upper approximations and boundaries of the domain members using definable multisets.

Let U be a nonempty set. The 6-tuple $\mathsf{MAS}(U) = \langle \mathcal{MS}^{<\infty}(U), \mathfrak{B}, \mathfrak{D}_{\mathfrak{B}}, \mathsf{l}, \mathsf{b}, \mathsf{u} \rangle$ is a *multiset approximation space* if

- (*domain*) $\mathcal{MS}^{<\infty}(U) \subseteq \mathcal{MS}(U)$;
- (*base system*) $\mathfrak{B}(\neq \emptyset) \subseteq \mathcal{MS}^{<\infty}(U)$ and if $B \in \mathfrak{B}$, then $B \neq \emptyset$;
- (*definable multisets*) $\mathfrak{B} \subseteq \mathfrak{D}_{\mathfrak{B}}$; $\emptyset \in \mathfrak{D}_{\mathfrak{B}}$; if $B \in \mathfrak{B}$, $\oplus_n B \in \mathfrak{D}_{\mathfrak{B}}$ $(n = 1, 2, \dots)$;
- (*approximation primitives*) functions $\mathsf{l}, \mathsf{b}, \mathsf{u} : \mathcal{MS}^{<\infty}(U) \to \mathcal{MS}^{<\infty}(U)$ meet the following requirements:
  - (*i*) $\mathsf{l}(\mathcal{MS}^{<\infty}(U)), \mathsf{b}(\mathcal{MS}^{<\infty}(U)), \mathsf{u}(\mathcal{MS}^{<\infty}(U)) \subseteq \mathfrak{D}_{\mathfrak{B}}$ (*definability*);
  - (*ii*) the functions l and u are monotone (*monotonicity*);
  - (*iii*) $\mathsf{u}(\emptyset) = \emptyset$ (*normality* of u);
  - (*iv*) if $M \in \mathcal{MS}^{<\infty}(U)$, then $\mathsf{l}(M) \sqsubseteq \mathsf{u}(M)$ (*weak approximation property*);
  - (*v*) $\mathsf{b}(M) \sqcap M \neq \emptyset$ but $\mathsf{b}(M) \not\sqsubseteq M$ and $\mathsf{b}(M) \ominus M \neq \emptyset$, provided that $\mathsf{b}(M) \neq \emptyset$ $(M \in \mathcal{MS}^{<\infty}(U))$ (*Janus-faced nature of boundary*).

By historical reasons, lower and upper approximations together is called the approximation pair and denoted by $\langle \mathsf{l}, \mathsf{u} \rangle$. With the above properties, it is said that $\langle \mathsf{l}, \mathsf{u} \rangle$ is a *weak approximation pair*.

A number of important and interesting variations of $\mathsf{MAS}(U)$ can be formed. For our aim, the most interesting case is when $\mathsf{MAS}(U)$ is Pawlakian type.

Let $\mathfrak{B}^{\oplus} = \{\oplus_n B \mid B \in \mathfrak{B},\ n = 1, 2, \dots\}$. $\mathsf{MAS}(U)$ is a *strictly set-union type* multiset approximation space if $\mathfrak{D}_{\mathfrak{B}}$ is given by the following inductive definition:

1. $\emptyset \in \mathfrak{D}_{\mathfrak{B}}$, $\mathfrak{B}^{\oplus} \subseteq \mathfrak{D}_{\mathfrak{B}}$, and
2. if $\mathfrak{B}' \subseteq \mathfrak{B}^{\oplus}$, then $\bigsqcup \mathfrak{B}' \in \mathfrak{D}_{\mathfrak{B}}$.

Let $\mathsf{MAS}(U)$ be a strictly set-union type multiset approximation space. Then, $\mathsf{l}, \mathsf{u}, \mathsf{b} : \mathcal{MS}^{<\infty}(U) \to \mathcal{MS}^{<\infty}(U)$ form a *Pawlakian multiset approximation pair* $\langle \mathsf{l}, \mathsf{u} \rangle$ and a *Pawlakian boundary* b if for any multiset $M \in \mathcal{MS}^{<\infty}(U)$

1. $\mathsf{l}(M) = \bigsqcup\{\oplus_n B \mid n \in \mathbb{N}^+, B \in \mathfrak{B} \text{ and } B \sqsubseteq^n M\}$,
2. $\mathsf{b}(M) = \bigsqcup\{\oplus_n B \mid B \in \mathfrak{B}, B \not\sqsubseteq M,\ B \sqcap M \neq \emptyset \text{ and } B \sqcap M \sqsubseteq^n M\}$,
3. $\mathsf{u}(M) = \mathsf{l}(M) \sqcup \mathsf{b}(M)$.

In this case, $\mathsf{MAS}(U)$ is called a *Pawlakian multiset approximation space*.

3.2. R functions in Pawlakian multiset approximation spaces

Let $\mathsf{MAS}(U)$ be a Pawlakian multiset approximation space. Any member of the domain $\mathcal{MS}^{<\infty}(U)$ can be represented in Parikh vector form by the concatenation function `c()` as usual.

Of course, it is assumed that the number of base multisets is finite. Base system is represented in matrix form. It can be formed in three steps with the help of R functions `c()` and `matrix()`:

1. defining base multisets by `c()` (it is assumed that the number of base multisets is n, where $n(>0) \in \mathbb{N}$);

2. forming a base vector from base multisets by `c()`;

3. building the base matrix from the base vector by `matrix()`.

Let $U = \{a, b, c, d, e\}$ as above. The previous process is illustrated by the following example:

1. Defining base multisets:

`> B1 <- c(2,0,0,0,0)`	$B_1 = a^2$
`> B2 <- c(1,1,0,0,0)`	$B_2 = ab$
`> B3 <- c(0,1,0,0,0)`	$B_3 = b$
`> B4 <- c(0,0,1,1,1)`	$B_4 = cde$
`> n <- 4`	$n = 4$

2. Forming the base vector:

```
> Base_vect <- c(B1,B2,B3,B4)
```

3. Building the base systems in matrix form from the base vector:

```
> B <- matrix(Base_vect, nrow=n, ncol=length(U), byrow=T) .
```

That is, the matrix B represents the base system as follows: B has 4 rows (the number of base multisets) and 5 columns (the cardinality of U). The ith row contains the components of the Parikh vector representation of the ith base multiset.

plow(mS, BASE, SU) – Lower approximation
Parameters: multiset **mS** ; base system **BASE**; the universe **SU**.
Description: This function computes the lower approximation of the multiset **mS** over the base system **BASE**.

Example 3.1. Lower approximation

`> plow(W1,B,U)`	
`[1] 0 0 0 0 0`	$\mathsf{l}(W_1) = \emptyset$
`> plow(W2,B,U)`	
`[1] 2 2 0 0 0`	$\mathsf{l}(W_2) = a^2b^2$
`> plow(W3,B,U)`	
`[1] 1 2 0 0 0`	$\mathsf{l}(W_3) = ab^2$
`> plow(W4,B,U)`	
`[1] 2 1 0 0 0`	$\mathsf{l}(W_4) = a^2b$
`> plow(W5,B,U)`	
`[1] 1 1 0 0 0`	$\mathsf{l}(W_5) = ab$

pbound(mS, BASE, SU) – Boundary
Parameters: multiset **mS**, base system **BASE**; the universe **SU**.
Description: This function computes the boundary of the multiset **mS** over the base system **BASE**.

Example 3.2. Boundary

`> pbound(W1,B,U)`	
`[1] 0 0 1 1 1`	$\mathsf{b}(W_1) = cde$
`> pbound(W2,B,U)`	
`[1] 0 0 1 1 1`	$\mathsf{b}(W_2) = cde$
`> pbound(W3,B,U)`	
`[1] 2 0 3 3 3`	$\mathsf{b}(W_3) = a^2c^3d^3e^3$
`> pbound(W4,B,U)`	
`[1] 0 0 0 0 0`	$\mathsf{b}(W_4) = \emptyset$
`> pbound(W5,B,U)`	
`[1] 2 0 2 2 2`	$\mathsf{b}(W_5) = a^2c^2d^2e^2$

pupp(mS, BASE, SU) – Upper approximation
Parameters: multiset **mS**, base system **BASE**; the universe **SU**.
Description: This function computes the upper approximation of the multiset **mS** over the base system **BASE**.

Example 3.3. Upper approximation

`> pupp(W1,B,U)`	
`[1] 0 0 1 1 1`	$\mathsf{u}(W_1) = cde$
`> pupp(W2,B,U)`	
`[1] 2 2 1 1 1`	$\mathsf{u}(W_2) = a^2b^2cde$
`> pupp(W3,B,U)`	
`[1] 2 2 3 3 3`	$\mathsf{u}(W_3) = a^2b^2c^3d^3e^3$
`> pupp(W4,B,U)`	
`[1] 2 1 0 0 0`	$\mathsf{u}(W_4) = a^2b$
`> pupp(W5,B,U)`	
`[1] 2 1 2 2 2`	$\mathsf{u}(W_5) = a^2bc^2d^2e^2$

4. Calculations of boundaries in membrane systems

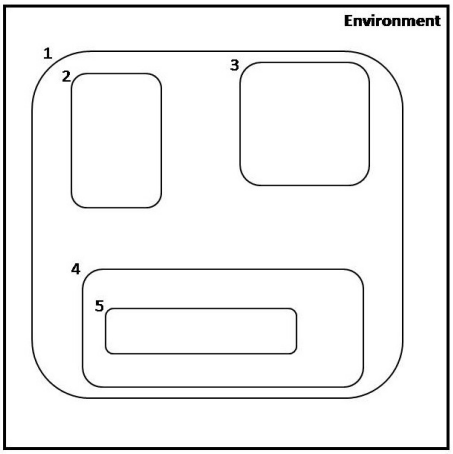
In this section the relationship between multiset approximation spaces and membrane systems is presented.

4.1. Membrane systems

Membrane system, or P system for short, was invented by Gheorghe Păun about 2000 [8, 9]. It was inspired by the architecture and functioning of living cells in order to formulate a model of computation.

Formally, a P system of degree $m(\geq 1)$ is a tuple

$$\Pi = \langle U, \mu, w_1, \ldots, w_m, R_1, \ldots, R_m \rangle.$$



Membranes delimit *regions* $w_1, \ldots, w_m$ separating "inside" from "outside".

Regions are arranged in a hierarchical structure μ.

Each region is
- represented by multisets over a finite set of objects U;
- endowed with two sets of rules.

Evolutions rules regulate the events taking place in the regions.

Communication rules regulate movements of objects through membranes.

Figure 1: A P system represented as a set of nested membranes ($m = 5$)

4.2. Membrane boundaries

Let the P system $\Pi = \langle U, \mu, w_1, w_2, \ldots, w_m, R_1, R_2, \ldots, R_m \rangle$ be given. Further, let $\mathsf{MAS}(\Pi) = \langle \mathcal{MS}^{<\infty}(U), \mathfrak{B}, \mathfrak{D}_{\mathfrak{B}}, \mathsf{l}, \mathsf{b}, \mathsf{u} \rangle$ be a Pawlakian mset approximation space. Then, $\mathsf{MAS}(\Pi)$ is called a *joint (multiset) approximation space* of Π. It should be noted that both the P system Π and the joint approximation space $\mathsf{MAS}(\Pi)$ are defined over the same universe.

Regions in P system Π are represented by multisets $w_1, w_2, \ldots, w_m$. Therefore, putting $w_1, w_2, \ldots, w_m$ into the joint approximation space of Π, they can be approximated, i.e., their Pawlakian lower/upper approximations and boundaries can be determined.

Pawlakian lower approximations of all regions follow the membrane structure. Furthermore, Pawlakian upper approximation and the boundary of the skin membrane completely lie within the environment. However, the upper approximations and boundaries of not skin membranes do not obey the membrane structure in general. Thus, these Pawlakian boundaries have to be adjusted to the membrane structure. This adjustment can be carried out as follows.

Let Π be a P system and $\mathsf{MAS}(\Pi)$ be its joint approximation space. First, let us determine the following quantities. If $B \in \mathfrak{B}$ and $i = 1, 2, \ldots, m$, let

$$N(B,i) = \begin{cases} 0, \text{if } B \sqsubseteq w_i \text{ or } B \sqcap w_i = \emptyset; \\ n, \text{if } i = 1 \text{ and } B \sqcap w_1 \sqsubseteq^n w_1; \\ \mathsf{min}\{k, n \mid B \sqcap w_i \sqsubseteq^k w_i,\ B \ominus w_i \sqsubseteq^n w_{\mathsf{parent}(i)}\}, \quad \text{otherwise.} \end{cases}$$

Then, the functions *membrane boundaries*, *outside* and *inside membrane boundaries* are defined as follows ($i = 1, \ldots, m$):

$$\mathsf{bnd}(w_i) = \bigsqcup\{\oplus_{N(B,i)} B \mid B \in \mathfrak{B}\};$$

$$\mathsf{bnd}^{\mathsf{out}}(w_i) = \mathsf{bnd}(w_i) \ominus w_i;$$

$$\mathsf{bnd}^{\mathsf{in}}(w_i) = \mathsf{bnd}(w_i) \ominus \mathsf{bnd}^{\mathsf{out}}(w_i).$$

4.3. Calculations of membrane boundaries

First, let us give the regions in matrix form. The membrane structure μ is given in vector form in which the ith element defines the parent of the ith region.

Let us illustrate this process by the following example (it is assumed that the multisets W_1, W_2, W_3, W_4, W_5 represent regions, and the multisets B_1, B_2, B_3, B_4 which were given earlier form the base system):

1. Giving regions:
```
> Region <- c(W1,W2,W3,W4,W5)
> m <- 5
> R <- matrix(Region, nrow=m, ncol=length(U), byrow=T)
```
2. Giving the membrane structure:
```
> MU <- c(0,1,1,1,4)
```

`MU` follows the membrane structure which is depicted in Figure 1: W_1 is the skin membrane; W_2, W_3, W_4 are nested in W_1, and W_5 is nested in W_4.

The first function is an auxiliary one in order to be able to calculate the quantities $N(B,i)$'s. It will be called the `NBi()` function.

NB(REGION, i, BASE, j, SU, SMU) – Calculating $N(B,i)$ for fixed base multiset and region
Parameters: regions in matrix form **REGION**, **i**th region, base system **BASE**, **j**th base multiset, the universe **SU**, membrane structure **SMU**.
Description: This function calculates the quantity $N(B,i)$ for the ith region and the jth base multiset.

Example 4.1. Calculating $N(B_1, 3)$

```
> NB(R,3,B,1,U,MU)
[1] 0
```

$N(B_1, 3) = 0$

NBi(i) – Calculating $N(B, i)$'s for the ith region and all base multisets
Parameters: region **i**.
Description: This function calculates the quantities $N(B, i)$'s for the ith region and all base multisets. It calls the **NB**() function.

Example 4.2. Calculating all $N(B, i)$'s

```
> NBi(1)
[1] 0 0 0 1
```

$N(B_1, 1) = 0, N(B_2, 1) = 0, N(B_3, 1) = 0, N(B_4, 1) = 1$

```
> NBi(2)
[1] 0 0 0 1
```

$N(B_1, 2) = 0, N(B_2, 2) = 0, N(B_3, 2) = 0, N(B_4, 2) = 1$

```
> NBi(3)
[1] 0 0 0 3
```

$N(B_1, 3) = 0, N(B_2, 3) = 0, N(B_3, 3) = 0, N(B_4, 3) = 3$

```
> NBi(4)
[1] 0 0 0 0
```

$N(B_1, 4) = 0, N(B_2, 4) = 0, N(B_3, 4) = 0, N(B_4, 4) = 0$

```
> NBi(5)
[1] 1 0 0 0
```

$N(B_1, 5) = 1, N(B_2, 5) = 0, N(B_3, 5) = 0, N(B_4, 5) = 0$

Having obtained the quantities $N(B, i)$'s, the membrane boundaries can be calculated.

bnd(REGION, i, BASE, SU) – Calculating membrane boundary
Parameters: regions in matrix form **REGION**, **i**th region, base system **BASE**; the universe **SU**.
Description: This function calculates the boundary of the ith region, i.e., the ith membrane boundary.

Example 4.3. Calculating all membrane boundaries

```
> bnd(R,1,B,U)
[1] 0 0 1 1 1
```

$\mathsf{bnd}(W_1) = cde$

```
> bnd(R,2,B,U)
[1] 0 0 1 1 1
```

$\mathsf{bnd}(W_2) = cde$

```
> bnd(R,3,B,U)
[1] 0 0 3 3 3
```

$\mathsf{bnd}(W_1) = c^3d^3e^3$

```
> bnd(R,4,B,U)
[1] 0 0 0 0 0
```

$\mathsf{bnd}(W_1) = \emptyset$

```
> bnd(R,5,B,U)
[1] 2 0 0 0 0
```

$\mathsf{bnd}(W_1) = a^2$

Last, the outside/inside membrane boundaries are calculated.

bndout(REGION, i, BASE, SU) – Calculating outside membrane boundary
Parameters: regions in matrix form **REGION**, **i**th region, base system **BASE**;

the universe **SU**.
Description: This function calculates the outside boundary of the ith region, i.e., the ith outside membrane boundary.

Example 4.4. Calculating all outside membrane boundaries

```
> bndout(R,1,B,U)
[1] 0 0 0 1 0
```
$\mathsf{bnd}^{\mathsf{out}}(W_1) = d$

```
> bndout(R,2,B,U)
[1] 0 0 1 0 1
```
$\mathsf{bnd}^{\mathsf{out}}(W_2) = ce$

```
> bndout(R,3,B,U)
[1] 0 0 0 0 3
```
$\mathsf{bnd}^{\mathsf{out}}(W_3) = e^3$

```
> bndout(R,4,B,U)
[1] 0 0 0 0 0
```
$\mathsf{bnd}^{\mathsf{out}}(W_4) = \emptyset$

```
> bndout(R,5,B,U)
[1] 1 0 0 0 0
```
$\mathsf{bnd}^{\mathsf{out}}(W_5) = a$

bndin(REGION, i, BASE, SU) – Calculating inside membrane boundary
Parameters: regions in matrix form **REGION**, **i**th region, base system **BASE**; the universe **SU**.
Description: This function calculates the inside boundary of the ith region, i.e., the ith inside membrane boundary.

Example 4.5. Calculating all inside membrane boundaries

```
> bndin(R,1,B,U)
[1] 0 0 1 0 1
```
$\mathsf{bnd}^{\mathsf{in}}(W_1) = ce$

```
> bndin(R,2,B,U)
[1] 0 0 0 1 0
```
$\mathsf{bnd}^{\mathsf{in}}(W_2) = d$

```
> bndin(R,3,B,U)
[1] 0 0 3 3 0
```
$\mathsf{bnd}^{\mathsf{in}}(W_3) = c^3d^3$

```
> bndin(R,4,B,U)
[1] 0 0 0 0 0
```
$\mathsf{bnd}^{\mathsf{in}}(W_4) = \emptyset$

```
> bndin(R,5,B,U)
[1] 1 0 0 0 0
```
$\mathsf{bnd}^{\mathsf{in}}(W_5) = a$

5. Summary

In this paper such R functions have been presented which allow us to carry out calculations in membrane systems combined with multiset approximation spaces. In this framework membrane boundaries (even inside and outside) can be determined. The calculations are illustrated with examples mainly coming from [6, 7]. The results presented in this paper also prove the usability of R language in membrane computing. Further research direction may be the implementation of membrane communication rules in R in order to show how maximal parallelism can actually be controlled with the help of generated membrane boundaries [1, 6].

References

[1] Csajbók, Z.E., Mihálydeák, T., Maximal parallelism in membrane systems with generated membrane boundaries. In: Beckmann, A., Csuhaj-Varjú, E., Meer, K. (eds.) Language, Life, Limits. 10th Conference on Computability in Europe, CiE 2014, Budapest, Hungary, June 23-27, 2014. Proceedings. LNCS, vol. 8493 (2014), Springer International Publishing, Switzerland, 103–112.

[2] Girish, K.P., John, S.J., Relations and functions in multiset context. Information Sciences **179**(6) (2009), 758–768.

[3] Kudlek, M., Martín-Vide, C., Păun, Gh., Toward a formal macroset theory. In Calude, C., Păun, Gh., Rozenberg, G., Salomaa, A., eds.: WMP. LNCS, vol. 2235 (2001), Berlin Heidelberg, Springer-Verlag, 123–134.

[4] Mihálydeák, T., Csajbók, Z.E., Membranes with boundaries. In: Csuhaj-Varjú, E., Gheorghe, M., Rozenberg, G., Salomaa, A., Vaszil, Gy. (eds.) Membrane Computing. CMC 2012, Budapest, Hungary, August 28-31, 2012, Revised Selected Papers. LNCS, vol. 7762 (2013), Springer-Verlag, Berlin Heidelberg, 277–294.

[5] Mihálydeák, T., Csajbók, Z.E., Partial approximation of multisets and its applications in membrane computing. In: Lingras, P., Wolski, M., Cornelis, C., Mitra, S., Wasilewski, P. (eds.) Rough Sets and Knowledge Technology, 8th International Conference, RSKT 2013, Halifax, NS, Canada, October 11-14, 2013, Proceedings. LNCS-LNAI, vol. 8171 (2013), Springer-Verlag, Berlin, Heidelberg, 99–108.

[6] Mihálydeák, T., Csajbók, Z.E., Takács, P., On the Membrane Computations in the Presence of Membrane Boundaries. Journal of Automata, Languages and Combinatorics 19(1-4) (2014), 227–238.

[7] Mihálydeák, T., Csajbók, Z.E., Takács, P., Communication rules controlled by generated membrane boundaries. In: Alhazov, A., Cojocaru, S., Gheorghe, M., Rogozhin, Y., Salomaa, A. (eds.) Membrane Computing, 14th International Conference, CMC 2013, Chişinău, Republic of Moldova, August 20-23, 2013, Revised Selected Papers. LNCS, vol. 8340 (2014), Springer, Berlin Heidelberg, 265–279.

[8] Păun, Gh., Computing with membranes. Journal of Computer and System Sciences 61(1) (2000), 108–143

[9] Păun, Gh., Membrane Computing. An Introduction. Springer-Verlag, Berlin, (2002)

[10] Păun, Gh., Rozenberg, G., An introduction to and an overview of membrane computing. In: Păun et al. [11], 1–27.

[11] Păun, Gh., Rozenberg, G., Salomaa, A., The Oxford Handbook of Membrane Computing. Oxford University Press, Inc., New York, NY, USA (2010)

[12] Pawlak, Z., Rough sets. International Journal of Computer and Information Sciences 11(5) (1982), 341–356.

[13] Pawlak, Z., Rough Sets: Theoretical Aspects of Reasoning about Data. Kluwer Academic Publishers, Dordrecht (1991)

[14] Syropoulos, A., Mathematics of Multisets, *Multiset Processing, Mathematical, Computer Science, and Molecular Computing Points of View,* Workshop on Multiset Processing. Curtea de Arges, Romania, August 21-25, 2000. (2000), 347–358.

[15] Meyer, D., Hornik, K., Generalized and Customizable Sets in R. *Journal of Statistical Software*, Vol. 31 (2009), 1–27.