Contents

ANNALES MATHEMATICAE ET INFORMATICAE 51. (2020)

ANNALES MATHEMATICAE ET INFORMATICAE

TOMUS 51. (2020)

EGER 1774
ESZTERHÁZY KÁROLY UNIVERSITY

HUNGARIA, EGER

ANNALES MATHEMATICAE ET INFORMATICAE

VOLUME 51. (2020)

INSTITUTE OF MATHEMATICS AND INFORMATICS
ESZTERHÁZY KÁROLY UNIVERSITY
HUNGARY, EGER

Selected papers of the 11^{th} International Conference on Applied Informatics

HU ISSN 1787-6117 (Online)

A kiadásért felelős az
Eszterházy Károly Egyetem rektora
Megjelent a Líceum Kiadó gondozásában
Kiadóvezető: Dr. Nagy Andor
Felelős szerkesztő: Dr. Domonkosi Ágnes
Műszaki szerkesztő: Dr. Tómács Tibor
Megjelent: 2020. július

Research papers

Annales Mathematicae et Informaticae
51 *(2020) pp. 7–15*
DOI: 10.33039/ami.2020.07.005
https://ami.uni-eszterhazy.hu

Simulation results on a triangle-based network evolution model*

István Fazekas, Attila Barta, Csaba Noszály

University of Debrecen
fazekas.istvan@inf.unideb.hu
barta.attila@inf.unideb.hu
noszaly.csaba@inf.unideb.hu

Submitted: February 4, 2020
Accepted: July 1, 2020
Published online: July 23, 2020

Abstract

We study a continuous time network evolution model. We consider the collaboration of three individuals. In our model, it is described by three connected vertices, that is by a triangle. During the evolution new collaborations, that is new triangles are created. The reproduction of the triangles is governed by a continuous time branching process. The long time behaviour of the number of triangles, edges and vertices is described. In this paper, we highlight the asymptotic behaviour of the network by simulation results.

Keywords: random graph, network, branching process, Malthusian parameter

MSC: 05C80, 90B15, 60J85

1. Introduction

In the past two decades network science became a popular and important topic, see [2]. It describes large real-life networks as the Internet, the WWW, social, biological and energy networks. Large networks have several common properties, therefore it is worth to study theoretical models of networks. Usually, networks are described by graphs. The nodes of the network are the vertices of the graph

*This work was supported by the construction EFOP-3.6.3-VEKOP-16-2017-00002. The project was supported by the European Union, co-financed by the European Social Fund.

and the connections are the edges. The meaning of connection can be cooperation or any interaction. A most cited paper in network science is [3]. It studies the famous preferential attachment model which leads to scale free networks. A deep mathematical study of discrete time network evolution models can be found in [6].

However, in our paper, we turn to a continuous time network evolution model. An interesting continuous time model is presented in [7]. In that paper the theory of general branching processes, so called Crump-Mode-Jagers processes (see [8]) is applied to obtain asymptotic theorems. In paper [9], the idea of preferential attachment is combined with the evolution mechanism of a multi-type continuous time branching process.

In this paper we apply certain ideas of papers [1] and [7]. Paper [1] describes the interaction (or co-operation) of three persons. It creates a discrete time network evolution model which relies on the preferential attachment rule and three-interactions. In [7], however, a continuous time network evolution model based on a branching process evolution rule is presented. In that model only the usual interaction of two vertices is included and triangles have no role in the evolution rule. Neither [1] nor [7] offer numerical results. In our paper we combine the above ideas of three interactions with the continuous time branching process evolution mechanism. We focus on numerical studies of our model.

In this paper, in Section 2, we offer a detailed description of the evolution rules of our network. Then, in Section 3, we give a brief summary of our theoretical results. Their detailed mathematical proofs are given in a separate paper (see [5]). Here, in Section 4, we present our numerical results. We show that our formulae are numerically tractable, so we can calculate the values of the important parameters and other features of our process. Then we show our simulation program and a certain part of our simulation results. These results support our mathematical theorems.

2. The network evolution model

We shall study the following evolving random graph model. At the initial time $t = 0$ we start with a single triangle. During the evolution new triangles are born. Every triangle has its own evolution process. We assume that during the evolution of the network the life processes of the triangles are identically distributed and independent of each other.

We denote the reproduction process of the generic triangle by $\xi(t)$ and its birth times by $\tau_1, \tau_2, \ldots$. We assume that $\tau_1, \tau_2, \ldots$ are the jumping time points of a Poisson process $\Pi(t), t \geq 0$, where the rate of Π is equal to 1. Then the point process $\xi(t)$ gives the total number of offspring up to time t. However, at a birth time not only triangles can be created but new edges and vertices can be added to the graph. Here we describe the details of an evolution step. At every birth time τ_i a new vertex is added to the graph which can be connected to its ancestor triangle with j edges, where $j = 0, 1, 2, 3$. The vertices of the ancestor triangle to be connected to the new vertex are chosen uniformly at random. Let p_j denote the probability that

the new vertex will be connected to j vertices of the ancestor triangle. It follows from the definition of the evolution process that at each birth step the possible number of the new triangles can be $0, 1$ or 3. On Figure 1 we represent these possibilities. The initial triangle is drawn by solid lines while the new ingredients by dashed lines. Denote the litter sizes belonging to the birth times $\tau_1, \tau_2, \dots$ by

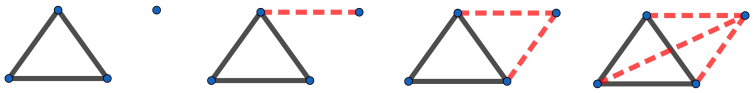


Figure 1: Possible birth events (0, 1, 2 or 3 new edges)

$\varepsilon_1, \varepsilon_2, \dots$. Then $\varepsilon_1, \varepsilon_2, \dots$ are independent identically distributed discrete random variables with distribution $\mathbb{P}(\varepsilon_i = j) = q_j, j \geq 0$. In our model the distribution of the litter size ε_i is given by

$$\mathbb{P}(\varepsilon_i = 0) = q_0 = p_0 + p_1, \ \mathbb{P}(\varepsilon_i = 1) = q_1 = p_2, \ \mathbb{P}(\varepsilon_i = 3) = q_3 = p_3,$$

$$\mathbb{P}(\varepsilon_i = j) = q_j = 0, \quad \text{if} \quad j \notin \{0, 1, 3\}.$$

We assume that the litter sizes are independent of the birth times $\tau_1, \tau_2, \dots$, too. Denote by λ the life length of the generic triangle. λ is a finite nonnegative random variable. After its death the triangle does not produce offspring, therefore $\xi(t) = \xi(\lambda)$ when $t > \lambda$. Then the reproduction process of a triangle is

$$\xi(t) = \sum_{\tau_i \leq t \wedge \lambda} \varepsilon_i = S_{\Pi(t \wedge \lambda)},$$

where $S_n = \varepsilon_1 + \dots + \varepsilon_n$ gives the total number of offspring before the $(n+1)$th birth event and $x \wedge y$ denotes the minimum of $\{x, y\}$.

Let $L(t)$ be the distribution function of λ. We assume that the survival function of the triangle's life length is

$$1 - L(t) = \mathbb{P}(\lambda > t) = \exp\left(- \int_0^t l(u) \, \mathrm{d}u \right),$$

where $l(t)$ is the hazard rate of the life span λ. Moreover, we assume that the hazard rate depends on the number of offspring as

$$l(t) = b + c\xi(t)$$

with positive constants b and c.

The whole evolution process is the following. The life and the reproduction process of the initial triangle is the same as that of the above described generic triangle. When a child triangle is born, then it starts its own life and reproduction process which is also defined by the same way as its parent triangle. The same applies to

the grandchildren, etc. Therefore the evolution of the network is described by a continuous time branching process. We underline that the life and reproduction process of any triangle have the same distribution as those of the generic triangle, but the reproduction processes of different triangles are independent.

3. Theoretical results

Here we summarize the theoretical results of our paper [5].

Let $\mu(t) = \mathbb{E}\xi(t)$ be the expectation of the number of offspring of a triangle up to time t. The total number of offspring of a triangle is $\xi(\infty)$. The expected offspring number of a triangle can be calculated as

$$\mu(\infty) = \mathbb{E}\xi(\infty) = (q_1 + 3q_3)\mathbb{E}(\lambda) =$$

$$(q_1 + 3q_3)\frac{1}{c}\int_0^1 (1-u)^{\frac{b+1-q_0}{c}-1} e^{\frac{u}{3c}(q_3u^2 - 3q_3u + 3(q_1+q_3))}\,\mathrm{d}u.$$

The probability of extinction is 1 if $\mu(\infty) \leq 1$.

Theorem 3.1. *If $\mu(\infty) > 1$, then the probability of the extinction of the triangles is the smallest non-negative solution of equation*

$$\frac{q_1 + q_3(y^2+y+1)}{c}\int_0^1 (1-u)^{\frac{1+b-q_0}{c}-1} e^{(\frac{q_1y+q_3y^3}{c}u - \frac{q_3y^3}{c}u^2 + \frac{q_3y^3}{3c}u^3)}\,\mathrm{d}u = 1. \quad (3.1)$$

Assume that $\mu(\infty) > 1$, that is our branching process is supercritical. Then the Malthusian parameter α is the only positive solution of equation $\int_0^\infty e^{-\alpha t}\mu(dt) = 1$. We can see that

$$q_1 + 3q_3 - b - 1 < \alpha < q_1 + 3q_3 - b.$$

In our model the Malthusian parameter α satisfies the equation

$$1 = \frac{(q_1 + 3q_3)}{c}\int_0^1 (1-u)^{\frac{\alpha+(b+1)}{c} - \frac{q_0}{c} - 1} e^{\frac{3q_1u + q_3u(u^2-3u+3)}{3c}}\,\mathrm{d}u. \quad (3.2)$$

Now we give the asymptotic behaviour of the number of triangles. Let us denote by $Z(t)$ the number of triangles alive at time t. Let α be the Malthusian parameter.

Theorem 3.2. *We have*

$$\lim_{t\to\infty} e^{-\alpha t}Z(t) = Y_\infty m_\infty$$

almost surely and in L^1, where the random variable Y_∞ is nonnegative, it is positive on the event of non-extinction, it has expectation 1 and

$$m_\infty = \frac{1}{(q_1+3q_3)^2\int_0^\infty te^{-\alpha t}(1-L(t))dt}.$$

Now we turn to the asymptotic behaviour of vertices and edges. Let us denote by $V(t)$ the total number of vertices (dead or alive) up to time t. Let $W(t)$ be the number of edges (dead or alive) up to time t. Let γ denote the number of new edges at a birth. Then its distribution is $\mathbb{P}(\gamma = j) = p_j,\ j = 0, 1, 2, 3$.

Theorem 3.3. *We have*

$$\frac{V(t)}{Z(t)} \to \frac{1}{\alpha} \qquad \textit{and} \qquad \frac{W(t)}{Z(t)} \to \frac{\mathbb{E}\gamma}{\alpha}$$

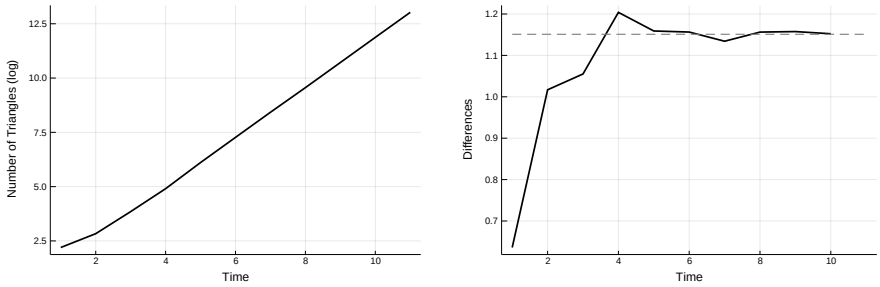
as $t \to \infty$ almost surely on the event of non-extinction.

4. Numerical and simulation results

To get a closer look on the theoretical results, we made some simulations about them. We generated our code in Julia language [4]. We chose Julia, because of the great implementation of priority queues. The simulation time of our code was significantly faster in Julia than in other programming languages. We handled the main objects (the triangles) of our model as arrays with 3 elements. The elements were the indices of the edges that formed an individual for the process. We put all triangles in a priority queue with the priority of its birth time, because we can pop out the element with the lowest priority. After we have got the triangle with the lowest birth time, we can handle its birth process with the predefined parameters b, c, q_1, q_3. In the birth process we generated an exponential time step for the next birth step of our triangle. After that we checked if the triangle is still alive by calculating the survival function. If the triangle is dead, we move to the next one. If it is alive, then we generate 1 or 3 new triangles and put them into the priority queue with the calculated birth time priorities. After this step we moved to the next birth event. The pseudocode of the birth process is seen at Algorithm 1.

We made several simulation experiments. Here we show only some typical results. For the above demonstration first we used the parameter set $b = 0.2,\ c = 0.2,\ p_0 = 0.05,\ p_1 = 0.05,\ p_2 = q_1 = 0.6,\ p_3 = q_3 = 0.3$. On Figure 2a the solid curve shows the number of triangles. According to Theorem 3.2 it has asymptotic rate $e^{\alpha t}$. Therefore we put logarithmic scale on the vertical axis, so the function $Z(t)$ is a straight line for large values of t. On the figure one can see that the shape of the curve is close to a straight line, so it supports our Theorem 3.2.

Then we checked the value of the Malthusian parameter α. We can find it in two ways. On the one hand, the slope of the line on Figure 2a is α for large values of the time. This slope can be approximated by the differences of the function. So on Figure 2b we present these differences (solid line). On the other hand, α can be calculated numerically from equation (3.2). This α value is shown of Figure 2b by the horizontal dashed line. The fit of the differences to this α can be seen for large values of t. To get a closer look on the Malthusian parameter α, we fixed 5 parameter sets. Then we calculated α from equation (3.2) for each case. It is shown in the fifth column of Table 1. Then for each of the parameter sets we



(a) The number of triangles in the evolving graph

(b) The differences of the number of triangles

Figure 2: Simulation results for $b = 0.2,\ c = 0.2,\ q_1 = 0.6,\ q_3 = 0.3$

simulated our process $Z(t)$ five times. Then we calculated the differences of $\log Z(t)$ which should be good approximations of α according to Theorem 3.2. In Table 1, $\widehat{\alpha_1},\ \widehat{\alpha_2},\ \widehat{\alpha_3},\ \widehat{\alpha_4},\ \widehat{\alpha_5}$ show the values of these approximations for large t. One can see that each $\widehat{\alpha_i}$ is close to the corresponding α. We calculated numerically the

b	c	q_1	q_3	α	$\widehat{\alpha_1}$	$\widehat{\alpha_2}$	$\widehat{\alpha_3}$	$\widehat{\alpha_4}$	$\widehat{\alpha_5}$
0.2	0.4	0.7	0.1	0.5628	0.5651	0.5730	0.5701	0.5611	0.5594
0.2	0.4	0.8	0.1	0.6531	0.6537	0.6497	0.6570	0.6510	0.6589
0.4	0.4	0.8	0.1	0.4531	0.4503	0.4519	0.4584	0.4541	0.4524
0.4	0.4	0.7	0.2	0.6545	0.6533	0.6517	0.6548	0.6534	0.6574
0.4	0.4	0.6	0.3	0.8535	0.8519	0.8489	0.8559	0.8547	0.8566

Table 1: α from equation (3.2) and $\widehat{\alpha_i}$ from simulations

probability of extinction from equation (3.1). It is shown in the column 'Numerical' of Table 2. In the column 'Simulation' the relative frequency of the extinction is shown using our computer experiment. For each parameter sets, we simulated 10^4 processes and counted the number of extinctions occured. The value of the relative frequency is close to the corresponding value of the probability in each case. So Table 2 supports the result of Theorem 3.1. To investigate how our difference process approximates α for large values of time t, we simulated around 500 independent processes with the same parameters $b = 0.2,\ c = 0.2,\ p_0 = 0.05,\ p_1 = 0.05,\ p_2 = q_1 = 0.6,\ p_3 = q_3 = 0.3$ and same running time. Then we checked the differences of the last two values in the numbers of triangles that we simulated and made a histogram, seen in Figure 3. From equation (3.2) we obtained that the value of α is 0.3365. We see that the values of the differences are in $[0.332, 0.340]$, so they are very close to 0.3365.

b	c	q_1	q_3	**Simulation**	**Numerical**
0.0	0.2	0.4	0.4	0.0	0.0
0.1	0.2	0.4	0.4	0.1304	0.1282
0.1	0.2	0.5	0.4	0.1165	0.1158
0.1	0.2	0.5	0.5	0.1097	0.1025
0.2	0.2	0.5	0.4	0.2227	0.2180
0.2	0.2	0.6	0.4	0.2038	0.2002
0.3	0.3	0.5	0.4	0.3231	0.3185
0.4	0.4	0.5	0.4	0.3966	0.4020

Table 2: The relative frequency and the probability of the extinction of the triangles

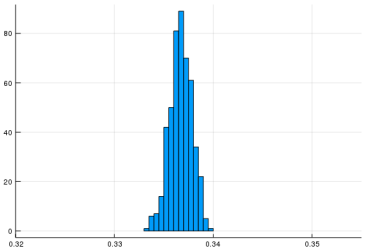
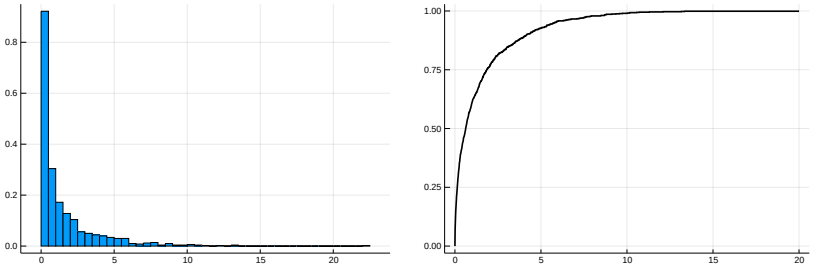


Figure 3: Histogram of differences

To get some information about the random variable $Y_\infty m_\infty$ represented in Theorem 3.2, we calculated the value of $Z(t)e^{-\alpha t}$ for 10^3 independent repetitions of the process for the same time t and same parameters $q_1 = 0.3,\ q_3 = 0.6,\ b = 0.2,\ c = 0.2$. On Figure 4 we represent the histogram and the empirical cumulative distribution function (ECDF) calculated from the simulation. We obtained that the empirical cumulative distribution function of $Y_\infty m_\infty$ fits to a gamma distribution, as the Kolmogorov–Smirnov test gave us p value 0.6713.



(a) Histogram of $Z(t)e^{-\alpha t}$ (b) ECDF of $Z(t)e^{-\alpha t}$

Figure 4: Simulation results for $Z(t)e^{-\alpha t}$

5. Summary

In this paper and in [5] we offer a network evolution model which describes collaborations of 3 persons. Our model grasps certain features of real collaborations as emerging and disappearing collaborations, moreover collaborations of 2 persons are also allowed. Our numerical results confirm the theoretical ones. The present results prepare our future research on more complicated collaborations.

Algorithm 1 Birth process of a triangle

```
1:  procedure BIRTH PROCESS
2:      Y ← non-empty Priority Queue
3:      b, c, q_1, q_3 ← parameters of the survival function
4:      x ← dequeue Y
5:      if x is a new triangle then
6:          t_0 ← the birth time of x in the whole process
7:          t ← 0, lifetime of x
8:          l ← 1, life variable
9:          while l = 1 do
10:             t ← t + Exp(1)
11:             p ← the calculated survival function
12:             if p > Uni(0, 1) then
13:                 p_0 ← Uni(0, 1)
14:                 if p_0 < q_1 then
15:                     take a new triangle with t_0 + t birth time to Y
16:                     offspring number is 1 at birth time t
17:                 else if p_0 > 1 − q_3 then
18:                     take three new triangles with t_0 + t birth times to Y
19:                     offspring number is 3 at birth time t
20:             else
21:                 l ← 0
22:                 take t as the death time of x to Y
```

References

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

[2] A.-L. Barabási: *Network science*, Cambridge: Cambridge University Press, 2018.

[3] A.-L. Barabási, R. Albert: *Emergence of scaling in random networks*, Science 286.5439 (1999), pp. 509–512,
doi: https://doi.org/10.1126/fscience.286.5439.509.

[4] J. Bezanson, A. Edelman, S. Karpinski, V. B. Shah: *Julia: A Fresh Approach to Numerical Computing*, 2014, arXiv: 1411.1607 [cs.MS].

[5] I. Fazekas, A. Barta, C. Nószály, B. Porvázsnyik: *A continuous time evolution model describing 3-interactions*, in: Manuscript, Debrecen, Hungary, 2020.

[6] R. van der Hofstad: *Random Graphs and Complex Networks Vol. 1.* Cambridge: Cambridge Series in Statistical and Probabilistic Mathematics, 2017,
doi: https://doi.org/10.1017/9781316779422.

[7] T. F. Móri, R. Sándor: *A random graph model driven by timedependent branching dynamics*, Ann. Univ. Sci. Budapest, Sect. Comp. 46.3 (2017), pp. 191–213.

[8] O. Nerman: *On the convergence of supercritical general (C-M-J) branching processes*, Z. Wahrscheinlichkeitstheorie verw Gebiete 57 (1981), pp. 365–395,
doi: https://doi.org/10.1007/BF00534830.

[9] S. Rosengren: *A Multi-type Preferential Attachment Tree*, Internet Math. 2018 (2018).

Annales Mathematicae et Informaticae
***51** (2020) pp. 17–28*
DOI: 10.33039/ami.2020.07.001
https://ami.uni-eszterhazy.hu

Numerical results on noisy blown-up matrices*

István Fazekas, Sándor Pecsora

University of Debrecen, Faculty of Informatics
fazekas.istvan@inf.unideb.hu
pecsora.sandor@inf.unideb.hu

Submitted: February 4, 2020
Accepted: July 1, 2020
Published online: July 23, 2020

Abstract

We study the eigenvalues of large perturbed matrices. We consider an Hermitian pattern matrix P of rank k. We blow up P to get a large block-matrix B_n. Then we generate a random noise W_n and add it to the blown up matrix to obtain the perturbed matrix $A_n = B_n + W_n$. Our aim is to find the eigenvalues of B_n. We obtain that under certain conditions A_n has k 'large' eigenvalues which are called structural eigenvalues. These structural eigenvalues of A_n approximate the non-zero eigenvalues of B_n. We study a graphical method to distinguish the structural and the non-structural eigenvalues. We obtain similar results for the singular values of non-symmetric matrices.

Keywords: Eigenvalue, symmetric matrix, blown-up matrix, random matrix, perturbation of a matrix, singular value

MSC: 15A18, 15A52

1. Introduction

Spectral theory of random matrices has a long history (see e.g. [1, 5–8] and the references therein). This theory is applied when the spectrum of noisy matrices is

*This work was supported by the construction EFOP-3.6.3-VEKOP-16-2017-00002. The project was supported by the European Union, co-financed by the European Social Fund.

considered. In [2] and [3] the eigenvalues and the singular values of large perturbed block matrices were studied. In [4] we extended the results of [2] and [3] and we suggested a graphical method to study the asymptotic behaviour of the eigenvalues. In this paper we consider the following important question for the numerical behaviour of the eigenvalues. How large the blown-up matrix should be in order to show the asymptotic behaviour given in the mathematical theorems? We also study the influence of the signal to noise ratio for our results.

Thus we consider a fixed deterministic pattern matrix P, we blow up P to obtain a 'large' block-matrix B_n, then we add a random noise matrix W_n. We present limit theorems for the eigenvalues of $A_n = B_n + W_n$ as $n \to \infty$ and show corresponding numerical results. We also consider a graphical method to distinguish the structural and the non-structural eigenvalues. This test is important, because in real-life only the perturbed matrix A_n is observed, but we are interested in eigenvalues or the singular values of B_n which are approximated by the above mentioned structural eigenvalues/singular values of A_n.

In Section 2 we list some theoretical results of [4]. In Section 3 the numerical results are presented. We obtained that the asymptotic behaviour of the eigenvalues can be seen for relatively low values of n, that is if the block sizes are at least 50, then we can use our asymptotic results. We also study the influence of the signal/noise ratio on the gap between the structural and non-structural singular values. Our numerical results support our graphical test visualised by Figure 1.

2. Eigenvalues of perturbed symmetric matrices

In this section we study the perturbations of Hermitian (resp. symmetric) blown-up matrices. We would like to examine the eigenvalues of perturbed matrices.

We use the following notation:

- P is a fixed complex Hermitian (in the real valued case symmetric) $k \times k$ pattern matrix of rank r
- p_{ij} is the (i,j)'th entry of P
- $n_1, \dots, n_k$ are positive integers, $n = \sum_{i=1}^{k} n_i$
- $\tilde{B}_n$ is an $n \times n$ matrix consisting of k^2 blocks, its block (i,j) is of size $n_i \times n_j$ and all elements in that block are equal to p_{ij}
- B_n is called a blown-up matrix if it can be obtained from $\tilde{B}_n$ by rearranging its rows and columns using the same permutation

Following [2], we shall use the growth rate condition

$$n \to \infty \quad \text{so that} \quad n_i/n \geq c \quad \text{for all} \quad i, \tag{2.1}$$

where $c > 0$ is a fixed constant. Here we list those theorems of [4] which will be tested by numerical methods.

Proposition 2.1. *Let P be a symmetric pattern matrix, that is a fixed complex Hermitian (in the real valued case symmetric) $k \times k$ matrix of rank r. Let B_n be the blown-up matrix of P. Then B_n has r non-zero eigenvalues. If condition* (2.1) *is satisfied, then the non-zero eigenvalues of B_n are of order n in absolute value.*

Now, we assume that $rank(P) = k$. We shall consider the eigenvalues in descending order, so we have $|\lambda_1(B_n)| \geq \cdots \geq |\lambda_n(B_n)|$. Since k eigenvalues of B_n are non-zero and the remaining ones are equal to zero, we shall call the first k ones structural eigenvalues of B_n. Similarly, we shall call structural eigenvalue that eigenvalue of A_n, which corresponds to a structural eigenvalue of B_n. This correspondence will be described by Theorem 2.2 and Corollary 2.3. We shall see that the magnitude of any structural eigenvalue is large and it is small for the other eigenvalues.

First we consider perturbations by Wigner matrices. Next theorem is a generalization of Theorem 2.3 of [2] where the real valued case and uniformly bounded perturbations were considered.

Theorem 2.2. *Let B_n, $n = 1, 2, \ldots$, be a sequence of complex Hermitian matrices. Let the Wigner matrices W_n, $n = 1, 2, \ldots$, be complex Hermitian $n \times n$ random matrices satisfying the following assumptions. Let the diagonal elements w_{ii} of W_n be i.i.d. (independent and identically distributed) real, let the above diagonal elements be i.i.d. complex random variables and let all of these be independent. Let W_n be Hermitian, that is $w_{ij} = \bar{w}_{ji}$ for all i, j. Assume that $\mathbb{E}w_{11}^2 < \infty$, $\mathbb{E}w_{12} = 0$, $\mathbb{E}|w_{12} - \mathbb{E}w_{12}|^2 = \sigma^2$ is finite and positive, $\mathbb{E}|w_{12}|^4 < \infty$. Then*

$$\limsup_{n\to\infty} \frac{|\lambda_i(B_n + W_n) - \lambda_i(B_n)|}{\sqrt{n}} \leq 2\sigma$$

for all i almost surely.

Corollary 2.3. *Let B_n, $n = 1, 2, \ldots$, be blown-up matrices of a complex Hermitian matrix P having rank k. Assume that condition* (2.1) *is satisfied. Let the Wigner matrices W_n, $n = 1, 2, \ldots$, satisfy the conditions of Theorem 2.2. Then Theorem 2.2 and Proposition 2.1 imply that $B_n + W_n$ has k eigenvalues of order n and the remaining eigenvalues are of order $\sqrt{n}$ almost surely.*

So the structural eigenvalues have magnitude n while the non-structural eigenvalues have magnitude $\sqrt{n}$.

3. Singular values of perturbed matrices

In this section we study the perturbations of arbitrary blown-up matrices. We are interested in the singular values of matrices perturbed by certain random matrices. We use the following notation:

- P is a fixed complex $a \times b$ pattern matrix of rank r

- p_{ij} is the (i,j)'th entry of P
- $m_1,\dots,m_a$ are positive integers, $m=\sum_{i=1}^{a} m_i$
- $n_1,\dots,n_b$ are positive integers, $n=\sum_{i=1}^{b} n_i$
- $\tilde{B}_n$ is an $m\times n$ matrix consisting of $a\times b$ blocks, its block (i,j) is of size $m_i\times n_j$ and all elements in that block are equal to p_{ij}
- B_n is called blown-up matrix if it can be obtained from $\tilde{B}$ by rearranging its rows and columns

Following [3], we shall use the growth rate condition

$$m,n\to\infty \ \text{ so that } \ m_i/m\geq c \ \text{ and } n_i/n\geq d \ \text{ for all } \ i, \tag{3.1}$$

where $c,d>0$ are fixed constants. The following proposition is an extension of Proposition 6 of [3] to the complex valued case.

Proposition 3.1. *Let P be a fixed complex $a\times b$ matrix of rank r. Let B be the $m\times n$ blown-up matrix of P. If condition* (3.1) *is satisfied, then the non-zero singular values of B are of order $\sqrt{mn}$.*

Now we consider perturbation with matrices having independent and identically distributed (i.i.d.) complex entries. Let x_{jk}, $j,k=1,2,\dots,$ be an infinite array of i.i.d. complex valued random variables with mean 0 and variance σ^2. Let $X=(x_{jk})_{j=1,\ k=1}^{m,\ n}$ be the left upper block of size $m\times n$.

Theorem 3.2. *For each m and n let $B=B_{mn}$ be a complex matrix of size $m\times n$ and let $X=X_{mn}$ be the above complex valued random matrix of size $m\times n$ with i.i.d. entries. Moreover, assume that the entries of X have finite fourth moments. Assume that $m,n\to\infty$ so that $K_1\leq\frac{m}{n}\leq K_2$, where $0<K_1\leq K_2<\infty$ are fixed constants. Denote by s_i and z_i the singular values of B and $B+X$, respectively, $s_1\geq\cdots\geq s_{\min\{m,n\}}$, $z_1\geq\cdots\geq z_{\min\{m,n\}}$. Then for all i*

$$|s_i-z_i|=\mathrm{O}(\sqrt{n})$$

as $m,n\to\infty$ almost surely.

Corollary 3.3. *Proposition 3.1 and Theorem 3.2 imply the following. Let P be a fixed complex $a\times b$ matrix of rank r. Let B be the $m\times n$ blown-up matrix of P. Assume that condition* (3.1) *is satisfied. Let X be complex valued perturbation matrices satisfying the assumptions of Theorem 3.2. Denote by z_i the singular values of $B+X$, $z_1\geq\cdots\geq z_{\min\{m,n\}}$. Then z_i are of order n for $i=1,\dots,r$ and $z_i=\mathrm{O}(\sqrt{n})$ for $i=r+1,\dots,\min\{m,n\}$ almost surely as $m,n\to\infty$ so that $K_1\leq\frac{m}{n}\leq K_2$, where $0<K_1\leq K_2<\infty$ are fixed constants.*

So the structural singular values (i.e. the largest r values) are 'large', and the remaining ones are 'small'.

4. Numerical results

4.1. Eigenvalues of a symmetric matrix perturbed with Wigner noise

In [4] we concluded the following facts. In the case when the perturbation matrix has zero mean random entries, then the structural eigenvalues are 'large' and the non-structural ones are 'small'. More precisely let $|\lambda_1| \geq |\lambda_2| \geq \dots$ be the absolute values of the eigenvalues of the perturbed blown-up matrix in descending order. Then the structural eigenvalues $|\lambda_1| \geq |\lambda_2| \geq \cdots \geq |\lambda_l|$ are 'large' and they rapidly decrease. The other eigenvalues $|\lambda_{l+1}| \geq |\lambda_{l+2}| \geq \dots$ are relatively small and they decrease very slowly. To obtain the structural eigenvalues we can use the following numerical (graphical) procedure. Calculate some eigenvalues of A_n starting with the largest ones in absolute value. Stop when the last 5-10 eigenvalues are close to zero and they are almost the same in absolute value. Then we obtain the increasing sequence $|\lambda_t| \leq |\lambda_{t-1}| \leq \cdots \leq |\lambda_1|$. Plot their values in the above order, then find the first abrupt change. If, say,

$$0 \approx |\lambda_t| \approx |\lambda_{t-1}| \approx \cdots \approx |\lambda_{l+1}| \ll |\lambda_l| < \cdots < |\lambda_1|,$$

that is the first abrupt change is at l, then $\lambda_l, \lambda_{l-1}, \dots, \lambda_1$ can be considered as the structural eigenvalues. The typical abrupt change after the non-structural eigenvalues can be seen in Figure 1.

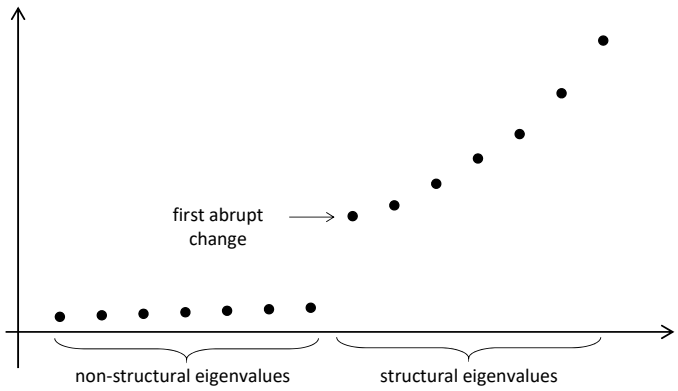


Figure 1: The abrupt change after the non-structural eigenvalues

Our first example supports Theorem 2.2 and Corollary 2.3 in the real valued case. The following results were obtained using the Julia programming language version 1.1.1. The simulations were divided into four steps. Let P be the real

symmetric pattern matrix

$$\begin{bmatrix} 8 & 7 & 2 & 5 & 3 & 2 & 4 & 0 & 3 & 1 \\ 7 & 9 & 6 & 3 & 4 & 0 & 2 & 5 & 2 & 0 \\ 2 & 6 & 7 & 6 & 5 & 4 & 2 & 0 & 3 & 4 \\ 5 & 3 & 6 & 8 & 7 & 6 & 0 & 5 & 4 & 2 \\ 3 & 4 & 5 & 7 & 9 & 8 & 8 & 6 & 5 & 1 \\ 2 & 0 & 4 & 6 & 8 & 7 & 6 & 8 & 0 & 4 \\ 4 & 2 & 2 & 0 & 8 & 6 & 9 & 7 & 6 & 6 \\ 0 & 5 & 0 & 5 & 6 & 8 & 7 & 8 & 8 & 4 \\ 3 & 2 & 3 & 4 & 5 & 0 & 6 & 8 & 9 & 6 \\ 1 & 0 & 4 & 2 & 1 & 4 & 6 & 4 & 6 & 5 \end{bmatrix}.$$

In the initial step we have matrix P. It has the following eigenvalues: 45.302, 15.497, 10.914, 7.677, −7.245, 6.188, 4.696, −3.789, −3.381 and 3.141. This matrix is blown up with the help of a vector containing the size informations of the blocks. If the vector is $\mathbf{n} = [n_1, n_2, \ldots, n_{10}]$, then the first row of blocks is built with the following block sizes: $n_1 \times n_1$, $n_1 \times n_2$, $\ldots n_1 \times n_{10}$, the second row of blocks: $n_2 \times n_1$, $n_2 \times n_2$, $\ldots n_2 \times n_{10}$ and we continue this pattern till the last row. In different simulations we used different vectors $\mathbf{n}$ to blow up the matrix, it will be detailed separately for each simulation.

The next step is to generate the noise matrix, which is a symmetric real Wigner matrix, as defined in Section 2. The entries are generated in the following way: let the diagonal elements w_{ii} of W_n be i.i.d. real with standard normal distribution, let the above diagonal elements be i.i.d. real random variables and let all of these be independent. The size of the noise matrix is equal to the size of the blown up matrix. After that, in order to obtain the noisy matrix, in each iteration we generate a new Wigner matrix and add to the blown up matrix.

As the last step we calculate the eigenvalues of the perturbed matrix. To do so we used the *eigvals* function from the *LinearAlgebra* package. Julia provides native implementations of many common and useful linear algebra operations which can be loaded with `using LinearAlgebra`, to install the package we used `using Pkg` and then with the `Pkg.add("LinearAlgebra")` command we installed the package.

We studied 4 different schemes to blow up matrix P, in each of these 4 schemes we applied 6 different block size vectors. These block series are given in Table 1.

We realised 1000 simulations for all block series in Table 1 and checked if the abrupt change like in Figure 1 was seen. For very low values of n_i we did not find the abrupt change. In each case the first well distinguishable abrupt change appeared when n_1 was 50. This result is shown in Figures 2, 3, 4, and 5. At each figure the left side: $|\lambda_{20}(A_n)| < \cdots < |\lambda_1(A_n)|$; right side: $|\lambda_{20}(A_n)| < \cdots < |\lambda_6(A_n)|$ in a typical realization in our example. The right side figures show only a part of the sequence around the change, so the jumps are clearly seen.

Each of the figures shows that there is an abrupt change between the 11^{th} and 10^{th} eigenvalues. So we can decide that the 10 largest eigenvalues are the structural

ones (which is the true value, since we used matrix P having rank 10).

Name of the scheme	Series of the blocks
Equal	$n_1 = n_2 = \cdots = n_{10} = 10$
	$n_1 = n_2 = \cdots = n_{10} = 20$
	$n_1 = n_2 = \cdots = n_{10} = 50$
	$n_1 = n_2 = \cdots = n_{10} = 100$
	$n_1 = n_2 = \cdots = n_{10} = 500$
	$n_1 = n_2 = \cdots = n_{10} = 1000$
Two types	$n_1 = n_2 = \cdots = n_5 = 10, n_6 = \cdots = n_{10} = 20$
	$n_1 = n_2 = \cdots = n_5 = 20, n_6 = \cdots = n_{10} = 40$
	$n_1 = n_2 = \cdots = n_5 = 50, n_6 = \cdots = n_{10} = 100$
	$n_1 = n_2 = \cdots = n_5 = 100, n_6 = \cdots = n_{10} = 200$
	$n_1 = n_2 = \cdots = n_5 = 500, n_6 = \cdots = n_{10} = 1000$
	$n_1 = n_2 = \cdots = n_5 = 1000, n_6 = \cdots = n_{10} = 2000$
Arithmetic progression	$n_1 = 10, n_2 = 20, \ldots, n_{10} = 100$
	$n_1 = 20, n_2 = 30, \ldots, n_{10} = 110$
	$n_1 = 50, n_2 = 60, \ldots, n_{10} = 140$
	$n_1 = 100, n_2 = 110, \ldots, n_{10} = 190$
	$n_1 = 500, n_2 = 510, \ldots, n_{10} = 590$
	$n_1 = 1000, n_2 = 1010, \ldots, n_{10} = 1090$
Four types	$n_1 = n_2 = 10, n_3 = n_4 = n_5 = 20,$ $n_6 = n_7 = n_8 = 30, n_9 = n_{10} = 40$
	$n_1 = n_2 = 20, n_3 = n_4 = n_5 = 40,$ $n_6 = n_7 = n_8 = 60, n_9 = n_{10} = 80$
	$n_1 = n_2 = 50, n_3 = n_4 = n_5 = 100,$ $n_6 = n_7 = n_8 = 150, n_9 = n_{10} = 200$
	$n_1 = n_2 = 100, n_3 = n_4 = n_5 = 200,$ $n_6 = n_7 = n_8 = 300, n_9 = n_{10} = 400$
	$n_1 = n_2 = 500, n_3 = n_4 = n_5 = 1000,$ $n_6 = n_7 = n_8 = 1500, n_9 = n_{10} = 2000$
	$n_1 = n_2 = 1000, n_3 = n_4 = n_5 = 2000,$ $n_6 = n_7 = n_8 = 3000, n_9 = n_{10} = 4000$

Table 1: Schemes used to blow up matrix P

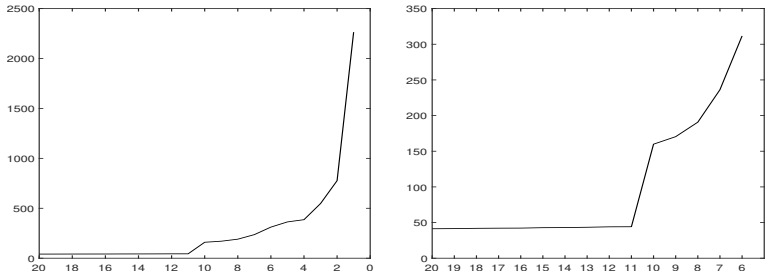


Figure 2: Equal: $n_1 = n_2 = \cdots = n_{10} = 50$

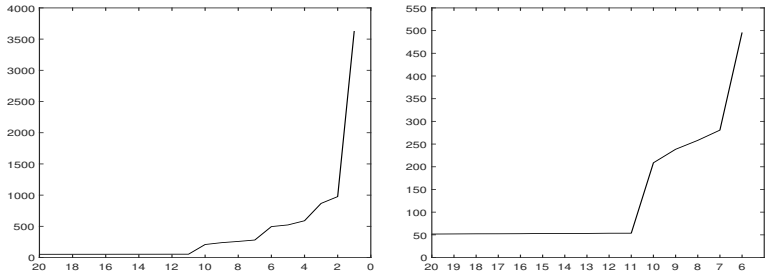


Figure 3: Two types: $n_1 = \cdots = n_5 = 50$, $n_6 = \cdots = n_{10} = 100$

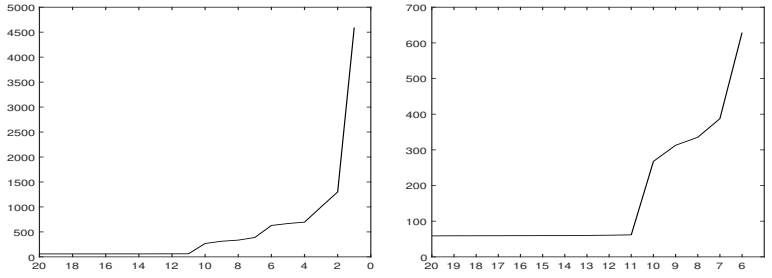


Figure 4: Arithmetic progression: $n_1 = 50, n_2 = 60, \ldots, n_{10} = 140$

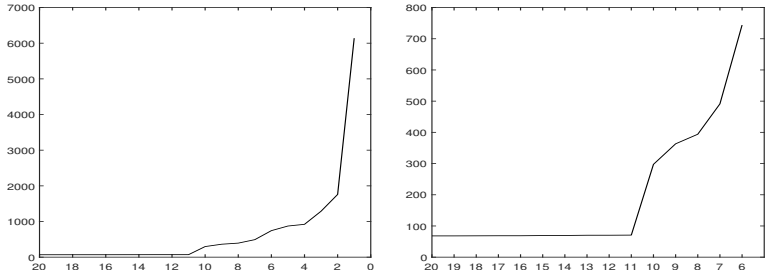


Figure 5: Four types: $n_1 = n_2 = 50, n_3 = n_4 = n_5 = 100$,
$n_6 = n_7 = n_8 = 150, n_9 = n_{10} = 200$

As the block's sizes are increasing, the border between the structural and non-structural eigenvalues is getting more and more significant, as it is shown in Figure 6. We see, that when $n_i = 1000$ (for all i), then the gap is much larger than in the case of $n_i = 50$.

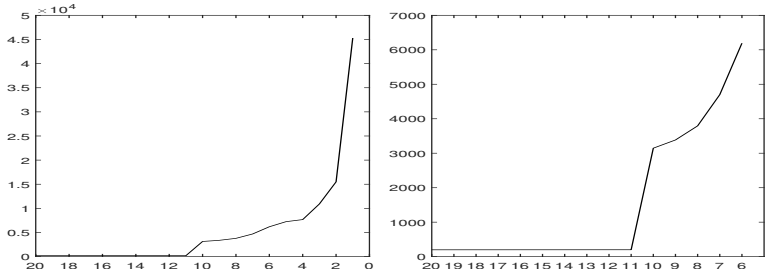


Figure 6: Equal: $n_1 = n_2 = \cdots = n_{10} = 1000$

In Table 2, we list the averages of the absolute values of the eigenvalues making 1000 repetitions in each case. More precisely, n_i means that all blocks were $n_i \times n_i$ and in the column below n_i there are the 20 largest of the averages of the absolute values of the corresponding eigenvalues of different A_n matrices. We see, that there is a gap between the 10^{th} and 11^{th} values even in the relatively small value of $n_i = 10$. The larger the value of n_i, the larger the gap. This table shows that our test is very reliable for moderate (e.g. $n_i = 50$) sizes of the blocks.

j	$n_i = 10$	$n_i = 20$	$n_i = 50$	$n_i = 100$	$n_i = 200$	$n_i = 500$	$n_i = 1000$
1	453.26	906.28	2265.40	4530.50	9060.70	22652.00	45303.00
2	155.64	310.65	775.43	1550.31	3100.00	7748.90	15497.00
3	110.06	219.28	546.65	1092.40	2183.90	5458.00	10915.00
4	77.99	154.82	385.08	768.93	1536.70	3839.60	7677.90
5	73.76	146.21	363.63	725.90	1450.40	3624.10	7246.90
6	63.36	125.32	310.93	620.41	1239.20	3095.40	6189.40
7	48.86	95.95	236.92	471.66	941.21	2349.90	4697.80
8	40.51	78.39	192.04	381.54	760.54	1897.10	3791.70
9	36.30	70.32	171.90	340.95	678.98	1693.20	3383.40
10	34.06	65.84	160.18	317.23	631.47	1573.90	3144.70
11	18.56	27.21	44.03	62.73	89.04	141.14	199.78
12	17.96	26.70	43.58	62.33	88.67	140.84	199.50
13	17.45	26.27	43.20	61.99	88.37	140.58	199.27
14	17.03	25.92	42.90	61.72	88.14	140.38	199.09
15	16.65	25.59	42.62	61.47	87.92	140.18	198.92
16	16.33	25.28	42.36	61.25	87.72	140.01	198.77
17	15.99	25.00	42.11	61.03	87.53	139.85	198.63
18	15.69	24.74	41.89	60.83	87.34	139.69	198.49
19	15.40	24.47	41.67	60.64	87.17	139.54	198.36
20	15.12	24.23	41.47	60.45	87.01	139.40	198.23

Table 2: Eigenvalues in the case of equal size blocks

4.2. Singular values of a non-symmetric perturbed matrix

This example supports Theorem 3.2 and Corollary 3.3. Let P be the 7×8 real non-symmetric pattern matrix

$$\begin{bmatrix} 6 & 5 & 6 & 5 & 3 & 2 & 1 & 2 \\ 3 & 9 & 6 & 7 & 4 & 5 & 6 & 1 \\ 4 & 8 & 9 & 8 & 3 & 4 & 2 & 1 \\ 5 & 7 & 6 & 8 & 7 & 5 & 3 & 2 \\ 2 & 5 & 7 & 8 & 9 & 6 & 5 & 3 \\ 1 & 3 & 4 & 5 & 6 & 7 & 6 & 4 \\ 2 & 1 & 4 & 6 & 7 & 8 & 9 & 9 \end{bmatrix}.$$

In the previous example we showed the effect of the blocks' sizes, now we show the effect of the signal to noise ratio (snr) on the singular values of P. To blow up P, we chose block heights as $[m_1, \ldots, m_a] = [500, 750, 500, 600, 750, 550, 500]$ and block widths as $[n_1, \ldots, n_b] = [500, 500, 600, 1000, 550, 500, 550, 500]$. Then we blew up P, and we added a noise to get the perturbed matrix. Like in Section 3, we denote by B the blown-up matrix which is the signal, by X the noise matrix, and by $A = B + X$ the perturbed matrix. We shall use the following definition of the signal to noise ratio

$$\texttt{snr} = \frac{\frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n} b_{i,j}^2}{\frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n} x_{i,j}^2}.$$

Here $m = \sum_{i=1}^{a} m_i$ is the number of rows, $n = \sum_{j=1}^{b} n_j$ is the number of columns in B and X, $b_{i,j}$ is the element of the signal matrix B, while $x_{i,j}$ is the element of the noise matrix X in the i^{th} row and j^{th} column.

In this example, the entries of the noise matrix X are independent and, at the initial step, they are from standard normal distribution. In each of the following steps we increased the noise, by multiplying each of the elements of the noise matrix by certain multipliers. The first multiplier was 1.002, the second one was 1.004, then $1.006, \ldots, 3.000$.

So, during the experiment, we amplify the noise step-by-step and check at each iteration the snr and the ratio of the last (smallest) structural singular value and the first (largest) non-structural one. If the structural/non-structural ratio reaches a certain point, the signal gets so noisy, that it is not possible any more to distinguish the structural singular values from the non-structural ones. On Figure 7 the dashed line is the signal to noise ratio and the solid line is the ratio of the the last (smallest) structural singular value and the first (largest) non-structural singular value. On the horizontal axis the scale is given by the variance of the noise. One can see, that when the noise grows, then the snr decreases quickly, but the structural/non-structural ratio decreases quite slowly. If the structural/non-structural ratio decreases close to 1, then we reach a point, where the structural and non-structural singular values are not well distinguishable any more. However, Figure 7 shows that our method is reliable, that is if the noise in not higher than

25% of the signal, then the structural singular values are well distinguishable from the non-structural ones.

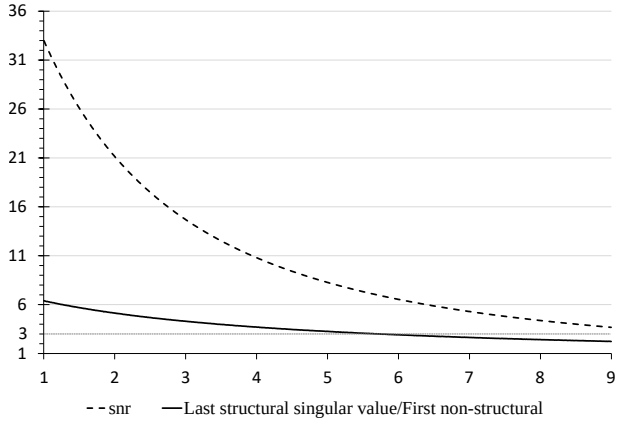


Figure 7: Signal to noise ratio compared to structural/non-structural singular value ratio

5. Conclusion

The shown graphical method works appropriately to distinguish the structural and non-structural eigenvalues and singular values. The size of the blocks has an influence on the 'jump' between the non-structural and the structural eigenvalues. As the block's sizes are increasing, the border between the structural and non-structural eigenvalues is getting more and more significant. If the block sizes are at least 50, then the structural and the non-structural eigenvalues are well distinguishable. Our test is reliable, if the noise is less than 25%, but above this noise level it can be unreliable.

References

[1] Z. D. Bai: *Methodologies in spectral analysis of large dimensional random matrices. A review.* Statistica Sinica 9 (1999), pp. 611–677.

[2] M. Bolla: *Recognizing linear structure in noisy matrices.* Linear Algebra and its Applications 402 (2005), pp. 228–244,
doi: https://doi.org/10.1016/j.laa.2004.12.023.

[3] M. Bolla, A. Krámli, K. Friedl: *Singular value decomposition of large random matrices (for two-way classification of microarrays).* J. Multivariate Analysis 101.2 (2010), pp. 434–446,
doi: https://doi.org/10.1016/j.jmva.2009.09.006.

[4] I. Fazekas, S. Pecsora: *On the spectrum of noisy blown-up matrices.* Special Matrices 8.1 (2020), pp. 104–122,
doi: https://doi.org/10.1515/spma-2020-0010.

[5] Z. Füredi, J. Komlós: *The eigenvalues of random symmetric matrices.* Combinatorica 1 (1981), pp. 233–241,
doi: https://doi.org/10.1007/BF02579329.

[6] S. O'Rourke, D. Renfrew: *Low rank perturbations of large elliptic random matrices.* Electron. J. Probab 19 (2014), pp. 1–65,
doi: https://doi.org/10.1214/EJP.v19-3057.

[7] T. Tao: *Outliers in the spectrum of iid matrices with bounded rank perturbations.* Probability Theory and Related Fields 155 (2013), pp. 231–263,
doi: https://doi.org/10.1007/s00440-011-0397-9.

[8] H. V. Van: *Spectral norm of random matrices.* Combinatorica 27 (2007), pp. 721–736,
doi: https://doi.org/10.1007/s00493-007-2190-z.

Annales Mathematicae et Informaticae
***51** (2020) pp. 29–39*
DOI: 10.33039/ami.2020.07.002
https://ami.uni-eszterhazy.hu

Indoor localization simulation framework for optimized sensor placement to increase the position estimation accuracy

Ádám Kaló, Zoltán Kincses, László Schäffer, Szilveszter Pletl

University of Szeged, Department of Technical Informatics
Kalo.Adam@stud.u-szeged.hu,{kincsesz,schaffer,pletl}@inf.u-szeged.hu

Submitted: February 4, 2020
Accepted: July 1, 2020
Published online: July 23, 2020

Abstract

Indoor position estimation is an important part of any indoor application which contains object tracking or environment mapping. Many indoor localization techniques (Angle of Arrival – AoA, Time of Flight – ToF, Return Time of Flight – RToF, Received Signal Strength Indicator – RSSI) and technologies (WiFi, Ultra Wideband – UWB, Bluetooth, Radio Frequency Identification Device – RFID) exist which can be applied to the indoor localization problem. Based on the measured distances (with a chosen technique), the position of the object can be estimated using several mathematical methods. The precision of the estimated position crucially depends on the placement of the anchors, which makes the position estimate less reliable. In this paper a simulation framework is presented, which uses genetic algorithm and the multilateral method to determine an optimal anchor placement for a given pathway in an indoor environment. In order to make the simulation more realistic, the error characteristics of the DWM1001 UWB ranging module were measured and implemented in the simulation framework. Using the proposed framework, various measurements with an optimal and with a reference anchor placement were carried out. The results show that using an optimal anchor placement, a higher position estimation accuracy can be achieved.

Keywords: Anchor placement, Ultra Wide Band, Multilateral, Genetic Algorithm, Indoor Localization, Non-linear Measurement Error Model

1. Introduction

The location of a device or user can be effectively obtained outdoor using the Global Positioning System (GPS), but it could be challenging in an indoor environment. During the last decade, indoor localization has been investigated mainly for wireless sensor networks and robotics. However, nowadays, the wide-scale usage of mobile phones and wearable devices has enabled localization in a wide range of applications like health-care, industry, surveillance and home management.

In the literature many localization technologies and techniques are available [9]. A Received Signal Strength Indicator (RSSI), which is the strength of the signal received usually measured in decibel-milliwatts (dBm), and a wireless Ethernet based localization approach is used in [4]. Using a path-loss model and the RSS, the distance between the sender and receiver can be estimated. In [7] an Angle of Arrival (AoA) and Wireless LAN (Wifi) based method is applied using an antennae array for the estimation of the angle by computing the difference between the arrival times at the individual elements of the array. A Time of Flight (ToF) and 2.4 GHz radio based approach is presented in [5], using signal propagation time to compute the distance between the transmitter and the receiver. A similar technique, the Return Time of Flight (RToF) is used in conjunction with RSSI in a Wifi-based method in [10].

RToF is a two-way ranging method where the transmitter sends a ranging message to the receiver at t_1 time. The receiver sends it back with a delay of t_{proc} time and it arrives to the original transmitter at t_2 time. The time of flight is $t_2 - t_1 - t_{\text{proc}}$, and the distance can be calculated with the speed of the signal, depending on the technology. The accuracy of the measurement highly depends on t_{proc}.

The UWB is a recently researched communication technology providing more accurate ToF and RToF estimations. It uses ultra-short pulses with a time period less than a nanosecond, resulting in a low duty cycle which leads to lower power consumption. Its frequency range is from 3.1 to 10.6 GHz with a bandwidth of 500 MHz. Since the UWB usually operates at a low energy level, typically between -40 and -70 dB, most of the other technologies detect it as background noise. This makes it practically immune to interference with other systems since it has a radically different signal type and radio spectrum. Moreover, the signal (especially in its lower frequencies) can penetrate through walls because signal pulses are very short. Utilizing this attribute, it is easier to differentiate the main path from the multi-paths, providing more accurate estimations [8].

Once the point-to-point distances between the objects are measured, the unknown position of the object can be estimated. There are various algebraic methods to estimate the position from the point-to-point distances like triangulation or multilateration. Most of them require a few devices with known fixed positions (anchor nodes) to calculate the actual position of the moving device (mobile node). In case of error-free distance measurements, these methods theoretically give an exact position. But real distance measurements contain errors which depend on the relative

position of the devices, the orientation of the antennas of the devices, and also the technique and technology used. These result in a varying reliability of the position estimation.

In this paper, a genetic-algorithm-based simulation framework is presented, which takes the specific error characteristics of the chosen localization system (DWM1001 a commercially available UWB localization system using RToF) into consideration and uses a 2-dimensional (2D) version of the multilateral method to determine an optimal anchor placement for a given pathway in a given environment. This framework can also be extended to 3-dimensional (3D) space in the future.

2. Methods

During the research, several methods were used together. To calculate the position of the object from the measured point-to-point distances, a 2D version of the original 3D multilateral algorithm [6] was used. The evaluation of the calculated positions was based on the Root-Mean-Squared Error (RMSE) of distances between the original and the calculated positions. To minimize this error, the genetic algorithm with a special fitness function (Subsection 4.1) was used. In this section, the various methods are presented.

2.1. Multilateral algorithm

The multilateral algorithm [6] can be used to determine the unknown position in a 3D space – assuming an adequate number of reference points in a common Descartes coordinate system spanning the space – by solving the following linear equation system:

$$(x - x_i)^2 + (y - y_i)^2 + (z - z_i)^2 = d_i^2; \quad i = 1, \ldots, N, \tag{2.1}$$

where N is the number of anchor nodes, (x_i, y_i, z_i) is the coordinate of the i^{th} anchor node, (x, y, z) is the coordinate of the mobile node and d_i is the point-to-point distance between the i^{th} anchor node and the mobile node respectively. Rearranging (2.1) in 2D case, it can be written in the following matrix representation:

$$\begin{pmatrix} -1 & -2x_1 & -2y_1 \\ -1 & -2x_2 & -2y_2 \\ \vdots & \vdots & \vdots \\ -1 & -2x_N & -2y_N \end{pmatrix} \cdot \begin{pmatrix} x^2 + y^2 \\ x \\ y \end{pmatrix} = \begin{pmatrix} d_1^2 - x_1^2 - y_1^2 \\ d_2^2 - x_2^2 - y_2^2 \\ \vdots \\ d_N^2 - x_N^2 - y_N^2 \end{pmatrix}. \tag{2.2}$$

Rewriting (2.2) in a short form:

$$\mathbf{A} \cdot \boldsymbol{\eta} = \mathbf{b}; \quad \mathbf{A} \in \mathbb{R}^{N \times 3}, \quad \boldsymbol{\eta} \in \mathbb{R}^3, \quad \mathbf{b} \in \mathbb{R}^N. \tag{2.3}$$

The solution of (2.3) for $\boldsymbol{\eta}$ is given by:

$$\boldsymbol{\eta} = \mathbf{A}^+ \cdot \mathbf{b}, \tag{2.4}$$

where $\mathbf{A}^+$ denotes the Moore-Penrose pseudo-inverse of matrix $\mathbf{A}$. Assuming that the positions of anchor nodes are fixed during the measurement, it is enough to calculate A^+ once offline thus speeding up the position estimation process.

2.2. Root-Mean-Squared error

There are several methods for qualifying models. In our case, to determine the quality of the anchor topography in relation to the position estimation accuracy, an appropriate quality factor is required which takes the standard deviation of the positioning error into account. Such factor is the RMSE or Root-Mean-Squared deviation (RMSD), which has the formula as follows [1]:

$$RMSE = \sqrt{\frac{\Sigma_{i=1}^{M}\left(\hat{d}_{\mathrm{err},i}^2\right)}{M}},$$

where M is the number of discrete points in the field of mobile nodes, and $\hat{d}_{\mathrm{err},i}$ is the Euclidean-distance between the estimated position $(\hat{x_i}, \hat{y_i})$ and the exact position (x_i, y_i) of point $i \in [1, M]$:

$$\hat{d}_{\mathrm{err},i} = \sqrt{(\hat{x}_i - x_i)^2 + (\hat{y}_i - y_i)^2}.$$

2.3. Genetic algorithm

The genetic algorithm (GA) is a metaheuristic, optimization algorithm based on the concept of Darwin's theory of evolution. During the optimization, the properties of the initial population are altered in such way that the value of the predefined fitness function should converge towards an optimal solution in every iteration. The genetic algorithm requires a genetic representation of the solution and a fitness function which can evaluate this representation. Since the GA usually works with hundreds of candidate solutions per iteration, choosing a compact representation and a fast evaluation method is crucial.

The GA typically starts with a randomly generated set of candidate solutions. The solutions are evaluated one-by-one, and a portion of the higher-ranking candidates are kept for the next generation.

After the first iteration, the generations will consist of higher-ranking candidates from the previous generations, cross-mutated individuals from the previous candidates, and also of new randomized individuals. Random mutations may also appear at any candidate. In the case of multi-population GA a migration step is also performed.

The optimization stops after the fitness value has reached a predefined limit or the number of maximum generations or stall-generations (i.e. generations, where the fitness value did not improved) has been reached [3].

3. Measurement setup

The environment of our experiment is a corridor in the University of Szeged. During the simulation and also in the validation a belt-type topography is used as can be seen in Fig. 1. The boundary conditions are derived from the physical capabilities of the corridor. For details of the anchors and mobile nodes used, see Subsection 3.1.

Two sets of measurements were carried out. The first set consisted of two measurements with 4 anchors using equidistant and optimised placement (see Section 4). The second set of measurement differed only in the number of anchors, since 8 anchors were used.

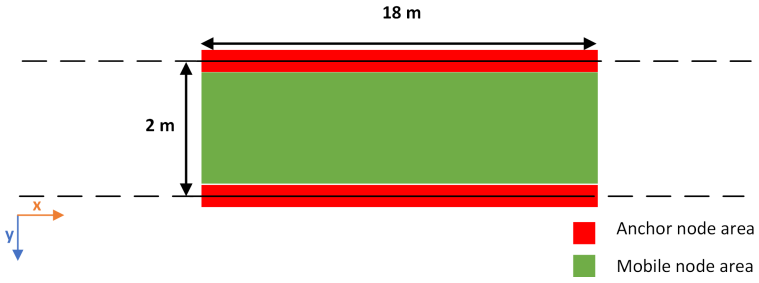


Figure 1: Measurement setup

During the measurements, the exact position of the discrete grid points (mobile node position) is determined using a professional laser rangefinder. In each grid point, the mobile node collected the distances from the anchors many times (at least 100 samples per point) and the aggregated data was sent to the data collector.

3.1. Hardware

The UWB based distance measurements were carried out by a commercially available localization system framework (MDEK1001), which consists of DWM1001 modules. The anchors were configured via the official mobile application (DecaWave DRTLS manager). The 8 anchors were organized in 2 networks, each network consisted of 4 anchors. On the measurements with 4 anchors, only one network was used.

The DWM1001 was connected to an STM32F746ZGT6 nucleo-144 development board (STM) via UART. The original program of the DWM1001 was modified to gather the 4 distances in both networks, and send the collected data towards the STM. The STM sent the data towards a mobile computer via UART trough an USB cable. On the mobile computer, the data was captured and saved via PuTTY, and were later evaluated with MATLAB.

4. Genetic algorithm based simulation

The simulation framework is based on the GA implementation of the optimtool toolbox of MATLAB. The implementation was slightly modified to run the multi-population version of GA. The genetic representation of the system (the phenotype) was the x coordinates of the anchors. The y coordinates were along the walls of the simulated area, and the distribution of the anchors between the two walls was equal, or its difference was 1. The phenotype had a lower limit of 0, and an upper limit of 18 (see Section 3), and they were free to move within this interval. The initial population range was 100 candidates. For each generation, the 20 highest-ranked candidates of the previous generation survived and further 60 were generated with cross-breeding and 20 new were randomly generated. The maximum number of generations, and the maximum number of stall-generations were both 100. The evaluation of the candidates was done by the special fitness function (Subsection 4.1). During the optimization process the number of anchors was 4 and 8.

4.1. Fitness function

The aim of the fitness function in a GA is to order the candidates of the population based on their phenotype. The input of the function is the x coordinates of the candidate anchors, and the output is a corresponding RMSE value. After that the exact positions of the mobile node have been generated along a given path in the simulated area, the point-to-point distances from these points and the anchor nodes can be calculated. For each distance value, a unique error (see Subsection 4.2) is added. Using the erroneous distance values and the anchor positions the simulated mobile node positions can be determined using (2.4). Since the physical hardware device distance horizon is limited to 10 metres, any distance value over this range is discarded. The RMSE value can be calculated from the exact and simulated positions (see Subsection 2.2), resulting a fitness value of the given candidate.

4.2. Determination of the 2D error characteristics

In order to implement a more realistic simulation, the UWB distance sensor calibration was performed using a more accurate (± 1 mm) class of laser rangefinder. Based on the measurements, the 2D error characteristics of the DWM1001 module with the built-in on-board antenna was determined. The measurement consisted of placing two modules in a known distance from each other, and performing a measurement with the UWB technology using RToF technique. The measured distance ($\hat{d}$) is the sum of the exact distance measured with a laser rangefinder (d) and the RToF measurement additive error (Δd) as follows:

$$\hat{d} = d + \Delta d.$$

The Δd was calculated in discrete points from 0.3 to 10 metres in steps of 0.3 meters. At each discrete point at least 100 measurements were conducted with

varying antenna orientation (0°, 90°, 180°) and the mean value was calculated.

After the measurements, at each measured distance, an error value was interpolated for every degree between 0°–180°. Since the error characteristic is symmetrical in this plane [2], the corresponding error values for degrees between 180° and 360° can be used from the interpolated ones. Each point of the 2D error characteristic is loaded into a Look-Up-Table (LUT), since in the optimization process the genetic algorithm reads the corresponding error value addressing the LUT by the distance and orientation parameters.

5. Results

The main results of the proposed method are creating a realistic (hardware- and environment-specific) 2D error characteristic of DWM1001, determining an optimal placement of anchor nodes and experimentally validating the results of the simulation.

5.1. Realistic 2D error characteristic

The visualization of the realistic 2D error characteristic can be seen in Fig. 2, where one slice of the surface represents the distance error of one degree of rotation of the DWM1001 module. In the data sheet of DWM1001 [2] it is claimed that the ranging accuracy of the module is within 10 cm. But the results in Fig. 2 showed that the accuracy is a nonlinear function of distance and orientation. Furthermore, between 0–1 m and 9–10 m, the error is significantly higher than 10 cm (17 cm ± 2 cm). The highest accuracy can be reached around 2 meters (7 cm ± 2 cm).

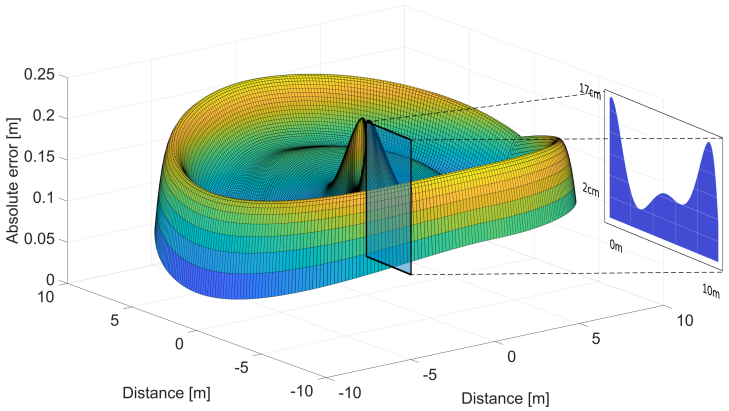


Figure 2: Realistic 2D error characteristic of DWM1001

5.2. Optimized anchor placements

The goal of the optimization process was to find an optimal anchor placement in case of 4 and 8 anchors. During this process, linear and non-linear paths of the mobile node was used. Table 1. shows the RMSE values of the position estimation using the optimally and equidistantly placed anchors. The results showed that the optimization significantly increases the accuracy of the position estimation in case of 4 anchors, but in case of 8 anchors, it has a lower impact. However, using only 4 anchors, the robustness of the system is lower.

	RMSE [m]			
	straight line	sine wave	arctangent wave	Average
4 anchor opt.	0.086	0.088	0.087	0.087
4 anchor eq.	1.392	1.392	1.391	1.392
8 anchor opt.	0.057	0.060	0.060	0.059
8 anchor eq.	0.072	0.072	0.073	0.072

Table 1: RMSE of the optimized and equidistant anchor placement

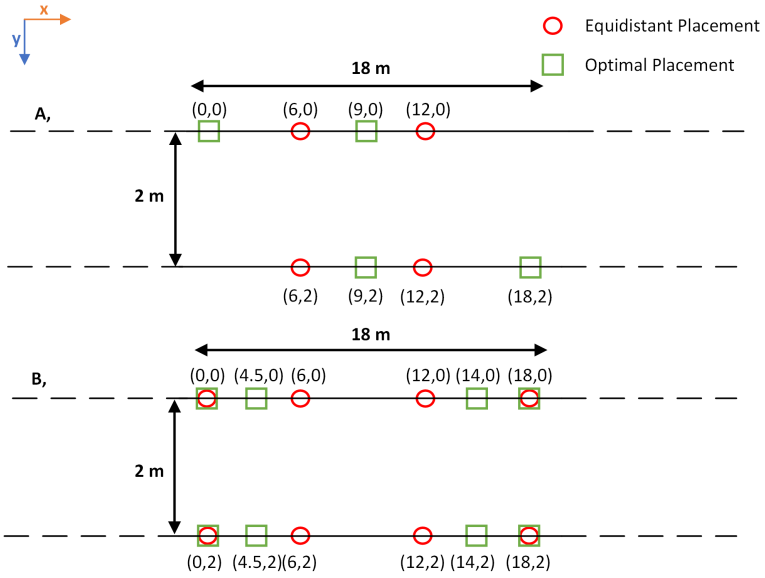


Figure 3: Optimized and reference anchor placements for 4 anchors (A) and 8 anchors (B)

In Fig. 3 the placement of the anchor nodes can be seen with and without optimization. The optimization process provides significantly different anchor placement in case of 4 anchors but just a slightly different in the 8-anchor case compared

to the equidistant placing. Analyzing the results of the optimization, it can be stated that the algorithm places the anchors considering two main conditions. It tries to cover the whole area to have at least 3 anchor nodes in the range of the mobile node and places as many anchors as possible in the border of the mobile node area since the accuracy of the DWM1001 is lower nearby its horizon.

5.3. Validation of the anchor placement

The purpose of the validation is to verify the localization accuracy by real measurements using the proposed anchor placement in case of 4 and 8 anchors. In Fig. 4 and Fig. 5 the anchor position, the accurate position using the laser rangefinder and the estimated position using the DWM1001 of the mobile node can be seen.

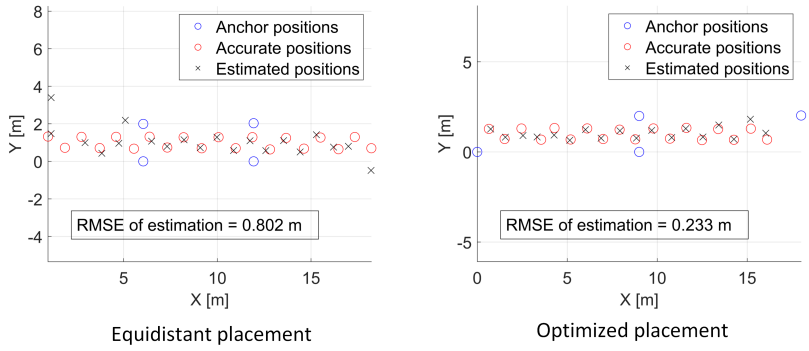


Figure 4: Experimental results for the 4 anchor measures

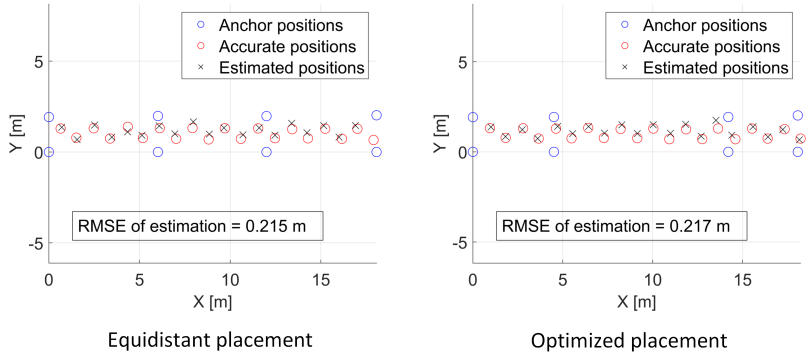


Figure 5: Experimental results for the 8 anchor measures

The results show that using an optimized anchor placement, the localization accuracy can be increased in case of 4 anchors but there is no significant improve-

ment using 8 anchors compared to the equidistant case. Using an optimal anchor placement in case of limited number of anchors the space of the mobile nodes can be effectively covered. Furthermore, the validation shows that using the simulation framework, the same positioning results can be achieved as with the real measurement.

6. Conclusion

In this paper, a genetic-algorithm-based simulation framework is presented to determine an optimal anchor placement in an indoor environment. To implement a realistic and precise simulation environment, the 2D error characteristics of the DWM1001 module was measured and implemented in this work. Using the proposed framework, various measurements with an optimal and with a reference anchor placement were carried out. The results show that the optimal anchor placement is crucial when the number of anchors is limited. It can also be concluded that if the number of anchors are increasing, their placement becomes less relevant. Furthermore, the validation shows that there is no significant difference between the simulation and the real experiments.

Acknowledgements. This research was supported by the projects "Extending the activities of the HU-MATHS-IN Hungarian Industrial and Innovation Mathematical Service Network" EFOP-3.6.2-16-2017-00015.

References

[1] T. Chai, R. R. Draxler: *Root mean square error (RMSE) or mean absolute error (MAE)? – Arguments against avoiding RMSE in the literature*, Geoscientific Model Development 7.3 (2014), pp. 1247–1250,
doi: 10.5194/gmd-7-1247-2014.

[2] DecaWave: *DWM1001 Datasheet, Version 1.10*, 2017.

[3] J. H. Holland: *Genetic Algorithms*, Scientific American 267.1 (1992), pp. 66–73,
doi: 10.1038/scientificamerican0792-66.

[4] A. Ladd, K. Bekris, A. Rudys, E. L. Kavraki, D. S. Wallach: *Robotics-Based Location Sensing Using Wireless Ethernet*, Wireless Networks 11 (2005), pp. 189–204,
doi: https://doi.org/10.1007/s11276-004-4755-8.

[5] S. Lanzisera, D. T. Lin, K. S. J. Pister: *RF Time of Flight Ranging for Wireless Sensor Network Localization*, in: International Workshop on Intelligent Solutions in Embedded Systems, 2006, pp. 1–12,
doi: 10.1109/WISES.2006.329127.

[6] A. Norrdine: *An algebraic solution to the multilateration problem*, in: International Conference on Indoor Positioning and Indoor Navigation, 2012, pp. 1–4,
doi: 10.13140/RG.2.1.1681.3602.

[7] J. Xiong, K. Jamieson: *ArrayTrack: A Fine-Grained Indoor Location System*, in: Symposium on Networked Systems Design and Implementation, 2013, pp. 71–84.

[8] K. Yu, I. Oppermann, in: UWB Theory and Applications, John Wiley and Sons, Ltd, 2005, pp. 175–196,
DOI: 10.1002/0470869194.ch8.

[9] F. Zafari, A. Gkelias, K. K. Leung: *A Survey of Indoor Localization Systems and Technologies*, IEEE Communications Surveys Tutorials 21.3 (2019), pp. 2568–2599,
DOI: 10.1109/COMST.2019.2911558.

[10] Zheng Sun, R. Farley, T. Kaleas, J. Ellis, K. Chikkappa: *Cortina: Collaborative context-aware indoor positioning employing RSS and RToF techniques*, in: 2011 IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops), 2011, pp. 340–343,
DOI: 10.1109/PERCOMW.2011.5766901.

Annales Mathematicae et Informaticae
***51** (2020) pp. 41–52*
DOI: 10.33039/ami.2020.07.010
https://ami.uni-eszterhazy.hu

Evaluation of colony formation dataset of simulated cell cultures

Dániel Kiss[a], Gábor Kertész[a], Máté Jaskó[a], Sándor Szénási[ac], Anna Lovrics[b], Zoltán Vámossy[a]

[a]John von Neumann Faculty of Informatics, Óbuda University, Hungary

[b]Membrane Protein Research Group, Research Centre for Natural Sciences, Hungarian Academy of Sciences, Hungary

[c]Faculty of Economics and Informatics, J. Selye University, Slovakia

Submitted: February 4, 2020
Accepted: July 1, 2020
Published online: July 23, 2020

Abstract

In vitro biological experiments and *in silico* individual-based computational models are widely used to understand the low-level behavior of cells and cellular functions. Many of these functions can not be directly observed, however, may be deduced from other properties that can be well measured and modeled. In this paper, we present a procedure to evaluate synthetic cell colony formation generated by an off-lattice individual-based model. The calculated shape features of the artificial cell aggregates can be related to the parameter values of the simulated agents, therefore this data can be used to quantify properties of real-life cells such as motility or binding affinity that can not be easily determined otherwise. Our experiments showed that only a few of these parameters are responsible for the difference in shape features of the colonies.

Keywords: modeling in vitro cells; colony formation; agent-based simulation; parameter inference; machine learning

1. Introduction

1.1. Background and motivation

Biological experiments frequently carried out on cell cultures, also known as *in vitro* cultures. These cells are usually kept alive in culture media added into different sized culture dishes. At ideal conditions, the cells start to proliferate and reproduce, which leads to an increased size of the initial populations. Depending on individual cellular-level properties, these cells may form tight, regularly-shaped colonies, loosely-connected aggregates with irregular edges or no distinguishable clusters at all.

An experienced researcher can recognize a change in some cellular functions or properties only by looking at the culture under a microscope and see the differences in the pattern of cell aggregates, their size, shape, etc. Analyzing microscopy images by specific image processing applications can also also provide useful information, such as the percentage area covered by cells, the number of cell aggregates detected or the statistical features of the shape descriptors of colonies. On the other hand, it is usually impossible to objectively and precisely define the changes in cellular-level functions only by evaluating microscopy photos.

1.2. Aims of the research

Our objective was to propose a method, which demonstrates how it is possible to relate some pre-selected cellular properties to the measured shape features of multi-cellular aggregates. To do so, we first created an individual-based model that captures the selected properties of a single cell *in vitro*. Then, a large number of input parameters were generated and multiple simulations were executed. The resulting dataset was processed by a shape feature extraction algorithm. Finally, we used a multi-layered neural network to relate the extracted shape features of the artificial cell-aggregates to the input parameters of each simulation.

2. Related work

In the last few decades, the so-called individual-based modeling technique became more and more popular in this field, partially because it can provide useful insights into cellular level features based on the emergent behavior of a large population of individuals, also called as agents. For instance, such techniques can be utilizes to study collective cell migration [14], to model the calcium dependent behaviour of epithelial cells [16] or malignant tumor growth [12], just to mention some.

Previous researches showed, that even a relatively simple individual-based computational model of individual cells is able to produce different colony morphologies when some of the input parameters were altered [6–8]. The growth dynamics and morphology features of multi-cellular aggregates can be also captured by statistical physics models [1, 9]. Such models suggest that the key mechanism in monolayer

colony formation is the surface diffusion of cells at the boundary of the aggregate [3], however, this problem is still not entirely solved [4].

3. Methods

3.1. Agent-based modeling

We used a simplified version of the model introduced by Drasdo et al. [8] and previously presented in [11]. This model uses the position, mean diameter, core ratio, adhesive ratio, adhesion factor and velocity data of each individual. Therefore, cells can be interpreted as partially overlapping, sticky disks with diameter d on a two-dimensional circular and bounded flat surface (the bottom of the culture dish). The position of agent i is stored in its coordinate vector $\vec{x_i}$. Core ratio $r^c < 1$ defines the diameter of an embedded disk (core). For two or more agents cores should not overlap, so values of $r^c \approx 0$ belong to highly elastic cells, while $r^c \approx 1$ to highly rigid ones. Adhesive ratio $r^a > 1$ defines the distance in which two agents form adhesive bonds. Interaction properties of agents i and j being in distance x are incorporated by an interaction potential function defined as

$$V_{ij}(x) = \begin{cases} \infty & \text{if } x \leq d_{ij}^c \\ -\varepsilon & \text{if } d_{ij}^c < x \leq d_{ij}^a \\ 0 & \text{otherwise} \end{cases} \tag{3.1}$$

where

$$d_{ij}^c = \frac{r_i^c d_i + r_j^c d_j}{2} \quad \text{and} \quad d_{ij}^a = \frac{r_i^a d_i + r_j^a d_j}{2}$$

are the core and adhesion distances of the agents, respectively. Value ε describes the agent-agent adhesion energy and can be directly linked to quantities such as cell membrane adhesion receptor density [2, 7]. To simulate this model, a Monte Carlo rejection sampling process [13] is executed, during which agent i is displaced by the vector $\delta\vec{u}$ with acceptance probability

$$\min\left\{1, \quad \exp^{-1}\left(\sum_{i\neq j} V_{ij}^{t+\Delta t} - \sum_{i\neq j} V_{ij}^{t}\right)\right\},$$

where $\vec{u}$ is a randomly directed unit vector, δ is a gamma distributed distance with shape parameter k and scale parameter θ and Δt is the unit of simulation time scale.

Duplication of agent i is based on its cell cycle time τ_i (the time duration between two division events) and time counter (internal clock) state t_i. When

$$t_i \geq \frac{\tau_i}{\Delta t}$$

cell duplication is performed by creating a copy i' of agent i and assigning new coordinates

$$\vec{x_i}^{\text{new}} = \vec{x_i}^{\text{old}} + \frac{1}{2} r_i^c d_i \vec{u}$$
$$\vec{x'_i}^{\text{new}} = \vec{x_i}^{\text{old}} - \frac{1}{2} r_i^c d_i \vec{u}$$

to the agents, where $\vec{u}$ is a uniformly distributed two-dimensional unit vector. If there is no sufficient space to locate both agents (that is, when the interaction energy defined by (3.1) is infinity), the duplication trial is rejected. Otherwise, both agents are set to their initial cell cycle state.

3.2. Input data generation and simulation

To simulate the model, we generated all possible input parameter combination to better explore the structure of the feature space. However, to decrease the number of individual combination, some model parameters were fixed by analyzing and evaluating real microscopy images, therefore we set $d = 25$ µm, $r^c = 0.8$, $r^a = 1.2$, $\tau = 24$ h and $\theta = 0.1$. All other input values were picked one by one from its possible range (see Table 2 for details). To minimize stochastic effects, threefold replication were used with all parameter combinations.

When a simulation is started, a given number of agents (approximately 250) are randomly placed into the simulation space which is a circular surface with a diameter of 6.4 mm (this is approximately equivalent to the diameter of a standard 96-well culture dish). The locations of each agent are saved periodically and a pseudo-microscopy image is rendered from the simulation data, where black disks represent the simulated cell on a white background (somewhat similar to in Fig. 1). This photo is later loaded into the image processing software for further evaluation.

3.3. Shape feature extraction

To evaluate the rendered pseudo-microscopy images, we created a batch feature extraction pipeline in CellProfiler [5]. This pipeline first smooths the image using a Gaussian filter with kernel size $\sigma = 1.0$ to make the artificial image more realistic. Then, an automated binarization using minimum cross-entropy thresholding separates the foreground (cell aggregates) from the background. To remove small artifacts and holes at the boundary of the colonies, we performed a closing operation with a disk structuring element. After that, all distinct foreground objects are marked and processed one by one. When shape features of all detected objects are determined, the data is saved as an output file, along with the corresponding simulation parameter values and identifiers such as object label, frame number, etc.

The most significant attributes along with their description and key statistical properties of the produced dataset is summarized on Table 1.

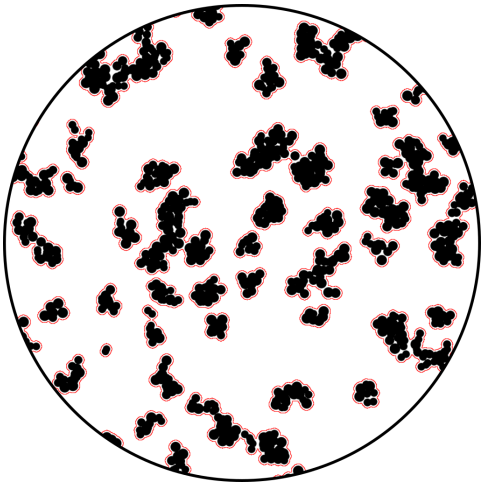


Figure 1: Representative image of a pseudo-microscopy image of simulated cells (black dots) in a small circular vessel. The boundaries of detected cell aggregate objects are shown red. Note, that cell diameters and vessel diameter are not realistic on this image.

Attribute	Unit	Mean	SD	Min	Max
Area: the number of pixels covered by the given object	pixel	1549.77	1995.13	259.0	228649.0
Perimeter: the total number of pixels around the boundary of each region in the image	pixel	157.11	119.8	64.42	9792.12
MeanRadius: the mean distance of any pixel in the object to the closest pixel outside of the object	pixel	5.74	1.61	3.0	26.65
Compactness: the mean squared distance of the object's pixels from the centroid divided by the area	dimensionless	1.15	0.21	1.0	5.29
Solidity: the proportion of the pixels in the convex hull that are also in the object	dimensionless	0.92	0.05	0.36	1.0
FormFactor: calculated as $4\pi \text{Area}/\text{Perimeter}^2$	dimensionless	0.77	0.16	0.03	0.98

Table 1: Most significant shape attributes of simulated cell aggregates along with their short description and basic statistical properties

3.3.1. Data filtering

Since the set contains time series data, measured values should be reviewed. An interesting phenomenon can be observed when multiple distinct cell aggregates merged into one large aggregate. In these cases, the segmentation method is not able to correctly distinguish these objects, therefore produce invalid shape measurements.

On the other hand, individual cell movements can result in breaking up these

Parameter	Unit	Value
ε: adhesion energy parameter ("stickyness")	dimensionless	$\varepsilon \in \{1, 3, 5\}$
k: distribution shape parameter of the mean displacement step size ("velocity")	dimensionless	$k \in \{2, 3, 4\}$
Δt: time resolution of the simulation	minute	$\Delta t \in \{1, 2, 4, 8, 16, 32, 64\}$
N_{MC}: number of repeated Monte Carlo displacement trials of an agent in a given time step.	dimensionless	$N_{MC} \in \{1, 2, 4, 8, 16, 32, 64\}$

Table 2: Input parameter data of the simulated agents (See 3.1 for a detailed description)

clusters into separate objects. These combined features cause outliers, resulting significant noise in the observations. Affected objects can be removed from the dataset based on the area sizes, using a simple algorithm. At time step 0, all objects are removed from the set where the area size is considered large, according to the input values. At every other time step i, the observed area size A_i^j of object number j is compared to the last observation A_{i-1}^j, and if $A_i^j > mA_{i-1}^j$, where m is a factor defined as $\sqrt{2}$, 2 or 3, the observation is considered as an outlier; therefore, it is removed.

The application of this method will result in serious imbalance, or – if all observations of an object are removed on detection – cropping of the dataset. In our experiments, approximately 99% of objects are removed because of a size mismatch at some point of their lifetime. It is important to point out, that object numbers are not unique identifiers: the numbering in each frame restarts. As a consequence, some objects which themselves are not affected by clustering error are removed by the dataset because of incorrect labeling caused by a nearby error.

However, cell aggregate object identification is possible based on the coordinates of their midpoint: an assumption can be made that object o_1 at observed time point t_k and object o_2 of observation t_{k+1} are the same for some k if the Euclidean distance $d(o_1, o_2)$ between the center points $c_{t_k}^{o_1}$ and $c_{t_{k+1}}^{o_2}$ are minimal:

$$\min_i d(c_{t_k}^{o_1}, c_{t_{k+1}}^{o_n}).$$

We would like to note, that other methods, such as the iterative closest point (ICP) method [15] could also be applied to extend or replace the described matching procedure. After identifying the cells through frames, the previously described outlier filtering method can be applied to filter the data, using the calculated object IDs.

3.4. Statistical analysis

To analyze the behavior of the model for different inputs, a machine learning-based model for regression was used. As a proof of concept, a classical multi-layered neural

network was trained to predict the area size of a cell at a given time for a given set of input parameters.

The input layer receives the normalized time and input features, a total of 11 features. The shallow network architecture (visualized on Fig. 2) consists of 7 hidden layers with parameter numbers between 100 and 25, followed by an output layer with one single neuron with a "leaky" Rectified Linear Unit activation function to predict the area. For training, the state-of-the-art Adam optimizer [10] was used to minimize the mean squared error. Results showed a mean absolute error of 13.5%.

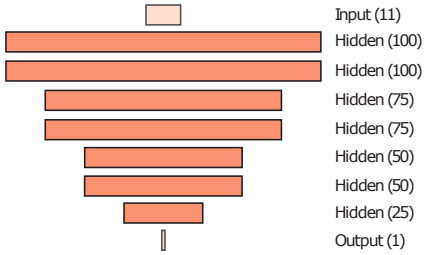


Figure 2: The structure of the fully connected neural network. The 11 input parameters are followed by a total of 7 hidden layers, and one single output is calculated. The activation functions are common ReLU activations, while the output neuron has a "leaky" ReLU activation to support negative values.

The relatively high error rate can be explained by the dependency of the adjacent cell objects, and the relatively small size of training data. We also would like to point out, that our future plans include the analysis of convolutional neural networks (CNN) to incorporate the features of adjacent cells, as well as recurrent neural networks (RNN) to take the past states of the objects into account. This paper shows a proof of concept, and the base idea of the procedure.

3.5. Input inference

The presented prediction technique of the output values is used as a basis of an input inference method. The pre-trained network is extended with a first layer with random trainable weights, while all other layers, including the trained parameters, are unchanged. As it is shown in Fig. 3, the neuron number of the inserted layer equals the number of input parameters, while other parts of the network – including the weights – remain unchanged.

For a given output value, all input values are set to constant 1, and using these values a few steps of back-propagation is done. The given values are flown through the network to get a prediction, the error is calculated from the expected value, this loss is then back-propagated to the first layer, and the weights are changed accordingly. A few of these training steps are done until the error rate descends to

a fixed value.

Afterwards, the final step is to calculate the so-called expected input parameters from the trained parameters. Layer weights can be represented as a matrix W with a size of 11×11, as all the 11 neurons of the layer are connected with all the 11 neurons of the input, resulting in 11 rows of weight vectors of length 11. The layer also has bias values for each neuron, resulting in a total of 11 values represented by vector b.

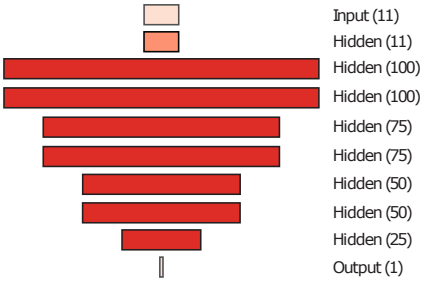


Figure 3: The structure of the input inference model. The hidden layers colored with red are "frozen", non-trainable, the weights are pre-trained. The fully connected layer inserted as the first layer is the only trainable layer in the network.

Based on the classic formula of activation, the sum of each row i is calculated as

$$\sum_{j=1}^{N} w_{i,j},$$

resulting in a vector of 11 values. This will be multiplied by the input values, which are constant ones, therefore, vector w is unchanged and the bias is added:

$$z_i = \sum_{j=1}^{N} w_{i,j} + b_i.$$

In case the expected input values are in the domain of $[0, 1]$, a sigmoid activation function could be applied as

$$a_i = \mathrm{sigm}(z_i)$$

to get the expected input values for a given output.

It is notable, that the function of the neural network is non-injective, the inputs can not be inverted, the inputs can only be inferred, while a set of multiple solutions might exist, the method defined here only results in the input set with the lowest error rate.

4. Results

Since the large number of possible features, we inspected the resulting dataset by performing a hierarchical clustering on the attributes and visualizing their dependence on a correlation heatmap (see Fig. 4). To build the dendrogram of the attributes, an agglomerative clustering process was used with Ward's minimum variance method and Euclidean distance function as a measure of dissimilarity. As it was revealed, the measured shape attributes are highly interconnected, therefore it is possible to reduce the number of shape features to a much smaller subset. We selected Area, Compactness and FormFactor as they fall into separate classes based on the hierarchical clustering.

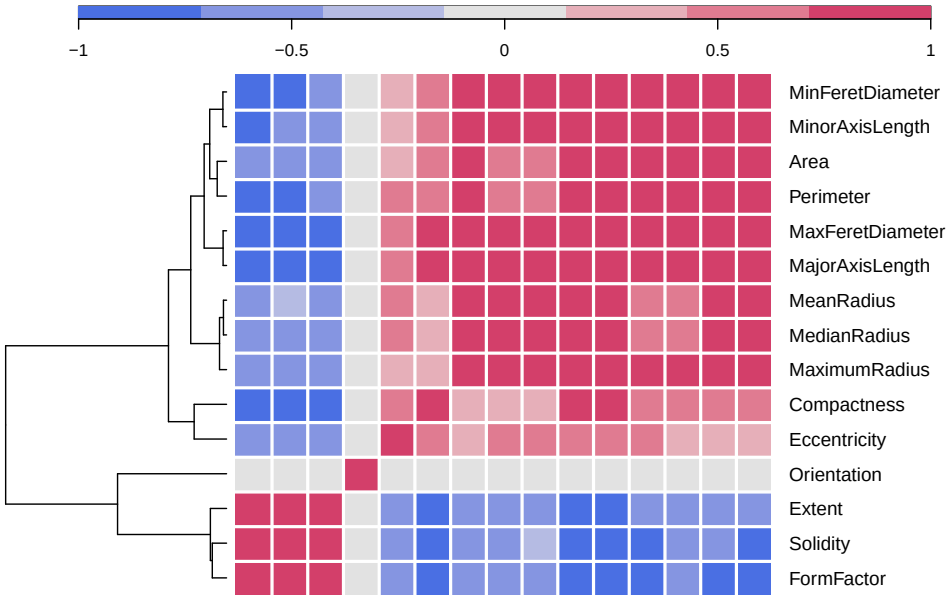


Figure 4: Hierarchical clustering and correlation heatmap of the dataset. As attributes are highly interconnected, a well-chosen subset is able to catch most of the differences in the shape features of the cell aggregates.

The final dataset contained approximately 250,000 records. The multi-layered neural network is trained on these data to predict a selected shape feature. For demonstration purposes, we chose Area, as it is easy to interpret and it is a good representative element of the feature set. During training, 70% of the original dataset was used for training, and the remaining 30% for validation, to detect overfitting.

The training of this baseline model was evaluated with a separated test set of 1522 synthetic test records: the measured average difference between the expected

and the predicted area size was 12.6%. After the model was trained, the parameters were used to infer the input variable, now based on the outcome area size of the cell aggregate, using the method defined in section 3.5. After freezing the original weights, training affects only the parameters of the appended first layer. Training concludes when the measured loss stops descending; during our experiments this happened with a loss value near zero. During our experiments, the inferred inputs were fed back to the original model, and the predicted area size is compared with the expected.

During the experiments, we created a novel embedding structure, where a selected input parameter can be predefined, and are not affected by training. This defined method is easily extendable, allowing the researchers to predict some input values for fixed input parameters. Future plans include the extension of the model to examine the behavior of multiple cells and time-series based on the previously mentioned CNN or RNN structures.

Our initial aim was to demonstrate the possibility of relating simulation input parameters to measured shape features. We inspected this relation on our simulation data. Using principal component analysis, we concluded that the two most significant input parameters are ε and k, i.e. the interaction potential well depth ("stickyness") and the shape parameter of the step size distribution ("velocity").

As depicted in Fig. 5, stronger attractive forces between the agents (larger ε values) results in smaller but more circular cell aggregates. This observation was confirmed statistically by performing a two-way ANOVA which showed that ε has a clear effect on those features ($p < 0.001$). On the other hand, we found that the velocity has statistically significant effect only on the FormFactor feature ($p < 0.001$) but not on the Area.

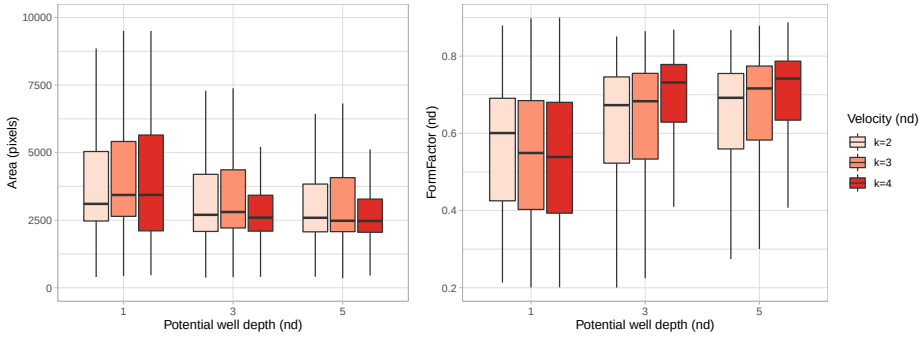


Figure 5: The effect of altered input parameters on shape features. There is a relation between the distribution of Area (left) and FormFactor (right) and the input parameters ε and k.

5. Conclusion

In this paper, we proposed a procedure to generate a dataset of colony formation process of simulated cell cultures. Using a simplified version of an existing agent-based model, multiple simulations were executed parallelly and the results were analyzed using an image processing pipeline. The details of the agent-based model, the feature extraction method as well as the data filtering technique were also discussed. Following the statistical analysis of the results, a proof-of-concept regression method was presented to predict an output of the simulation, based on input parameter data. Built on the regression model, an input inference method is introduced to produce possible input parameters from the received output.

As a preliminary result, we found that the measured shape features of the artificial cell aggregates (such as the area or the circularity) can be predicted by only a few input parameters, namely the simulated time t which is simple to understand, but also the adhesion energy ε and velocity distribution shape parameter k which both belong to the motility of a living cell. This observation is consistent with other published results. Our experiments show, that the proposed method – extended by sensitivity analysis and a precisely defined search – could be promising in case of parameter search for simulated environments.

We believe that the proposed procedure could serve as a useful base for creating and testing more accurate prediction models based on machine learning or developing advanced statistical methods that reveal some non-trivial patterns of *in vitro* cell pattern formation. This concept could also contribute to researches aiming to predict some hard-to-measure properties of living cells by creating and fitting a model with known parameters to the real phenomenon.

Acknowledgements. Dániel Kiss was supported by UNKP-18-3-I-OE-94 New National Excellence Program of the Ministry of Human Capacities. Anna Lovrics was supported by OTKA PD124467 grant. The authors acknowledge the financial support by the Hungarian State and the European Union under the EFOP-3.6.1-16-2016-00010 project. The authors thankfully acknowledge the support of the Doctoral School of Applied Informatics and Applied Mathematics of Óbuda University.

References

[1] M. Baraldi, A. Alemi, J. Sethna, et al.: *Growth and form of melanoma cell colonies*, J. Stat. Mech. (2013), doi: https://doi.org/10.1088/1742-5468/2013/02/P02032.

[2] D. A. Beysens, G. Forgacs, J. A. Glazier: *Cell sorting is analogous to phase ordering in fluids*, Proceedings of the National Academy of Sciences 97.17 (2000), pp. 9467–9471, doi: https://doi.org/10.1073/pnas.97.17.9467.

[3] A. Brú, S. Albertos, J. L. Subiza, J. L. García-Asenjo, I. Brú: *The Universal Dynamics of Tumor Growth*, Biophysical Journal 85 (2003), pp. 2948–2961, doi: https://doi.org/10.1016/S0006-3495(03)74715-8.

[4] J. Buceta, J. Galeano: *Comments on the Article "The Universal Dynamics of Tumor Growt" by A. Brú et al.* Biophysical Journal 85 (5 2005), pp. 3734–3736, doi: https://doi.org/10.1529/biophysj.104.043463.

[5] A. Carpenter, T. Jones, M. Lamprecht, et al.: *CellProfiler: image analysis software for identifying and quantifying cell phenotypes*, Genome Biology 7.100 (2006).

[6] D. Drasdo, S. Hoehme: *Individual-based approaches to birth and death in avascular tumors*, Mathematical and Computer Modelling 37 (2003), pp. 1163–1175, doi: https://doi.org/10.1016/S0895-7177(03)00128-6.

[7] D. Drasdo, S. Hoehme, M. Block: *On the Role of Physics in the Growth and Pattern Formation of Multi-Cellular Systems: What can we Learn from Individual-Cell Based Models?*, Journal of Statistical Physics 128.287 (2007), doi: https://doi.org/10.1007/s10955-007-9289-x.

[8] D. Drasdo, R. Kree, J. McCaskill: *Monte Carlo approach to tissue-cell populations*, Physical Review E. 52.6 (1995), pp. 6635–6657, doi: https://doi.org/10.1103/PhysRevE.52.6635.

[9] M. A. C. Huergo, M. A. Pasquale, P. H. González, A. E. Bolzán, A. J. Arvia: *Dynamics and morphology characteristics of cell colonies with radially spreading growth fronts*, Phys. Rev. E. 84 (2011), doi: https://doi.org/10.1103/PhysRevE.84.021917.

[10] D. Kingma, J. Ba: *Adam: A method for stochastic optimization*, arXiv preprint arXiv: 1412.6980 (2014).

[11] D. Kiss, A. Lovrics: *Performance analysis of a computational off-lattice tumor growth model*, in: IEEE 30th Jubilee Neumann Colloquium, 2017, pp. 141–146.

[12] P. Macklin, M. Edgerton, A. Thompson, V. Cristini: *Patient-calibrated agent-based modelling of ductal carcinoma in situ (DCIS): From microscopic measurements to macroscopic predictions of clinical progression*, Journal of Theoretical Biology 301 (2012), pp. 122–140, doi: https://doi.org/10.1016/j.jtbi.2012.02.002.

[13] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, E. Teller: *Equations of State Calculations by Fast Computing Machines*, Journal of Chemical Physics 21.6 (1953), pp. 1087–1092, doi: https://doi.org/10.1063/1.1699114.

[14] M. J. Plank, M. J. Simpson: *Models of collective cell behaviour with crowding effects: comparing lattice-based and lattice-free approaches*, J. R. Soc. Interface 9 (2012), pp. 2983–2996, doi: https://doi.org/10.1098/rsif.2012.0319.

[15] D. Stojcsics, Z. Domozi, A. Molnár: *Iterative Closest Point Based Volume Analysis on UAV Made Timeseries Large-scale Point Clouds*, in: IEEE 16th International Symposium on Intelligent Systems and Informatics: SISY 2018, 2018, pp. 69–74, doi: https://doi.org/10.1109/SISY.2018.8524645.

[16] D. Walker, J. Southgate, G. Hill, et al.: *The epitheliome: agent-based modelling of the social behaviour of cells*, Biosystems 76.1–3 (2004), pp. 89–100, doi: https://doi.org/10.1016/j.biosystems.2004.05.025.

Annales Mathematicae et Informaticae
51 *(2020) pp. 53–63*
DOI: 10.33039/ami.2020.07.008
https://ami.uni-eszterhazy.hu

Numerical analysis of finite source Markov retrial system with non-reliable server, collision, and impatient customers

Attila Kuki, Tamás Bérczes, Ádám Tóth, János Sztrik

University of Debrecen
{kuki.attila,berczes.tamas,toth.adam,sztrik.janos}@inf.unideb.hu

Submitted: February 4, 2020
Accepted: July 9, 2020
Published online: July 23, 2020

Abstract

A retrial queuing system with a single server is investigated in this paper. The server is subject to random breakdowns. The number of customers is finite and collision may take place. A collision occurs when a customer arrives to the busy server. In case of a collision both customers involved in the collision are sent back to the orbit. From the orbit the customers retry their requests after a random waiting time. The server can be down due to a failure. During the failed period the arriving customers are sent to the orbit, as well. The novelty of this analysis is the impatient behaviour of the customers. A customer waiting in the orbit may leave it after a random waiting time. The requests of these customers will not be served. All the random variables included in the model construction are assumed to be exponentially distributed and independent from each other.

The impatient property makes the model more complex, so the derivation of a direct algorithmic solution (which was provided for the non-impatient case) is difficult. For numerical calculations the MOSEL-2 tool can be used. This tool solves the Kolmogorov system equations, and from the resulting steady-state probabilities various system characteristics and performance measures can be calculated, i.e. mean response time, mean waiting time in the orbit, utilization of the server, probability of the unserved impatient requests. Principally the effect of the impatient property is investigated in

these results, which are presented graphically, as well.

Keywords: queueing, finite source, non-reliable, collision, impatient

MSC: 91B70

1. Introduction

Retrial queueing systems (RQ-systems) are very useful tools for modeling a large variety of problems of real life situations. An RQ-system can be described by the following characteristics: when an arriving job from the outside world (from the sources) or from the queue of the system finds the server busy, joins the orbit and after a random, usually exponentially distributed time retries to reach the server again. In case of an infinite source, the orbit is assumed to be infinitely large and jobs keep retrying until they are served. The call centers, telecommunication systems, computer networks, telephone switching systems and recently smart city networks etc. can effectively be modeled by RQ-systems. Instead of the infinite source models which have been investigated by many authors, the models with finite number of sources are more appropriate to describe the behaviour of the systems under consideration. The mobile networks, sensor networks, and cognitive radio systems can be mentioned as common example of these finite source systems. The random and multiple access protocols for these types of systems have been investigated, for example in [3, 12].

In real life situations, unfortunately, the reliability of the systems cannot be assumed and assured. The elements of the systems are subject to random breakdowns. These situations also have to be investigated, so the models contain random server breakdowns and repairs. The system characteristics and performance measures are very sensitive to the non-reliable operation of the systems. Finite-source RQ-systems with server breakdowns and repairs have been investigated in several recent papers, for example in [2, 8–10, 20, 22].

The goal of this paper is to give a stochastic model for describing the phenomenon of the impatient waiting customers. The customers may retry their requests, the environment is non-reliable, and during the service process collisions might occur. A single server $M/M/1//N$ retrial queueing system is useful and efficient for this task. The server is subject to random breakdowns, and the customers are subject to collisions at the service unit. This type of collisions are essential part of various implementations of telecommunication systems, computer networks. In case of busy communication channels there is large probability of conflict of signals. In these cases the signals involved into collision are lost, and retransmission is needed. The performance measures of these systems are under an optimal level. Consequently, the investigation of the systems subject to collisions has great interest nowadays. The best solution, namely building systems without collision is difficult to reach. The main effort of the investigations is to maximize the performance of the systems with collision. Previous years many authors have investigated queueing systems with conflict of customers, e.g. [1, 4, 11, 13–16, 18].

The novelty of this paper is the impatient behaviour of the waiting customers in the described environment, namely a single server unreliable system with a virtual waiting facility (orbit) and with possibility of collision of customers. This type of behaviour was also investigated by some authors, e.g. [5, 21]. A customer transferred to the orbit may retry its request several times. In case of unsuccessful retries after a random, exponentially distributed time the customer leaves the system (the orbit), and goes back to the source. This customer remains unserved. Our goal is to calculate the steady-state probabilities and the performance measures of these type of systems. The empirical distribution of the system probabilities and the effect of the impatient parameter are also investigated.

2. System model

The system under consideration is modeled by a finite source closed retrial queuing system of type $M/M/1//N$. The system has one server and the number of sources is N. In this paper two working characteristics of the server are distinguished:

- Non-reliable server and patient customers. The server is subject to random breakdowns. The breakdown times are exponentially distributed. The breakdown parameters for busy and idle server are γ_0 and γ_1, respectively. In case of breakdown the request under service is sent to the orbit. After the breakdown the repair starts immediately. The repair time is exponentially distributed with parameter γ_2. While the server is under repair, the sources are able to generate requests. These customers are transferred to the orbit, because the server is not available. The requests in the orbit may retry reaching the server again after an exponentially distributed time with parameter σ/N. The customers are patient, that is they keep retrying from the orbit until they are served.

- Non-reliable server and impatient customers. The breakdown behaviour of the server is the same, as in the previous point. The customers are impatient, that is a customer keeps retrying until it is served, or the customer leaves the orbit and goes back to the source after an exponentially distributed waiting time with parameter τ.

A job (customer) is generated in the source towards the server. The distribution of the inter-request times are exponential with parameter λ/N. The customer enters the system, and the source waits for a successful service. Until the end of service of the job the source cannot generate a new request. The new customer tries reaching the server. The state of the server can be busy or idle. When the server is idle, the service of the customer starts immediately. The distribution of service times is exponential with parameter μ. In case of a busy server state a conflict of customers can be occur: when an arriving job finds the server busy it involves into collision with customer under service and both customers are moved into the orbit. See the model on Figure 1.

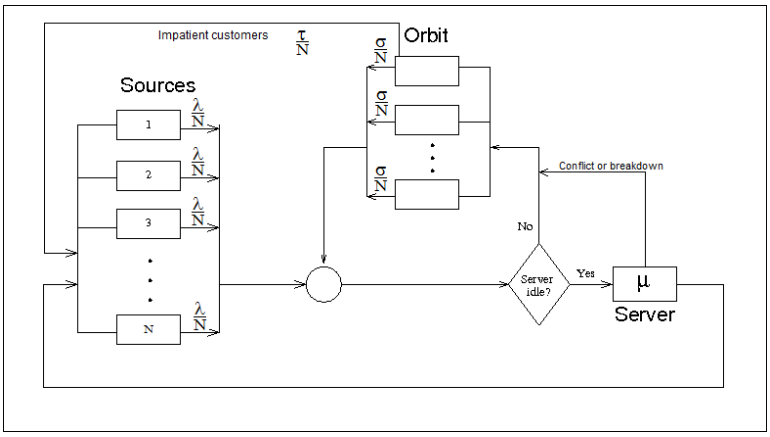


Figure 1: System model

Let $i(t)$ be the number of customers in the system. The customer can be either in the orbit or under service. Let $k(t)$ denote the status of the server:

$$k(t) = \begin{cases} 0, & \text{if the server is idle,} \\ 1, & \text{if the server is busy,} \\ 2, & \text{if the server is under repair.} \end{cases}$$

Let us denote the probability that at the time t there are i customers in "waiting" state and the server is in the state k by $P(k(t) = k, i(t) = i) = P_k(i,t)$. Under the above assumption the process $X(t) = \{k(t), i(t)\}$ is a 2-dimensional Markov-chain with a state space of $\{0,1,2\} \times \{0,1,\ldots,N\}$.

The successfully served customer goes back to the source. All the random variables involved in the model construction are assumed to be totally independent from each other.

For the non-impatient case the, the Kolmogorov differential-equations for probabilities $P_k(i,t)$ are the following (see in [14, 16]):

$$\begin{aligned}
\frac{\partial P_0(0,t)}{\partial t} &= -(\lambda + \gamma_0)P_0(0,t) + \mu P_1(1,t) + \gamma_2 P_2(0,t), \\
\frac{\partial P_1(1,t)}{\partial t} &= -\left(\lambda\frac{N-1}{N} + \mu + \gamma_1\right)P_1(1,t) + \lambda P_0(0,t) + \frac{\sigma}{N}P_0(1,t), \\
\frac{\partial P_2(0,t)}{\partial t} &= -(\lambda + \gamma_2)P_2(0,t) + \gamma_0 P_0(0,t), \\
\frac{\partial P_0(i,t)}{\partial t} &= -\left(\lambda\frac{N-1}{N} + \sigma\frac{i}{N} + \gamma_0\right)P_0(i,t) + \mu P_1(i+1,t) \\
&\quad + \lambda\frac{N-i+1}{N}P_1(i-1,t) + \sigma\frac{i-1}{N}P_1(i,t) + \gamma_2 P_2(i,t),
\end{aligned}$$

$$\frac{\partial P_1(i,t)}{\partial t} = -\left(\lambda\frac{N-1}{N} + \sigma\frac{i-1}{N} + \gamma_1 + \mu\right) P_1(i,t) + \lambda\frac{N-i+1}{N}P_0(i-1,t) + \sigma\frac{i}{N}P_0(i,t),$$

$$\frac{\partial P_2(i,t)}{\partial t} = -\left(\lambda\frac{N-1}{N} + \gamma_2\right) P_2(i,t) + \gamma_0 P_0(i,t) + \gamma_1 P_1(i,t) + \lambda\frac{N-i+1}{N}P_2(i-1,t).$$

Since $X(t) = \{k(t), i(t)\}$ is a finite state Markov-chain it can be assumed that it operates in steady-state that is: $P_k(i,t) = P_k(i)$.

Hence the steady-state Kolmogorov-equations can be written as

$$-(\lambda+\gamma_0)P_0(0) + \mu P_1(1) + \gamma_2 P_2(0) = 0,$$

$$-\left(\lambda\frac{N-1}{N} + \mu + \gamma_1\right) P_1(1) + \lambda P_0(0) + \frac{\sigma}{N}P_0(1) = 0,$$

$$-(\lambda+\gamma_2)P_2(0) + \gamma_0 P_0(0) = 0,$$

$$-\left(\lambda\frac{N-1}{N} + \sigma\frac{i}{N} + \gamma_0\right) P_0(i) + \mu P_1(i+1) + \lambda\frac{N-i+1}{N}P_1(i-1) + \sigma\frac{i-1}{N}P_1(i) + \gamma_2 P_2(i) = 0,$$

$$-\left(\lambda\frac{N-1}{N} + \sigma\frac{i-1}{N} + \gamma_1 + \mu\right) P_1(i) + \lambda\frac{N-i+1}{N}P_0(i-1) + \sigma\frac{i}{N}P_0(i) = 0,$$

$$-\left(\lambda\frac{N-1}{N} + \gamma_2\right) P_2(i) + \gamma_0 P_0(i) + \gamma_1 P_1(i) + \lambda\frac{N-i+1}{N}P_2(i-1) = 0.$$

Note, if all of the γ_2 parameters and P_2 probabilities are set to zero, we get the formulas for the system with conflict and reliable server.

By the help of the same method described above, the steady-state Kolmogorov-equations can be obtained for the system with conflict, non-reliable server and impatient customers:

$$-(\lambda+\gamma_0)P_0(0) + \mu P_1(1) + \gamma_2 P_2(0) + \frac{\tau}{n}P_0(1) = 0,$$

$$-\left(\lambda\frac{N-1}{N} + \mu + \gamma_1\right) P_1(1) + \lambda P_0(0) + \frac{\sigma}{N}P_0(1) + \frac{\tau}{n}P_1(2) = 0,$$

$$-(\lambda+\gamma_2)P_2(0) + \gamma_0 P_0(0) = 0,$$

$$-\left(\lambda\frac{N-i}{N} + \sigma\frac{i}{N} + \tau\frac{i}{N} + \gamma_0\right) P_0(i) + \mu P_1(i+1) + \lambda\frac{N-i+1}{N}P_1(i-1) + \sigma\frac{i-1}{N}P_1(i) + \tau\frac{i+1}{N}P_0(i+1) + \gamma_2 P_2(i) = 0,$$

$$-\left(\lambda\frac{N-i}{N} + \sigma\frac{i-1}{N} + \tau\frac{i-1}{N} + \gamma_1 + \mu\right) P_1(i) + \lambda\frac{N-i+1}{N}P_0(i-1)$$

$$+\sigma\frac{i}{N}P_0(i)+\tau\frac{i}{N}P_1(i+1)=0,$$
$$-\left(\lambda\frac{N-i}{N}+\gamma_2\right)P_2(i)+\gamma_0P_0(i)+\gamma_1P_1+\lambda\frac{N-i+1}{N}P_2(i-1)$$
$$+\tau\frac{i+1}{N}P_2(i+1)=0.$$

3. Performance Measures

The performance measures express the effect of the input parameters of the system. Let us define the most important characteristics which can be determined directly from the steady state probabilities.

- Mean number of customers in the system $\overline{Q}$ and in the orbit $\overline{O}$

$$\overline{Q}=\sum_{i=0}^{N}iP(i),\quad \overline{O}=\overline{Q}-P_1,$$

- Mean arrival rate $\overline{\lambda}$

$$\overline{\lambda}=\sum_{k=0}^{1}\sum_{i=0}^{N}(N-i)\frac{\lambda}{N}P_k(i),$$

- Mean response time $\overline{T}$ and mean waiting time $\overline{W}$ in the orbit can be obtained by the Little-formula

$$\overline{T}=\frac{\overline{Q}}{\overline{\lambda}},\quad \overline{W}=\frac{\overline{O}}{\overline{\lambda}},\quad \overline{O}=\overline{Q}-P_1,$$

- Mean total service time $E(T_S)$ and mean total sojourn time in the source $E(\kappa)$

$$E(T_S)=\overline{T}-\overline{W},\quad E(\kappa)=\frac{(N-\overline{Q})\overline{T}}{\overline{Q}},$$

- Mean number of trials from the source $E(N_{TS})$ and from the orbit $E(N_{TO})$

$$E(N_{TS})=\frac{\lambda}{N}E(\tau),\quad E(N_{TO})=\frac{\sigma}{N}\overline{W}.$$

4. Numerical solution

Before this model several other systems were investigated. Simple retrial queueing models, retrial models with conflict of customers, non-reliable retrial models with conflict of customers, retrial models with two-way communications. Obtaining the

system characteristics three different solutions were performed: recursive numerical calculations, solving the system equations (e.g. by MOSEL-2 tool), and run simulations. The results of the three different approaches were identical. Using these results, we were able to investigate models, where all of the mentioned solutions were not applicable. For example, systems with non-exponentially distributed service times can not be solved by MOSEL-2, but simulation and, in some cases, numerical solution proved useful.

The situation for this model is very similar. For the non-impatient case the equations can be solved recursively (described in [14, 16, 19]). The resulting steady-state probabilities $P_k(i)$ can be used for calculating the system performance measures. For double-checking the result, MOSEL-2 tool can also be applied here. For the impatient case we did our best, but such recursive solution cannot be obtained, because new variables enter into equations due to the impatient property. For this impatient case a software tool, MOSEL-2 is used to solve the system equations. The correctness of MOSEL-calculations was empirically proved in cases, when this tool and the numerical calculations were used simultaneously.

On Figure 2 the steady-state probabilities are displayed for the different models (non-conflict, conflict, unreliable, unreliable block, unreliable impatient). When the calculations are performed by MOSEL-2 tool (MOdeling Specification and Evaluation Language), see in [6], we run into a strict limitations, namely the state space grows extremely fast, consequently the number of sources cannot exceed 200. In Excel we can go far more above 200 (when the recursive calculations can be performed.

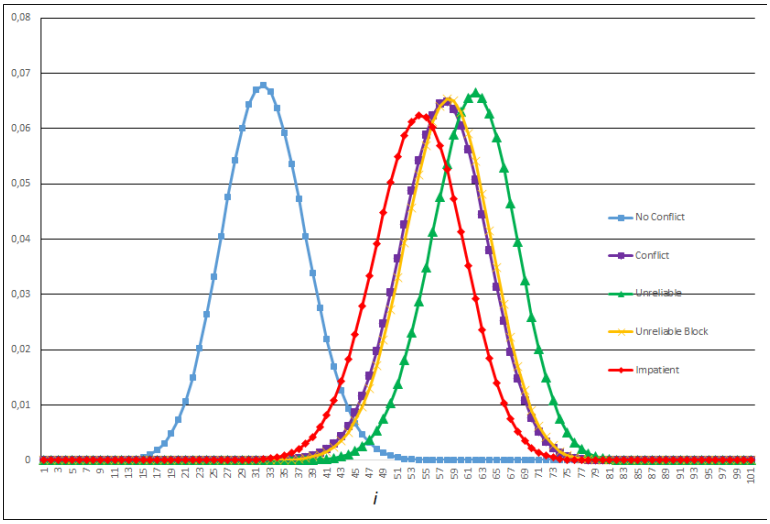


Figure 2: Different models

The first important question was the distribution of the system probabilities $P_k(i)$. Previous investigations the distribution was found very close to normal dis-

tribution. The normality is important, because in general only the average number of customers in the orbit or average waiting time of customers in the orbit can be calculated by the methods mentioned in this paper. But, if the normality of steady state probabilities can be assumed, the limiting probability distribution of the sojourn time/waiting time of the customer in the orbit can be obtained by asymptotic methods. See in [7, 17]. That's why is it important to find domains of parameters, where the steady-state system (or orbit) probabilities have normal or asymptotic normal distribution. Here the normality of the distribution was checked for different numbers of sources: $N = 50, 100$ and 200. Then the Kolmogorov-distance was computed. For the Kolmogorov-distance the theoretical normal distribution is calculated by using Excel built-in function. The parameters of the distribution is calculated from the steady-state probabilities. For example, in case of $N = 100$, a normal distribution is generated with mean of 52.9 and standard deviation of 6.37. The Kolmogorov-distance is defined as:

$$\Delta_N = \max_{0 \le k \le N} \left| \sum_{i=0}^{k} P_{\text{Theoretical}}(i) - \sum_{i=0}^{k} P_{\text{Mosel}}(i) \right|.$$

The following result were found: $\Delta_{50} = 0.03, \Delta_{100} = 0.02, \Delta_{200} = 0.003$. Thus the normality of the system probabilities can be accepted.

On Figure 3 the cumulative distribution function (CDF) of the normal (Gaussian) distribution and the empirical CDF are compared. As from the Kolmogorov-distance can be expected, the two distributions are almost identical.

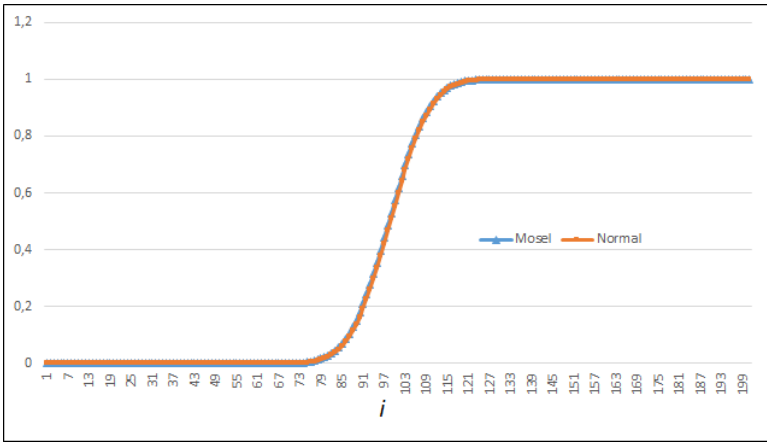


Figure 3: Normal CDF vs. empirical CDF

As described above, from the steady-state probabilities the performance measures (system characteristics) can be calculated.

On Figure 4 the mean response time, calculated by the help of formulas presented in Chapter 3 is displayed as a function of the overall generation rate. The

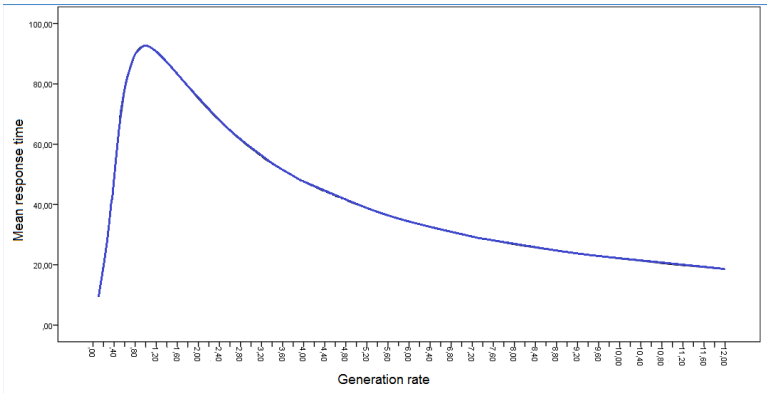


Figure 4: Mean response time vs. λ

expected maximum characteristic can be observed on this figure, as well. Under some parameter settings the finite-source retrial queueing systems have this maximum feature for several performance measures, e.g. response time. The reason is the special coincidence of the high generation rate and the low number of active tokens in the source (the number of jobs in the system is usually high at this situation).

5. Conclusion

The goal of this paper was to handle the impatient behaviour of customers in the environment of unreliable systems with collision. For non-impatient systems computing the steady-state system characteristics a recursive solution can be given. The impatient property makes the system equations more complex. New variables appear in the equations, so the recursive numeric solution cannot be performed. Because of this reason a software tool was used to solve the system equations. For this complex case there is no limit distribution of sojourn and waiting times of customers in the orbit. So, it is important to find domains of parameter, where the distribution of steady-state probabilities can be accepted as normal, to give the possibility of further theoretical investigations towards the limit distributions.

Acknowledgements. The research work was supported by the construction EFOP - 3.6.3 - VEKOP - 16-2017-00002. The project was supported by the European Union, co-financed by the European Social Fund.

References

[1] A.-A. Ali, S. Wei: *Modeling of coupled collision and congestion in finite source wireless access systems*, in: Wireless Communications and Networking Conference (WCNC), 2015 IEEE, IEEE, 2015, pp. 1113–1118.

[2] B. Almási, J. Roszik, J. Sztrik: *Homogeneous finite-source retrial queues with server subject to breakdowns and repairs*. English, Math. Comput. Modelling 42.5-6 (2005), pp. 673–682, ISSN: 0895-7177,
DOI: 10.1016/j.mcm.2004.02.046.

[3] J. Artalejo, A. G. Corral: *Retrial Queueing Systems: A Computational Approach*, Springer, 2008.

[4] S. Balsamo, G.-L. Dei Rossi, A. Marin: *Modelling retrial-upon-conflict systems with product-form stochastic Petri nets*, in: International Conference on Analytical and Stochastic Modeling Techniques and Applications, Springer, 2013, pp. 52–66.

[5] D. Barrer: *Queuing with impatient customers and ordered service*, Operation Research 5 (1957), pp. 650–656.

[6] T. Bérczes, J. Sztrik, Á. Tóth, A. Nazarov: *Performance Modeling of Finite-Source Retrial Queueing Systems with Collisions and Non-reliable Server Using MOSEL*, in: International Conference on Distributed Computer and Communication Networks, Springer, 2017, pp. 248–258.

[7] E. Y. Danilyuk, S. P. Moiseeva, J. Sztrik: *Asymptotic Analysis of Retrial Queueing System M/M/1 with Impatient Customers, Collisions and Unreliable Server*, Journal of Siberian Federal University. Mathematics & Physics 13.2 (2020), pp. 218–230,
DOI: 10.17516/1997-1397-2020-13-2-218-230.

[8] V. I. Dragieva: *Number of retrials in a finite source retrial queue with unreliable server*. English, Asia-Pac. J. Oper. Res. 31.2 (2014), p. 23, ISSN: 0217-5959; 1793-7019/e,
DOI: 10.1142/S0217595914400053.

[9] N. Gharbi, C. Dutheillet: *An algorithmic approach for analysis of finite-source retrial systems with unreliable servers*, English, Computers & Mathematics with Applications 62.6 (2011), pp. 2535–2546, ISSN: 0898-1221,
DOI: 10.1016/j.camwa.2011.03.109.

[10] A. Gómez-Corral, T. Phung-Duc: *Retrial queues and related models*, Annals of Operations Research 247.1 (2016), pp. 1–2, ISSN: 1572-9338,
DOI: 10.1007/s10479-016-2305-2.

[11] J. S. Kim: *Retrial queueing system with collision and impatience*, Communications of the Korean Mathematical Society 25.4 (2010), pp. 647–653.

[12] J. Kim, B. Kim: *A survey of retrial queueing systems*, Annals of Operations Research 247.1 (2016), pp. 3–36, ISSN: 1572-9338,
DOI: 10.1007/s10479-015-2038-7.

[13] A. Kuki, T.Bérczes, J. Sztrik, A. Kvach: *Numerical Analysis of Retrial Queueing Systems with Conflict of Customers*, Journal of Mathematical Sciences (2017), submitted.

[14] A. Kvach, A. Nazarov: *Sojourn Time Analysis of Finite Source Markov Retrial Queuing System with Collision*, in: Information Technologies and Mathematical Modelling - Queueing Theory and Applications: 14th International Scientific Conference, ITMM 2015, named after A. F. Terpugov, Anzhero-Sudzhensk, Russia, November 18-22, 2015, Proceedings, Cham: Springer International Publishing, 2015, chap. 8, pp. 64–72.

[15] T. V. Lyubina, A. A. Nazarov: *Research of the non-Markov dynamic retrial queue system with collision (In Russian)*, Herald of Kemerovo State University 1.49 (2012), pp. 38–44.

[16] A. Nazarov, A. Kvach, V. Yampolsky: *Asymptotic Analysis of Closed Markov Retrial Queuing System with Collision*, in: Information Technologies and Mathematical Modelling: 13th International Scientific Conference, ITMM 2014, named after A.F. Terpugov, Anzhero-Sudzhensk, Russia, November 20-22, 2014. Proceedings, Cham: Springer International Publishing, 2014, chap. 1, pp. 334–341, isbn: 978-3-319-13671-4.

[17] A. Nazarov, J. Sztrik, A. Kvach, Á. Tóth: *Asymptotic sojourn time analysis of finite-source M/M/1 retrial queueing system with collisions and server subject to breakdowns and repairs*, Annals of Operations Research 288.1 (2019), pp. 417–434, doi: 10.1007/s10479-019-03463-0.

[18] Y. Peng, Z. Liu, J. Wu: *An* M/G/1 *retrial G-queue with preemptive resume priority and collisions subject to the server breakdowns and delayed repairs.* English, J. Appl. Math. Comput. 44.1-2 (2014), pp. 187–213, issn: 1598-5865; 1865-2085/e, doi: 10.1007/s12190-013-0688-7.

[19] Á. Tóth, T. Bérczes, J. Sztrik, A. Kuki: *Comparison of two operation modes of finite-source retrial queueing systems with collisions and non-reliable server by using simulation*, Journal of Mathematical Sciences 237 (2019), pp. 846–857.

[20] J. Wang, L. Zhao, F. Zhang: *Analysis of the finite source retrial queues with server breakdowns and repairs*, English, Journal of Industrial and Management Optimization 7.3 (2011), pp. 655–676, issn: 1547-5816; 1553-166X/e, doi: 10.3934/jimo.2011.7.655.

[21] P. Wüchner, J. Sztrik, H. De Meer: *Finite-source M/M/S retrial queue with search for balking and impatient customers from the orbit*, Computer Networks 53.8 (2009), pp. 1264–1273.

[22] F. Zhang, J. Wang: *Performance analysis of the retrial queues with finite number of sources and service interruptions*, English, Journal of the Korean Statistical Society 42.1 (2013), pp. 117–131, issn: 1226-3192, doi: 10.1016/j.jkss.2012.06.002.

Annales Mathematicae et Informaticae
***51** (2020) pp. 65–76*
DOI: 10.33039/ami.2020.07.009
https://ami.uni-eszterhazy.hu

Closed Association Rules

Laszlo Szathmary

University of Debrecen, Faculty of Informatics
Debrecen, Hungary
szathmary.laszlo@inf.unideb.hu

Submitted: February 4, 2020
Accepted: July 10, 2020
Published online: July 23, 2020

Abstract

In this paper we present a new basis for association rules called Closed Association Rules ($\mathcal{CR}$). This basis contains all valid association rules that can be generated from frequent closed itemsets. $\mathcal{CR}$ is a lossless representation of all association rules. Regarding the number of rules, our basis is between all association rules ($\mathcal{AR}$) and minimal non-redundant association rules ($\mathcal{MNR}$), filling a gap between them. The new basis provides a framework for some other bases and we show that $\mathcal{MNR}$ is a subset of $\mathcal{CR}$. Our experiments show that $\mathcal{CR}$ is a good alternative for all association rules. The number of generated rules can be much less, and beside frequent closed itemsets nothing else is required.

1. Introduction

In data mining, frequent itemsets (FIs) and association rules play an important role [2]. Generating valid association rules (denoted by $\mathcal{AR}$) from frequent itemsets often results in a huge number of rules, which limits their usefulness in real life applications. To solve this problem, different concise representations of association rules have been proposed, e.g. generic basis ($\mathcal{GB}$), informative basis ($\mathcal{IB}$) [3], Duquennes-Guigues basis ($\mathcal{DG}$) [5], Luxenburger basis ($\mathcal{LB}$) [8], etc. A very good comparative study of these bases can be found in [7], where it is stated that a rule representation should be *lossless* (should enable the derivation of all valid rules), *sound* (should forbid the derivation of rules that are not valid) and *informative* (should allow the determination of rules parameters such as support and confidence).

In this paper we present a new basis for association rules called Closed Association Rules ($\mathcal{CR}$). The number of rules in $\mathcal{CR}$ is less than the number of all rules, especially in the case of dense, highly correlated data when the number of frequent itemsets is much more than the number of frequent closed itemsets. $\mathcal{CR}$ contains more rules than minimal non-redundant association rules ($\mathcal{MNR}$), but for the extraction of closed association rules we *only* need frequent closed itemsets, nothing else. On the contrary, the extraction of $\mathcal{MNR}$ needs much more computation since frequent generators also have to be extracted and assigned to their closures.[1]

The remainder of the paper is organized as follows. Background on pattern mining and concept analysis is provided in Section 2. All association rules, closed association rules and minimal non-redundant association rules are presented in Sections 3, 4 and 5, respectively. Experimental results are provided in Section 6, and Section 7 concludes the paper.

2. Basic concepts

In the following, we recall basic concepts from frequent pattern mining and formal concept analysis (FCA). The following 5×5 sample dataset: $\mathcal{D} = \{(1,\ ABDE),\ (2,\ AC),\ (3,\ ABCE),\ (4,\ BCE)\},\ (5,\ ABCE)\}$ will be used as a running example. Henceforth, we refer to it as dataset $\boldsymbol{\mathcal{D}}$.

Frequent itemsets. We consider a set of *objects* $O = \{o_1, o_2, \ldots, o_m\}$, a set of *attributes* $A = \{a_1, a_2, \ldots, a_n\}$, and a binary relation $R \subseteq O \times A$, where $R(o, a)$ means that the object o has the attribute a. In formal concept analysis the triple (O, A, R) is called a *formal context* [4]. The Galois connection for (O, A, R) is defined along the lines of [4] in the following way (here $B \subseteq O$, $D \subseteq A$):

$$B' = \{a \in A \mid R(o, a) \text{ for all } o \in B\}, \quad D' = \{o \in O \mid R(o, a) \text{ for all } a \in D\}.$$

In data mining applications, an element of A is called an *item* and a subset of A is called an *itemset*. Further on, we shall keep to these terms. An itemset of size i is called an i-itemset.[2] We say that an itemset $P \subseteq A$ *belongs* to an object $o \in O$, if $(o, p) \in R$ for all $p \in P$, or $P \subseteq o'$. The *support* of an itemset $P \subseteq A$ indicates the number of objects to which the itemset belongs: $supp(P) = |P'|$. An itemset is *frequent* if its support is not less than a given *minimum support* (denoted by *min_supp*). An itemset P is *closed* if there exists no proper superset with the same support. The closure of an itemset P (denoted by P'') is the largest superset of P with the same support. Naturally, if $P = P''$, then P is a closed itemset. The task of frequent itemset mining consists of generating all (closed) itemsets (with their supports) with supports greater than or equal to a specified *min_supp*.

Two itemsets $P, Q \subseteq A$ are said to be *equivalent* ($P \cong Q$) iff they belong to the same set of objects (i.e. $P' = Q'$). The set of itemsets that are equivalent to

[1] Concepts in this section are defined in Section 2.

[2] For instance, $\{A, B, E\}$ is a 3-itemset. Further on we use separator-free set notations, i.e. ABE stands for $\{A, B, E\}$.

an itemset P (P's *equivalence class*) is denoted by $[P] = \{Q \subseteq A \mid P \cong Q\}$. An itemset $P \in [P]$ is called a *generator*, if P has no proper subset in $[P]$, i.e. it has no proper subset with the same support. A *frequent generator* is a generator whose support is not less than a given minimum support.

Frequent association rules. An association rule is an expression of the form $P_1 \to P_2$, where P_1 and P_2 are arbitrary itemsets ($P_1, P_2 \subseteq A$), $P_1 \cap P_2 = \emptyset$ and $P_2 \neq \emptyset$. The left side, P_1 is called *antecedent*, the right side, P_2 is called *consequent*. The (absolute) support of an association rule r is defined as: $supp(r) = supp(P_1 \cup P_2)$. The *confidence* of an association rule $r\colon P_1 \to P_2$ is defined as the conditional probability that an object has itemset P_2, given that it has itemset P_1: $conf(r) = supp(P_1 \cup P_2)/supp(P_1)$. An association rule is *valid* if $supp(r) \geq min\_supp$ and $conf(r) \geq min\_conf$. The set of all valid association rules is denoted by $\mathcal{AR}$.

Minimal non-redundant association rules ($\mathcal{MNR}$) [3] have the following form: $P \to Q \setminus P$, where $P \subset Q$, P is a generator and Q is a closed itemset. That is, an $\mathcal{MNR}$ rule has a minimal antecedent and a maximal consequent. Minimal (resp. maximal) means that the antecedent (resp. consequent) is a minimal (resp. maximal) element in its equivalence class. Note that P and Q are not necessarily in the same equivalence class. As it was shown in [3], $\mathcal{MNR}$ rules contain the most information among rules with the same support and same confidence.

3. All association rules

From now on, by "all association rules" we mean all (frequent) *valid* association rules. The concept of association rules was introduced by Agrawal *et al.* [1]. Originally, the extraction of association rules was used on sparse market basket data. The first efficient algorithm for this task was *Apriori*. The generation of all valid association rules consists of two main steps:

1. Find all *frequent* itemsets P in a dataset, i.e. where $supp(P) \geq min\_supp$.
2. For each frequent itemset P_1 found, generate all confident association rules r of the form $P_2 \to (P_1 \setminus P_2)$, where $P_2 \subset P_1$ and $conf(r) \geq min\_conf$.

The more difficult task is the first step, which is computationally and I/O intensive.

Generating all valid association rules. Once all frequent itemsets and their supports are known, this step can be done in a relatively straightforward manner. The general idea is the following: for every frequent itemset P_1, all subsets P_2 of P_1 are derived, and the ratio $supp(P_1)/supp(P_2)$ is computed.[3] If the result is higher or equal to min_conf, then the rule $P_2 \to (P_1 \setminus P_2)$ is generated.

[3] $supp(P_1)/supp(P_2)$ is the confidence of the rule $P_2 \to (P_1 \setminus P_2)$.

The support of any subset P_3 of P_2 is greater than or equal to the support of P_2. Thus, the confidence of the rule $P_3 \rightarrow (P_1 \setminus P_3)$ is necessarily less than or equal to the confidence of the rule $P_2 \rightarrow (P_1 \setminus P_2)$. Hence, if the rule $P_2 \rightarrow (P_1 \setminus P_2)$ is not confident, then neither is the rule $P_3 \rightarrow (P_1 \setminus P_3)$. Conversely, if the rule $(P_1 \setminus P_2) \rightarrow P_2$ is confident, then all rules of the form $(P_1 \setminus P_3) \rightarrow P_3$ are confident. For example, if the rule $A \rightarrow BE$ is confident, then the rules $AB \rightarrow E$ and $AE \rightarrow B$ are confident as well.

Using this property for efficiently generating valid association rules, the algorithm works as follows [1]. For each frequent itemset P_1, all *confident* rules with one item in the consequent are generated. Then, using the `Apriori-Gen` function (from [1]) on the set of 1-long consequents, we generate consequents with 2 items. Only those rules with 2 items in the consequent are kept whose confidence is greater than or equal to min_conf. The 2-long consequents of the confident rules are used for generating consequents with 3 items, etc.

Example. Table 1 depicts which valid association rules ($\mathcal{AR}$) can be extracted from dataset $\mathcal{D}$ with $min\_supp = 3$ (60%) and $min\_conf = 0.5$ (50%). First, all frequent itemsets have to be extracted from the dataset. In $\mathcal{D}$ with $min\_supp = 3$ there are 12 frequent itemsets, namely A (supp: 4), B (4), C (4), E (4), AB (3), AC (3), AE (3), BC (3), BE (4), CE (3), ABE (3) and BCE (3).[4] Only those itemsets can be used for generating association rules that contain at least 2 items. Eight itemsets satisfy this condition. For instance, using the itemset ABE, which is composed of 3 items, the following rules can be generated: $BE \rightarrow A$ (supp: 3; conf: 0.75), $AE \Rightarrow B$ (3; 1.0) and $AB \Rightarrow E$ (3; 1.0). Since all these rules are confident, their consequents are used to generate 2-long consequents: AB, AE and BE. This way, the following rules can be constructed: $E \rightarrow AB$ (3; 0.75), $B \rightarrow AE$ (3; 0.75) and $A \rightarrow BE$ (3; 0.75). In general, it can be said that from an m-long itemset, one can potentially generate $2^m - 2$ association rules.

4. Closed Association Rules

In the previous section we presented all association rules that are generated from frequent itemsets. Unfortunately, the number of these rules can be very large, and many of these rules are redundant, which limits their usefulness. Applying concise rule representations (a.k.a. *bases*) with appropriate inference mechanisms can lessen the problem [7]. By definition, a *concise representation of association rules* is a subset of all association rules with the following properties: **(1)** it is much smaller than the set of all association rules, and **(2)** the whole set of all association rules can be restored from this subset (possibly with no access to the database, i.e. very efficiently) [6].

[4] Support values are indicated in parentheses.

$\mathcal{AR}$	supp.	conf.	$\mathcal{CR}$	$\mathcal{MNR}$
$B \to A$	3	0.75		
$A \to B$	3	0.75		
$C \to A$	3	0.75	+	+
$A \to C$	3	0.75	+	+
$E \to A$	3	0.75		
$A \to E$	3	0.75		
$C \to B$	3	0.75		
$B \to C$	3	0.75		
$E \Rightarrow B$	4	1.0	+	+
$B \Rightarrow E$	4	1.0	+	+
$E \to C$	3	0.75		
$C \to E$	3	0.75		
$BE \to A$	3	0.75	+	
$AE \Rightarrow B$	3	1.0	+	+
$AB \Rightarrow E$	3	1.0	+	+
$E \to AB$	3	0.75	+	+
$B \to AE$	3	0.75	+	+
$A \to BE$	3	0.75	+	+
$CE \Rightarrow B$	3	1.0	+	+
$BE \to C$	3	0.75	+	
$BC \Rightarrow E$	3	1.0	+	+
$E \to BC$	3	0.75	+	+
$C \to BE$	3	0.75	+	+
$B \to CE$	3	0.75	+	+

Table 1: Different sets of association rules extracted from dataset $\mathcal{D}$ with $min\_supp = 3$ (60%) and $min\_conf = 0.5$ (50%)

Related work. In addition to the first method presented in the previous section, there is another approach for finding all association rules. This approach was introduced in [9] by Bastide *et al.* They have shown that frequent closed itemsets are a lossless, condensed representation of frequent itemsets, since the whole set of frequent itemsets can be restored from them with the proper support values. They propose the following method for finding all association rules. First, they extract frequent closed itemsets[5], then they restore the set of frequent itemsets from them, and finally they generate all association rules. The number of FCIs is usually much less than the number of FIs, especially in dense and highly correlated datasets. In such databases the exploration of all association rules can be done more efficiently by this way. However, this method has some disadvantages: **(1)** the restoration of FIs from FCIs needs *lots of* memory, **(2)** the final result is still "all the association rules", which means lots of redundant rules.

[5]For this task they introduced a new algorithm called "Close". Close is a levelwise algorithm for finding FCIs.

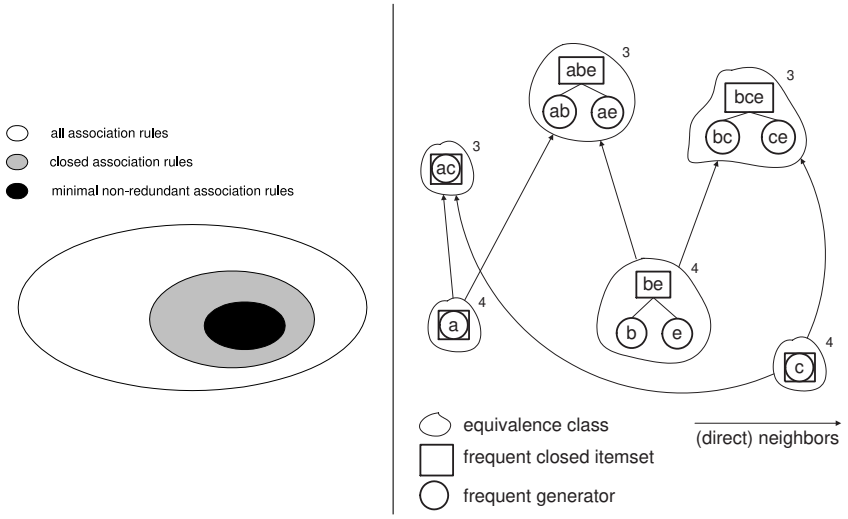


Figure 1: **Left:** position of Closed Rules; **Right:** equivalence classes of $\mathcal{D}$ with $min\_supp = 3$ (60%). Support values are indicated in the top right corners.

Contribution. We introduce a new basis called Closed Association Rules, or simply Closed Rules ($\mathcal{CR}$). This basis requires frequent closed itemsets only. The difference between our work and the work presented in [9] stems from the fact that although we also extract FCIs, instead of restoring all FIs from them, we use them *directly* to generate valid association rules. This way, we find less and probably more interesting association rules.

$\mathcal{CR}$ is a generating set for all valid association rules with their proper support and confidence values. Our basis fills a gap between all association rules and minimal non-redundant association rules ($\mathcal{MNR}$), as depicted in Figure 1 (left). $\mathcal{CR}$ contains all valid rules that are derived from frequent closed itemsets. Since the number of FCIs are usually much less than the number of FIs, the number of rules in our basis is also much less than the number of all association rules. Using our basis the restoration of all valid association rules can be done without any loss of information. It is possible to deduce efficiently, without access to the dataset, all valid association rules with their supports and confidences from this basis, since frequent closed itemsets are a lossless representation of frequent itemsets. Furthermore, we will show in the next section that minimal non-redundant association rules are a special subset of the Closed Rules, i.e. $\mathcal{MNR}$ can be defined in the framework of our basis. $\mathcal{CR}$ has the advantage that its rules can be generated very easily since only the frequent closed itemsets are needed. As there are usually much less FCIs than FIs, the derivation of the Closed Rules can be done much more efficiently than generating all association rules.

Before showing our algorithm for finding the Closed Rules, we present the essential definitions.

Definition 4.1 (closed association rule)**.** An association rule $r\colon P_1 \to P_2$ is called *closed* if $P_1 \cup P_2$ is a closed itemset.

This definition means that the rule is derived from a closed itemset.

Definition 4.2 (Closed Rules)**.** Let FC be the set of frequent closed itemsets. The set of Closed Rules contains *all* valid closed association rules:

$$\mathcal{CR} = \{r\colon P_1 \to P_2 \mid (P_1 \cup P_2) \in FC \wedge supp(r) \geq min\_supp \wedge conf(r) \geq min\_conf\}.$$

Property 4.3. *The support of an arbitrary frequent itemset is equal to the support of its smallest frequent closed superset [9].*

By this property, FCIs are a condensed lossless representation of FIs. This is also called the *frequent closed itemset representation* of frequent itemsets. Property 4.3 can be generalized the following way:

Property 4.4. *If an arbitrary itemset X has a frequent closed superset, then X is frequent and its support is equal to the support of its smallest frequent closed superset. If X has no frequent closed superset, then X is not frequent.*

The algorithm. The idea behind generating all valid association rules is the following. First we need to extract all frequent itemsets. Then rules of the form $X \setminus Y \to Y$, where $Y \subset X$, are generated for all frequent itemsets X, provided the rules have at least minimum confidence.

Finding closed association rules is done similarly. However, this time we only have frequent *closed* itemsets available. In this case the left side of a rule $X \setminus Y$ can be non-closed. For calculating the confidence of rules its support must be known. Thanks to Property 4.3, this support value can be calculated by only using frequent closed itemsets. It means that only FCIs are needed; all frequent itemsets do not have to be extracted. This is the principle idea behind this part of our work.

Example. Table 1 depicts which closed association rules ($\mathcal{CR}$) can be extracted from dataset $\mathcal{D}$ with $min\_supp = 3$ (60%) and $min\_conf = 0.5$ (50%). First, frequent closed itemsets must be extracted from the dataset. In $\mathcal{D}$ with $min\_supp = 3$ there are 6 FCIs, namely A (supp: 4), C (4), AC (3), BE (4), ABE (3) and BCE (3). Note that the total number of frequent itemsets by these parameters is 12. Only those itemsets can be used for generating association rules that contain at least 2 items. There are 4 itemsets that satisfy this condition, namely itemsets AC (supp: 3), BE (4), ABE (3) and BCE (3). Let us see which rules can be generated from the itemset BCE for instance. Applying the algorithm from [1], we get three rules: $CE \to B$, $BE \to C$ and $BC \to E$. Their support is known, it is equal to the support of BCE. To calculate the confidence values we need to know the support of the left sides too. The support of BE is known since it is a closed

itemset, but CE and BC are non-closed. Their supports can be derived by Property 4.3. The smallest frequent closed superset of both CE and BC is BCE, thus their supports are equal to the support of this closed itemset, which is 3. Then, using the algorithm from [1], we can produce three more rules: $E \rightarrow BC$, $C \rightarrow BE$ and $B \rightarrow CE$. Their confidence values are calculated similarly. From the four frequent closed itemsets 16 closed association rules can be extracted altogether, as depicted in Table 1.

5. Minimal non-redundant association rules

As seen in Section 2, minimal non-redundant association rules ($\mathcal{MNR}$) have the following form: $P \rightarrow Q \setminus P$, where $P \subset Q$, P is a generator and Q is a closed itemset.

In order to generate these rules efficiently, one needs to extract the frequent closed itemsets (FCIs), the frequent generators (FGs), and then these itemsets must be grouped together. That is, to generate these rules, one needs to explore all the frequent equivalence classes in a dataset (see Figure 1, right). Most algorithms address either FCIs or FGs, and only few algorithms can extract both types of itemsets.

Example. Table 1 depicts which $\mathcal{MNR}$ rules can be extracted from dataset $\mathcal{D}$ with $min\_supp = 3$ (60%) and $min\_conf = 0.5$ (50%). As can be seen, there are 14 $\mathcal{MNR}$ rules in the dataset. For instance, $BE \rightarrow A$ is not an $\mathcal{MNR}$ rule becasue its antecedent (BE) is not a generator (see Figure 1, right). To learn more about the $\mathcal{MNR}$ rules, please refer to [11].

Comparing $\mathcal{CR}$ and $\mathcal{MNR}$. As we have seen, $\mathcal{CR}$ is a maximal set of closed association rules, i.e. it contains *all* closed association rules. As a consequence, we cannot say that this basis is minimal, or non-redundant, but by all means it is a smaller set than $\mathcal{AR}$, especially in the case of dense, highly correlated datasets. Moreover, $\mathcal{CR}$ is a framework for some other bases. For instance, minimal non-redundant association rules are also closed association rules, since by definition the union of the antecedent and the consequent of such a rule forms a frequent closed itemset. Thus, $\mathcal{MNR}$ is a special subset of $\mathcal{CR}$, which could also be defined the following way:

Definition 5.1. Let CR be the set of Closed Rules. The set of minimal non-redundant association rules is:

$$\mathcal{MNR} = \{r\colon P_1 \rightarrow P_2 \mid r \in CR \wedge P_1 \text{ is a frequent generator}\}.$$

This is equivalent to the following definition:

$$\mathcal{MNR} = \{r\colon P_1 \rightarrow P_2 \mid (P_1 \cup P_2) \in FC \wedge P_1 \text{ is a frequent generator}\},$$

where FC stands for the set of frequent closed itemsets.

6. Experimental results

For comparing the different sets of association rules ($\mathcal{AR}$, $\mathcal{CR}$ and $\mathcal{MNR}$), we used the multifunctional *Zart* algorithm [11] from the CORON[6] system [10]. *Zart* was implemented in Java. The experiments were carried out on an Intel Pentium IV 2.4 GHz machine running Debian GNU/Linux with 2 GB RAM. All times reported are real, wall clock times as obtained from the Unix *time* command between input and output. For the experiments we have used the following datasets: T20I6D100K, C20D10K and MUSHROOMS.[7] It has to be noted that T20 is a sparse, weakly correlated dataset imitating market basket data, while the other two datasets are dense and highly correlated. Weakly correlated data usually contain few frequent itemsets, even at low minimum support values, and almost all frequent itemsets are closed. On the contrary, in the case of highly correlated data the difference between the number of frequent itemsets and frequent closed itemsets is significant.

6.1. Number of rules

Table 2 shows the following information: minimum support and confidence; number of all association rules; number of closed rules; number of minimal non-redundant association rules. We attempted to choose significant min_supp and min_conf thresholds as observed in other papers for similar experiments.

In T20 almost all frequent itemsets are closed, thus the number of all rules and the number of closed association rules is almost equal. For the other two datasets that are dense and highly correlated, the reduction of the number of rules in the Closed Rules is considerable.

The size of the $\mathcal{MNR}$ set is almost equal to the size of $\mathcal{AR}$ in sparse datasets, but in dense datasets $\mathcal{MNR}$ produces much less rules.

6.2. Execution times of rule generation

Figure 3 shows for each dataset the execution times of the computation of all, closed and minimal non-redundant association rules. For the extraction of the necessary itemsets we used the multifunctional *Zart* algorithm [11] that can generate all three kinds of association rules. Figure 3 does not include the extraction time of itemsets, it only shows the time of rule generation.

For datasets with much less frequent closed itemsets (C20, MUSHROOMS), the generation of closed rules is more efficient than finding all association rules. As seen before, we need to look up the closed supersets of frequent itemsets very often when extracting closed rules. For this procedure we use the trie data structure that shows its advantage on dense, highly correlated datasets. On the contrary, when almost all frequent itemsets are closed (T20), the high number of superset operations cause that all association rules can be extracted faster.

[6] `http://coron.loria.fr`

[7] `https://github.com/jabbalaci/Talky-G/tree/master/datasets`

dataset (min_supp)	min_conf	$\mathcal{AR}$	$\mathcal{CR}$	$\mathcal{MNR}$
$\mathcal{D}$ (40%)	50%	50	30	25
T20I6D100K (0.5%)	90%	752,715	726,459	721,948
	70%	986,058	956,083	951,572
	50%	1,076,555	1,044,086	1,039,575
	30%	1,107,258	1,073,114	1,068,603
C20D10K (30%)	90%	140,651	47,289	9,221
	70%	248,105	91,953	19,866
	50%	297,741	114,245	25,525
	30%	386,252	138,750	31,775
Mushrooms (30%)	90%	20,453	5,571	1,496
	70%	45,147	11,709	3,505
	50%	64,179	16,306	5,226
	30%	78,888	21,120	7,115

Table 2: Comparing sizes of different sets of association rules

dataset (min_supp)	min_conf	$\mathcal{AR}$	$\mathcal{CR}$	$\mathcal{MNR}$
T20I6D100K (0.5%)	90%	114.43	120.30	394.14
	70%	147.69	152.31	428.59
	50%	165.48	167.07	441.52
	30%	169.66	170.06	449.47
C20D10K (30%)	90%	15.72	12.49	1.68
	70%	26.98	21.10	2.77
	50%	34.74	24.24	3.35
	30%	41.40	27.36	4.04
Mushrooms (30%)	90%	1.93	1.49	0.54
	70%	3.99	2.44	0.78
	50%	5.63	2.98	1.00
	30%	6.75	3.31	1.28

Table 3: Execution times of rule generation (given is seconds)

Experimental results show that $\mathcal{CR}$ can be generated more efficiently than $\mathcal{MNR}$ on sparse datasets. However, on dense datasets $\mathcal{MNR}$ can be extracted much more efficiently.

7. Conclusion

In this paper we presented a new basis for association rules called Closed Rules ($\mathcal{CR}$). This basis contains all valid association rules that can be generated from frequent closed itemsets. $\mathcal{CR}$ is a lossless representation of all association rules. Regarding the number of rules, our basis is between all association rules ($\mathcal{AR}$) and minimal non-redundant association rules ($\mathcal{MNR}$), filling a gap between them. The

new basis provides a framework for some other bases. We have shown that $\mathcal{MNR}$ is a subset of $\mathcal{CR}$. The number of extracted rules is less than the number of all rules, especially in the case of dense, highly correlated data when the number of frequent itemsets is much more than the number of frequent closed itemsets. $\mathcal{CR}$ contains more rules than $\mathcal{MNR}$, but for the extraction of closed association rules we *only* need frequent closed itemsets, nothing else. On the contrary, the extraction of minimal non-redundant association rules needs much more computation since frequent generators also have to be extracted and assigned to their closures.

As a summary, we can say that $\mathcal{CR}$ is a good alternative for all association rules. The number of generated rules can be much less, and beside frequent closed itemsets nothing else is required.

Acknowledgement

This work was supported by the construction EFOP-3.6.3-VEKOP-16-2017-00002. The project was supported by the European Union, co-financed by the European Social Fund.

References

[1] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, A. I. Verkamo: *Fast discovery of association rules*, in: Advances in knowledge discovery and data mining, American Association for Artificial Intelligence, 1996, pp. 307–328, isbn: 0-262-56097-6.

[2] R. Agrawal, R. Srikant: *Fast Algorithms for Mining Association Rules in Large Databases*, in: Proc. of the 20th Intl. Conf. on Very Large Data Bases (VLDB '94), San Francisco, CA: Morgan Kaufmann, 1994, pp. 487–499, isbn: 1-55860-153-8.

[3] Y. Bastide, R. Taouil, N. Pasquier, G. Stumme, L. Lakhal: *Mining Minimal Non-Redundant Association Rules Using Frequent Closed Itemsets*, in: Proc. of the Computational Logic (CL '00), vol. 1861, LNAI, Springer, 2000, pp. 972–986.

[4] B. Ganter, R. Wille: *Formal concept analysis: mathematical foundations*, Berlin / Heidelberg: Springer, 1999, p. 284, isbn: 3540627715.

[5] J. L. Guigues, V. Duquenne: *Familles minimales d'implications informatives résultant d'un tableau de données binaires*, Mathématiques et Sciences Humaines 95 (1986), pp. 5–18.

[6] B. Jeudy, J.-F. Boulicaut: *Using condensed representations for interactive association rule mining*, in: Proc. of PKDD '02, volume 2431 of LNAI, Helsinki, Finland, Springer-Verlag, 2002, pp. 225–236.

[7] M. Kryszkiewicz: *Concise Representations of Association Rules*, in: Proc. of the ESF Exploratory Workshop on Pattern Detection and Discovery, 2002, pp. 92–109.

[8] M. Luxenburger: *Implications partielles dans un contexte*, Mathématiques, Informatique et Sciences Humaines 113 (1991), pp. 35–55.

[9] N. Pasquier, Y. Bastide, R. Taouil, L. Lakhal: *Efficient mining of association rules using closed itemset lattices*, Inf. Syst. 24.1 (1999), pp. 25–46, issn: 0306-4379, doi: http://dx.doi.org/10.1016/S0306-4379(99)00003-4.

[10] L. Szathmary: *Symbolic Data Mining Methods with the Coron Platform*, PhD Thesis in Computer Science, Univ. Henri Poincaré – Nancy 1, France, Nov. 2006.

[11] L. Szathmary, A. Napoli, S. O. Kuznetsov: *ZART: A Multifunctional Itemset Mining Algorithm*, in: Proc. of the 5th Intl. Conf. on Concept Lattices and Their Applications (CLA '07), Montpellier, France, Oct. 2007, pp. 26–37,
url: `http://hal.inria.fr/inria-00189423/en/`.

Annales Mathematicae et Informaticae
***51** (2020) pp. 77–87*
DOI: 10.33039/ami.2020.07.006
https://ami.uni-eszterhazy.hu

Improving the simultaneous application of the DSN-PC and NOAA GFS datasets*

Ádám Vas[a], Oluoch Josphat Owino[a], László Tóth[ab]

[a]Faculty of Informatics, University of Debrecen
vas.adam@inf.unideb.hu
josphatowino@gmail.com

[b]SciTech Műszer Kft, Debrecen, Hungary
laszlo.toth@scitechmuszer.com

Submitted: February 4, 2020
Accepted: July 1, 2020
Published online: July 23, 2020

Abstract

Our surface-based sensor network, called Distributed Sensor Network for Prediction Calculations (DSN-PC) obviously has limitations in terms of vertical atmospheric data. While efforts are being made to approximate these upper-air parameters from surface-level, as a first step it was necessary to test the network's capability of making distributed computations by applying a hybrid approach. We accessed public databases like NOAA Global Forecast System (GFS) and the initial values for the 2-dimensional computational grid were produced by using both DSN-PC measurements and NOAA GFS data for each grid point. However, though the latter consists of assimilated and initialized (smoothed) data the stations of the DSN-PC network provide raw measurements which can cause numerical instability due to measurement errors or local weather phenomena. Previously we simultaneously interpolated both DSN-PC and GFS data. As a step forward, we wanted for our network to have a more significant role in the production of the initial values. Therefore it was necessary to apply 2D smoothing algorithms on the initial conditions. We found significant difference regarding numerical stability between calculating with raw and smoothed initial data. Applying the smoothing algorithms greatly improved the prediction reliability compared

*This work was supported by the construction EFOP-3.6.3-VEKOP-16-2017-00002. The project was co-financed by the Hungarian Government and the European Social Fund.

to the cases when raw data were used. The size of the grid portion used for smoothing has a significant impact on the goodness of the forecasts and it's worth further investigation. We could verify the viability of direct integration of DSN-PC data since it provided forecast errors similar to the previous approach. In this paper we present one simple method for smoothing our initial data and the results of the weather prediction calculations.

Keywords: sensor network, distributed computing, weather prediction, data assimilation, data smoothing

MSC: 86A10, 68M14

1. Introduction

Sensor networks have generated a great interest in scientific areas and becoming more popular as the devices used for building such a network are available at a low price while their reliability and capabilities are higher than ever before. In the early days sensor networks consisted of simple data logger devices equipped with sensors. Their only role was to collect and store the measured data. However, recent sensor networks usually make real-time data available through the Internet, and they can be used for purposes other than data logging.

Such sensor networks are already widely used by meteorological agencies, although their network nodes (sensor stations) are of a much developed and industrial category. However, a lot of other competitors entered the business worldwide, and it looks like they can produce significant results, too. One reason for that is the higher spatial resolution in terms of surface-level measurements. With the evolution of wireless technologies and IoT, their role is expected to grow even further.

It's worth mentioning that the computing potential that's available at large meteorological agencies is unquestionably a great advantage. Still, there is potential in sensor networks in this matter, because the network nodes can also be used for computational tasks [25, 27]. This way a central supercomputer can be eliminated because the calculations can be performed in a fully distributed way. Previously we followed a mixed approach by integrating our own Distributed Sensor Network for Prediction Calculations (DSN-PC) nodes and NOAA GFS data into a hybrid sensor- and computational network [28]. Following that, we wanted to step further by increasing the involvement of our own measurements in the final hybrid network's initial data. This approach arose some problems that we tried to mitigate, such as big differences (spikes) between two geographically adjacent measurements. Due to the simplicity of our currently used numerical model these caused numerical instability and incorrect results in the forecast calculations. This is a widely researched area in meteorology and several data assimilation [1, 16, 17] and data smoothing [14, 21, 31] techniques exist to address that problem. Among them, the spline methods have been the most widely used which have been discussed in several articles [2, 4–7, 20, 24, 29] and algorithms are already available to implement them in a distributed form [22]. Applications in atmospheric and geosciences also show the viability of these smoothing methods [11, 12, 15]. A subtype

called thin-plate smoothing spline procedure described by Hutchinson [10] has been widely used in geosciences and performed well in global studies [18, 19] as well as in comparative tests of multiple interpolation techniques [9, 13]. Our final goal is to implement it on our system in a distributed form – however, before moving to these advanced techniques we aimed for a much simpler distributed algorithm to check if smoothing alone is enough to achieve numerical stability and satisfactory prediction results.

In this paper we introduce a minor improvement over our previous approach by directly injecting our measurements into the computational grid's initial data. A simple smoothing algorithm was applied to the initial data which can be performed distributedly by the nodes communicating with each other. Below the results of these numerical weather prediction calculations are shown.

2. System and model description

2.1. The geographical area and data sources

We implemented a virtual sensor network which covers a European area and consists of 20×20 nodes forming a regular grid on a map using polar stereographic projection. In order to produce the results in this paper the network nodes were simulated as Java threads. Figure 1 shows the locations of the grid points. The detailed properties of the grid are described in our previous paper [28].

The initial values for the computations are from 2 data sources: our 5 DSN-PC weather stations installed in Hungary and data from the publicly available GFS-ANL database [8]. DSN-PC weather stations are equipped with temperature, pressure and relative humidity sensors [26]. For our currently used model they calculate the 500 hPa geopotential height from the hypsometric equation [28, 30]. Regarding GFS sources, in the current calculations the 0.5° resolution dataset was used.

2.2. Integrating DSN-PC and GFS data

As a first step we performed natural neighbor interpolation [23, 28] on the z_{500} values of the GFS grid points to calculate the z_{500} values for the 20×20 computational grid. For the interpolation the latitude(°) and longitude(°) coordinates of the grid points were converted to (x,y) coordinates based on polar stereographic map projection [3]:

$$r = \frac{\cos(\text{latitude})}{1 + \sin(\text{latitude})} \cdot 2a,$$

where $a = 4 \cdot 10^7 / 2\pi$ is the radius of the Earth (m). Then

$$x = r \cdot \sin(\text{longitude}),$$
$$y = -r \cdot \cos(\text{longitude}).$$

After the interpolation 5 grid points' initial values were replaced by DSN-PC measurements. For that purpose we installed our weather stations to geographical locations near those 5 grid points. Table 1 shows the latitude(°) and longitude(°) coordinates of the grid points and their respective DSN-PC stations.

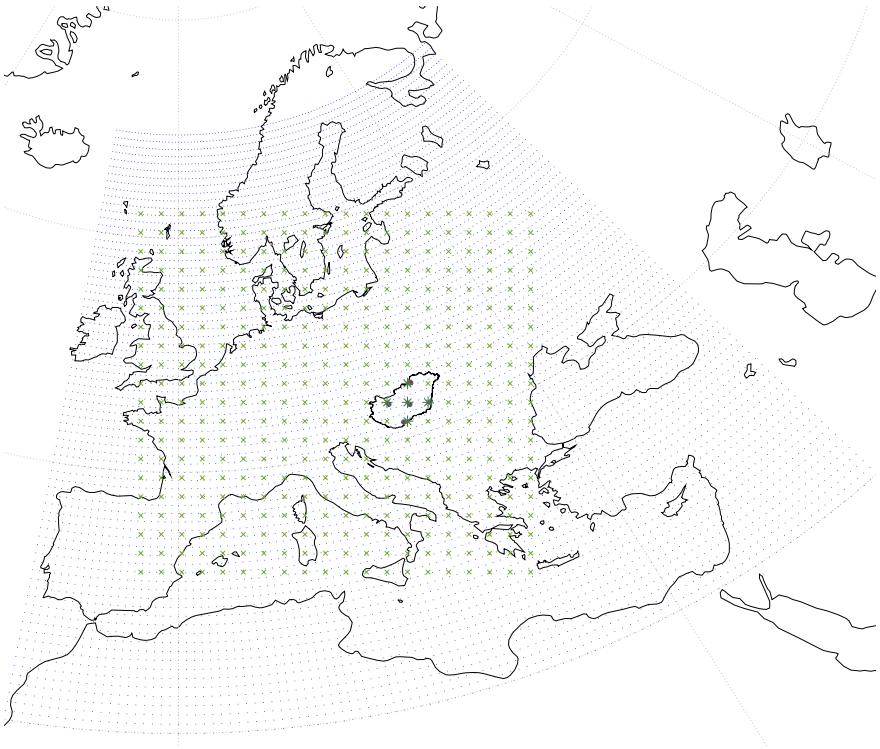


Figure 1: The regular grid of the 20×20 computational network (marked with +), the grid points of the NOAA GFS dataset (marked with ·), the geographical locations of our DSN-PC weather stations in Hungary (marked with o) and the 5 grid points whose data were replaced by the nearest DSN-PC stations' data (marked with *)

ID	lat (°N)	lon (°E)	grid point lat (°N)	grid point lon (°E)
1	48.17	20.42	48.18	20.14
2	46.92	19.67	47.08	19.55
3	46.65	21.29	46.67	21.15
4	47.31	18.01	47.46	17.91
5	46	18.68	45.98	18.99

Table 1: The geographic locations of our currently operational DSN-PC weather stations and their respective grid points

2.3. The distributed smoothing algorithm

Sometimes the DSN-PC measurements contain outlier values. The reason for these can be measurement errors or local weather phenomena. If we do not handle these errors, our simple model cannot handle the big differences between adjacent grid points and will produce incorrect results. To smooth out those spikes we applied a simple averaging algorithm which is executed by each node before starting the forecast algorithm. The nodes get the initial values from their adjacent nodes and calculate the average of those and their own values. The adjacency distance can vary between 1–3 hops, thence near neighbours are always queried and distant neighbors may be queried as well. On Figure 2 the algorithm's operation is visualized on a 20×20 grid. On Figure 3 the flowchart diagram of the algorithm is shown.

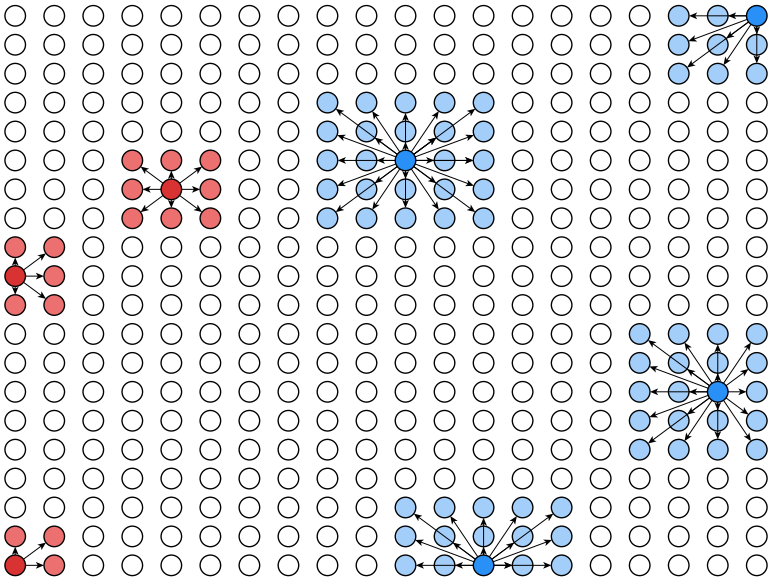


Figure 2: Examples of smoothing areas for different nodes with Np = 1 (red) and Np = 2 (blue)

2.4. The forecast algorithm

The applied distributed algorithm is based on the barotropic vorticity equation and was developed by Charney, Fjørtoft, and von Neumann (CFvN) [3]. Later, during our previous work we altered it so that it operates on a distributed sensor network. The details of how the algorithm operates on the network nodes were covered previously [25]. Like in the previous case [28] we ran calculations on data from the period between 2019.03.21. and 2019.03.27. For each day the measurements taken at 00:00 UTC were chosen as initial values. The Mean Absolute Error (MAE) was

calculated for each forecast by

$$\mathrm{MAE} = \frac{1}{18 \cdot 18} \sum_{i=1}^{18} \sum_{j=1}^{18} \left| z_{500,i,j} - z'_{500,i,j} \right| ,$$

where $z'_{500,i,j}$ is the predicted and $z_{500,i,j}$ is the measured 500 hPa geopotential 24 hours later. Due to the special way of handling the boundary grid points [3] we did not include them in the calculation of MAE.

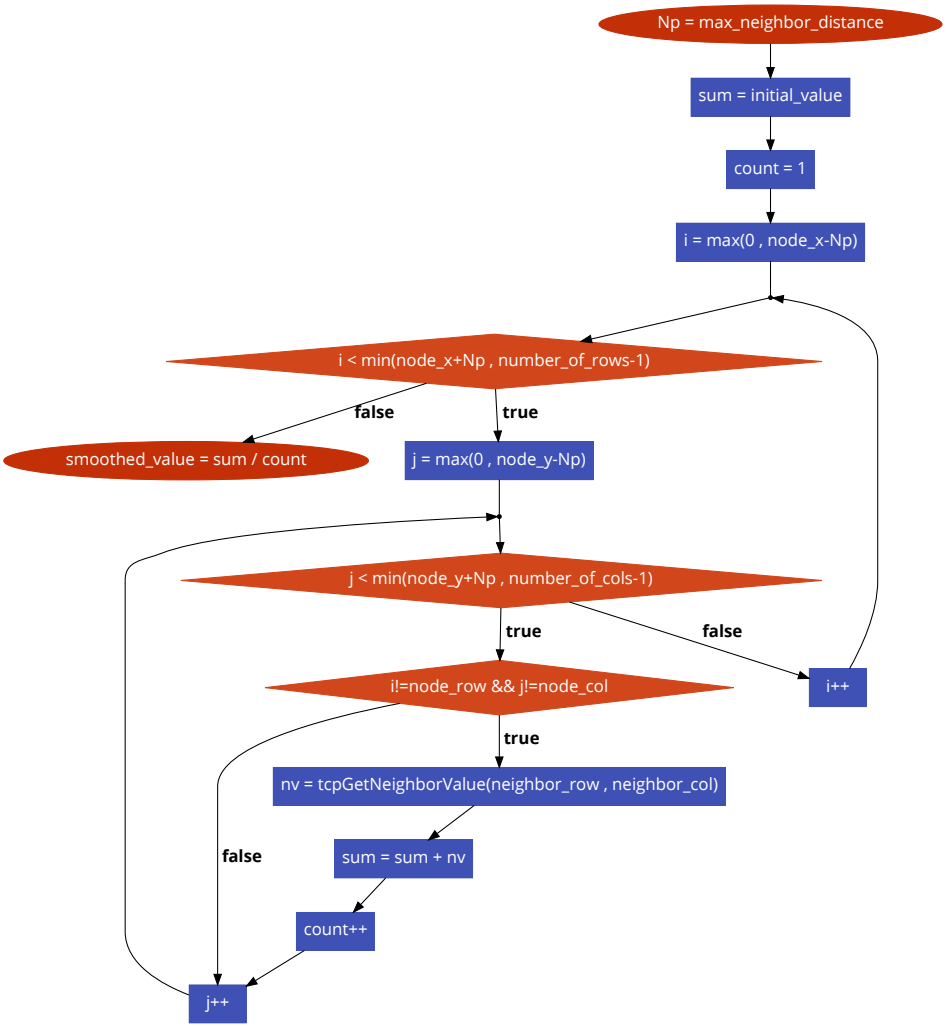
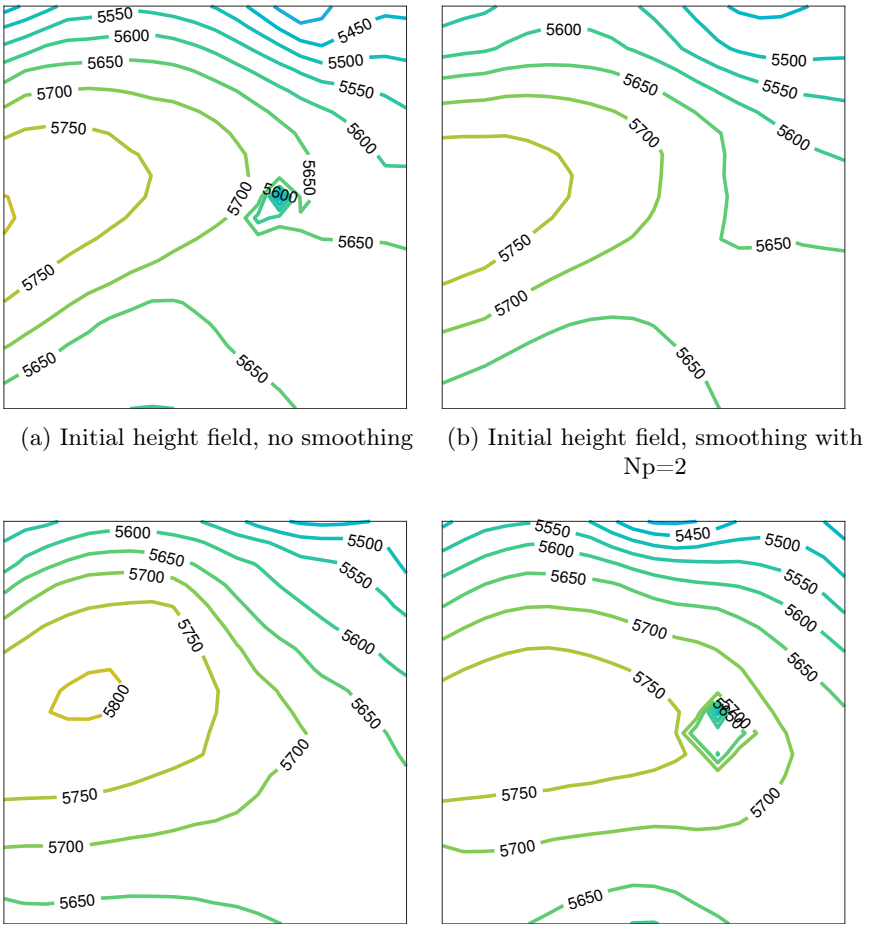


Figure 3: The smoothing algorithm as executed on one node

3. Results

The MAE values of the forecast calculations are summarized in Table 2 where the previous (simultaneous interpolation) results [28] are also included for comparison. The CFvN algorithm remained numerically unstable in cases where the initial data were unsmoothed and stable when smoothing was applied. The adjacency distance ($Np = 1, 2, 3$) value used during the smoothing phase had a significant impact on the goodness of the forecast. As a general rule, a minimum of $Np = 2$ seems to be necessary to achieve satisfactory results. On Figure 4 the initial, the analysis and the forecast height fields are shown for 2019.03.21. 00:00 UTC.



(a) Initial height field, no smoothing

(b) Initial height field, smoothing with Np=2

(c) Forecast height field

(d) Height field measured 24 hours later

Figure 4: Initial height fields and the result of the CFvN forecast performed on 2019.03.21. 00:00 UTC data

date	previous method		current method				
			no smoothing		smoothing		
					Np = 1	Np = 2	Np = 3
	CFvN	pers	CFvN	pers	CFvN	CFvN	CFvN
2019.03.21.	59.52	31.29	NaN	31.14	206.06	22.82	34.91
2019.03.22.	50.63	44.33	NaN	45.26	182.30	45.52	38.89
2019.03.23.	73.76	49.23	81.19	50.00	202.92	87.90	61.31
2019.03.24.	37.88	94.54	NaN	93.87	41.81	96.91	79.26
2019.03.25.	85.71	101.07	89.95	100.12	206.17	134.39	77.60
2019.03.26.	46.33	60.31	NaN	59.79	253.58	57.04	65.44
2019.03.27.	35.87	61.54	NaN	61.10	207.80	40.87	76.99

Table 2: Mean Absolute Error (m) values of the forecast calculations performed by the CFvN algorithm and the persistence method

Regarding performance, the smoothing algorithm's execution time strongly depends on the network conditions, but shouldn't take more than a few seconds. This, in our current hardware environment, is negligible compared to the forecast algorithm's typical execution time which is a few minutes.

4. Conclusion

We succeeded in integrating DSN-PC measurements and GFS datasets into a common dataset used as initial conditions for the CFvN weather forecast algorithm. It is a step forward from the simultaneous interpolation because of the direct integration of our data. As can be seen from the results, handling the outliers in the 2D grid is necessary while we use the CFvN algorithm. For that purpose, we successfully implemented a simple data smoothing algorithm on the virtual network nodes that can compute it by communicating with each other. This way we could achieve numerical stability in every investigated case. As a next step this smoothing method can be refined, especially on the boundaries. More complex smoothing methods can also be tested on the input data, although their conversion into distributed form can be challenging. However, the best way to improve our system would be the application of more advanced forecast models that can handle small-scale phenomena. Currently our system is not eligible to compete with more advanced and complex weather prediction models. Instead, our long-term goal is to make highly distributed weather prediction calculations possible.

Acknowledgements. We wish to thank Ficsor Endre, Perlaki Csaba, Szabó Sándor, Vas Ferenc and the Baptist Church of Kecskemét for providing place for our DSN-PC weather stations and thus supporting our research. We also thank SciTech Műszer Kft. for providing the possibility to use their facilities and for supporting

our work financially; and Gábor Nagy for his contribution in designing, manufacturing and testing our weather stations' electronic circuits.

References

[1] E. Andersson, J. N. Thépaut: *Assimilation of operational data*, in: Data Assimilation: Making Sense of Observations, 2010, ISBN: 9783540747024, DOI: 10.1007/978-3-540-74703-1_11.

[2] V. Baramidze, M. J. Lai, C. K. Shum: *Spherical Splines for Data Interpolation and Fitting*, SIAM Journal on Scientific Computing 28.1 (Jan. 2006), pp. 241–259, ISSN: 1064-8275, DOI: 10.1137/040620722, URL: http://epubs.siam.org/doi/10.1137/040620722.

[3] J. G. Charney, R. Fjørtoft, J. V. Neumann: *Numerical Integration of the Barotropic Vorticity Equation*, Tellus 2.4 (1950), pp. 237–254, ISSN: 0040-2826, DOI: 10.3402/tellusa.v2i4.8607, URL: https://www.tandfonline.com/doi/full/10.3402/tellusa.v2i4.8607.

[4] W. Freeden, W. Törnig: *On spherical spline interpolation and approximation*, Mathematical Methods in the Applied Sciences 3.1 (1981), pp. 551–575, ISSN: 01704214, DOI: 10.1002/mma.1670030139, URL: http://doi.wiley.com/10.1002/mma.1670030139.

[5] W. Freeden, M. Z. Nashed, M. Schreiner: *Spherical Harmonics Interpolatory Sampling*, in: 2018, pp. 267–300, DOI: 10.1007/978-3-319-71458-5_11, URL: http://link.springer.com/10.1007/978-3-319-71458-5%7B%5C_%7D11.

[6] W. Freeden, M. Schreiner: *Special Functions in Mathematical Geosciences: An Attempt at a Categorization*, in: Handbook of Geomathematics, Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 2455–2482, ISBN: 9783642545511, DOI: 10.1007/978-3-642-54551-1_31, URL: http://link.springer.com/10.1007/978-3-642-54551-1%7B%5C_%7D31.

[7] W. Freeden, W. Törnig: *Spline methods in geodetic approximation problems*, Mathematical Methods in the Applied Sciences 4.1 (1982), pp. 382–396, ISSN: 01704214, DOI: 10.1002/mma.1670040124, URL: http://doi.wiley.com/10.1002/mma.1670040124.

[8] *Global Forecast System (GFS) | National Centers for Environmental Information (NCEI) formerly known as National Climatic Data Center (NCDC)*, URL: https://www.ncdc.noaa.gov/data-access/model-data/model-datasets/global-forcast-system-gfs (visited on 05/20/2020).

[9] A. D. Hartkamp, K. de Beurs, A. Stein, J. White: *Interpolation Techniques for Climate Variables*, Geographic Information Systems Series 99-01. International Maize and Wheat Improvement Center (CIMMYT), Mexico 1999. ISSN: 1405-7484 (1999).

[10] M. F. Hutchinson: *Interpolating mean rainfall using thin plate smoothing splines*, International Journal of Geographical Information Systems (1995), ISSN: 02693798, DOI: 10.1080/02693799508902045.

[11] M. F. Hutchinson: *Interpolation of Rainfall Data with Thin Plate Smoothing Splines -Part II: Analysis of Topographic Dependence*, Journal of Geographic Information and Decision Analysis 2.2 (1998), pp. 152–167.

[12] M. F. Hutchinson: *Interpolation of rainfall data with thin plate smoothing splines. Part I: Two dimensional smoothing of data with short range correlation*, Journal of Geographic Information and Decision Analysis 2.2 (1998), pp. 139–151.

[13] C. H. Jarvis, N. Stuart: *A comparison among strategies for interpolating maximum and minimum daily air temperatures. Part II: Interaction between number of guiding variables and the type of interpolation method*, Journal of Applied Meteorology (2001), issn: 08948763, doi: 10.1175/1520-0450(2001)040<1075:ACASFI>2.0.CO;2.

[14] B. I. Justusson: *Median filtering: statistical properties.* Two-dimensional digital signal processing II. (1981), doi: 10.1007/bfb0057597.

[15] M. Kiani: *Template-based smoothing functions for data smoothing in Geodesy*, Geodesy and Geodynamics (Mar. 2020), issn: 16749847, doi: 10.1016/j.geog.2020.03.003, url: https://linkinghub.elsevier.com/retrieve/pii/S1674984720300197.

[16] W. Lahoz, B. Khattatov, R. Ménard: *Data assimilation and information*, in: Data Assimilation: Making Sense of Observations, 2010, isbn: 9783540747024, doi: 10.1007/978-3-540-74703-1_1.

[17] P. Lynch, X. Y. Huang: *Initialization*, in: Data Assimilation: Making Sense of Observations, 2010, isbn: 9783540747024, doi: 10.1007/978-3-540-74703-1_9.

[18] M. New, M. Hulme, P. Jones: *Representing Twentieth-Century Space–Time Climate Variability. Part I: Development of a 1961–90 Mean Monthly Terrestrial Climatology*, Journal of Climate 12.3 (Mar. 1999), pp. 829–856, issn: 0894-8755, doi: 10.1175/1520-0442(1999)012<0829:RTCSTC>2.0.CO;2, url: http://journals.ametsoc.org/doi/abs/10.1175/1520-0442%7B%5C%%7D281999%7B%5C%%7D29012%7B%5C%%7D3C0829%7B%5C%%7D3ARTCSTC%7B%5C%%7D3E2.0.CO%7B%5C%%7D3B2.

[19] M. New, D. Lister, M. Hulme, I. Makin: *A high-resolution data set of surface climate over global land areas*, Climate Research 21 (2002), pp. 1–25, issn: 0936-577X, doi: 10.3354/cr021001, url: http://www.int-res.com/abstracts/cr/v21/n1/p1-25/.

[20] L. L. S., G. Wahba: *Spline Models for Observational Data.* Mathematics of Computation 57.195 (July 1991), p. 444, issn: 00255718, doi: 10.2307/2938687, url: https://www.jstor.org/stable/2938687?origin=crossref.

[21] A. Savitzky, M. J. Golay: *Smoothing and Differentiation of Data by Simplified Least Squares Procedures*, Analytical Chemistry (1964), issn: 15206882, doi: 10.1021/ac60214a047.

[22] Z. Shang, G. Cheng: *Computational Limits of A Distributed Algorithm for Smoothing Spline*, Journal of Machine Learning Research 18.108 (2017), pp. 1–37, url: http://jmlr.org/papers/v18/16-289.html.

[23] R. Sibson: *A Brief Description of Natural Neighbour Interpolation*, in: Interpreting multivariate data, 1981, pp. 21–36, isbn: 0471280399.

[24] W. Van Assche: *W. Freeden, T. Gervens, and M. Schreiner, Constructive Approximation on the Sphere, with Application to Geomathematics*, Journal of Approximation Theory 112.2 (Oct. 2001), pp. 324–325, issn: 00219045, doi: 10.1006/jath.2001.3607, url: https://linkinghub.elsevier.com/retrieve/pii/S002190450193607X.

[25] Á. Vas, Á. Fazekas, G. Nagy, L. Tóth: *Distributed Sensor Network for meteorological observations and numerical weather Prediction Calculations*, Carpathian Journal of Electronic and Computer Engineering 6.1 (2013), pp. 56–63.

[26] Á. Vas, G. Nagy, L. Tóth: *Networkable Sensor Station for DSN-PC System*, Carpathian Journal of Electronic and Computer Engineering 8.2 (2015), pp. 37–40, url: http://cjece.ubm.ro/vol/8-2015/n2/1512.22-8208.pdf.

[27] Á. Vas, L. Tóth: *Evaluation of a simulated distributed sensor- and computational network for numerical prediction calculations*, in: Communications in Computer and Information Science, ed. by V. M. Vishnevskiy, D. V. Kozyrev, vol. 919, Cham: Springer International Publishing, 2018, pp. 9–20, isbn: 9783319994468,
doi: 10.1007/978-3-319-99447-5_2.

[28] Á. Vas, L. Tóth: *Investigation of a Hybrid Sensor- and Computational Network for Numerical Weather Prediction Calculations*, in: Communications in Computer and Information Science, ed. by V. M. Vishnevskiy, D. V. Kozyrev, vol. 1141, Cham: Springer International Publishing, 2019, pp. 510–523, isbn: 9783030366247,
doi: 10.1007/978-3-030-36625-4_41.

[29] G. Wahba: *Spline Interpolation and Smoothing on the Sphere*, SIAM Journal on Scientific and Statistical Computing 2.1 (Mar. 1981), pp. 5–16, issn: 0196-5204,
doi: 10.1137/0902002,
url: http://epubs.siam.org/doi/10.1137/0902002.

[30] J. M. Wallace, P. V. Hobbs: *Atmospheric Thermodynamics*, in: Atmospheric Science, 2006, pp. 63–111,
doi: 10.1016/b978-0-12-732951-2.50008-9.

[31] C. H. Woodford: *An algorithm for data smoothing using spline functions*, BIT 10.4 (Dec. 1970), pp. 501–510, issn: 00063835,
doi: 10.1007/BF01935569.

Annales Mathematicae et Informaticae
***51** (2020) pp. 89–99*
DOI: 10.33039/ami.2020.07.004
https://ami.uni-eszterhazy.hu

Performance evaluation of finite-source Cognitive Radio Networks with impatient customers

Mohamed Hedi Zaghouani, János Sztrik

Doctoral School of Informatics, University of Debrecen, Debrecen, Hungary
zaghoueni.hedi@gmail.com, sztrik.janos@inf.unideb.hu

Submitted: February 4, 2020
Accepted: July 1, 2020
Published online: July 23, 2020

Abstract

The current paper takes into consideration a cognitive radio network with impatient customers, by the help of finite-source retrial queueing system. We consider two different types of customers (Primary and Secondary) assigned to two interconnected frequency bands. A first frequency band with a priority queue and a second one with an orbit, both are respectively dedicated for the Primary Users (PUs) and Secondary Users (SUs). In case the servers are busy, both customers (Licensed and Unlicensed) join either the queue or the orbit. Before joining the orbit, secondary customers receive a random retrial time according to Exponential distribution, which is the holding time before the next retry. Unlicensed users (impatient) are obliged to leave the system once their total waiting time exceeds a given maximum waiting time.

The novelty of this work is the investigation of the abandonment and its impact on several performance measures of the system such as the mean response time and waiting time of users, probability of abandonment of SU, etc. Several figures illustrate the problem in question by the help of simulation.

Keywords: Retrial queuing systems, simulation, cognitive radio networks, performance and reliability measures, Impatient customers, Tandem queue, Abandonment.

1. Introduction

Cognitive Radio (CR) is a smart technique capable to get rid of the under-utilization spectrum issues, by allowing secondary customers to use opportunistically the primary channel without performing any interference to the communication of the primary customers to enhance the performance of the network. This intelligent technology is skilled to modify its transmitter parameters in compliance with the interaction of the environment in which it operates. The main objective of CRN is to exploit the unused sections of the primary frequency bands for the favour of unlicensed customers, more details can be found in [1, 6, 8, 24].

The expression cognitive radio was presented for the first time in 1999 by Mitola [13], explaining that this technology is conscious of the surrounding environment where performers and can adjust its parameters to improve customers performance. Several studies and researches, as [25, 28] reveal that often many parts of the channels are unused in time and space. These parts (white spaces) are not occupied by any licensed users. Secondary users in these parts of the service unit can detect this disuse and communicate freely with each other without performing any harmful consequences on the primary users. Nowadays, two types of Cognitive Radio Network exist. The first type is termed as (underlay network) in which unlicensed users might use the primary channels simultaneously with the licensed users, based on some predetermined conditions. The second type is called (overlay networks) where the unlicensed users are allowed at any given time to use the Primary Service Unit whilst it is unoccupied by licensed customers, more information was introduced by the authors of [18, 22, 26, 29] . However, the current paper deals with overlay CRN, by modelling a system that contains two finite-source subsystems(Primary and Secondary).

In this queuing system, we take into account two elements, a first subsystem is allocated for the jobs of Primary Users (PU), with a finite number of sources. In this subsystem, each source generates a primary call for the PUs after an exponentially distributed time, the latter requests are forwarded to a single server Primary Channel Service (PCS) with a preemptive discipline (FIFO queue) to start the service, supposing that the service time is exponentially distributed as well. The second component of the model is built for the requests of Secondary Users (SU), coming from a fi nite-source and heading to the Secondary Channel Service (SCS), knowing that the source and service times of the secondary users are exponentially distributed.

The generated primary tasks are targeting the PCS to check its accessibility. If this service unit is unoccupied, the service starts immediately. However, if the PCS is busy by another primary job, this last task joins a First In First Out (FIFO) queue. Nevertheless, if a secondary job is being treated in the primary unit, this job disconnects right away and should be routed back to the Secondary Channel Service.

Per the status of the secondary channel, the cancelled task either begins again the service on its original server (SCS) or joins the retrial queue (orbit). Besides,

the secondary channel will receive the low priority requests. If the aimed unit is idle, the service might start immediately, otherwise, these secondary requests will try to join the primary unit. If it is free the secondary requests have the opportunity to begin. If not, they will join automatically the orbit. From the orbit, the postponed requests retry to get the service after an exponentially distributed random interval, more details can be found in [6, 8, 18, 22, 24, 29].

In this study, we assume that impatient customers in the orbit who their total waiting times exceed a given maximum waiting time need to leave the system, which is the novelty of this work.

Several studies have investigated the CRN based on different scenarios. Taking [20] as an example the authors have used some methods of queuing theoretical on a finite source cognitive radio network with two service units (primary and secondary) in order to examine the main performances of this system by the help of tool-supported approach.

However, in a similar work [2] authors studied a single server network, which is subject to breakdowns and repairs. This kind of networks suffer from many difficulties while dealing with the requests, as the breakdown of the only server affects the whole network.

By the help of retrial queuing model and using Primary and Secondary service channels [14] supposed that both units are subject to random breakdowns and repairs. The authors used several distributions (Exponential, Hypo and Hyper Exponential) in order to show the impact of these distributions on the performance measures of the system.

As extended work, in [27] Gamma distribution was added to the above mentioned above distributions.

In papers [15] and [16] authors used as well Hypo, Hyper and Exponential distributions supposing that the secondary users of the system are subject to collisions and the two services in the system are unreliable, respectively.

However, after a deep dive in many similar investigations and studies, we figured out that none of them dealt with this model taking into consideration the abandonment phenomena.

Instead, many probes analysed abandonment in other types of network, endorsing that customers can leave systems from queues, server units, while getting services and while waiting, more details are found in [5, 9–11, 23]. However, in the present paper, we suppose that the impatient users (Secondary ones) are forced to leave the system from the orbit only while waiting.

Several figures will show the impact of the abandonment on the performance measures of the system, by the help of simulation.

2. System's operation model

Fig.1 demonstrates a finite source queuing system that models the considered cognitive radio network. Our queuing system consists of two not independent, interconnected sub-systems. A first part is allocated to primary requests, with $N1$ the

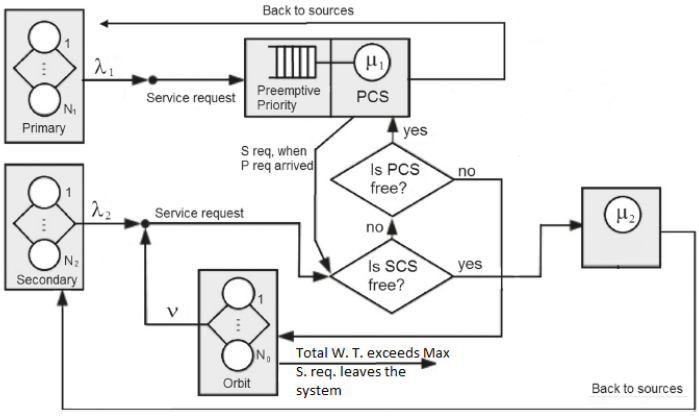


Figure 1: Finite-source retrial queuing system: Modeling the Cognitive Radio Network with impatient customers

number of sources. These sources will be responsible for generating a high priority requests with an inter-request time exponentially distributed, using a parameter λ_1. All the produced requests are directed to a single server unit (PCS) with a FIFO queue. The service times of the primary tasks are supposed to be exponentially distributed as well with rate μ_1.

The second subsystem is devoted for the low-priority requests. The number of sources is denoted by $N2$, the inter-arrival times and service times in this subsystem are assumed to be exponentially distributed with parameter λ_2 and μ_2, respectively.

Both servers can be in two states: idle or busy. Per the server's state, the generated primary packet goes to the primary server (if the server is idle) or joins the FIFO queue (if the service unit is busy with a PU). However, if the PCS is occupied by an unlicensed user, its service is instantly stopped and the interrupted secondary request is sent back to the Secondary unit.

Depending on the availability of the secondary unit, the aborted task is addressed either to the server or the retrial queue (orbit) and reties again its service from the beginning after an exponentially distributed time with parameter ν.

In the other hand, requests from SUs are directed to SCS. If it is idle, the service begins, if not, this unlicensed task will sens the PCS. In case of an idle status for PCS, this service may opportunistically join the high priority channel. If the PCS is engaged, the request goes to the orbit. It should be noted that Secondary Users in the orbit are obliged to leave the system once their total waiting time will exceed a specified maximum waiting time.

We introduce the following notations, to create a stochastic process describing the behaviour of the system:

- $k1(t)$: represents the number of licensed sources at given time t;
- $k2(t)$: refers to the number of unlicensed at time given t;

- $q(t)$: is the number of primary requests in the queue at certain time t;
- $o(t)$: denotes the number of tasks in the orbit at time t;
- $y(t) = 0$, if the primary channel is idle, $y(t) = 1$, if the primary channel is processing (busy) a high-priority request and $y(t) = 2$, if the primary service unit is processing (busy) a low-priority request at time t;
- $c(t) = 0$, if the secondary service unit is idle(free) and $c(t) = 1$, if the secondary service unit is busy at given time t.

As consequence we can see that:

$$k1(t) = \begin{cases} N1 - q(t), & \text{y(t)=0,2,} \\ N1 - q(t) - 1, & \text{y(t)=1.} \end{cases}$$

As a result we can see that:

$$k2(t) = \begin{cases} N2 - o(t) - c(t), & \text{y(t)=0,1,} \\ N2 - o(t) - c(t) - 1, & \text{y(t)=2.} \end{cases}$$

We assume that all the random variables used in the model construction are exponentially distributed, therefore we decided to use a stochastic simulation by the help of C coding language with GSL stochastic library.

All the numerical results of this were collected by the validation of the simulation outputs. Speaking of exponentially distributed inter-event time, we can construct continuous-time Markov chain and the main steady-state performance measures can be obtained, see for example [19]. The input parameters are displayed in Table 1.

Parameters	Value at moment t	Maximum Value
Primary sources	$k1(t)$	N1
Secondary Sources	$k2(t)$	N2
Primary arrival rate	λ_1	
Secondary arrival rate	λ_2	
Number of requests at the queue (FIFO)	$q(t)$	N1-1
Number of requests at the orbit	$o(t)$	N2-1
Primary service rate	μ_1	
Secondary service rate	μ_2	

Table 1: Parameters of the simulation

3. Simulation results

The batch-mean method was used in the simulation to estimate the mean response times of the requests. This method is a common confidence interval technique

which is applied for steady-state simulation output analysis. See for example [3, 4, 7, 12, 21].

Based on the values shown in Table 2 we could generate several figures supposing that the maximum waiting time of SU is constant, it should be noted as well, that in our simulation the Secondary Users were divided into two categories (Successful and Abandoned).

Figure No.	N1	N2	λ_1	λ_2	μ_1	μ_2	ν	$MaxW.T.$
Figure 2,3	7	12	0.5	0.2	2	1	10	x-axis
Figure 4	6	6	0.6	x-axis	4	4	0.4	10
Figure 5	8	10	0.5	0.2	1	0.5	20	x-axis

Table 2: Numerical values of model parameters

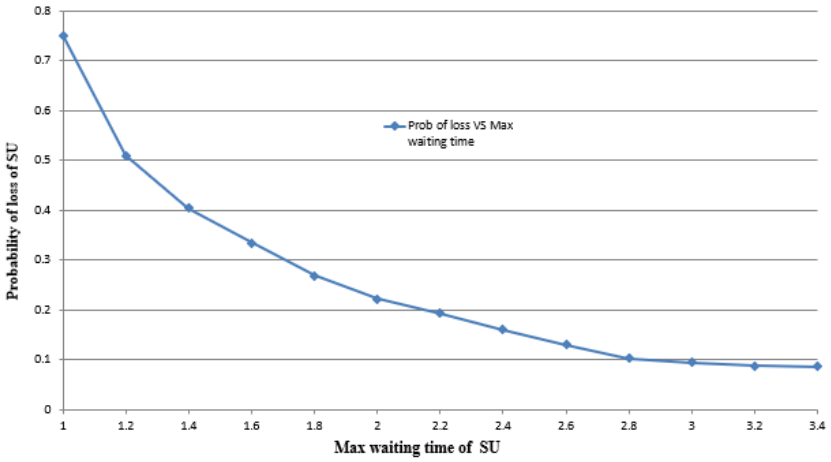


Figure 2: The effect of the Maximum waiting time of SU on the Probability of loss of SU

Figure 2 illustrates the impact of the maximum waiting time of SU on the probability of loss, as anticipated, by increasing the abandonment time, the probability of loss decreases as more secondary customers have the chance to get served without leaving the system, which makes the SCS busier.

Using the following formula we could generate Figure 3:

$$\overline{W_a} = P_{abon}.C + (1 - P_{abon})\overline{W_{succ}}$$

- $\overline{W_a}$: Mean waiting time of an arbitrary (patient or impatient) SU
- P_{abon}: Probability of abandonment
- C: Maximum waiting time of SU

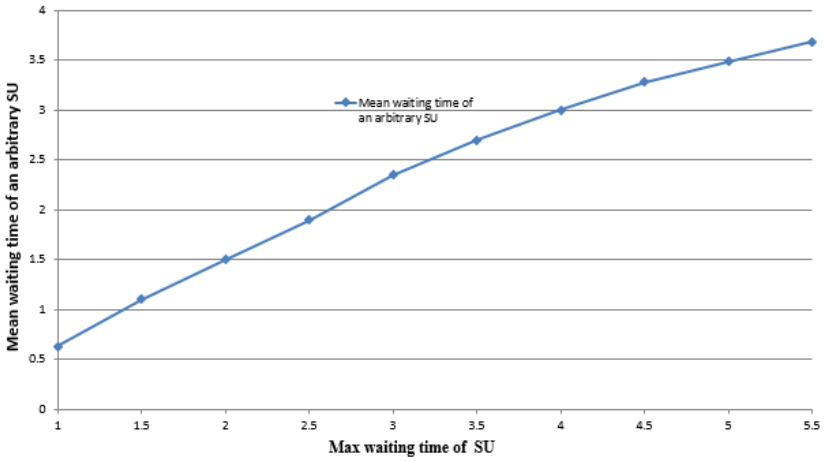


Figure 3: The effect of the Maximum waiting time of SU on the mean waiting time of an arbitrary SU

- $\overline{W_{succ}}$: Mean waiting time of successful Secondary user.

In Figure 3 the effect of abandonment time of SU on the mean waiting time of an arbitrary SU (Patient or Impatient) was displayed. This figure confirms the expectation that is increasing the maximum waiting time for SU involves higher waiting times for the two categories of unlicensed customers.

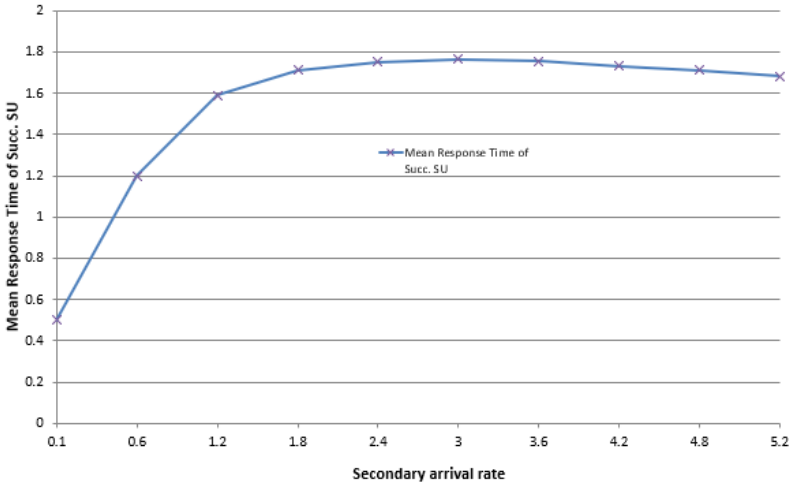


Figure 4: The effect of the secondary arrival rate on the mean response time of the successful Secondary Users

Figure 4 shows the effect of the request generation rate on the mean response time of the secondary users. The result presents the phenomenon of having a maximum value of the mean response time which was noticed in [17]. The abandonment of impatient SU from the orbit provides shorter response time for the patient users.

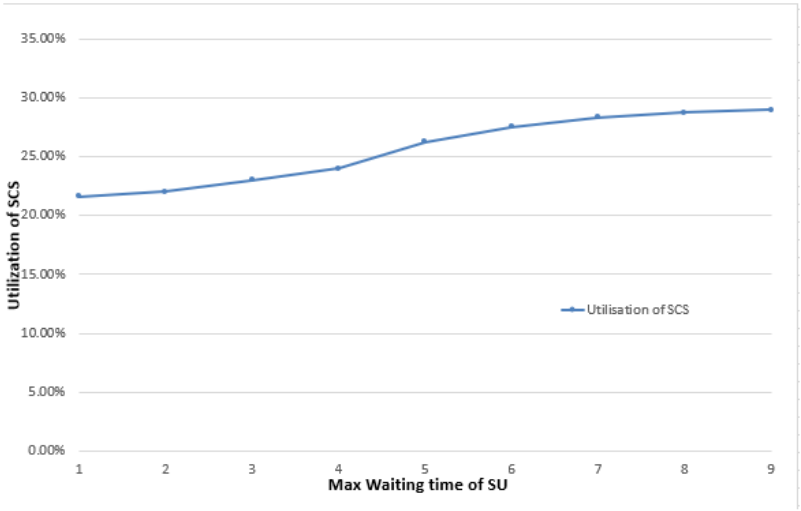


Figure 5: The effect of the Maximum waiting time of SU on the Utilization of SCS

The last Figure exhibits the effect of the abandonment time of the secondary users on the utilization of the secondary server. The innovation of abandonment contributes less utilization for the server when the maximum waiting time is too small, as a consequence, only the SU with a small amount of waiting time will benefit the service.

4. Conclusion

In this paper a finite-source retrial queuing model was introduced using two not independent, interconnected channels servicing licensed and unlicensed users in a cognitive radio network with abandonment from the orbit. Licensed users have preemptive priority over the unlicensed ones in servicing at the primary channel. However, at the secondary channel, an orbit was established for the secondary jobs finding the secondary service unit occupied upon arrival. SU may leave the system from the orbit, once their total waiting time reaches a given maximum. By the help of simulation, several sample examples were obtained, showing the effect of the abandonment on the different performance measures of the system.

Lastly, as future work, we will keep investigating the impact of the abandon-

ment on a such system assuming that the maximum waiting time is a generally distributed random variable.

Acknowledgements. The research work of János Sztrik and Mohamed Hedi Zaghouani is supported by the construction EFOP-3.6.3-VEKOP-16-2017-00002 and by the Stipendium Hungaricum Scholarship, respectively.

References

[1] I. F. Akyildiz, W.-Y. Lee, M. C. Vuran, S. Mohanty: *Next generation/dynamic spectrum access/cognitive radio wireless networks: A survey*, Computer networks 50.13 (2006), pp. 2127–2159,
doi: https://doi.org/10.1016/j.comnet.2006.05.001.

[2] B. Almási, T. Bérczes, A. Kuki, J. Sztrik, J. Wang: *Performance modeling of finite-source cognitive radio networks*, Acta Cybernetica 22.3 (2016), pp. 617–631,
doi: https://doi.org/10.14232/actacyb.22.3.2016.5.

[3] E. Carlstein et al.: *The use of subseries values for estimating the variance of a general statistic from a stationary sequence*, The annals of statistics 14.3 (1986), pp. 1171–1179,
doi: https://doi.org/10.1214/aos/1176350057.

[4] E. J. Chen, W. D. Kelton: *A procedure for generating batch-means confidence intervals for simulation: Checking independence and normality*, Simulation 83.10 (2007), pp. 683–694,
doi: https://doi.org/10.1177/0037549707086039.

[5] E. Danilyuk, O. Vygoskaya, S. Moiseeva: *Retrial queue M/M/N with impatient customer in the orbit*, in: International Conference on Distributed Computer and Communication Networks, Springer, 2018, pp. 493–504,
doi: https://doi.org/10.1007/978-3-319-99447-5_42.

[6] N. Devroye, M. Vu, V. Tarokh: *Cognitive radio networks*, IEEE Signal Processing Magazine 25.6 (2008), pp. 12–23,
doi: https://doi.org/10.1109/MSP.2008.929286.

[7] G. S. Fishman, L. S. Yarberry: *An implementation of the batch means method*, INFORMS Journal on Computing 9.3 (1997), pp. 296–310,
doi: https://doi.org/10.1287/ijoc.9.3.296.

[8] S. Gunawardena, W. Zhuang: *Modeling and Analysis of Voice and Data in Cognitive Radio Networks*, Springer, 2014,
doi: https://doi.org/10.1007/978-3-319-04645-7.

[9] Q.-M. He, H. Zhang, Q. Ye: *An M/PH/K queue with constant impatient time*, Mathematical Methods of Operations Research 87.1 (2018), pp. 139–168,
doi: https://doi.org/10.1007/s00186-017-0612-2.

[10] R. Ibrahim: *Managing queueing systems where capacity is random and customers are impatient*, Production and Operations Management 27.2 (2018), pp. 234–250,
doi: https://doi.org/10.1111/poms.12796.

[11] R. Kulshrestha et al.: *Channel allocation and ultra-reliable communication in CRNs with heterogeneous traffic and retrials: A dependability theory-based analysis*, Computer Communications (2020).

[12] A. M. Law, W. D. Kelton, W. D. Kelton: *Simulation modeling and analysis*, vol. 3, McGraw-Hill New York, 2000.

[13] J. Mitola, G. Q. Maguire: *Cognitive radio: making software radios more personal*, IEEE personal communications 6.4 (1999), pp. 13–18,
doi: https://doi.org/10.1109/98.788210.

[14] H. Nemouchi, J. Sztrik: *Performance Simulation of Finite-Source Cognitive Radio Networks with Servers Subjects to Breakdowns and Repairs*, Journal of Mathematical Sciences 237.5 (2019), pp. 702–711,
doi: https://doi.org/10.1007/s10958-019-04196-y.

[15] H. Nemouchi, J. Sztrik: *Performance evaluation of finite-source cognitive radio networks with collision using simulation*, in: 2017 8th IEEE International Conference on Cognitive Infocommunications (CogInfoCom), IEEE, 2017, pp. 000127–000131,
doi: https://doi.org/10.1109/CogInfoCom.2017.8268228.

[16] H. Nemouchi, J. Sztrik: *Performance evaluation of finite-source cognitive radio networks with non-reliable services using simulation*, in: Annales Mathematicae et Informaticae, vol. 49, Eszterházy Károly University Institute of Mathematics and Informatics, 2018, pp. 109–122,
doi: https://doi.org/10.33039/ami.2018.12.001.

[17] H. Nemouchi, J. Sztrik: *Performance Simulation of Non-reliable Servers in Finite-Source Cognitive Radio Networks with Collision*, in: International Conference on Information Technologies and Mathematical Modelling, Springer, 2017, pp. 194–203,
doi: https://doi.org/10.1007/978-3-319-68069-9_16.

[18] F. Paluncic, A. S. Alfa, B. T. Maharaj, H. M. Tsimba: *Queueing models for cognitive radio networks: A survey*, IEEE Access 6 (2018), pp. 50801–50823,
doi: https://doi.org/10.1109/ACCESS.2018.2867034.

[19] J. Sztrik: *On the finite-source G/M/r queue*, European Journal of Operational Research 20.2 (1985), pp. 261–268,
doi: https://doi.org/10.1016/0377-2217(85)90068-2.

[20] J. Sztrik, B. Almási, J. Roszik: *Heterogeneous finite-source retrial queues with server subject to breakdowns and repairs*, Journal of Mathematical Sciences 132.5 (2006), pp. 677–685,
doi: https://doi.org/10.1007/s10958-006-0014-0.

[21] H. C. Tijms: *A first course in stochastic models*, John Wiley and sons, 2003,
doi: https://doi.org/10.1002/047001363X.

[22] T. Van Do, N. H. Do, Á. Horváth, J. Wang: *Modelling opportunistic spectrum renting in mobile cellular networks*, Journal of Network and Computer Applications 52 (2015), pp. 129–138,
doi: https://doi.org/10.1016/j.jnca.2015.02.007.

[23] J. Wang, H. Abouee Mehrizi, O. Baron, O. Berman: *Staffing Tandem Queues with Impatient Customers–Application in Financial Service Operations*, Rotman School of Management Working Paper 3116815 (2018),
doi: https://doi.org/10.2139/ssrn.3116815.

[24] L. Wang, C. Wang, F. Adachi: *Load-Balancing Spectrum Decision for Cognitive Radio Networks*, IEEE Journal on Selected Areas in Communications 29.4 (2011), pp. 757–769,
doi: https://doi.org/10.1109/JSAC.2011.110408.

[25] T. A. Weiss, F. K. Jondral: *Spectrum pooling: an innovative strategy for the enhancement of spectrum efficiency*, IEEE communications Magazine 42.3 (2004), S8–14,
doi: https://doi.org/10.1109/MCOM.2004.1273768.

[26] E. W. Wong, C. H. Foh: *Analysis of cognitive radio spectrum access with finite user population*, IEEE Communications Letters 13.5 (2009), pp. 294–296,
doi: https://doi.org/10.1109/LCOMM.2009.082113.

[27] M. H. Zaghouani, J. Sztrik, A. Uka: *Simulation of the performance of Cognitive Radio Networks with unreliable servers*, in: Annales Mathematicae et Informaticae.

[28] S. A. Zekavat, X. Li: *User-central wireless system: ultimate dynamic channel allocation*, in: First IEEE International Symposium on New Frontiers in Dynamic Spectrum Access Networks, 2005. DySPAN 2005. IEEE, 2005, pp. 82–87.

[29] Y. Zhao, L. Bai: *Performance analysis and optimization for cognitive radio networks with classified secondary users and impatient packets*, Mobile Information Systems 2017 (2017), doi: https://doi.org/10.1155/2017/3613496.

Review papers

Annales Mathematicae et Informaticae
***51** (2020) pp. 103–111*
DOI: 10.33039/ami.2020.07.007
https://ami.uni-eszterhazy.hu

A comprehensive review on software comprehension models*

Anett Fekete, Zoltán Porkoláb

Eötvös Loránd University, Faculty of Informatics
{hutche,gsd}@inf.elte.hu

Submitted: February 4, 2020
Accepted: July 9, 2020
Published online: July 23, 2020

Abstract

Software comprehension is one of the most important among software development tasks since most developers do not start a brand new software every time they switch jobs or get transferred from one project to another but join long-running software projects. Every experienced and expert developer has their own established methods of understanding complex software systems. These methods might be different for everyone but they still have common aspects by which multiple well-defined code comprehension models can be constructed. Furthermore, the degree of understanding of a software can be categorized as well, according to the ability of the programmer to modify or develop a certain part of the software system. This paper is intended to provide a review of the cognitive software comprehension models established by extensive research in this topic as well as describe the dimensions of understanding software. It also determines the editor support of cognition models by examining common editor functionalities and categorizing code editors based on the availability of functionalities of each cognition approach.

Keywords: code comprehension, comprehension model, code cognition, taxonomy

MSC: 68N99

*This work was supported by the construction EFOP-3.6.3-VEKOP-16-2017-00002. The project was supported by the European Union, co-financed by the European Social Fund.

1. Introduction

Since the very inception of computer science, software comprehension has been an ongoing challenge for everyone that ever tried to understand any unfamiliar code. Each programmer, even the ones with the least experience possesses some kind of – either conscious or subconscious – way of understanding source code. Naturally, more experienced developers have more sophisticated methods and workflows, since they are more familiar with the language, framework and architecture they work with.

Throughout the history of computer science, several researchers tried to grasp how a programmer approaches software comprehension and constructed high-level abstraction models from the collected information. There have been several empirical experiments done in this area where researchers observed the process of understanding unfamiliar source code and tried to draw comprehension patterns from the results. The acquired information was used to define mental models that determine a certain direction of thinking while a developer is executing code comprehension tasks. As a result of decades of research, several complete cognition models were constructed which can be classified as being one of two determinant approaches, or a combination of them.

Cognition models are easily applicable in everyday software development. If we consider code editor functionalities, we can see that they can be categorized into one or both approaches according to their individual purpose. Based on this classification, we can also determine which approach is supported by code editors.

Sec. 2 provides an overview of the common elements of comprehension models, followed by a comprehensive review on comprehension approaches illustrated by the most well-designed mental models of each category. We classify the common software comprehension features in standalone comprehension tools and code editors in Sec. 3 by the categories discussed in Sec. 2 and decide about the most popular editors which category they fit in based on their available features. Sec. 4 concludes our paper.

2. Models of code comprehension

Although computer science is a relatively new discipline, it has always been a great interest of software programmers to find a decent way to understand program code that has been written by fellow developers. One of the most frequent activities of a programmer is comprehending the code others wrote from the very beginning of their careers. Several prestigious companies tried to measure exactly how much time is spent looking at or searching for code: IBM found in 1989 that more than 50% of working hours are consumed by static analysis [3]. A research conducted by Microsoft in 2007 [2] showed that 95% of developers thought code comprehension was an important part of their daily tasks while 65% said that they engage in software comprehension activities at least once a day.

Every programmer, no matter how little experience they have, has their own, conscious or subconscious way of getting familiar with unknown source code. Young programmers at the beginning of their career often try to understand code in an ad hoc way, e.g. jumping from one part of the code to another while running the program. On the other hand, experienced programmers usually don't just hop into the middle of a code but tries understanding in a more structured, thus more effective way.

The methods applied by different programmers provide us with information about the workflow of software comprehension, from which several different structures can be extracted. A structure constitutes a mental model which is an abstraction of the software comprehension process. Several comprehension models have been constructed since the beginning of software production and multiple excellent research papers tried to collect and classify them based on similar viewpoints. Comprehensive research was done by von Mayrhauser [19] who also constructed their own mental model (see Sec. 2.4). Another similar review was done by Storey [18]. The paper of O'Brien [12] presents a somewhat different review as he compares the models based on their data collection methods. They all determined two main directions for software comprehension: top-down (see Sec. 2.2) and bottom-up (see Sec. 2.3).

As noticed by Levy [10], top-down models serve the purpose of learning about architecture and system components first, then move onto finer details. On the other hand, the bottom-up approach is the exact opposite, intended to obtain knowledge about smaller code snippets of a feature. However, the two directions can be switched between in an opportunistic way, thus creating combined comprehension models.

2.1. Common elements in comprehension models

All research on the subject revealed that comprehension models have common elements of which the models' components are built up. Practically, the elements are telltale code snippets and conjectures about the programming goals, and activities that brings the developer closer to the complete mental model.

- Static elements include recognizable patterns and clues in the source code as well as domain knowledge and conjectures.
  - Beacons [20] are signs standing close to human thinking that may give a hint for the programmer about the purpose of the examined code, e.g. function or variable identifiers.
  - Chunks [4] are coherent code snippets that describe some level of abstraction in the program (e.g. an algorithm).
  - Hypotheses [9] are assumptions about the source code based on domain knowledge that are a result of applying various comprehension techniques. They are classified by Letovsky [9] according to whether they

are aimed at the purpose (*why*), implementation method (*how*) or type (*what*) of a source code detail like a function.

- Plans [17] include characteristic features of the source code that are so frequently used that they are easily recognizable.
  - Domain plans include high-level abstractions about the problem domain and its environment. It contains the mapping of real-world objects to programming objects (not necessarily meaning the objects of the object-oriented paradigm).
  - Programming plans describe typical practical concepts, e.g. data structures and their operations or significant details of algorithms.
- Rules of programming discourse are the consensus about coding that are intended to facilitate comprehension by not having to adapt to other programmers' coding habits like coding style. Rules may determine coding conventions or data structure and algorithm implementations.
- Text-structure knowledge [19] contains information about statements and commands in the source code and their relationships. It includes familiarity with control statement syntax and semantics.

• Dynamic elements are comprehension activities that bring the developer closer to the complete mental model.

- Strategies include methods in the comprehension process to move from low-level abstractions to high-level ones.
  - Chunking [4] is the process of producing higher level chunks from lower level chunks. After repeating the process multiple times, high-level abstractions can be built.
  - Cross-referencing connects different abstraction levels by mapping the elements of source code to level description elements. Cross-referencing is the key step in building a mental model of the existing abstractions.

2.2. Top-down approach

Generally speaking, when applying a model that belongs to the set of top-down approach models means that the programmer starts the comprehension process from "the big picture" and gradually moves on to the smaller details of the project. The first step is to acquire a comprehensive system overview e.g. by running the program and placing breakpoints through which the programmer can trace the running process and locate the significant parts.

The developer in this case is usually equipped with some previous domain knowledge. In the theory of Brooks [1], comprehension is built upon the domain knowledge by constituting an initial hypothesis about the source code. This is later refined into follow-up hypotheses that are either proved right or wrong.

When the developer comes across a familiar algorithm, the same algorithm should be easily understandable for them in a different programming language or framework. This serves as a base to the cognitive model of Soloway, Adelson and Ehrlich [17], who also focus on the hierarchical structure of programming plans and goals. The plans are also ordered in their own hierarchical upbuilding. They say that programmers also make use of beacons and rules of programming discourse during the comprehension process.

2.3. Bottom-up approach

The bottom-up approach is the opposite of the top-down approach, as in when applied, programmers first try to understand the details of the code, then move towards the larger units by chunking the code statements. Shneiderman and Mayer [16] present a theory that consists of two main knowledge areas: the language dependent syntactic knowledge and the semantic knowledge that, although independent of any particular programming language, relies heavily on general programming knowledge. The semantic knowledge is built up of hierarchically structured layers from low-level details to the actual, high-level mental model.

Pennington [14] describes a similar, two-component model in her paper. However, unlike the previous model, the components here are rather coordinative than completing each other like the syntactic and semantic knowledge. According to Pennington, a program model is built first in the programmer's mind by observing the control-flow of the program. Then, a situation model is built while refining the program model in parallel, which incorporates the programming goals.

2.4. Combined approaches

Some cognition models apply both the top-down and bottom-up approach in some form; either they have a component that opportunistically applies one direction (or switch between them if needed) or utilize elements of other models from both approaches in their own components. An example for the former case is Letovsky's [9] model. Beside the knowledge base and the internal representation, it consists of a third component, the assimilation process which follows the discursive human thinking as it tries to acquire the most knowledge possible in the shortest possible time. During the assimilation process, the developer soaks up as much information as possible with the help of the knowledge base and external representations of the code (like documentation).

Another similarly high-level combined mental model was described by von Mayrhauser et al. [19], called the integrated metamodel. Four major components build up this model, two of them borrowed from Pennington's bottom-up model [14] (the program and situation model) and one borrowed from the top-down model of Soloway, Adelson and Ehrlich [17] (the top-down model). These three components are supported by a knowledge base. Any of the components can be activated at any time during the comprehension process.

3. Tool support for understanding the comprehension process

Other than domain knowledge like language syntax or coding conventions, programmers are aided by various software features during code comprehension activities. There are standalone tools that were made for specifically this purpose such as CodeCompass [15], CodeSurveyor [6] or OpenGrok [13]. These software provide a wide range of textual and/or visual information about the source code. However, most modern integrated development environments (IDEs) and code editors are also rich in code comprehension supporting functionalities [11, 18]. These can be categorized according to the cognition approach they support, top-down, bottom-up, or even both, when a functionality serves multiple actions in the comprehension process.

3.1. Editor functionalities

- **Call hierarchy views** support the top-down approach since they offer a well-structured view of the program's control-flow.
- **Code browsing**: top-down comprehension is intuitively helped by searching for previously captured beacons in the software files. On the other hand, control-flow and data-flow is also supported by code browsing which are key elements of bottom-up comprehension.[1]
- **Find all references** is an obvious tool for bottom-up comprehension since it serves as a navigation tool when the developer tries to get a hint of the usage of a symbol thus helps in chunking. Control-flow is also supported by this feature.
- **Go to definition** supports top-down comprehension because its main purpose is to find the definition (source) of a beacon thus helps the programmer move from higher to lower abstraction level.
- **Intelligent code completion** supports top-down comprehension as it offers the possibility to capture beacons by providing intuitive perspective of the various classes, functions and variables of a program.
- **Split view** provides top-down perspective as it makes the developer able to grasp beacons from multiple files at the same time. This functionality also supports typical bottom-up elements like data-centric views.
- **UML diagrams** are in support of top-down comprehension as their purpose is to provide a high-level visualization of the code structure (e.g. class diagram, activity diagram) and the program domain (e.g. use-case diagram).

[1] By code browsing we mean high-level navigation across files, classes, symbols etc., not a simple text search.

Functionality	Top-down	Bottom-up
Call graphs	✓	✗
Code browsing	✓	✓
Find all references	✗	✓
Go to definition	✓	✗
Code completion	✓	✗
Split view	✓	✓
UML diagrams	✓	✗

Table 1: Classification of editor functionalities based on cognition approaches

3.2. Editors and comprehension models

As Table 1 shows, most functionalities primarily support top-down comprehension. If we investigate the available tools in the most popular IDEs [11], we can determine which cognition approach is supported by a certain IDE.

	Top-down				Bottom-up	Both	
Editor	Call graphs	Go to definition	Code completion	UML diagrams	Find all references	Code browsing	Split view
Atom	✗	✓	✓	✗	✓	✗	✓
Eclipse	✓	✓	✓	✓	✓	✓	✓
JetBrains	✓	✓	✓	✓	✓	✓	✓
NetBeans	✓	✓	✓	✗	✓	✓	✓
Notepad++	✗	✗	✓	✗	✗	✗	✓
Sublime Text	✗	✓	✓	✗	✓	✓	✓
vim	✗	✓	✓	✗	✓	✓	✓
Visual Studio Code	✓	✓	✓	✗	✓	✓	✓

Table 2: Editor support of cognition approaches

Table 2 shows that the most popular IDEs and code editors support every software comprehension approach in general by providing the previously classified functionalities. All-purpose IDEs support most if not all examined features. Open-source IDEs like Eclipse and Visual Studio code are further extendable by software comprehension supporting plugins [5, 11]. It is also worth noticing that IDEs generally perform poorly regarding visual features such as call graphs or diagrams.

4. Conclusion

In this paper, we gave a comprehensive review on the various types of code comprehension models and their influence on software editors. We discussed their common elements, and categorized the cognition models based on their components and the direction of their workflow. We investigated several widespread code editor features and classified them considering whether they support a comprehension approach or not. This investigation allowed us to determine which of the most popular IDEs support the described approaches based on the availability of functionalities. We determined that the majority of IDEs support all approaches.

It is worth considering that well-defined cognition models seem to be overly strict regarding the human thinking process. Multiple research has shown that developers do not usually follow a rigorous pattern or a concrete model during their understanding activities, they rather apply opportunistic strategies [7]. For example, Koenemann et al. [8] concluded that programmers perform best in comprehension tasks when they try to understand only the relevant parts of the code in an as-needed manner. This means that cognition models work well as abstractions about the comprehension process and are also provide a good basis for IDE functionalities but they shouldn't be considered the only ways of 'correct' comprehension workflows.

References

[1] R. Brooks: *Towards a theory of the cognitive processes in computer programming*, International Journal of Man-Machine Studies 9.6 (1977), pp. 737–751,
doi: https://doi.org/10.1016/S0020-7373(77)80039-4.

[2] M. Cherubini, G. Venolia, R. DeLine, A. J. Ko: *Let's go to the whiteboard: how and why software developers use drawings*, in: Proceedings of the SIGCHI conference on Human factors in computing systems, 2007, pp. 557–566,
doi: https://doi.org/10.1145/1240624.1240714.

[3] T. A. Corbi: *Program understanding: Challenge for the 1990s*, IBM Systems Journal 28.2 (1989), pp. 294–306,
doi: https://doi.org/10.1147/sj.282.0294.

[4] J. S. Davis: *Chunks: A basis for complexity measurement*, Information Processing & Management 20.1-2 (1984), pp. 119–127,
doi: https://doi.org/10.1016/0306-4573(84)90043-8.

[5] L. Hattori, M. D'Ambros, M. Lanza, M. Lungu: *Software evolution comprehension: Replay to the rescue*, in: 2011 IEEE 19th International Conference on Program Comprehension, IEEE, 2011, pp. 161–170,
doi: https://doi.org/10.1109/ICPC.2011.39.

[6] N. Hawes, S. Marshall, C. Anslow: *Codesurveyor: Mapping large-scale software to aid in code comprehension*, in: 2015 IEEE 3rd Working Conference on Software Visualization (VISSOFT), IEEE, 2015, pp. 96–105,
doi: https://doi.org/10.1109/VISSOFT.2015.7332419.

[7] A. J. Ko, B. Uttl: *Individual differences in program comprehension strategies in unfamiliar programming systems*, in: 11th IEEE International Workshop on Program Comprehension, 2003. IEEE, 2003, pp. 175–184,
doi: https://doi.org/10.1109/WPC.2003.1199201.

[8] J. Koenemann, S. P. Robertson: *Expert problem solving strategies for program comprehension*, in: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, ACM, 1991, pp. 125–130,
doi: https://doi.org/10.1145/108844.108863.

[9] S. Letovsky: *Cognitive processes in program comprehension*, Journal of Systems and software 7.4 (1987), pp. 325–339,
doi: https://doi.org/10.1016/0164-1212(87)90032-X.

[10] O. Levy, D. G. Feitelson: *Understanding large-scale software: a hierarchical view*, in: Proceedings of the 27th International Conference on Program Comprehension, IEEE Press, 2019, pp. 283–293,
doi: https://doi.org/10.1109/ICPC.2019.00047.

[11] M. Mészáros, M. Cserép, A. Fekete: *Delivering comprehension features into source code editors through LSP*, in: 2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), IEEE, 2019, pp. 1581–1586,
doi: https://doi.org/10.23919/MIPRO.2019.8756695.

[12] M. P. O'brien: *Software comprehension–a review & research direction*, Department of Computer Science & Information Systems University of Limerick, Ireland, Technical Report (2003).

[13] Oracle: *OpenGrok: "A wicked fast source browser"*,
url: https://oracle.github.io/opengrok/.

[14] N. Pennington: *Stimulus structures and mental representations in expert comprehension of computer programs*, Cognitive psychology 19.3 (1987), pp. 295–341,
doi: https://doi.org/10.1016/0010-0285(87)90007-7.

[15] Z. Porkoláb, T. Brunner, D. Krupp, M. Csordás: *Codecompass: an open software comprehension framework for industrial usage*, in: Proceedings of the 26th Conference on Program Comprehension, 2018, pp. 361–369,
doi: https://doi.org/10.1145/3196321.3197546.

[16] B. Shneiderman, R. Mayer: *Syntactic/semantic interactions in programmer behavior: A model and experimental results*, International Journal of Computer & Information Sciences 8.3 (1979), pp. 219–238,
doi: https://doi.org/10.1007/BF00977789.

[17] E. Soloway, K. Ehrlich: *Empirical studies of programming knowledge*, IEEE Transactions on software engineering 5 (1984), pp. 595–609,
doi: https://doi.org/10.1109/TSE.1984.5010283.

[18] M.-A. Storey: *Theories, methods and tools in program comprehension: Past, present and future*, in: 13th International Workshop on Program Comprehension (IWPC'05), IEEE, 2005, pp. 181–191,
doi: https://doi.org/10.1109/WPC.2005.38.

[19] A. Von Mayrhauser, A. M. Vans: *Program comprehension during software maintenance and evolution*, Computer 28.8 (1995), pp. 44–55,
doi: https://doi.org/10.1109/2.402076.

[20] S. Wiedenbeck: *Beacons in computer program comprehension*, International Journal of Man-Machine Studies 25.6 (1986), pp. 697–709,
doi: https://doi.org/10.1016/S0020-7373(86)80083-9.

Annales Mathematicae et Informaticae
***51** (2020) pp. 113–121*
DOI: 10.33039/ami.2020.07.003
https://ami.uni-eszterhazy.hu

Loop optimizations in C and C++ compilers: an overview

Réka Kovács, Zoltán Porkoláb

Eötvös Loránd University
rekanikolett@caesar.elte.hu
gsd@inf.elte.hu

Submitted: February 4, 2020
Accepted: July 1, 2020
Published online: July 23, 2020

Abstract

The evolution of computer hardware in the past decades has truly been remarkable. From scalar instruction execution through superscalar and vector to parallel, processors are able to reach astonishing speeds – if programmed accordingly. Now, writing programs that take all the hardware details into consideration for the sake of efficiency is extremely difficult and error-prone. Therefore we increasingly rely on compilers to do the heavy-lifting for us.

A significant part of optimizations done by compilers are loop optimizations. Loops are inherently expensive parts of a program in terms of run time, and it is important that they exploit superscalar and vector instructions. In this paper, we give an overview of the scientific literature on loop optimization technology, and summarize the status of current implementations in the most widely used C and C++ compilers in the industry.

Keywords: loops, optimization, compilers, C, C++

MSC: 68N20 Compilers and interpreters

1. Introduction

The Illiac IV, completed in 1966, was the first massively parallel supercomputer [13]. It marked the first milestone in a decades-long period that would see computing machines become unbelievably fast and increasingly complex. To harness the capabilities of such ever more parallel systems, researchers started writing tools

that could transform sequential programs (typically written in Fortran for scientific applications) to their own parallel equivalents.

By the mid 70's, a group of University of Illinois researchers led by David Kuck developed the Parafrase [18] tool, which pioneered the most influential ideas on automatic program transformations including dependence testing [19] and the first loop transformations [32].

In the late 70's, researchers led by Ken Kennedy at Rice University started the Parallel Fortran Compiler (PFC) [2] project. The authors' initial goal was to extend Parafrase, but they ended up implementing a completely new system, furthering the theory of data dependence [4] and inventing effective algorithms for a number of transformations including vector code generation, the handling of conditional statements [3], loop interchange, and other new transformations [5].

In this paper, we take a step aside, and instead of discussing the optimization of Fortran programs, for which most of the classical algorithms have been invented, we take a look at how C and C++ compiler writers are coping with the challenge. C family languages are very similar to Fortran, but notorious for the lack of constraints imposed on the programmer, which makes their analysis and optimization undoubtedly more difficult.

The structure of this paper is as follows. Section 2 describes the challenges C and C++ compiler developers face in contrast to classic Fortran optimization, and lists some of the strategies used to mitigate these issues. In Section 3, we give a status report of loop optimizations in the two open-source compilers most heavily used in the industry. Finally, in Section 4, we survey the latest research papers in the field of loop optimizations.

2. Adapting classic algorithms for C/C++

2.1. Challenges in optimizing C/C++

In their 1988 paper, [6] Allen and Johnson pointed out a number of considerations that make the vectorization and parallelization of C code difficult, as opposed to Fortran, the language that inspired most of the classic loop transformations in the literature. These concerns pose a challenge in optimizing programs written in C and C++ to the present day:

- **Pointers instead of subscripts.** C/C++ programs often address memory using pointer variables rather than arrays and explicit subscripts. Because of this, it is extremely difficult to decide whether two statements refer to the same section of memory or not.
- **Sequential operators.** Conditional operators and operators with side effects (e.g. ++) are inherently sequential. Vectorization of such operations require them to be either transformed or removed.
- **Loops.** The *for* statement used in C family languages is less restricted than the *DO* loop of Fortran, for which most of the classic loop transformations

were developed. This makes its vectorization considerably more difficult. The loop can contain operations that almost arbitrarily change the loop variable, and the loop body can contain branching statements.

- **Small functions.** Function calls can hide information that is necessary for optimization. Modern compilers often run optimizations together with inlining in an iterative fashion, trying to regain some information lost to the fine modularity encouraged in C and C++.

- **Argument aliasing.** Unline Fortran, function parameters in C and C++ are allowed to point into the same section of memory. Aliasing prohibits vectorization, but can only be checked at run-time, resulting in a high run-time cost.

- **Volatile.** In C, *volatile* variables represent values that may change outside of the context of the program, even if the change cannot be "seen" from the source code. Obviously, such language constructs are very difficult to optimize.

- **Address-of operator.** The & operator allows the programmer to take the address of any variable and modify it. This greatly increases the analysis needed for optimization.

2.2. Compiler strategies for C/C++ optimization

Inlining. The problems listed in the previous section are relatively hard to handle in compilers. Suprisingly, a large portion of the problems can be removed by the judicious inlining of function calls. Some of the benefits of inlining:

- If the body of a function call is available in the caller function, the compiler no longer has to calculate its effects conservatively, assuming the worst case. It can use the actual function body, making the analysis more precise, and allowing more optimizations to happen.

- Some of the argument aliasing problems disappear when the origins of arrays become visible.

- Function calls are inherently sequential operations. Their removal helps vectorization in its own right.

A high-level IR. Low-level intermediate representations had long been the norm for C [16] and Fortran [29] compilers before their vectorizing versions began gaining ground. Lowering the code too early can introduce unnecessary complexity in the analyses that precede optimizations. For example, the ability to analyze loops and subscripts is crucial for loop optimizations, and breaking down the loop into `gotos` and pointers would make it considerably more difficult. Other information such as the `volatile` modifier also get lost or obfuscated after the lowering phase.

Loop conversion. The C `for` loop is a fairly unconstrained language construct: the increment and termination conditions can have side effects, bounds and strides can change during execution, and control can enter and leave the loop body. Because of this, most C compilers perform a *doloop conversion*, when they attempt to transform the unconstrained `for` loop into a more regular `DO`-like form. It often makes sense to do the high-level transformations on this representation as well.

3. Loop optimization in modern compilers

The most widely used C and C++ compilers include both open-source (GCC, LLVM) and commercial (Microsoft Visual C++, IBM XL, Intel C Compiler) products. Unfortunately, there is limited information available for closed-source application, thus in this section we decided to review the status of loop transformations in GCC and LLVM. Both of these compilers are heavily used in the industry, and have a populous base of active contributors.

3.1. LLVM

The LLVM optimization pipeline [24] consists of 3 stages. The *Canonicalization* phase removes and simplifies the IR as much as possible by running scalar optimizations. The second part is called the *Inliner cycle*, as it runs a set of scalar simplification passes, a set of simple loop optimization passes, and the inliner itself iteratively. The primary goal of this part is to simplify the IR, through a cycle of exposing and exploiting simplification opportunities. After the Inliner cycle, the *Target Specialization* phase is run, which deliberately increases code complexity in order to take advantage of target-specific features as make the code as fast as possible on the target.

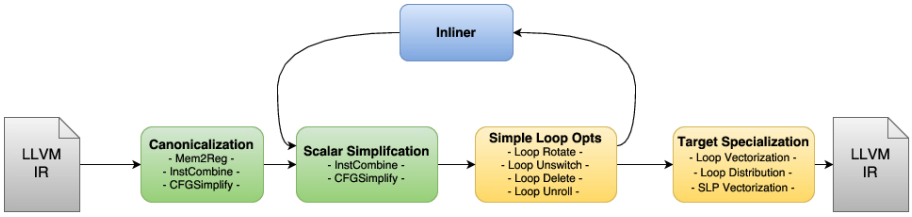


Figure 1: The LLVM optimization pipeline

LLVM supports various *pragmas* that allow users to guide the optimization process, which saves them the trouble of performing the optimizations by hand. Alternatively, they can rely on the *heuristics* in the compiler that strive to achieve similar performance gains.

The optimization infrastructure is modular, passes can be switched on and off on demand [17]. The currently available loop transformation passes are the following: loop unrolling, loop unswitching, loop interchange, detection of `memcpy` and `memset`

idioms, deletion side-effect-free loops, loop distribution, and loop vectorization. Members of the open-source community are working on adding loop fusion to the list [7].

As part of the modular structure of the optimizer, a common infrastructure is available to optimizations in the form of certain passes that perform analyses (`LoopInfo`, `ScalarEvolution`) and normalizing transformations (`LoopRotate`, `LoopSimplify`, `IndVarSimplify`) on the loops.

Many of the mentioned loop transformations are disabled by default, as they are either experimental in nature or not mature enough to be used by the wide public. Such transformations are e.g. `LoopInterchange` and `LoopUnrollAndJam`.

The order of the loop optimizations is fixed within the pipeline. This may result in conflicts or less profitable sequences of transformations. Additionally, because scalar and loop passes are run in cycles, they often interfere with each other by destroying canonical structures and invalidating analysis results.

A recent proposal plans to switch to a single integrated `LoopOptimizationPass` that would not interact with scalar optimizations, making it simpler. Similarly, the introduction of a loop tree intermediate representation could make loop modifications easier and might also help the profitability analysis. This idea is inspired by red-green trees [23] used in the Roslyn C# compiler.

3.2. GCC

As the early days of GCC date back to the 80's, its old monolithic structure made it hard to keep it aligned with the forefront of optimization research for a long time. However, its loop optimizer was almost completely re-written in the early 2000's [11]. Its new modular structure is similar to that of LLVM's, starting with an *initialization* pass, followed by several *optimization* passes, and ended with a trivial *finalization* phase that de-allocates any data structures used.

During initialization, the optimizer runs the induction variable, scalar evolution, and data dependence analyses [8] to gather necessary information about the loop, and performs preliminary transformations that simplify and canonicalize it. The optimization passes include loop unswitching, loop distribution, and two types of auto-vectorization [22]. The middle block of `pass_graphite` transformations refers to the polyhedral framework GRAPHITE that comes with GCC, but is unfortunately turned off by default, due to a lack of resources for maintenance.

In spite of its long history, GCC was still not very good at optimizing loop nests in 2017 [9]. This was mainly caused by

- a lack of traditional loop nest transformations, including loop interchange, unroll-and-jam, loop fusion and scalar expansion,
- transformations in need of a revision, and possibly
- a suboptimal arrangement of passes.

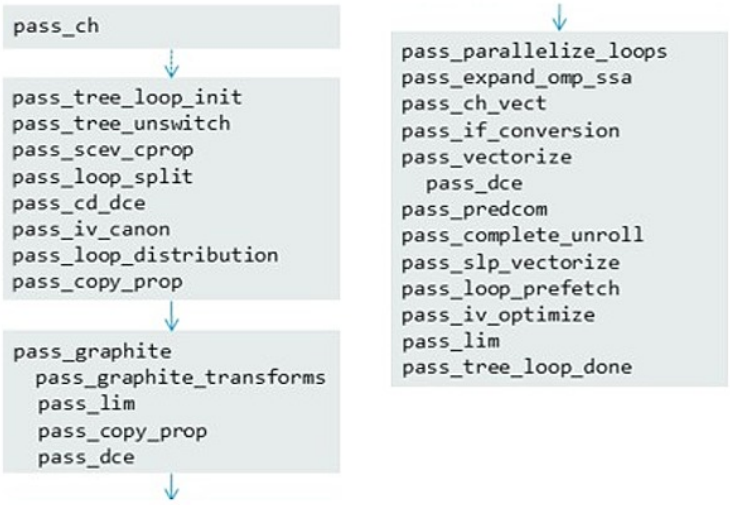


Figure 2: The GCC optimization pipeline

Since 2017, members of the community have started adding some of the missing transformations to the compiler, e.g. loop interchange [14], but others e.g. loop fusion and scalar expansion are still future work.

Similarly to LLVM, the latest proposal to the pass arrangement problem is a single loop transformation pass with a unified cost model.

4. Current trends in optimization research

Transformation ordering. One of the main research directions in the past few years concerns the choice of loop transformations and their ordering. With ever more complex machines, the performance gap between hand-tuned and compiler-generated code is getting wider. [31] presents a system and language named Locus that uses empirical search to automatically generate valid transformation sequences and then return the list of steps to the best variant. The source code needs to be annotated. [33] gives a template of scheduling algorithms with configurable constraints and objectives for the optimization process. The template considers multiple levels of parallelism and memory hierarchies and models both temporal and spatial effects. [10] describes a similar loop transformation scheduling approach using dataflow graphs. [30] recognizes that some combinations of loop optimizations can create memory access patterns that interfere with hardware prefetching. They give an algorithm to decide whether a loop nest should be optimized mainly for temporal or mainly for spatial locality, taking hardware prefetching into account.

Straight-line code vectorization. The past few years saw significant new developments in the field of straight-line code vectorization. The original Superword-Level Parallelism algorithm (SLP) [20] was designed for contiguous memory access patterns that can be packed greedily into vector instructions, without expensive data reordering movements. Throttled SLP [26] attempts to identify statements harmful to vectorization and stop the process earlier if that leads to better results. SuperGraph SLP [25] operates on larger code regions and is able to vectorize previously unreachable code. Look-ahead SLP [28] extends SLP to commutative operations, and is implemented in both LLVM and GCC. The latest development, SuperNode SLP [27] enables straight-line vectorization for expressions involving a commutative operator and its inverse.

Improving individual transformations. Other research efforts target the improvement of individual optimizations. [21] gives an algorithm to locate where to perform code duplication in order to enable optimizations that are limited by control flow merges. [1] describes a software prefetching algorithm for indirect memory accesses. [12] shows how to discover scalar reduction patterns and how it was implemented in an LLVM pass. [15] created a framework to enable collaboration between different kinds of dependency analyses.

5. Conclusion

In the age when hardware evolution makes machines ever more complex, compiler optimizations become ever more important, even for the simplest applications. This paper gave a short history of parallel hardware and compiler optimizations, followed by a discussion of hardships that the C and C++ languages pose to compiler writers. We gave a status report on the loop optimizing capabilities of the most popular open-source compilers for these languages, GCC and LLVM. In the end, we reviewed the latest research directions in the field of loop optimization research.

Acknowledgements. The publication of this paper is supported by the European Union, co-financed by the European Social Fund (EFOP-3.6.3-VEKOP-16-2017-00002).

References

[1] S. Ainsworth, T. M. Jones: *Software prefetching for indirect memory accesses*, in: 2017 IEEE/ACM International Symposium on Code Generation and Optimization (CGO), IEEE, 2017, pp. 305–317,
doi: https://doi.org/10.1145/3319393.

[2] J. R. Allen, K. Kennedy: *PFC: A program to convert Fortran to parallel form*, tech. rep., 1982.

[3] J. R. Allen, K. Kennedy, C. Porterfield, J. Warren: *Conversion of control dependence to data dependence*, in: Proceedings of the 10th ACM SIGACT-SIGPLAN symposium on Principles of programming languages, 1983, pp. 177–189, doi: https://doi.org/10.1145/567067.567085.

[4] J. R. Allen: *Dependence Analysis for Subscripted Variables and Its Application to Program Transformations*, AAI8314916, PhD thesis, USA, 1983, doi: https://doi.org/10.5555/910630.

[5] R. Allen: *K. Kennedy*, Automatic translation of FORTRAN programs to vector form. A CM Transactzons on Programming Languages and Systems 9.2 (1987), pp. 491–542, doi: https://doi.org/10.1145/29873.29875.

[6] R. Allen, S. Johnson: *Compiling C for vectorization, parallelization, and inline expansion*, ACM SIGPLAN Notices 23.7 (1988), pp. 241–249.

[7] K. Barton: *Loop Fusion, Loop Distribution and their Place in the Loop Optimization Pipeline*, LLVM Developers' Meeting, 2019, url: https://www.youtube.com/watch?v=-JQr9aNagQo.

[8] D. Berlin, D. Edelsohn, S. Pop: *High-level loop optimizations for GCC*, in: Proceedings of the 2004 GCC Developers Summit, Citeseer, 2004, pp. 37–54.

[9] B. Cheng: *Revisit the loop optimization infrastructure in GCC*, GNU Tools Cauldron, 2017, url: https://slideslive.com/38902330/revisit-the-loop-optimization-infrastructure-in-gcc.

[10] E. C. Davis, M. M. Strout, C. Olschanowsky: *Transforming loop chains via macro dataflow graphs*, in: Proceedings of the 2018 International Symposium on Code Generation and Optimization, 2018, pp. 265–277, doi: https://doi.org/10.1145/3168832.

[11] Z. Dvořák: *A New Loop Optimizer for GCC*, in: GCC Developers Summit, Citeseer, 2003, p. 43.

[12] P. Ginsbach, M. F. O'Boyle: *Discovery and exploitation of general reductions: a constraint based approach*, in: 2017 IEEE/ACM International Symposium on Code Generation and Optimization (CGO), IEEE, 2017, pp. 269–280.

[13] R. M. Hord: *The Illiac IV: the first supercomputer*, Springer Science & Business Media, 2013.

[14] *Introduce loop interchange pass and enable it at -O3.* https://gcc.gnu.org/ml/gcc-patches/2017-12/msg00360.html, Accessed: 2020-05-24.

[15] N. P. Johnson, J. Fix, S. R. Beard, et al.: *A collaborative dependence analysis framework*, in: 2017 IEEE/ACM International Symposium on Code Generation and Optimization (CGO), IEEE, 2017, pp. 148–159.

[16] S. C. Johnson: *A portable compiler: theory and practice*, in: Proceedings of the 5th ACM SIGACT-SIGPLAN symposium on Principles of programming languages, 1978, pp. 97–104, doi: https://doi.org/10.1145/512760.512771.

[17] M. Kruse: *Loop Optimizations in LLVM: the Good, the Bad, and the Ugly*, LLVM Developers' Meeting, 2018, url: https://www.youtube.com/watch?v=QpvZt9w-Jik.

[18] D. J. Kuck: *Automatic program restructuring for high-speed computation*, in: International Conference on Parallel Processing, Springer, 1981, pp. 66–84, doi: https://doi.org/10.1007/BFb0105110.

[19] D. J. Kuck, R. H. Kuhn, D. A. Padua, B. Leasure, M. Wolfe: *Dependence graphs and compiler optimizations*, in: Proceedings of the 8th ACM SIGPLAN-SIGACT symposium on Principles of programming languages, 1981, pp. 207–218, doi: https://doi.org/10.1145/567532.567555.

[20] S. Larsen, S. Amarasinghe: *Exploiting superword level parallelism with multimedia instruction sets*, Acm Sigplan Notices 35.5 (2000), pp. 145–156,
doi: https://doi.org/10.1145/349299.349320.

[21] D. Leopoldseder, L. Stadler, T. Würthinger, et al.: *Dominance-based duplication simulation (DBDS): code duplication to enable compiler optimizations*, in: Proceedings of the 2018 International Symposium on Code Generation and Optimization, 2018, pp. 126–137,
doi: https://doi.org/10.1145/3168811.

[22] D. Naishlos: *Autovectorization in GCC*, in: Proceedings of the 2004 GCC Developers Summit, 2004, pp. 105–118.

[23] *Persistence, Facades and Roslyn's Red-Green Trees*, https://docs.microsoft.com/en-gb/archive/blogs/ericlippert/persistence-facades-and-roslyns-red-green-trees, Accessed: 2020-05-24.

[24] *Polly: The Architecture.* https://polly.llvm.org/docs/Architecture.html, Accessed: 2020-05-24.

[25] V. Porpodas: *Supergraph-slp auto-vectorization*, in: 2017 26th International Conference on Parallel Architectures and Compilation Techniques (PACT), IEEE, 2017, pp. 330–342,
doi: https://doi.org/10.1109/PACT.2017.21.

[26] V. Porpodas, T. M. Jones: *Throttling automatic vectorization: When less is more*, in: 2015 International Conference on Parallel Architecture and Compilation (PACT), IEEE, 2015, pp. 432–444,
doi: https://doi.org/10.1109/PACT.2015.32.

[27] V. Porpodas, R. C. Rocha, E. Brevnov, L. F. Góes, T. Mattson: *Super-Node SLP: optimized vectorization for code sequences containing operators and their inverse elements*, in: 2019 IEEE/ACM International Symposium on Code Generation and Optimization (CGO), IEEE, 2019, pp. 206–216,
doi: https://doi.org/10.1109/CGO.2019.8661192.

[28] V. Porpodas, R. C. Rocha, L. F. Góes: *Look-ahead SLP: Auto-vectorization in the Presence of Commutative Operations*, in: Proceedings of the 2018 International Symposium on Code Generation and Optimization, 2018, pp. 163–174,
doi: https://doi.org/10.1145/3168807.

[29] R. G. Scarborough, H. G. Kolsky: *A vectorizing Fortran compiler*, IBM Journal of Research and Development 30.2 (1986), pp. 163–171,
doi: https://doi.org/10.1109/10.1147/rd.302.0163.

[30] S. Sioutas, S. Stuijk, H. Corporaal, T. Basten, L. Somers: *Loop transformations leveraging hardware prefetching*, in: Proceedings of the 2018 International Symposium on Code Generation and Optimization, 2018, pp. 254–264,
doi: https://doi.org/10.1145/3168823.

[31] S. T. Teixeira, C. Ancourt, D. Padua, W. Gropp: *Locus: a system and a language for program optimization*, in: 2019 IEEE/ACM International Symposium on Code Generation and Optimization (CGO), IEEE, 2019, pp. 217–228,
doi: http://doi.acm.org/10.1145/2737924.2738003.

[32] M. J. Wolfe: *High performance compilers for parallel computing*, Addison-Wesley Longman Publishing Co., Inc., 1995.

[33] O. Zinenko, S. Verdoolaege, C. Reddy, et al.: *Modeling the conflicting demands of parallelism and temporal/spatial locality in affine scheduling*, in: Proceedings of the 27th International Conference on Compiler Construction, 2018, pp. 3–13,
doi: https://doi.org/10.1145/3178372.3179507.