Computer Science Journal of Moldova, vol.**29**, no.1(85), 2021

A Sublinear Sudoku Solution in cP Systems and its Formal Verification*

Yezhou Liu✉, Radu Nicolescu, Jing Sun, Alec Henderson

Abstract

Sudoku is known as a NP-complete combinatorial number-placement puzzle. In this study, we propose the first cP system solution to generalised Sudoku puzzles with $m \times m$ cells grouped in m blocks. By using a fixed constant number of rules, our cP system can solve all Sudoku puzzles in sublinear steps. We evaluate the cP system and discuss its formal verification.

Keywords: cP systems, P systems, Sudoku problem, NP-complete problem, Formal verification.

MSC 2020: 68Q07, 68N30.

1 Introduction

Sudoku is a famous number-placement puzzle designed for a single player, which has $m \times m$ cells divided into m blocks. A solvable Sudoku puzzle may have one or multiple solutions. In a valid Sudoku solution, each row, column and block contains exactly one of each number from 1 to m. For all the classic 9×9 Sudoku puzzles, there are approximately 6.67×10^{21} valid solutions [1].

Different algorithms can be used to solve Sudoku puzzles, which include: backtracking [2], stochastic search [3], evolutionary algorithms [4]–[7], propositional satisfiability inference techniques [8], constraint algorithms [9], [10] and rewriting rules [11]. Many existing solvers are designed to solve specific Sudoku instances, and their performance is related to the difficulties of these instances.

*This paper is based on our presentation at the 6th annual Rogojin lectures, November 15, 2019, Chisinau, Moldova.

Membrane computing systems (P systems) were inspired by the biological structure of living cells [12]. Major P system variants have theoretically unlimited computational power and memory, and can compute multiple tasks in a maximally parallel manner, which include but not limited to: P system with active membranes [13], tissue P systems [14], spiking neural P systems [15], kernel P systems [16], and P systems with complex objects (cP systems) [17], [18]. According to the membrane structure, these P system variants can be roughly classified into three categories: cell-like P systems, tissue-like P systems and neural-like P systems.

The first P study of Sudoku used a family of P systems containing enzymatic, dissolution and send-out rules [19]. The proposed P systems can either solve a Sudoku instance, or detect the drawback and stop the computation. Later, the work was extended by adding some solving strategies and a brute-force algorithm [20]. Another Sudoku solution used cell-like P systems with the rules of particle swarm optimization to solve Sudoku instances [21]. The authors classified the instances based on the difficulty, and evaluated their P systems on success rate and running time.

As a recently proposed P system variant, cP systems support complex symbols and generic rules, which can use a fixed constant number of rules to solve computationally hard problems efficiently. In this study, we provide a cP system solution to general Sudoku puzzles. Compared to other P system solutions, our cP system solution consists of only 16 rules (by using relations as special promoters), which can solve all $m \times m$ Sudoku puzzles in $3m + 7$ steps regardless of their difficulty. Considering an input size of m^2, our cP solution is sublinear. We use the model checker PAT3 [22] to formally verify the cP solution.

We organize this paper as follows. Section 2 reviews cP systems' syntax and the model checker PAT3. Section 3 introduces our Sudoku solution. Section 4 shows a worked example of the solution. Section 5 discusses its formal verification. Section 6 discusses existing P system Sudoku solutions. Section 7 concludes the work with future directions.

2 Background

cP systems share advantages of both traditional cell-like P systems [12] and tissue P systems [14]. One cP system may have one or multiple top-cells, where each top-cell can contain a number of nested sub-cells. Top-cells in cP systems have multiset rewriting rules, and sub-cells are only used to represent local data [18]. In previous studies, cP systems were used to solve multiple computationally hard problems, which include the subset sum problem [23], Hamiltonian cycle problem [24], travelling salesman problem [24], and quantified SAT problem [25].

To formally verify cP systems, we can apply model checking techniques. A model checker can simulate a finite-state system, and exhaustively search its state-space to check if the system can meet some given specifications. We use the model checker PAT3 to verify our cP Sudoku solution, where its rules are translated into CSP# descriptions, and its properties are specified in temporal logic.

2.1 cP system notation

The syntax of cP systems is shown in Fig. 1. Our basic vocabulary consists of *atoms* and *variables*, collectively known as *simple terms*. We use lowercase letters to represent atoms and uppercase letters to denote variables. For instance, a, b, c are atoms and X, Y, Z are variables. To represent numbers, we use a unity symbol *1*, which can be treated as a special atom. Underscores (_) are used to denote anonymous variables.

Compound terms are recursively built by labelled multisets of other compound or simple terms with *functors*, where functors are atoms. For instance, $a(b)$, $a(b(1)c(X))$, $f(g(a)b\_)$ are compound terms. In cP systems, cells and their contents are represented as compound terms. For example, a cell with label a which contains two atoms b and c can be presented as $a(bc)$; a cell with label d which contains three unity symbols can be represented as $d(111)$ or $d(1^3)$. We can either represent numbers in unary as *1*, *11*, *111*..., 1^n; or in decimal as 1, 2, 3..., n.

Since all the terms in cP system are multiset-based, the order of their sub-terms does not matter. For example, to represent a cell la-

$< term > ::= < simple\text{-}term > | < compound\text{-}term >$
$< simple\text{-}term > ::= < atom > | < variable >$
$< compound\text{-}term > ::= < functor > (< argument >)$
$< functor > ::= < atom >$
$< state > ::= < l\text{-}state > | < r\text{-}state >$
$< l\text{-}state > ::= < atom >$
$< r\text{-}state > ::= < atom >$
$< argument > ::= < term > ...$
$< rule > ::= < lhs > \rightarrow_{\alpha} < rhs > < promoters >$
$< lhs > ::= < l\text{-}state > < term > ...$
$< rhs > ::= < r\text{-}state > < term > ...$
$< promoters > ::= \ | < term > ... | < relation > ...$

Figure 1. cP system syntax (lhs = left-hand-side, rhs = right-hand-side, α = rule application model)

belled as f which contains two simple terms g and h, we can either write $f(gh)$ or $f(hg)$, where the order of g and h does not matter. Similarly, the terms $a(bcd)$, $a(bdc)$ and $a(dcb)$ are identical.

Only top-cells in cP systems contain rewriting rules, each top-cell may have one or multiple rules. A rule consists of a lhs, a rhs and an *application model* α; both its lhs and rhs contain a state and zero or more terms. For example: $s_1\ a\ \rightarrow_1\ s_2\ bc$ is a cP rule, where s_1 is its *l-state*, s_2 is its *r-state*, a is a term of its lhs, b and c are terms of its rhs. The application model of this rule is 1, which means "exactly-once". The rule can consume a term a and produce two terms b and c – in other words, it can rewrite a as bc.

Every cP top-cell has a *state* alternatively named *system state*. States are atoms, to distinguish them with lhs and rhs terms in rules, we often use atom s with subscripts to represent states, for example: s_1, s_2 and s_3.

A rule is applicable if and only if its *l-state* matches the system state, and its lhs terms can be found in the system. After applying it, the system state will be changed to the rule's *r-state*. Suppose we have a cP system at state s_1, which has a term a and a rule $s_1\ a\ \rightarrow_1\ s_2\ bc$. Since the rule's *l-state* matches the system state, and the system does

contain a term a, the rule is applicable. After it is being applied, the term a will be consumed, two terms b and c will be generated, and the system state will be changed to s_2.

For generic rules with variable terms such as $a(X)$, $b(Y)$, a *one-way unification* (pattern matching) is supported in cP systems. Before applying a rule, its variable terms must be unified against terms in the system. Suppose a cP system at state s_1 that has two terms $a(1)$, $b(11)$ and a generic rule $s_1\ a(X)\ b(Y) \rightarrow_1 s_2\ c(XY)$. Variable terms $a(X)$ and $b(Y)$ will be unified against terms $a(1)$ and $b(11)$. In this example, we can get $X \mapsto 1$, $Y \mapsto 11$. So the rule will be unified as $s_1\ a(1)\ b(11) \rightarrow_1 s_2\ c(111)$. By applying it, the two terms $a(1)$ and $b(11)$ will be consumed, and a term $c(111)$ will be generated. The system state will be changed from s_1 to s_2.

Two major *application models* are supported in cP systems, which are "*exactly-once* (1)" and "*max-parallel* (+)". In the exactly-once model, a rule will only apply once. In the max-parallel model, a rule will apply to all possible terms simultaneously. Suppose a cP system at state s_1 that has three terms $a(1^2)$, $a(1^3)$, $a(1^3)$ and a rule $s_1\ a(1X) \rightarrow_\alpha s_2\ a(X)$. The rule can be unified to three ground rules (ground here means "without variables"), which are $s_1\ a(11) \rightarrow_\alpha s_2\ a(1)$, $s_1\ a(11^2) \rightarrow_\alpha s_2\ a(1^2)$ and $s_1\ a(11^2) \rightarrow_\alpha s_2\ a(1^2)$, where the variable X is unified as 1, 1^2 and 1^2, respectively. When the application model α of the rule is "1" (exactly-once model), the system will *non-deterministically* choose one unified rule to apply. The computation result will be $a(1)$, $a(1^3)$, $a(1^3)$ or $a(1^2)$, $a(1^2)$, $a(1^3)$. If α in the rule is "+" (max-parallel model), the system will apply all three unified rules, the computation result will be $a(1)$, $a(1^2)$, $a(1^2)$.

For a cP system with a max-parallel rule, the unified rules which can be applied together are called *compatible*. Suppose a cP system at state s_1 has four terms $a(c)$, $a(d)$, $b(e)$, $b(f)$, and a rule $s_1\ a(X)b(Y) \rightarrow_+ s_1\ g(XY)$. To unify the rule against the system terms, we can get the following unified rules: *r1*: $s_1\ a(c)b(e) \rightarrow_+ s_1\ g(ce)$, *r2*: $s_1\ a(c)b(f) \rightarrow_+ s_1\ g(cf)$, *r3*: $s_1\ a(d)b(e) \rightarrow_+ s_1\ g(de)$, and *r4*: $s_1\ a(d)b(f) \rightarrow_+ s_1\ g(df)$. When applicable, these unified rules will be non-deterministically chosen by the system. Suppose it chooses

r2 to apply, the system terms $a(c)$ and $b(f)$ will be "locked" by *r2*, which will be used to consume and to produce the term $g(cf)$ later. So they cannot be used by other unified rules any more. The only free terms in the system are $a(d)$ and $b(e)$, which can be used in *r3*. Thus, *r2* and *r3* can be applied together – they are compatible. Similarly, *r1* and *r4* are compatible. In the max-parallel model, the system will non-deterministically choose *r2*, *r3* OR *r1*, *r4* to apply.

cP systems apply rules following a *weak priority* order – i.e., rules are sequentially considered in the top-down order. The first applied rule commits the target state, any subsequent rule that indicates a different target state is then disabled. This way, the weak priority order can be used to simulate *if-then-else* structures of traditional programming.

Suppose a cP system at state s_1 that has two terms $a(c)$, $b(d)$ and three rules: *r1*: $s_1\ a(X) \rightarrow_1 s_2\ o(XX)$, *r2*: $s_1\ b(X) \rightarrow_1 s_3\ p(X)$, and *r3*: $s_1\ b(X) \rightarrow_1 s_2\ q(XXX)$. The system will first consider *r1* and find if it is applicable. Thus, the target state will be confirmed as $s2$. Since *r2* commits to a different target state s_3, it is not applicable. *r3* commits to s_2, and it is compatible with *r1*, so it will be applied with *r1* together in the same *step*. The computational result of the system will be $o(cc)$, $q(ddd)$.

In each step, new generated terms will be temporarily put into a "product membrane", which will not be available until the next step. For example, a cP system has two terms $a(c)$, $b(d)$ and two rules *r1*: $s_1\ a(X) \rightarrow_1 s_2\ b(X)$ and *r2*: $s_1\ b(X) \rightarrow_+ s_2\ c(X)$. *r1* and *r2* will be applied in the same step. The term $b(c)$ generated by *r1* will be sent to the product membrane, which will not be consumed by *r2* in the same step. After applying them, the system state will be changed to s_2, then none of them are applicable. The computational result of the system will be $b(c)$, $c(d)$.

To apply a rule with promoters, the promoters must exist in the system, and will not be consumed. Suppose a cP system has a rule $s_1 \rightarrow_1 s_2\ x(X) \mid y(XZ)\ z(Z)$, and two terms $y(6)$ and $z(4)$. The rule can be unified as $s_1 \rightarrow_1 s_2\ x(2) \mid y(6)\ z(4)$. By applying it, a term $x(2)$ will be generated. $y(6)$ and $z(4)$ are promoters, they will be checked by the rule, but will not be consumed.

For readability, we can use two kinds of delimiters in cP terms as needed, which are blank space " " and comma ",". Adding delimiters to a term will not affect its meaning. For example, $a(bc)$, $a(b\ c)$ and $a(b, c)$ represent the same term.

To simplify the cP encoding of Sudoku, we can use the following abbreviation: $a(X)(Y) \equiv a(a_1(X)a_2(Y))$, when the sub-cell names a_1 and a_2 are unimportant. Similarly, we can use $b(X)(Y)(Z)$ to represent $b(b_1(X)b_2(Y)b_3(Z))$ as needed.

In this study, to make our rules more readable, we use logic *relations* – including *multiset inclusion* ($\subseteq$), *multiset NOT inclusion* ($\nsubseteq$), and *multiset inequality* ($\neq$) – as special promoters. This design can be translated into classical cP systems by adding a few more rules.

For example, to test the inclusion relation ($\subseteq$) – e.g., $aab \subseteq aabcd$, we can use a rule fragment (stateless) $\rightarrow_1 \quad 1 \mid aab$. If the system contains the multiset *aabcd*, then the rule fragment is applicable, and it will generate a symbol *1* to indicate that *aab* is a submultiset of *aabcd*. If the multiset which needs to be checked is not a supermultiset of *aab*, for instance $cdef$, the rule fragment will not be applicable, and the symbol *1* will not be generated.

Another example is to test the NOT inclusion relation ($\nsubseteq$) – e.g. $aab \nsubseteq abcd$, we can use two rules: *r1*: $s_1 \rightarrow_1 s_2 \mid aab$ and *r2*: $s_1 \rightarrow_1 s_3\ 1$. Suppose the cP system starts at state s_1, and contains the multiset *abcd*. It will consider *r1* first, but it cannot find all *r1*'s promoters – thus *r1* is not applicable. Then the system will consider *r2*, which is applicable. *r2* will generate a symbol *1* to indicate that *aab* is not a submultiset of *abcd*. By using the same rules, if the multiset that needs to be checked is a supermultiset of *aab*, for instance *aabbc*, rule *r1* – which appears before *r2*, thus has a higher priority – is applicable, it will change the system state to s_2, then no more rules are applicable, and the symbol *1* will not be generated.

2.2 The model checker PAT3

PAT3 provides an extensible framework for simulating and verifying different systems in multiple application domains [22]. Previous research

showed that PAT3 can effectively verify cP models, and transformation guidelines from cP syntax to CSP# were proposed [23].

PAT3 supports several semantic models and modelling languages, which include Communicating Sequential Processes (CSP), Real-Time Systems (RTS), Labeled Transition Systems (LTS), and Timed Automata (TA). To improve PAT3's performance, the authors implemented a number of model checking algorithms, state reduction techniques and abstraction techniques in it.

A specification language called CSP# (Communicating Sequential Programs) is supported by PAT3. CSP#, an extension of CSP, combines high-level modeling operators (e.g. choices, interrupt, parallel composition, asynchronous message passing) and low-level constructs (e.g. data structures and conditional statements) together. CSP# is especially good at representing cP systems, which has great potential for simulating the cell communication among multiple top-cells.

To describe features of cP systems, we can either use Linear Temporal Logic (LTL) or Computation Tree Logic (CTL). In addition to specifying features by users, some commonly checked features are pre-implemented in PAT3.

Similar to other model checkers, PAT3 also has a simulator, which can be used to visualize its checking processes. Selecting different simulation engines, PAT3 can traverse the system's state-space using different heuristics.

3 Solving Sudoku in a cP system

Our strategy of solving Sudoku is to generate all possible solutions (matrices), eliminate invalid ones, and filter them by comparing them to the input puzzle. For an $m \times m$ Sudoku, the cP system will first generate all valid m-size row candidates, where each candidate is a permutation of $[1..m]$. Then the system will use these row candidates to build templates of $m \times m$ matrices. After getting all the matrix templates, the system will filter them by columns and blocks. After that, it will contain all the valid $m \times m$ Sudoku solutions. Then it can match these matrix templates to a particular instance, and find its

solutions.

The cP system starts at state s_1 with terms $p()$, $t()$, $s(S)$, $a(1)$, $a(2)$,..., $a(m)$, $n(n)$, $m(m)$, and $l(1)$. The term $p(\_)$ is used to build and store the row candidates, $t(\_)$ is used to store matrix templates, and $s(S)$ is the cP encoding of a Sudoku puzzle instance. Terms $a(1)$, $a(2)$ ..., $a(m)$ store the numbers from 1 to m, which can be used to fill the blank cells of the puzzle. $n(n)$ stores the block size and $m(m)$ stores the problem size of the puzzle, where $m = n^2$. The system uses $l(\_)$ as a counter.

A simple Sudoku example ($m = 4$) is shown in Fig. 2. In its cP representation, we use two terms $m(4)$ and $n(2)$ to represent its problem size and block size. Four numbers $a(1)$, $a(2)$, $a(3)$ and $a(4)$ can be used to fill the puzzle. The puzzle instance is encoded as $s(r(1)(c(3)(2), c(4)(4)),\ r(2)(c(1)(2), c(2)(4), c(4)(3)),\ r(3)(c(2)(1)),$ $r(4)(c(3)(3)))$. In the encoding, the term s stores all the existing numbers of the puzzle, where sub-cell names r and c refer to "row" and "column" respectively. The sub-cell $r(2)(c(1)(2), c(2)(4), c(4)(3))$ can be interpreted as "the value in row 2 column 1 is 2, in row 2 column 2 is 4, and in row 2 column 4 is 3".

		2	4
2	4		3
	1		
		3	

Figure 2. A Sudoku puzzle, $m = 4$

3.1 Generating row candidates

To build valid solutions, the cP system first generates all the row candidates. Each row candidate contains all the numbers from 1 to m, and each number only appears once. A ruleset with four rules can be used to generate row candidates in a column by column manner (Fig. 3).

$s_1\ l(M1)$	$\rightarrow_1$	s_2	$l(1) \mid m(M)$	(1)
s_1	$\rightarrow_+$	s_1	$p(X, c(L)(V)) \mid l(L), a(V), p(X), (c(\_)(V) \not\subseteq X)$	(2)
$s_1\ p(\_)$	$\rightarrow_+$	s_1		(3)
$s_1\ l(L)$	$\rightarrow_1$	s_1	$l(L1)$	(4)

Figure 3. Ruleset (1): generating row candidates

Rule (1) uses a counter $l(\_)$ to track the progress of generating row candidates. When the value of $l(M1)$ is greater than the puzzle size $m(M)$, it means all the row candidates have been successfully generated. Then the cP system resets the counter to $l(1)$, changes its state to s_2 and moves to the next ruleset.

Rule (2) works in the max-parallel model, so all the compatible unified rules will be applied (all solution will be generated). When rule (2) is applied, it adds a number V at column L to each row candidate $p(X)$. The relation $c(\_)(V) \not\subseteq X$ guarantees the number V has not been used in the same row candidate. At the beginning of the computation, $p()$ was empty. By applying rule (2) once, the system creates m different terms, which are $p(c(1)(1))$, $p(c(1)(2))$..., $p(c(1)(m))$. By applying it again, the system will generate $m \times (m-1)$ terms including $p(c(1)(1), c(2)(2))$, $p(c(1)(1), c(2)(3))$..., $p(c(1)(m), c(2)(m-1))$. After applying it m times, the cP system will generate all the $m!$ row candidates.

Rule (3) is another max-parallel rule, which cleans the out-of-date $p(\_)$ terms in the system. As mentioned, rule (2), (3) and (4) commit to the same target state, so if applicable, they will be applied in one step. Thus $p(\_)$ terms generated by rule (2) will not be immediately consumed by rule (3) in the same step.

Rule (4) increases the counter $l(\_)$ by 1 in every step, by consuming the existing counter $l(L)$, and producing a new counter $l(L1)$.

To generate all the row candidates for a $m \times m$ Sudoku, the cP system needs to apply the ruleset $m+1$ times, and the state of the system will be changed from s_1 to s_2.

3.2 Generating matrix templates

The ruleset to build matrix templates is shown in Fig. 4. Using the row candidates generated by ruleset (1), the cP system builds matrix templates row by row.

$s_2\ l(M1)$	$\to_1$	s_3	$l(1) \mid m(M)$	(5)
s_2	$\to_+$	s_2	$t(X, r(L)(P)) \mid l(L), p(P), t(X), (r(\_)(P) \not\subseteq X)$	(6)
$s_2\ t(\_)$	$\to_+$	s_2		(7)
$s_2\ l(L)$	$\to_1$	s_2	$l(L1)$	(8)

Figure 4. Ruleset (2): generating matrix templates

The cP system uses the counter $l(\_)$ to track the working row. Once all the m rows of matrix templates are filled with row candidates – when $l(M1)$ is greater than $m(M)$ – rule (5) is applicable. It changes the system state to s_3, and resets the counter to $l(1)$.

Rule (6) generates matrix templates row by row. In every step, the system adds exactly one row candidate $p(P)$ at row L to each matrix template $t(X)$. After m steps, the system will finish generating all $m!!/(m! - m)!$ matrix templates. Rule (7) cleans out-of-date $t(\_)$ terms, and rule (8) increments the counter $l(\_)$ by 1 in each step.

Ruleset (2) takes $m + 1$ steps in total. The matrix templates generated by ruleset (2) do not have any number conflicts in each row, since every row candidate is a permutation of $[1..m]$; but they may have number conflicts in columns and blocks (Fig. 5).

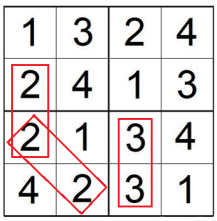

Figure 5. Number conflicts in a matrix template

3.3 Filtering matrix templates by columns

To delete the matrix templates with number conflicts in columns, the cP system only needs one max-parallel rule (Fig. 6). The rule works as a filter, which is applied to all the matrix templates simultaneously. In a matrix template $t(\_)$, if there are two cells in the same column – row A column C and row B column C – share the same value V, the template will be consumed (deleted). Ruleset (3) only needs 1 step to run. After applying it, all the matrix templates that remain in the cP system do not have any number conflicts in rows and columns.

$$s_3 \quad t(r(A)(c(C)(V)\_),\ r(B)(c(C)(V)\_),\ \_) \qquad \rightarrow_+ \qquad s_3 \qquad (9)$$

Figure 6. Ruleset (3): filtering matrix templates by columns

3.4 Filtering matrix templates by blocks

To check if matrix templates have number conflicts in blocks, we need to create some supporting terms to indicate the relationship among rows, columns and blocks. For example, when $m = 9$, we can build terms $b(1)(1)$, $b(2)(1)$, $b(3)(1)$, $b(4)(2)$, $b(5)(2)$, $b(6)(2)$, $b(7)(3)$, $b(8)(3)$ and $b(9)(3)$ in the cP system. To check if two Sudoku cells are in the same block, we only need to compare their rows and columns with the supporting terms. Suppose we want to check if two cells – row 4 column 3 and row 6 column 1 – are in the same block (Fig. 7). First, we check terms $b(4)(A)$ and $b(6)(B)$ in the supporting terms, we can find $A = 2$ and $B = 2$. Then we check $b(3)(X)$ and $b(1)(Y)$, find $X = 1$ and $Y = 1$. If $A = B$ and $X = Y$, the two cells are in the same block; otherwise they are not. In this example, row 4 column 3 and row 6 column 1 are in the same block.

The ruleset we use to build the supporting terms is shown in Fig. 8. Rule (10) creates two terms $v(1)$ and $k(N)$, and changes the state to s_4. Term $v(V)$ holds a value to fill in current supporting term $b(\_)(\_)$, and $k(K)$ tracks the boundary of blocks. Rule (11) monitors the progress of building supporting terms. When $l(M1)$ is greater than $m(M)$, the

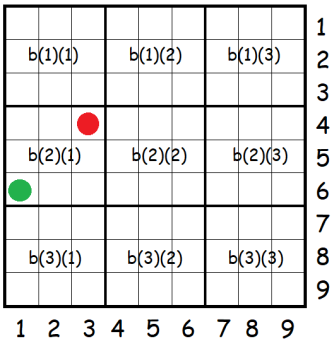

Figure 7. Checking if two cells are in the same block

system has finished creating supporting terms, it changes the state to s_5. Rule (12) creates a supporting term $b(L)(V)$ based on the counter $l(L)$ and value $v(V)$. Rule (13) updates terms $v(\_)$ and $k(\_)$ after the counter $l(\_)$ moved to the next block. Rule (14) increases the counter $l(L)$ by 1 in each step.

s_3	$\rightarrow_1$	s_4	$v(1),\ k(N) \mid n(N)$	(10)
$s_4\ l(M1)$	$\rightarrow_1$	s_5	$\mid m(M)$	(11)
s_4	$\rightarrow_1$	s_4	$b(L)(V) \mid l(L),\ v(V)$	(12)
$s_4\ k(K),\ v(V)$	$\rightarrow_1$	s_4	$k(KN),\ v(V1) \mid n(N),\ l(K)$	(13)
$s_4\ l(L)$	$\rightarrow_1$	s_4	$l(L1)$	(14)

Figure 8. Ruleset (4a): creating block checking supporting terms

After having the supporting terms in the system, rule (15) can filter the matrix templates by blocks (Fig. 9). If the system detects a matrix template $t(\_)$ has two cells (row X column A and row Y column B) in the same block that share the same value V, it will consume the template. Rule (15) runs in the max-parallel model, which can filter all the matrix templates in 1 step.

Rulesets (4a) and (4b) take $m + 3$ steps in total. After applying them, all the matrix templates that remain in the cP system are valid Sudoku solutions; and all the valid solutions are contained in the

$$s_5 \quad t(r(X)(c(A)(V)\_), r(Y)(c(B)(V)\_), \_) \quad \rightarrow_+ \quad s_5 \quad | \, b(X)(W), b(Y)(W), b(A)(C), b(B)(C) \qquad (15)$$

Figure 9. Ruleset (4b): filtering matrix templates by blocks

cP system!

3.5 Matching matrix templates to a Sudoku instance

One max-parallel rule can be used to match the matrix templates to a Sudoku instance (Fig. 10). Rule (16) compares all matrix templates $t(\_)$ to the instance $s(S)$. If the system finds any conflicts between a matrix template $t(\_)$ and $s(S)$, it deletes the template. Rule (16) takes 1 step. After applying it, the $t(T)$ terms in the system are solutions to the instance. A Sudoku instance may have multiple valid solutions, the cP system is guaranteed to find all of them at the same step.

$$s_5 \quad t(r(R)(c(C)(V)\_)\_) \quad \rightarrow_+ \quad s_5 \quad | \, s(r(R)(c(C)(U)\_)\_), \, U \neq V \qquad (16)$$

Figure 10. Ruleset (5): matching matrix templates to a Sudoku instance

The cP solution consists of 16 rules, which can solve any $m \times m$ Sudoku instances in $3m + 7$ steps. Considering the size of the input of a Sudoku puzzle is m^2, the complexity of the solution is sublinear (square root time).

4 A worked example

In this section we use the Sudoku puzzle ($m = 4$, $n = 2$) shown in Fig. 2 to illustrate how our cP system works. To represent the puzzle, we create 10 terms in the cP system (Table 1). The initial state of the system is s_1.

Table 1. Initial terms in the cP system

$m(4),\ n(2),\ a(1),\ a(2),\ a(3),\ a(4),\ l(1),\ p(),\ t(),$ $s(r(1)(c(3)(2), c(4)(4)),\ r(2)(c(1)(2), c(2)(4), c(4)(3)),\ r(3)(c(2)(1)),\ r(4)(c(3)(3))).$

The cP system uses ruleset (1) to generate all valid row candidates. A hand simulation can be found in Table 2. Since there are many terms in the system, the table only shows terms directly related to the ruleset. In each step, the system adds exactly one number into each $p(\_)$ term. After $m + 1 = 5$ steps, the cP system will successfully generate all the $m! = 24$ row candidates, which are the permutations of [1..4]. Then the system will reset the counter to $l(1)$, and change its state to s_2.

Ruleset (2) will be applied when the system is at state s_2. It uses row candidates to build matrix templates row by row. After $m+1 = 5$ steps, the system will generate all $m!!/(m! - m)! = 255024$ matrix templates, and change its state to s_3.

To filter the matrix templates by columns, the cP system applies ruleset (3). If it finds one value that appears twice in a column of a template, it consumes that template. Ruleset (3) takes one max-parallel step to apply to all matrix templates in the system, and does not change the system state. After applying it, all the matrix templates with number conflicts in their columns will be deleted from the cP system.

The cP system uses rulesets (4a) and (4b) to check and delete matrix templates with number conflicts in blocks. The hand simulation of building supporting terms (ruleset (4a)) can be found in Table 4. System terms which are related to ruleset (4a) are shown in the table.

By applying rule (10), the system will generate terms $v(1)$ and $k(2)$. Rule (11) compares terms $l(L)$ and $m(4)$, it is only applied when L is greater than 4. Rule (12) keeps building $b(\_)$ terms in every step. Rule (13) updates $k(\_)$ and $v(\_)$ values when the counter $l(\_)$ moves to a new block. Rule (14) increases $l(\_)$ by 1 in each step. Because rules (12), (13) and (14) are committing to the same state s_4, the cP system will apply them in the same step if possible.

Terms $b(1)(1)$, $b(2)(1)$, $b(3)(2)$, $b(4)(2)$ show the relationship among

Table 2. Generating row candidates

Step	State	Available terms	Generated terms	Rule to apply
0	s_1	$m(4)$, $l(1)$, $p()$.		(2)
1	s_1	$m(4)$, $l(1)$.	$p(c(1)(1))$, $p(c(1)(2))$, $p(c(1)(3))$, $p(c(1)(4))$.	(4)
1	s_1	$m(4)$.	$p(c(1)(1))$, $p(c(1)(2))$, $p(c(1)(3))$, $p(c(1)(4))$, $l(2)$.	(2)
2	s_1	$m(4)$, $p(c(1)(1))$, $p(c(1)(2))$, $p(c(1)(3))$, $p(c(1)(4))$, $l(2)$.	$p(c(1)(1), c(2)(2))$, $p(c(1)(1), c(2)(3))$,..., $p(c(1)(4), c(2)(3))$.	(3)
2	s_1	$m(4)$, $l(2)$.	$p(c(1)(1), c(2)(2))$, $p(c(1)(1), c(2)(3))$,..., $p(c(1)(4), c(2)(3))$.	(4)
2	s_1	$m(4)$.	$p(c(1)(1), c(2)(2))$, $p(c(1)(1), c(2)(3))$,..., $p(c(1)(4), c(2)(3))$, $l(3)$.	(2)
3	s_1	$m(4)$, $p(c(1)(1), c(2)(2))$, $p(c(1)(1), c(2)(3))$,..., $p(c(1)(4), c(2)(3))$, $l(3)$.	$p(c(1)(1), c(2)(2), c(3)(3))$, $p(c(1)(1), c(2)(2), c(3)(4))$,..., $p(c(1)(4), c(2)(3), c(3)(2))$.	(3)
3	s_1	$m(4)$, $l(3)$.	$p(c(1)(1), c(2)(2), c(3)(3))$, $p(c(1)(1), c(2)(2), c(3)(4))$,..., $p(c(1)(4), c(2)(3), c(3)(2))$.	(4)
3	s_1	$m(4)$.	$p(c(1)(1), c(2)(2), c(3)(3))$, $p(c(1)(1), c(2)(2), c(3)(4))$,..., $p(c(1)(4), c(2)(3), c(3)(2))$, $l(4)$.	(2)
4	s_1	$m(4)$, $p(c(1)(1), c(2)(2), c(3)(3))$, $p(c(1)(1), c(2)(2), c(3)(4))$,..., $p(c(1)(4), c(2)(3), c(3)(2))$, $l(4)$.	$p(c(1)(1), c(2)(2), c(3)(3), c(4)(4))$, $p(c(1)(1), c(2)(2), c(3)(4), c(4)(3))$,..., $p(c(1)(4), c(2)(3), c(3)(2), c(4)(1))$.	(3)
4	s_1	$m(4)$, $l(4)$.	$p(c(1)(1), c(2)(2), c(3)(3), c(4)(4))$, $p(c(1)(1), c(2)(2), c(3)(4), c(4)(3))$,..., $p(c(1)(4), c(2)(3), c(3)(2), c(4)(1))$.	(4)
4	s_1	$m(4)$.	$p(c(1)(1), c(2)(2), c(3)(3), c(4)(4))$, $p(c(1)(1), c(2)(2), c(3)(4), c(4)(3))$,..., $p(c(1)(4), c(2)(3), c(3)(2), c(4)(1))$, $l(5)$.	(1)
5	s_2	$m(4)$, $p(c(1)(1), c(2)(2), c(3)(3), c(4)(4))$, $p(c(1)(1), c(2)(2), c(3)(4), c(4)(3))$,..., $p(c(1)(4), c(2)(3), c(3)(2), c(4)(1))$.	$l(1)$.	

rows, columns and blocks. To check if two cells row 1 column 2 and row 2 column 3 are in the same block, we need to compare their rows

Table 3. A matrix template with number conflicts (cP representation)

$t(r(1)(c(1)(1), c(2)(3), c(3)(2), c(4)(4)),$
$r(2)(c(1)(2), c(2)(4), c(3)(1), c(4)(3)),$
$r(3)(c(1)(2), c(2)(1), c(3)(4), c(4)(3)),$
$r(4)(c(1)(4), c(2)(1), c(3)(3), c(4)(2))).$

Table 4. Building supporting terms (for block check)

Step	**State**	**Available terms**	**Generated terms**	**Rule to apply**
0	s_3	$n(2)$, $m(4)$, $l(1)$.		(10)
1	s_4	$n(2)$, $m(4)$, $l(1)$.	$v(1)$, $k(2)$.	(12)
2	s_4	$n(2)$, $m(4)$, $l(1)$, $v(1)$, $k(2)$.	$b(1)(1)$.	(14)
2	s_4	$n(2)$, $m(4)$, $v(1)$, $k(2)$.	$b(1)(1)$, $l(2)$.	(12)
3	s_4	$n(2)$, $m(4)$, $v(1)$, $k(2)$, $b(1)(1)$, $l(2)$.	$b(2)(1)$.	(13)
3	s_4	$n(2)$, $m(4)$, $b(1)(1)$, $l(2)$.	$b(2)(1)$, $v(2)$, $k(4)$.	(14)
3	s_4	$n(2)$, $m(4)$, $b(1)(1)$.	$b(2)(1)$, $v(2)$, $k(4)$, $l(3)$.	(12)
4	s_4	$n(2)$, $m(4)$, $b(1)(1)$, $b(2)(1)$, $v(2)$, $k(4)$, $l(3)$.	$b(3)(2)$.	(14)
4	s_4	$n(2)$, $m(4)$, $b(1)(1)$, $b(2)(1)$, $v(2)$, $k(4)$.	$b(3)(2)$, $l(4)$	(12)
5	s_4	$n(2)$, $m(4)$, $b(1)(1)$, $b(2)(1)$, $v(2)$, $k(4)$, $b(3)(2)$, $l(4)$.	$b(4)(2)$.	(13)
5	s_4	$n(2)$, $m(4)$, $b(1)(1)$, $b(2)(1)$, $b(3)(2)$, $l(4)$.	$b(4)(2)$, $v(3)$, $k(6)$.	(14)
5	s_4	$n(2)$, $m(4)$, $b(1)(1)$, $b(2)(1)$, $b(3)(2)$.	$b(4)(2)$, $v(3)$, $k(6)$, $l(5)$.	(11)
6	s_5	$n(2)$, $m(4)$, $b(1)(1)$, $b(2)(1)$, $b(3)(2)$, $b(4)(2)$, $v(3)$, $k(6)$.		

and columns separately. To check their rows, we need to find terms $b(1)(A)$ and $b(2)(B)$ in the cP system – which are $b(1)(1)$ and $b(2)(1)$, thus $A = B = 1$. To check their columns, we search for $b(2)(X)$ and $b(3)(Y)$ in the system, and we will get $b(2)(1)$, $b(3)(2)$, thus we have $X = 1$, $Y = 2$, where $X \neq Y$. Then we know that row 1 column 2 and row 2 column 3 are not in the same block.

Ruleset (4a) takes $m + 2 = 6$ steps. After building the support-

ing terms, the system will take one max-parallel step to delete matrix templates with number conflicts in blocks (ruleset (4b)), then all the matrix templates left in the cP system are valid solutions to different Sudoku puzzles.

By applying ruleset (5) – which also takes one max-parallel step – the system can find the solutions to the given Sudoku instance (Table 5), by deleting other matrix templates in it. The entire solution takes $3m + 7 = 19$ steps.

Table 5. The solution of the example puzzle in Fig. 2

$t(r(1)(c(1)(1), c(2)(3), c(3)(2), c(4)(4)),$
$r(2)(c(1)(2), c(2)(4), c(3)(1), c(4)(3)),$
$r(3)(c(1)(3), c(2)(1), c(3)(4), c(4)(2)),$
$r(4)(c(1)(4), c(2)(2), c(3)(3), c(4)(1))).$

5 Verifying the cP Sudoku solution

We verified the two core rulesets (1 and 2) of our solution in PAT3. To model our cP system in CSP#, we mainly followed the transformation guidelines proposed in our previous work [23]. A translation example from cP syntax of ruleset (1) to CSP# is shown in Figure 11.

Our cP system's properties including deadlock-freeness (safety, weak-liveness), terminating (safety, liveness), divergence-freeness (safety) and non-deterministic (fairness) were formally verified. The CSP# code of this study can be found in *https://github.com/YezhouLiu/cP-Sudoku*.

Table 6 shows the model checking result. The cP system is deadlockfree, divergencefree, terminating and non-deterministic. In model checking, a *deadlock* state is a state with no further move (except expected final states). The "state" here refers to the state in the Kripke structure in the model checker, which is different from states in cP systems. Deadlockfree can be written as $A\Box(E \bigcirc (true))$ in CTL, which means "a non-deadlock state must have at least one successor". The symbol $\Box$ refers to *globally* and $\bigcirc$ is the *next operator*. In cP systems,

```
#define M 9; //problem size
var a = [1,2,3,4,5,6,7,8,9];    //term a
var c[M];       //index: L-1, value: V
var l = 1; //term l
var state = 1;
var promoter_valid = true;

R1() = rule1{
        if(state == 1 && l == M + 1){
                l = 1;
                state = 2;
        }
} -> if (state == 2){Skip} else {R2()};

R2() = []i:{0..(M-1)}@ rule2{
        if (state == 1){
                promoter_valid = true;
                var j = 0;
                while (j < M){
                        if (c[j] == a[i]){
                                promoter_valid = false;
                        }
                        j++;
                }
                if(promoter_valid){c[l-1] = a[i];}
                state = 1;
        }
} -> if (promoter_valid){R3()} else{Skip};

R3() = rule3{
        if (state == 1){state = 1;}
} -> R4();

R4() = rule4{
        if(state == 1){
                l = l + 1;
                state = 1;
        }
} -> R1();
```

Figure 11. The CSP# representation of ruleset (1)

a deadlock indicates that the system terminates somewhere unexpectedly. For example, if a rule's $r - state$ is miswritten as another state, which is unused anywhere else, after applying the rule, the system may

encounter a deadlock. The deadlock check can tell us if the cP system will get stuck.

Divergence in PAT3 means a process may perform internal transitions forever, which is often undesirable. The divergence check can be used to detect badly designed rules in cP systems (e.g. unnecessary self-looping rules). *Terminating* means the system will eventually halt, which can often be written as a reachability property. For example, a termination of ruleset (1) can be defined as: $A\Diamond(at_{s_2})$, where $\Diamond$ means "eventually". When a cP system applies ruleset (1), after generating all the row candidates, the system state will be changed to s_2. Then ruleset (1) terminates, because none of its rules are applicable at s_2. Performing the terminating check, we can find out if a cP system will run forever, or eventually halt at an expected state.

As a fairness property, *deterministic* means for a state there is no more than one out-going transition. cP systems are often non-deterministic because of unification. When designing a non-deterministic cP system, we often need to make it *confluent* – i.e. all its evolutions need to yield the same result. For example, when building a row candidate, the cP system will randomly choose a number to fill its working column. All the possible numbers have equal chance to be selected, none of the numbers will be neglected by the system. The system will finally generate $m!$ row candidates which represent all the permutations of $[1..m]$.

Table 6. Model checking result of the cP Sudoku rulesets

Ruleset	**Problem Size**	**Deadlock-free**	**Divergence-free**	**Terminating**	**Deterministic**
(1)	4	True	True	True	False
(1)	9	True	True	True	False
(2)	4	True	True	True	False

A major limitation of applying model checking to our cP system is state explosion. We verified the system's properties with $m = 4$ and $m = 9$. When the problem size $m = 9$, PAT3 encounters a memory explosion issue when verifying ruleset (2). To model check ruleset (1), PAT3 generated $9! = 362880$ row candidates and successfully checked

its state-space. To verify ruleset (2), PAT3 needs to simulate the generation of the $9!!/(9! - 9)!$ matrix templates, which is impossible in practice. Even though PAT3 implements abstraction algorithms and can generate its state-space on the fly (without keeping the entire state-space in memory), it still cannot check that many states.

Because of the combinatorial explosion and limited languages features supported by existing general purpose model checkers, max-parallel filtering cP rulesets including ruleset (3), (4a), (4b), and (5) are not suitable to be verified via model checking. There is no straight forward way to manually release terms' memory in model checkers such as PAT3, so it is hard to emulate the term consumption in cP systems.

6 Related work

The first P system study on solving the Sudoku problem was published in 2010 [19]. To solve Sudoku puzzles, the authors used a family of P systems with enzymatic rules, dissolution rules, and send-out rules. In the P solution, 13 rulesets of $O(n^6)$ rules were used, where n is the block size of the puzzle. In most P system variants, to solve computationally hard problems, a number of P rules related to the problem size are needed. Instead of solving Sudoku in a brute-force manner, it used a human-style strategy. It is not guaranteed that the P system can solve all the Sudoku instances, but the system will indicate failure if it cannot solve an instance.

Later, the P system in [19] was extended by other researchers [20]. Additional algorithms including pruning were added to the solution. The authors also introduced a brute-force algorithm to the system. In this extended version of the P system, if no valid Sudoku solution can be found by the human-style strategy, the system will try the brute-force algorithm. $O(n^6)$ rules were used in the extended version of P system with the brute-force approach. The authors tested their solution in the P-Lingua simulator [26], but only for small-scale problems ($n = 2$).

Another P study used particle swarm optimization incorporated with P systems to solve Sudoku puzzles [21]. No P system rule was predefined in the system. The system generated the evolution rules

and communication rules by using particle swarm optimization. The authors also proposed another P system solution to solve Sudoku [27], where some evolution and communication rules were used. The ruleset of the P solution was not specified in the paper, too. These solutions cannot guarantee to solve all the Sudoku puzzles. The two algorithms designed by the authors only borrowed some shallow ideas from the cell-like structure of P systems, which did not make use of the theoretical unlimited computational power and memory of membrane computing models.

7 Conclusion

Although membrane computing models are suitable for solving computationally hard problems, to model practical problems in P system models is often non-trivial. In this study, we solved one of the most famous NP-complete puzzles – Sudoku – in cP systems. Our cP system only contains 16 rules, which can solve all $m \times m$ Sudoku puzzles in $3m + 7$ steps by using relations as special promoters. Considering the input size of a Sudoku puzzle is m^2, the cP solution is sublinear. To the best of our knowledge, our solution is the most efficient P system solution to the Sudoku problem in time complexity. We formally verified the core rulesets of our solution using the PAT3 model checker, and checked their safety, liveness and fairness properties.

Similar to other P system solutions, practical implementations of our cP solution will encounter a memory explosion issue. The search for finding efficient software or hardware implementations of P systems is one of the most important challenges in membrane computing.

In addition to the brute force algorithm, future work may include the design of practically useful cP solutions, based on practical Sudoku solving strategies. Although it is not guaranteed that the practical strategies can solve all Sudoku instances, they usually can find solutions for real-life Sudoku puzzles quickly.

To solve the existing issues on simulating and verifying cP systems, we also plan to design and implement a cP model checker, which can effectively simulate cP systems (including our cP Sudoku solution) and

automatically check their properties.

References

[1] D. Berend, "On the number of Sudoku squares," *Discrete Mathematics*, vol. 341, no. 11, pp. 3241–3248, 2018.

[2] D. E. Knuth, "Dancing links," *arXiv preprint cs/0011047*, 2000.

[3] R. Lewis, "Metaheuristics can solve Sudoku puzzles," *Journal of Heuristics*, vol. 13, no. 4, pp. 387–401, 2007.

[4] A. Moraglio, J. Togelius, and S. Lucas, "Product geometric crossover for the Sudoku puzzle," in *2006 IEEE International Conference on Evolutionary Computation*. IEEE, 2006, pp. 470–476.

[5] A. Moraglio and J. Togelius, "Geometric particle swarm optimization for the Sudoku puzzle," in *GECCO*, vol. 7, 2007, pp. 118–125.

[6] T. Mantere and J. Koljonen, "Solving, rating and generating Sudoku puzzles with GA," in *2007 IEEE Congress on Evolutionary Computation*. IEEE, 2007, pp. 1382–1389.

[7] Z. W. Geem, "Harmony search algorithm for solving Sudoku," in *International Conference on Knowledge-Based and Intelligent Information and Engineering Systems*. Springer, 2007, pp. 371–378.

[8] I. Lynce and J. Ouaknine, "Sudoku as a SAT problem." in *ISAIM*, 2006.

[9] H. Simonis, "Sudoku as a Constraint Problem," *Modelling and Reformulating Constraint Satisfaction Problems*, p. 13, 2005.

[10] T. K. Moon, J. H. Gunther, and J. J. Kupin, "Sinkhorn solves Sudoku," *IEEE Transactions on Information Theory*, vol. 55, no. 4, pp. 1741–1746, 2009.

[11] G. Santos-García and M. Palomino, "Solving Sudoku puzzles with rewriting rules," *Electronic Notes in Theoretical Computer Science*, vol. 176, no. 4, pp. 79–93, 2007.

[12] G. Păun, "Computing with membranes," *Journal of Computer and System Sciences*, vol. 61, no. 1, pp. 108–143, 2000.

[13] A. Păun, "On P systems with active membranes," in *Unconventional Models of Computation, UMC'2K*. Springer, 2001, pp. 187–201.

[14] C. Martín-Vide, G. Păun, J. Pazos, and A. Rodríguez-Patón, "Tissue P systems," *Theoretical Computer Science*, vol. 296, no. 2, pp. 295–326, 2003.

[15] M. Ionescu, G. Păun, and T. Yokomori, "Spiking neural P systems," *Fundamenta informaticae*, vol. 71, no. 2, 3, pp. 279–308, 2006.

[16] M. Gheorgue, F. Ipate, C. Dragomir, L. Mierla, L. Valencia Cabrera, M. García Quismondo, and M. d. J. Pérez Jiménez, "Kernel P systems-version 1," *Proceedings of the Eleventh Brainstorming Week on Membrane Computing, 97-124. Sevilla, ETS de Ingeniería Informática, 4-8 de Febrero, 2013,*, 2013.

[17] R. Nicolescu, F. Ipate, and H. Wu, "Programming P systems with complex objects," in *International Conference on Membrane Computing*. Springer, 2013, pp. 280–300.

[18] R. Nicolescu and A. Henderson, "An introduction to cP systems," in *Enjoying natural computing*. Springer, 2018, pp. 204–227.

[19] D. Díaz-Pernil, C. M. Fernández-Mírquez, M. García-Quismondo, M. A. Gutiérrez-Naranjo, and M. A. Martínez-del Amor, "Solving Sudoku with membrane computing," in *2010 IEEE Fifth International Conference on Bio-inspired Computing: Theories and Applications (BIC-TA)*. IEEE, 2010, pp. 610–615.

[20] D. Deodhare, S. Sonone, and A. Gupta, "A generic membrane computing-based Sudoku solver," in *2014 International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT)*. IEEE, 2014, pp. 89–99.

[21] G. Singh and K. Deep, "A new membrane algorithm using the rules of Particle Swarm Optimization incorporated within the framework of cell-like P-systems to solve Sudoku," *Applied Soft Computing*, vol. 45, pp. 27–39, 2016.

[22] Y. Liu, J. Sun, and J. S. Dong, "Pat 3: An extensible architecture for building multi-domain model checkers," in *2011 IEEE 22nd International Symposium on Software Reliability Engineering*. IEEE, 2011, pp. 190–199.

[23] Y. Liu, R. Nicolescu, and J. Sun, "Formal verification of cP systems using PAT3 and ProB," *Journal of Membrane Computing*, pp. 1–15, 2020.

[24] J. Cooper and R. Nicolescu, "The Hamiltonian cycle and travelling salesman problems in cP systems," *Fundamenta Informaticae*, vol. 164, no. 2-3, pp. 157–180, 2019.

[25] A. Henderson, R. Nicolescu, and M. J. Dinneen, "Solving a PSPACE-complete problem with cP systems," *Journal of Membrane Computing*, pp. 1–12, 2020.

[26] I. Pérez-Hurtado, D. Orellana-Martín, G. Zhang, and M. J. Pérez-Jiménez, "P-lingua in two steps: flexibility and efficiency," *Journal of Membrane Computing*, vol. 1, no. 2, pp. 93–102, 2019.

[27] G. Singh and K. Deep, "Cell-like P-systems using deterministic update rules to solve Sudoku," *International Journal of System Assurance Engineering and Management*, vol. 8, no. 2, pp. 857–866, 2017.

Yezhou Liu, Radu Nicolescu,
Jing Sun, Alec Henderson

Received October 22, 2020
Revised December 22, 2020

Yezhou Liu
Institution: The University of Auckland
Address: SCIENCE CENTRE 303S - Bldg 303S, Level 5, Room 596, 38 PRINCES ST, AUCKLAND, New Zealand 1010
E–mail: yliu442@aucklanduni.ac.nz

Radu Nicolescu
Institution: The University of Auckland
Address: SCIENCE CENTRE 303S - Bldg 303S, Level 5, Room 587, 38 PRINCES ST, AUCKLAND, New Zealand 1010
E–mail: r.nicolescu@auckland.ac.nz

Jing Sun
Institution: The University of Auckland
Address: SCIENCE CENTRE 303 - Bldg 303, Level 5, Room 522, 38 PRINCES ST, AUCKLAND, New Zealand 1010
E–mail: jing.sun@auckland.ac.nz

Alec Henderson
Institution: The University of Auckland
Address: SCIENCE CENTRE 303S - Bldg 303S, Level 5, Room 596, 38 PRINCES ST, AUCKLAND, New Zealand 1010
E–mail: ahen386@aucklanduni.ac.nz

Computer Science Journal of Moldova, vol.**29**, no.1(85), 2021

On a modified M/M/m/n queueing model *

Mario Lefebvre

Abstract

The classic M/M/m/n queueing model is modified by allowing a given task to require up to m servers to be performed. Moreover, the maximum time that a task can wait in the queue before being executed is a random variable having an exponential distribution. Both FIFO (First In, First Out) and priority disciplines are considered. The case when $m = 2$ is treated: the state space needed to fully describe the system is given, its size is calculated and the balance equations are presented when $n = 1, 2$ and 3. The queueing process can be used to model cluster-type multiprocessor computing systems.

Keywords: State space, balance equations, queue discipline, priority, cluster-type multiprocessor computing systems.

MSC 2010: 68M20, 60K25, 60K30, 90B22.

1 Introduction

We consider a system consisting of m (≥ 1) servers. For instance, it could be a computing system having m processors. Tasks arrive at the system according to a Poisson process with rate λ. For every server, the service time is an exponential random variable with mean μ. The times between the arrivals and the service times are assumed to be independent random variables. The capacity of the system is n (≥ 1). Finally, the queue discipline is FIFO (First In, First Out). Therefore, using Kendall's notation, the system can be denoted by M/M/m/n.

* This work was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC)

In a recent paper, Vardanyan and Sahakyan [5] (see also [4]) proposed the above model for cluster-type multiprocessor computing systems, with the following modifications: firstly, any task can require up to m servers to be executed. More precisely, the number of servers required to perform a task is a random variable K having a discrete uniform distribution over the set $\{1, 2, \ldots, m\}$. Secondly, the maximum time Ω that a task can wait in the queue before being executed is an exponentially distributed random variable with parameter ω.

The authors defined the states (i, j), where i is the number of tasks being serviced and j is the number of tasks waiting for service in the queue at a given time instant. They assumed that $0 \leq i \leq m$ and $0 \leq j \leq n$. There are $m \times (n+1) + 1$ possible states in the state space (since $i = 0$ implies that j is also equal to zero). Notice that n is taken to be the capacity of the *waiting queue*, rather than that of the system in their paper.

Remark 1.1. *(i) The authors of [5] actually wrote that there are* $m \times n + 1$ *possible states. But if* $m = n = 1$*, the possible states are* $(0, 0)$*,* $(1, 0)$ *and* $(1, 1)$*. Thus,* $3 = 1 \times (1 + 1) + 1$*.*
(ii) For the classic M/M/m/n model, $m = n$ *means that there is no waiting space, and the system is a so-called* loss system*. Here, because any task can require more than one server (if* $m > 1$*), the incoming tasks will only be lost (on arrival) when the system is in state* (m, m)*, so that all the* (m) *servers are servicing a different task. In particular, all the servers could be servicing the same task, and there could be* $m-1$ $(= n-1)$ *tasks waiting for service in the queue. We could also assume, as in [5], that* n *is the capacity of the waiting queue, so that the case* $n = 0$ *would indeed correspond to a loss system. Finally, note that there is no waiting if* $n < m$*.*

In [5], the authors computed the limiting probabilities of the system. However, their definition of the various states does not fully describe the different possible states of the system. Indeed, the value of i only gives us the *number of tasks* being serviced, so that the number of servers working on each task is unknown. With this important addi-

tional information, the number of possible states is much larger than $m \times (n+1)+1$.

In this note, we first generalise the model considered by Vardanyan and Sahakyan in [5]: the service time is an exponential random variable with mean μ_k, for $k = 1, 2, \ldots, m$. That is, the service time may depend on the server. Moreover, if two or more servers are working on a task, the service time may also be different. Next, instead of assuming that the random variable K has a discrete uniform distribution over the set $\{1, 2, \ldots, m\}$, we denote the probability $P[K = k]$ by $p_k \in [0, 1]$ for $k = 1, \ldots, m$. Finally, the maximum time Ω_k that a task requiring k servers can wait in the queue before being executed is an exponential random variable with parameter ω_k.

Remark 1.2. *(i) Contrary to the definition in [5], here the system capacity, n, is the maximum total number of tasks in the system at any one time; that is, the number of tasks being serviced or awaiting service. (ii) When a task requiring the two servers is completed and the first two tasks in the queue only require one server each, then they will both go into service at the same time.*

This problem was also considered by Green [1], but with different assumptions: firstly, $n = \infty$; secondly, there is no maximum time that a task can wait for service in the queue; thirdly, the mean service time is equal to $1/\mu$ for every server; finally, and more importantly, servers working together on a given task complete service at independent times, so that they do not end service together. It follows that the service times are not exponentially distributed random variables; instead, they are distributed as the maximum of a random number of exponentially distributed random variables. Consequently, the stochastic process $\{X(t), t \geq 0\}$, where $X(t)$ is the state of the system at time t, is not a continuous-time Markov chain.

In the next section, the case when $m = 2$ will be treated. We will give a state space that is exhaustive, as well as the full balance equations in simple cases. Then, the queue discipline will be modified: instead of the FIFO discipline, we will consider priority disciplines. Again, the balance equations will be presented in a simple case.

2 A generalised model

We consider the M/M/m/n queueing model with the modifications mentioned in the previous section. The tasks are generated by a Poisson process with rate λ, and the probability that a given task will require k servers to be performed is $P[K = k] = p_k \in [0, 1]$ for $k = 1, \ldots, m$, with $p_1 + \ldots + p_m = 1$. It follows that tasks requiring k servers arrive in the system according to a Poisson process with rate $\lambda_k := p_k \lambda$. We could instead assume that the various tasks are generated by m independent Poisson processes with rates $\lambda_1, \ldots, \lambda_m$.

Because (with the FIFO discipline) a task requiring more than one server can prevent a task behind it in the waiting queue to enter service, even if a server is free, we need to determine the type of each task in the system. Therefore, the number of possible states is very large. For this reason, we will limit ourselves in this note to the case when there are two servers. The results obtained below can be generalised to any value of m.

Thus, we assume that the queueing model is an M/M/2/n queue, where $n \geq 1$ is the capacity of the system. Moreover, an arriving task requires one server with probability $p_1 \in [0, 1]$ and two servers with probability $p_2 = 1 - p_1 \in [0, 1]$. The two servers do not necessarily execute the various tasks at the same rate: the service time for server i is an exponential random variable with parameter μ_i, for $i = 1, 2$. When the system is empty, an arriving (or waiting) task requiring only one server (which we call a type 1 task) will be executed by server 1 or 2 with respective probabilities $r_1 \in [0, 1]$ and $r_2 = 1 - r_1$. Furthermore, when a task requires the two servers, the service time becomes an exponential random variable with parameter μ_3. It is indeed more realistic to suppose that the service time depends both on the server and on the type of the task. Finally, tasks requiring i server(s) can spend an exponential amount of time with parameter ω_i, for $i = 1, 2$, before entering service. Note that we assume that once a task is taken into service, it will remain in the system until it has been fully completed.

Let us start with the case when $n = 1$. Then, as mentioned above, there is no waiting. Even if a server is free, if the other server is

occupied then an incoming task will be rejected. In this case, which is the simplest possible, there are four different states of the system:

0: the system is empty
1: server 1 is executing a task, server 2 is free
2: server 2 is executing a task, server 1 is free
3: servers 1 and 2 are executing a type 2 task

Let π_j denote the limiting probability that the system will be in state j, for $j = 0, 1, 2, 3$. Note that because the number of states is finite, the limiting probabilities do exist. The balance equations of the system (see Ross [3] or Lefebvre [2]) are the following:

State j	Departure rate from j	=	Arrival rate to j
0	$\lambda\pi_0$	=	$\mu_1\pi_1 + \mu_2\pi_2 + \mu_3\pi_3$
1	$\mu_1\pi_1$	=	$r_1 p_1 \lambda\pi_0$
2	$\mu_2\pi_2$	=	$r_2 p_1 \lambda\pi_0$
3	$\mu_3\pi_3$	=	$p_2\lambda\pi_0$

To obtain the limiting probabilities, we can solve the balance equations, together with the condition $\sum_{j=0}^{3}\pi_j = 1$. Notice that we have four equations in four unknowns, plus the preceding condition. Therefore, we can drop one of the balance equations. Once the limiting probabilities have been computed, we should check that they also satisfy the equation that was neglected. We easily find that

$$\begin{aligned}
\pi_0 &= \left\{1 + \frac{r_1 p_1 \lambda}{\mu_1} + \frac{r_2 p_1 \lambda}{\mu_2} + \frac{p_2\lambda}{\mu_3}\right\}^{-1}, \\
\pi_1 &= \frac{r_1 p_1 \lambda}{\mu_1}\pi_0, \quad \pi_2 = \frac{r_2 p_1 \lambda}{\mu_2}\pi_0 \quad \text{and} \quad \pi_3 = \frac{p_2\lambda}{\mu_3}\pi_0.
\end{aligned}$$

When $n = 2$, in addition to the above defined states (for which we

must now mention that the waiting queue is empty), we have:

4: servers 1 and 2 are executing type 1 tasks
5: server 1 is executing a task, server 2 is free, a type 2 task is waiting for service
6: server 2 is executing a task, server 1 is free, a type 2 task is waiting for service
7: servers 1 and 2 are executing a type 2 task, a type 1 task is waiting for service
8: servers 1 and 2 are executing a type 2 task, a type 2 task is waiting for service

Thus, there are now nine possible states. The balance equations of the system become

State j	Departure rate	=	Arrival rate
0	$\lambda\pi_0$	=	$\mu_1\pi_1 + \mu_2\pi_2 + \mu_3\pi_3$
1	$(\mu_1+\lambda)\pi_1$	=	$r_1 p_1 \lambda\pi_0 + \mu_2\pi_4 + \omega_2\pi_5 + \mu_3 r_1\pi_7$
2	$(\mu_2+\lambda)\pi_2$	=	$r_2 p_1 \lambda\pi_0 + \mu_1\pi_4 + \omega_2\pi_6 + \mu_3 r_2\pi_7$
3	$(\mu_3+\lambda)\pi_3$	=	$p_2\lambda\pi_0 + \mu_1\pi_5 + \mu_2\pi_6 + \omega_1\pi_7 + (\mu_3+\omega_2)\pi_8$
4	$(\mu_1+\mu_2)\pi_4$	=	$p_1\lambda(\pi_1+\pi_2)$
5	$(\mu_1+\omega_2)\pi_5$	=	$p_2\lambda\pi_1$
6	$(\mu_2+\omega_2)\pi_6$	=	$p_2\lambda\pi_2$
7	$(\mu_3+\omega_1)\pi_7$	=	$p_1\lambda\pi_3$
8	$(\mu_3+\omega_2)\pi_8$	=	$p_2\lambda\pi_3$

Next, when $n = 3$, we can add a waiting task to the states 4 to 8 of the system. This waiting task can be either a type 1 or type 2 task. It follows that the number of possible states increases to $9 + 5 \times 2 = 19$. In the general case $n \geq 1$, we have the following proposition.

Proposition 2.1. *The size of the state space of the modified M/M/2/n queueing model considered in this note is given by* $4 + 5(2^{n-1} - 1)$ *for* $n \geq 1$.

Proof. We can use mathematical induction. The result holds true for $n = 1$ (as well as $n = 2$ and $n = 3$). Assume that the formula is valid for a fixed $n \geq 2$. Then, for $n + 1$ the number of states is obtained by adding any waiting task to the states that correspond to a full system when the capacity is n. There are

$$[4 + 5(2^{n-1} - 1)] - [4 + 5(2^{n-2} - 1)] = 5 \times 2^{n-2}$$

such states. Hence, we add $5 \times 2^{n-1}$ states to the state space. It implies that the total number of states is

$$4 + 5(2^{n-1} - 1) + 5 \times 2^{n-1} = 4 + 5(2^n - 1),$$

which proves the result. □

Remark 2.1. *(i) If we assume that n is the capacity of the waiting queue, rather than that of the system, then we find that the number of possible states is $5+6(2^n-1)$ for $n \geq 1$, compared to $2(n+1)+1 = 2n+3$ for the problem considered in [5].*
(ii) Maple *is able to solve the nine balance equations of the system when $n = 2$. However, the solution it provides is very involved. In the particular case when $p_1 = 3/4$, $r_1 = 1/2$, $\lambda = 5$, $\mu_1 = 2$, $\mu_2 = 2$, $\mu_3 = 3$ and $\omega_1 = \omega_2 = 0$, the solution is*

$$\begin{aligned} &\pi_0 \simeq 0.1469, \quad \pi_1 \simeq 0.1277, \quad \pi_2 \simeq 0.1277, \\ &\pi_3 \simeq 0.0745, \quad \pi_4 \simeq 0.2394, \quad \pi_5 \simeq 0.0798, \\ &\pi_6 \simeq 0.0798, \quad \pi_7 \simeq 0.0931, \quad \pi_8 \simeq 0.0310. \end{aligned}$$

We see that, in this example, state 4 is by far the most likely state in stationary regime. Moreover, $\pi_1 = \pi_2$ and $\pi_5 = \pi_6$, which follows from the fact that $\mu_1 = \mu_2$ and $r_1 = 1/2$.

Now for $n \geq 3$, with the FIFO discipline, sometimes a server is free while a type 1 task is in the waiting queue, but behind a type 2 task that is blocking access to the free server. If we then allow the type 1

task to get serviced, it will reduce the average time that type 1 tasks spend in the system and conversely increase this average time for type 2 tasks. When m and n are large, a task requiring m servers could block the system for a long time. It is therefore worth examining the effect of giving priority to type 1 or to type 2 tasks in our case.

Let us first assume that type 2 tasks always have priority over type 1 tasks in the waiting queue. We can consider the general case when $n \geq 1$. We will calculate the number of states needed to fully describe the state of the system. Because our aim is to determine the effect of giving priority to a type of task, we could assume that $\mu_1 = \mu_2$.

If $n = 1$, priority does not make any difference, because there is no waiting. Similarly, if $n = 2$, then there is at most one task in the waiting queue, which again implies that priority is irrelevant. Let $n \geq 3$. There are five possible states when the waiting queue is empty, that we denote by $0, 1, 2, 3$ and 4 as defined above. Then, the other states of the system can be characterised by a triple (s, n_2, n_1), where $s \in \{1, 2, 3, 4\}$ and n_i is the number of type i tasks in the waiting queue, for $i = 1, 2$. We assume that a server can be free even if there is at least one type 1 task in the waiting queue. However, we cannot have triples of the form $(s, 0, n_1)$ with $s \in \{1, 2\}$ and $n_1 > 0$.

When $s = 1$ or 2, we can write that $n_2 \geq 1$ and $n_1 + n_2 \leq n - 1$. The number of states is given by

$$\underbrace{n-1}_{n_2=1} + \underbrace{n-2}_{n_2=2} + \cdots + \underbrace{1}_{n_2=n-1} = \frac{(n-1)n}{2}.$$

If $s = 3$, we have $n - 1$ additional states of the form $(3, 0, n_1)$, with $1 \leq n_1 \leq n - 1$, so that the number of states is

$$\frac{(n-1)n}{2} + (n-1) = (n-1)\frac{n+2}{2}.$$

Finally, the number of states of the form $(4, n_2, n_1)$ is obtained by replacing n by $n-1$ in the above formula: $(n-2)(n+1)/2$. Therefore, we can state the following proposition.

Proposition 2.2. *The size of the state space of the queueing model considered in this note, when type 2 tasks have priority over type 1*

tasks in the waiting queue, is given by

$$5 + 2\frac{(n-1)n}{2} + (n-1)\frac{n+2}{2} + (n-2)\frac{n+1}{2} = 2n^2 - n + 3$$

for $n \geq 3$.

Remark 2.2. *(i) The formula is actually valid for* $n \geq 1$.

(ii) If we assume that $\mu_1 = \mu_2$, *then there is no need to distinguish between* 1 *and* 2, *and between* $(1, n_2, n_1)$ *and* $(2, n_2, n_1)$. *The number of states is reduced to*

$$4 + \frac{(n-1)n}{2} + (n-1)\frac{n+2}{2} + (n-2)\frac{n+1}{2} = \frac{3n^2 - n + 4}{2}.$$

(iii) Compared with the FIFO discipline, the size of the state space is greatly reduced when n *is relatively large: for* $n = 3$, *it goes from 19 to 18, but from 639 to 123 if* $n = 8$.

(iv) When type 1 tasks have priority over type 2 tasks in the waiting queue, we have the states $0, 1, 2, 3$ *and* 4 *as above, and triples of the form* (s, n_1, n_2), *where* $s \in \{1, 2, 3, 4\}$. *We cannot have states of the form* $(1, n_1, n_2)$ *or* $(2, n_1, n_2)$ *with* $n_1 > 0$. *For* $s = 3$ *and* $s = 4$, *the number of different states is the same as when type 2 tasks have priority. It follows that the total number of states is*

$$5 + 2(n-1) + (n-1)\frac{n+2}{2} + (n-2)\frac{n+1}{2} = n^2 + 2n + 1$$

for $n \geq 3$ *(*$n \geq 1$, *actually). If* $\mu_1 = \mu_2$, *we can reduce the number of states to*

$$4 + (n-1) + (n-1)\frac{n+2}{2} + (n-2)\frac{n+1}{2} = n^2 + n + 1.$$

We give next the balance equations when $n = 3$ and $\mu_1 = \mu_2$, so that there are 14 states. If we assume further, for the sake of simplicity, that

$\omega_1 = \omega_2 = 0$, then the system of equations is obviously much simpler.

State	Departure rate	=	Arrival rate
0	$\lambda\pi_0$	=	$\mu_1\pi_1 + \mu_3\pi_3$
$1\,(=2)$	$(\lambda+\mu_1)\pi_1$	=	$p_1\lambda\pi_0 + 2\mu_1\pi_4 + \omega_2\pi_5 + \mu_3\pi_8$
3	$(\lambda+\mu_3)\pi_3$	=	$p_2\lambda\pi_0 + \mu_1\pi_5 + \omega_1\pi_8 + (\mu_3+\omega_2)\pi_9$
4	$(\lambda+2\mu_1)\pi_4$	=	$p_1\lambda\pi_1 + \omega_2(\pi_7+\pi_{14}) + \mu_3\pi_{11} + (2\mu_1+\omega_1)\pi_{13}$
$(1,1,0) := 5$	$(\lambda+\mu_1+\omega_2)\pi_5$	=	$p_2\lambda\pi_1 + 2\omega_2\pi_6 + \omega_1\pi_7 + 2\mu_1\pi_{14}$
$(1,2,0) := 6$	$(\mu_1+2\omega_2)\pi_6$	=	$p_2\lambda\pi_5$
$(1,1,1) := 7$	$(\mu_1+\omega_1+\omega_2)\pi_7$	=	$p_1\lambda\pi_5$
$(3,0,1) := 8$	$(\lambda+\mu_3+\omega_1)\pi_8$	=	$p_1\lambda\pi_3 + \mu_1\pi_7 + 2\omega_1\pi_{11} + (\mu_3+\omega_2)\pi_{12}$
$(3,1,0) := 9$	$(\lambda+\mu_3+\omega_2)\pi_9$	=	$p_2\lambda\pi_3 + \mu_1\pi_6 + (\mu_3+2\omega_2)\pi_{10} + \omega_1\pi_{12}$
$(3,2,0) := 10$	$(\mu_3+2\omega_2)\pi_{10}$	=	$p_2\lambda\pi_9$
$(3,0,2) := 11$	$(\mu_3+2\omega_1)\pi_{11}$	=	$p_1\lambda\pi_8$
$(3,1,1) := 12$	$(\mu_3+\omega_1+\omega_2)\pi_{12}$	=	$p_2\lambda\pi_8 + p_1\lambda\pi_9$
$(4,0,1) := 13$	$(2\mu_1+\omega_1)\pi_{13}$	=	$p_1\lambda\pi_4$
$(4,1,0) := 14$	$(2\mu_1+\omega_2)\pi_{14}$	=	$p_2\lambda\pi_4$

3 Concluding remarks

In this note, a modified M/M/m/n queueing model that can be used for cluster-type multiprocessor computing systems was considered. The main modification is that any task can require up to m servers to be executed. Moreover, each task can spend a random (exponentially distributed) maximum amount of time in the waiting queue before being taken into service. The default queue discipline is FIFO.

We saw that to fully describe the state of the system, we need a very large number of states. For this reason, we limited ourselves to the case

when $m = 2$. There are then two types of tasks: type 1 (respectively 2) tasks require only one (respectively two) servers. If priority in the waiting queue is given to either type 1 or type 2 tasks, then the size of the state space is greatly reduced, at least when n is relatively large. We presented the detailed balance equations in simple cases.

In any particular case, if n is small enough, we can use a mathematical software such as *Maple* to solve the balance equations, which are a system of linear equations. For n relatively large, if the number of states is too large to obtain precise numerical solutions, then one could perhaps use simulation to estimate the various limiting probabilities.

Finally, an important question is whether one should use the FIFO discipline or a priority discipline, particularly because in the general case, with m large enough, a task requiring the m servers could block the system for a large amount of time. The answer to this question depends on the aim of the system administrators. Do they prioritise the average number of tasks being serviced, or the average time spent in the system for each (or a particular) type of task? Another important criterion is the percentage of rejected tasks.

Acknowledgements

The author wishes to thank the anonymous reviewer of this note for his/her constructive comments.

References

[1] L. Green, "A queueing system in which customers require a random number of servers," *Operations Research*, vol. 28, no. 6, pp. 1335–1346, 1980.

[2] M. Lefebvre, *Applied Stochastic Processes,* New York, NY: Springer, 2007, 382 p.

[3] S. M. Ross, *Introduction to Probability Models,* 12th ed., Amsterdam: Elsevier/Academic Press, 2019, 826 p.

[4] V. G. Sahakyan, Y. H. Shoukourian, and H. V. Astsatryan, "About some queueing models for computational grid systems," *Mathematical Problems of Computer Science*, vol. 46, pp. 55–58, 2016.

[5] A. P. Vardanyan and V. G. Sahakyan, "The queue distribution in multiprocessor systems with the waiting time restriction," *Mathematical Problems of Computer Science*, vol. 51, pp. 82–89, 2019.

Mario Lefebvre

Received November 25, 2020
Accepted March 11, 2021

Polytechnique Montréal
Department of Mathematics and Industrial Engineering
2500, chemin de Polytechnique
Montréal (Québec), Canada H3T 1J4
E–mail: mlefebvre@polymtl.ca

Computer Science Journal of Moldova, vol.**29**, no.1(85), 2021

Properties of Finitely Supported Self - Mappings on the Finite Powerset of Atoms

Andrei Alexandru

Abstract

The theory of finitely supported algebraic structures represents a reformulation of Zermelo-Fraenkel set theory in which every classical structure is replaced by a finitely supported structure according to the action of a group of permutations of some basic elements named atoms. It provides a way of representing infinite structures in a discrete manner, by employing only finitely many characteristics. In this paper we present some (finiteness and fixed point) properties of finitely supported self-mappings defined on the finite power set of atoms.

Keywords: finitely supported structures, atoms, finite powerset, injectivity, surjectivity, fixed points.

MSC 2010: 03E30, 03E25, 03B70.

1 Introduction

Finitely Supported Mathematics (FSM) is a general name for the theory of finitely supported sets equipped with finitely supported internal operations or with finitely supported relations [2]. Finitely supported sets are related to the recent development of the Fraenkel-Mostowski axiomatic set theory, to the theory of admissible sets of Barwise (particularly by generalizing the theory of hereditary finite sets) and to the theory of nominal sets. Fraenkel-Mostowski set theory (FM) represents an axiomatization of the Fraenkel Basic Model of the Zermelo-Fraenkel set theory with atoms (ZFA); its axioms are the ZFA axioms together with an axiom of finite support claiming that any set-theoretical construction has to be finitely supported modulo a canonical hierarchically

defined permutation action. An alternative approach for FM set theory that works in the classical Zermelo-Fraenkel (ZF) set theory (i.e. without being necessary to consider an alternative set theory obtained by weakening the ZF axiom of extensionality) is related to the theory of nominal sets that are defined as usual ZF sets equipped with canonical permutation actions of the group of all one-to-one and onto transformations of a fixed infinite, countable ZF set formed by basic elements (i.e. by elements whose internal structure is not taken into consideration, called 'atoms') satisfying a finite support requirement (meaning that 'for every element x in a nominal set there should exist a finite subset of basic elements S such that any one-to-one and onto transformation of basic elements that fixes S pointwise also leaves x invariant under the effect of the permutation action with who the nominal set is equipped').

Nominal sets [5] are related to binding, freshness and renaming in the computation of infinite structures containing enough symmetries such that they can be concisely manipulated. Ignoring the requirement regarding the countability of A in the definition of a nominal set, and motivated by Tarski's approach regarding logicality (a logical notion is defined by Tarski as one that is invariant under the one-to-one transformations of the universe of discourse onto itself), we introduce *invariant sets*. A finitely supported set is defined as a finitely supported element in the power set of an invariant set. Equipping finitely supported sets with finitely supported mappings and relations, we get finitely supported algebraic structures that form FSM.

In this paper we collect specific properties of finitely supported mappings defined of the finite power set of atoms [2]–[4] and we present some other new properties. We are particularly focused on proving the equivalence between injectivity and surjectivity for such mappings, together with some fixed point properties. Therefore, although the finite power set of atoms is infinite, it has some finiteness properties. Furthermore, although the finite power set of atoms is not a complete lattice in FSM, some fixed points of Tarski type hold. Particularly, finitely supported self-mappings defined on the finite powerset of atoms have infinitely many fixed points if they satisfy some properties (such

as strict monotony, injectivity or surjectivity).

2 Preliminary Results

A finite set (without other specification) is referred to a set that can be represented as $\{x_1, \ldots, x_n\}$ for some $n \in \mathbb{N}$. An infinite set (without other specification) means "a set which is not finite". We consider a fixed infinite ZF set A (called 'the set of atoms' by analogy with ZFA set theory; however, despite classical set theory with atoms, we do not need to modify the axiom of extensionality in order to define A). The atoms are entities whose internal structure is ignored and which are considered as basic for a higher-order construction. This means atoms can be checked only for equality.

A *transposition* is a function $(a\,b) : A \to A$ that interchanges only a and b. A *permutation* of A in FSM is a bijection of A generated by composing finitely many transpositions. We denote by S_A the group of all permutations of A. According to Proposition 2.11 and Remark 2.2 in [2], an arbitrary bijection on A is finitely supported if and only if it is a permutation of A.

Definition 1.

1. *Let X be a ZF set. An* S_A-action *on X is a group action $\cdot$ of S_A on X. An* S_A-set *is a pair $(X, \cdot)$, where X is a ZF set, and $\cdot$ is an S_A-action on X.*

2. *Let $(X, \cdot)$ be an S_A-set. We say that $S \subset A$* supports *x whenever for each $\pi \in Fix(S)$ we have $\pi \cdot x = x$, where $Fix(S) = \{\pi \mid \pi(a) = a, \forall a \in S\}$. The least finite set (w.r.t. the inclusion relation) supporting x (which exists according to [2]) is called* the support of *x and is denoted by $supp(x)$. An empty supported element is called* equivariant.

3. *Let $(X, \cdot)$ be an S_A-set. We say that X is an* invariant set *if for each $x \in X$ there exists a finite set $S_x \subset A$ which supports x.*

Proposition 1. *[2], [5] Let $(X, \cdot)$ and $(Y, \diamond)$ be S_A-sets.*

1. *The set A of atoms is an invariant set with the S_A-action $\cdot : S_A \times A \to A$ defined by $\pi \cdot a := \pi(a)$ for all $\pi \in S_A$ and $a \in A$. Furthermore, $supp(a) = \{a\}$ for each $a \in A$.*

2. *Let $\pi \in S_A$. If $x \in X$ is finitely supported, then $\pi \cdot x$ is finitely supported and $supp(\pi \cdot x) = \{\pi(u) \,|\, u \in supp(x)\} := \pi(supp(x))$.*

3. *The Cartesian product $X \times Y$ is also an S_A-set with the S_A-action $\otimes : S_A \times (X \times Y) \to (X \times Y)$ defined by $\pi \otimes (x, y) = (\pi \cdot x, \pi \diamond y)$ for all $\pi \in S_A$ and all $x \in X$, $y \in Y$. If $(X, \cdot)$ and $(Y, \diamond)$ are invariant sets, then $(X \times Y, \otimes)$ is also an invariant set.*

4. *The powerset $\wp(X) = \{Z \,|\, Z \subseteq X\}$ is also an S_A-set with the S_A-action $\star : S_A \times \wp(X) \to \wp(X)$ defined by $\pi \star Z := \{\pi \cdot z \,|\, z \in Z\}$ for all $\pi \in S_A$, and all $Z \subseteq X$. For each invariant set $(X, \cdot)$, we denote by $\wp_{fs}(X)$ the set of elements in $\wp(X)$ which are finitely supported according to the action $\star$. $(\wp_{fs}(X), \star|_{\wp_{fs}(X)})$ is an invariant set.*

5. *The finite powerset of X denoted by $\wp_{fin}(X) = \{Y \subseteq X \,|\, Y finite\}$ and the cofinite powerset of X denoted by $\wp_{cofin}(X) = \{Y \subseteq X \,|\, X \setminus Y finite\}$ are both S_A-sets with the S_A-action $\star$ defined as in the previous item. If X is an invariant set, then both $\wp_{fin}(X)$ and $\wp_{cofin}(X)$ are invariant sets.*

6. *We have $\wp_{fs}(A) = \wp_{fin}(A) \cup \wp_{cofin}(A)$. If $X \in \wp_{fin}(A)$, then $supp(X) = X$. If $X \in \wp_{cofin}(A)$, then $supp(X) = A \setminus X$.*

7. *Any ordinary (non-atomic) ZF-set X (such as $\mathbb{N}, \mathbb{Z}, \mathbb{Q}$ or $\mathbb{R}$ for example) is an invariant set with the single possible S_A-action $\cdot : S_A \times X \to X$ defined by $\pi \cdot x := x$ for all $\pi \in S_A$ and $x \in X$.*

Definition 2. *Let $(X, \cdot)$ be an S_A-set. A subset Z of X is called* finitely supported *if and only if $Z \in \wp_{fs}(X)$. A subset Z of X is* uniformly supported *if all the elements of Z are supported by the same set S (and so Z is itself supported by S).*

From Definition 1, a subset Z of an invariant set $(X, \cdot)$ is finitely supported by a set $S \subseteq A$ if and only if $\pi \star Z \subseteq Z$ for all $\pi \in Fix(S)$, i.e. if and only if $\pi \cdot z \in Z$ for all $\pi \in S_A$ and all $z \in Z$. This is because any permutation of atoms should have finite order, and so the relation $\pi \star Z \subseteq Z$ is equivalent to $\pi \star Z = Z$.

Proposition 2. *[2] Let X be a uniformly supported (particularly, a finite) subset of an invariant set $(U, \cdot)$. Then X is finitely supported and $supp(X) = \cup\{supp(x) \,|\, x \in X\}$.*

Definition 3. *Let X and Y be invariant sets.*

1. *A function $f : X \to Y$ is* finitely supported *if $f \in \wp_{fs}(X \times Y)$.*
2. *Let Z be a finitely supported subset of X and T a finitely supported subset of Y. A function $f : Z \to T$ is* finitely supported *if $f \in \wp_{fs}(X \times Y)$. The set of all finitely supported functions from Z to T is denoted by T^Z_{fs}.*

Proposition 3. *[2], [5] Let $(X, \cdot)$ and $(Y, \diamond)$ be two invariant sets.*

1. *Y^X (i.e. the set of all functions from X to Y) is an S_A-set with the S_A-action $\widetilde{\star} : S_A \times Y^X \to Y^X$ defined by $(\pi \widetilde{\star} f)(x) = \pi \diamond (f(\pi^{-1} \cdot x))$ for all $\pi \in S_A$, $f \in Y^X$ and $x \in X$. A function $f : X \to Y$ is finitely supported (in the sense of Definition 3) if and only if it is finitely supported with respect the permutation action $\widetilde{\star}$.*
2. *Let Z be a finitely supported subset of X and T a finitely supported subset of Y. A function $f : Z \to T$ is supported by a finite set $S \subseteq A$ if and only if for all $x \in Z$ and all $\pi \in Fix(S)$ we have $\pi \cdot x \in Z$, $\pi \diamond f(x) \in T$ and $f(\pi \cdot x) = \pi \diamond f(x)$.*

3 Finitely Supported Self-Mappings on the Finite Powerset of A

This section collects surprising finiteness and fixed point properties of finitely supported self mappings defined on $\wp_{fin}(A)$. We involve specific

FSM proving techniques, especially properties of uniformly supported sets. Details regarding these aspects can be found in [2]–[4].

Theorem 1. *A finitely supported function $f : \wp_{fin}(A) \to \wp_{fin}(A)$ is injective if and only if it is surjective.*

Proof. 1. For proving the direct implication, assume, by contradiction, that $f : \wp_{fin}(A) \to \wp_{fin}(A)$ is a finitely supported injection having the property that $Im(f) \subsetneq \wp_{fin}(A)$. This means that there exists $X_0 \in \wp_{fin}(A)$ such that $X_0 \notin Im(f)$. We can construct a sequence of elements from $\wp_{fin}(A)$ which has the first term X_0 and the general term $X_{n+1} = f(X_n)$ for all $n \in \mathbb{N}$. Since $X_0 \notin Im(f)$, it follows that $X_0 \neq f(X_0)$. Since f is injective and $X_0 \notin Im(f)$, according to the injectivity of f we obtain that $f^n(X_0) \neq f^m(X_0)$ for all $n, m \in \mathbb{N}$ with $n \neq m$. Furthermore, X_{n+1} is supported by $supp(f) \cup supp(X_n)$ for all $n \in \mathbb{N}$. Indeed, let $\pi \in Fix(supp(f) \cup supp(X_n))$. According to Proposition 3, $\pi \star X_{n+1} = \pi \star f(X_n) = f(\pi \star X_n) = f(X_n) = X_{n+1}$. Since $supp(X_{n+1})$ is the least set supporting X_{n+1}, we obtain $supp(X_{n+1}) \subseteq supp(f) \cup supp(X_n)$ for all $n \in \mathbb{N}$. By induction on n, we have $supp(X_n) \subseteq supp(f) \cup supp(X_0)$ for all $n \in \mathbb{N}$. Thus, all X_n are supported by the same set of atoms $S = supp(f) \cup supp(X_0)$, which means the family $(X_n)_{n\in\mathbb{N}}$ is infinite and uniformly supported, contradicting the fact that $\wp_{fin}(A)$ has only finitely many elements supported by S, namely the subsets of S.

2. In order to prove the reverse implication, let us consider a finitely supported surjection $f : \wp_{fin}(A) \to \wp_{fin}(A)$. Let $X \in \wp_{fin}(A)$. Then $supp(X) = X$ and $supp(f(X)) = f(X)$ according to Proposition 2. Since $supp(f)$ supports f and $supp(X)$ supports X, for any π fixing pointwise $supp(f) \cup supp(X) = supp(f) \cup X$ we have $\pi \star f(X) = f(\pi \star X) = f(X)$ which means $supp(f) \cup X$ supports $f(X)$, that is $f(X) = supp(f(X)) \subseteq supp(f) \cup X$ (1).

For a fixed $m \geq 1$, let us fix m (arbitrarily considered) atoms $b_1, \ldots, b_m \in A \backslash supp(f)$. Let $\mathcal{F} = \{\{a_1, \ldots, a_n, b_1, \ldots, b_m\} \mid a_1, \ldots, a_n \in supp(f), n \geq 1\} \cup \{\{b_1, \ldots, b_m\}\}$. The set $\mathcal{F}$ is finite since $supp(f)$ is finite and the elements $b_1, \ldots, b_m \in A \backslash supp(f)$ are fixed. Let us consider an arbitrary $Y \in \mathcal{F}$, that is $Y \setminus supp(f) = \{b_1, \ldots, b_m\}$. There exists

$Z \in \wp_{fin}(A)$ such that $f(Z) = Y$. According to (1), Z must be either of form $Z = \{c_1, \ldots, c_k, b_{i_1}, \ldots, b_{i_l}\}$ with $c_1, \ldots, c_k \in supp(f)$ and $b_{i_1}, \ldots, b_{i_l} \in A \setminus supp(f)$ or of form $Z = \{b_{i_1}, \ldots, b_{i_l}\}$ with $b_{i_1}, \ldots, b_{i_l} \in A \setminus supp(f)$. In both cases we have $\{b_1, \ldots, b_m\} \subseteq \{b_{i_1}, \ldots, b_{i_l}\}$. We should prove that $l = m$ and hence the above sets are equal. Assume, by contradiction, that there exists b_{i_j} with $j \in \{1, \ldots, l\}$ such that $b_{i_j} \notin \{b_1, \ldots, b_m\}$. Then $(b_{i_j}\ b_1) \star Z = Z$ since both $b_{i_j}, b_1 \in Z$ and Z is a finite subset of A (b_{i_j} and b_1 are interchanged in Z under the effect of the transposition $(b_{i_j}\ b_1)$, while the other atoms belonging to Z are left unchanged, meaning that the entire Z is left invariant under the action $\star$). Furthermore, since $b_{i_j}, b_1 \notin supp(f)$, we have that the transposition $(b_{i_j}\ b_1)$ fixes $supp(f)$ pointwise, and, because $supp(f)$ supports f, from Proposition 3, we get $f(Z) = f((b_{i_j}\ b_1) \star Z) = (b_{i_j}\ b_1) \star f(Z)$, which is a contradiction, because $b_1 \in f(Z)$, while $b_{i_j} \notin f(Z)$. Thus, $\{b_{i_1}, \ldots, b_{i_l}\} = \{b_1, \ldots, b_m\}$, and so $Z \in \mathcal{F}$. Therefore, $\mathcal{F} \subseteq f(\mathcal{F})$, which means $|\mathcal{F}| \leq |f(\mathcal{F})|$. However, because f is a function and $\mathcal{F}$ is a finite set, we obtain $|f(\mathcal{F})| \leq |\mathcal{F}|$. We finally get $|\mathcal{F}| = |f(\mathcal{F})|$ and, because $\mathcal{F}$ is finite with $\mathcal{F} \subseteq f(\mathcal{F})$, we obtain $\mathcal{F} = f(\mathcal{F})$ (2), which means that $f|_{\mathcal{F}} : \mathcal{F} \to \mathcal{F}$ is surjective. Since $\mathcal{F}$ is finite, $f|_{\mathcal{F}}$ should be injective, i.e. $f(F_1) \neq f(F_2)$ whenever $F_1, F_2 \in \mathcal{F}$ with $F_1 \neq F_2$ (3).

Whenever $d_1, \ldots, d_u \in A \setminus supp(f)$ with $\{d_1, \ldots, d_u\} \neq \{b_1, \ldots, b_m\}$, $u \geq 1$, and considering $\mathcal{U} = \{\{a_1, \ldots, a_n, d_1, \ldots, d_u\} \,|\, a_1, \ldots, a_n \in supp(f), n \geq 1\} \cup \{\{d_1, \ldots, d_u\}\}$, we conclude that $\mathcal{F}$ and $\mathcal{U}$ are disjoint. Whenever $F_1 \in \mathcal{F}$ and $U_1 \in \mathcal{U}$, we have $f(F_1) \in \mathcal{F}$ and $f(U_1) \in \mathcal{U}$ by using the same arguments used to prove (2), and so $f(F_1) \neq f(U_1)$ (4). If $\mathcal{T} = \{\{a_1, \ldots, a_n\} \,|\, a_1, \ldots, a_n \in supp(f)\}$ and $Y \in \mathcal{T}$, then there is $T' \in \wp_{fin}(A)$ such that $Y = f(T')$. Similarly as in (2), we should have $T' \in \mathcal{T}$. Otherwise, if T' belonged to some $\mathcal{U}$ considered above, i.e. if T' contained an element outside $supp(f)$, we would get the contradiction $Y = f(T') \in \mathcal{U}$. Hence $\mathcal{T} \subseteq f(\mathcal{T})$ from which $\mathcal{T} = f(\mathcal{T})$ since $\mathcal{T}$ is finite (using similar arguments as those involved to prove (3) from $\mathcal{F} \subseteq f(\mathcal{F})$). Thus, $f|_{\mathcal{T}} : \mathcal{T} \to \mathcal{T}$ is surjective. Since $\mathcal{T}$ is finite, $f|_{\mathcal{T}}$ should be also injective, namely $f(T_1) \neq f(T_2)$ whenever $T_1, T_2 \in \mathcal{T}$ with $T_1 \neq T_2$ (5). The case $supp(f) = \emptyset$ is contained in the above analysis; it leads to $f(\emptyset) = \emptyset$ and $f(X) = X$ for all $X \in \wp_{fin}(A)$. We

also have $f(T_1) \neq f(U_1)$ whenever $T_1 \in \mathcal{T}$ and $U_1 \in \mathcal{U}$ since $f(T_1) \in \mathcal{T}$, $f(U_1) \in \mathcal{U}$ and $\mathcal{T}$ and $\mathcal{U}$ are disjoint (6). Since $b_1, \ldots, b_m$ and $d_1, \ldots, d_u$ were arbitrarily chosen from $A \setminus supp(f)$, the injectivity of f leads from the claims (3), (4), (5) and (6) covering all the possible cases for two different finite subsets of atoms and comparison of the values of f over the related subsets of atoms. □

Proposition 4. *Let $f : \wp_{fin}(A) \to \wp_{fin}(A)$ be finitely supported and injective. For each $X \in \wp_{fin}(A)$ we have $X \setminus supp(f) \neq \emptyset$ if and only if $f(X) \setminus supp(f) \neq \emptyset$. Furthermore, $X \setminus supp(f) = f(X) \setminus supp(f)$. Moreover, if f is monotone (i.e. order preserving), then $X \setminus supp(f) = f(X \setminus supp(f))$ for all $X \in \wp_{fin}(A)$, and $f(supp(f)) = supp(f)$.*

Proof. Let us consider $Y \in \wp_{fin}(A)$. Then we have $supp(Y) = Y$. According to Proposition 3, for any permutation $\pi \in Fix(supp(f) \cup supp(Y)) = Fix(supp(f) \cup Y)$ we have $\pi \star f(Y) = f(\pi \star Y) = f(Y)$ meaning that $supp(f) \cup Y$ supports $f(Y)$, that is $f(Y) = supp(f(Y)) \subseteq supp(f) \cup Y$ (1). If $Y \subseteq supp(f)$, we have $f(Y) \subseteq supp(f)$ (2). Let $X \in \wp_{fin}(X)$ with $X \subseteq supp(f)$. From (2) we get $f(X) \subseteq supp(f)$. Conversely, assume $f(X) \subseteq supp(f)$. By successively applying (2), we obtain $f^n(X) \subseteq supp(f)$ for all $n \in \mathbb{N}^*$ (3). Since $supp(f)$ is finite, there should exist $l, m \in \mathbb{N}^*$ with $l \neq m$ such that $f^l(X) = f^m(X)$. Assume $l > m$. Since f is injective, we obtain $f^{l-m}(X) = X$, and so by (3) we conclude that $X \subseteq supp(f)$. Therefore, $X \subseteq supp(f)$ if and only if $f(X) \subseteq supp(f)$, and hence $X \setminus supp(f) \neq \emptyset$ if and only if $f(X) \setminus supp(f) \neq \emptyset$.

Let $T \in \wp_{fin}(A)$ such that $f(T) \setminus supp(f) \neq \emptyset$ or, equivalently, $T \setminus supp(f) \neq \emptyset$. Thus, T should have the form $T = \{a_1, \ldots, a_n, b_1, \ldots, b_m\}$ with $a_1, \ldots, a_n \in supp(f)$ and $b_1, \ldots, b_m \in A \setminus supp(f)$, $m \geq 1$, or the form $T = \{b_1, \ldots, b_m\}$ with $b_1, \ldots, b_m \in A \setminus supp(f)$, $m \geq 1$. According to (1), we should have $f(T) = \{c_1, \ldots, c_k, b_{i_1}, \ldots, b_{i_l}\}$ with $c_1, \ldots, c_k \in supp(f)$ and $b_{i_1}, \ldots, b_{i_l} \in A \setminus supp(f)$, or $f(T) = \{b_{i_1}, \ldots, b_{i_l}\}$ with $b_{i_1}, \ldots, b_{i_l} \in A \setminus supp(f)$, having in any case the property that $\{b_{i_1}, \ldots, b_{i_l}\}$ is non-empty (i.e. it should contain at least one element, say b_{i_1}) and $\{b_{i_1}, \ldots, b_{i_l}\} \subseteq \{b_1, \ldots, b_m\}$. If $m = 1$, then $l = 1$, $b_{i_1} = b_1$, and we are done, so let $m > 1$.

Assume by contradiction that there exists $j \in \{1, \ldots, m\}$ such that $b_j \notin \{b_{i_1}, \ldots, b_{i_l}\}$. Then $(b_{i_1}\ b_j) \star T = T$ since both $b_{i_1}, b_j \in T$ and T is a finite subset of atoms (b_{i_1} and b_j are interchanged in T under the effect of the transposition $(b_{i_1}\ b_j)$, but the whole T is left invariant). Furthermore, since $b_{i_1}, b_j \notin supp(f)$ we have that the transposition $(b_{i_1}\ b_j)$ fixes $supp(f)$ pointwise, and hence by Proposition 3 we obtain $f(T) = f((b_{i_1}\ b_j) \star T) = (b_{i_1}\ b_j) \star f(T)$ which is a contradiction because $b_{i_1} \in f(T)$ while $b_j \notin f(T)$. Thus, $\{b_{i_1}, \ldots, b_{i_l}\} = \{b_1, \ldots, b_m\}$, and so $T \setminus supp(f) = f(T) \setminus supp(f)$.

Assume now that f is monotone. Let us fix $X \in \wp_{fin}(A)$, and consider the case $X \setminus supp(f) \neq \emptyset$, that is $X = \{a_1, \ldots, a_n, b_1, \ldots, b_m\}$ with $a_1, \ldots, a_n \in supp(f)$ and $b_1, \ldots, b_m \in A \setminus supp(f)$, $m \geq 1$, or $X = \{b_1, \ldots, b_m\}$ with $b_1, \ldots, b_m \in A \setminus supp(f)$, $m \geq 1$. Therefore we get $X \setminus supp(f) = \{b_1, \ldots, b_m\}$, and by involving the above arguments, we should have $f(X \setminus supp(f)) = \{x_1, \ldots, x_i, b_1, \ldots, b_m\}$ with $x_1, \ldots, x_i \in supp(f)$ or $f(X \setminus supp(f)) = \{b_1, \ldots, b_m\}$. In either case we obtain $X \setminus supp(f) \subseteq f(X \setminus supp(f))$, and since f is monotone we construct an ascending chain $X \setminus supp(f) \subseteq f(X \setminus supp(f)) \subseteq \ldots \subseteq f^k(X \setminus supp(f)) \subseteq \ldots$. Since for any $k \in \mathbb{N}$ we have that $f^k(X \setminus supp(f))$ is supported by $supp(f) \cup supp(X \setminus supp(f)) = supp(f) \cup supp(X)$ and $\wp_{fin}(A)$ does not contain an infinite uniformly supported subset, the related chain should be stationary, that is there exists $n \in \mathbb{N}$ such that $f^n(X \setminus supp(f)) = f^{n+1}(X \setminus supp(f))$, which, according to the injectivity of f, leads to $X \setminus supp(f) = f(X \setminus supp(f))$.

It remains to analyze the case $X \subseteq supp(f)$ or, equivalently, $X \setminus supp(f) = \emptyset$. We have $f(\emptyset) \subseteq supp(f)$. In the finite set $supp(f)$ we can define the chain of subsets $\emptyset \subseteq f(\emptyset) \subseteq f^2(\emptyset) \subseteq \ldots \subseteq f^m(\emptyset) \subseteq \ldots$ which is uniformly supported by $supp(f)$. Therefore the related chain should be stationary, meaning that there should exist $k \in \mathbb{N}$ such that $f^k(\emptyset) = f^{k+1}(\emptyset)$. According to the injectivity of f, we get $X \setminus supp(f) = \emptyset = f(\emptyset) = f(X \setminus supp(f))$.

According to (2), we have $f(supp(f)) \subseteq supp(f)$, and because f preserves the inclusion relation, we construct in $supp(f)$ the chain $\ldots \subseteq f^m(supp(f)) \subseteq \ldots \subseteq f(supp(f)) \subseteq supp(f)$. Since $supp(f)$ is finite, the chain should be stationary, and so $f^{k+1}(supp(f)) = f^k(supp(f))$

for some positive integer k, which, because f is injective, conduces to $f(supp(f)) = supp(f)$. □

Remark 1. *From the proof of Proposition 4, if $f : \wp_{fin}(A) \to \wp_{fin}(A)$ is finitely supported (even if it is not injective) with $X \subseteq supp(f)$, we have $f(X) \subseteq supp(f)$. If $f(X) \setminus supp(f) \neq \emptyset$, then $X \setminus supp(f) = f(X) \setminus supp(f)$.*

Corollary 1. *Let $f : \wp_{fin}(A) \to \wp_{fin}(A)$ be finitely supported and surjective. Then for each $X \in \wp_{fin}(A)$ we have $X \setminus supp(f) \neq \emptyset$ if and only if $f(X) \setminus supp(f) \neq \emptyset$. In either of these cases $X \setminus supp(f) = f(X) \setminus supp(f)$. If, furthermore, f is monotone, then $X \setminus supp(f) = f(X \setminus supp(f))$ for all $X \in \wp_{fin}(A)$, and $f(supp(f)) = supp(f)$.*

Proof. From Theorem 1, a finitely supported surjective function $f : \wp_{fin}(A) \to \wp_{fin}(A)$ should be injective. The result now follows from Proposition 4. □

Theorem 2. *Let $f : \wp_{fin}(A) \to \wp_{fin}(A)$ be finitely supported and strictly monotone (i.e. f has the property that $X \subsetneq Y$ implies $f(X) \subsetneq f(Y)$). Then we have $X \setminus supp(f) = f(X \setminus supp(f))$ for all $X \in \wp_{fin}(A)$.*

Proof. Let $X \in \wp_{fin}(A)$. According to Proposition 2, we have $supp(X) = X$ and $supp(f(X)) = f(X)$. According to Proposition 3, for any permutation $\pi \in Fix(supp(f) \cup supp(X)) = Fix(supp(f) \cup X)$ we get $\pi \star f(X) = f(\pi \star X) = f(X)$ meaning that $supp(f) \cup X$ supports $f(X)$, that is $f(X) = supp(f(X)) \subseteq supp(f) \cup X$ (1).

If $supp(f) = \emptyset$, we obtain $f(X) \subseteq X$ for all $X \in \wp_{fin}(A)$. If there exists $Y \in \wp_{fin}(A)$ with $f(Y) \subsetneq Y$, then we can construct the sequence $\ldots \subsetneq f^k(Y) \subsetneq \ldots \subsetneq f^2(Y) \subsetneq f(Y) \subsetneq Y$ which is infinite and uniformly supported by $supp(Y) \cup supp(f)$. This is a contradiction because the finite set Y cannot contain infinitely many distinct subsets, and so $f(X) = X$ for all $X \in \wp_{fin}(A)$.

Assume now that $supp(f)$ is non-empty. If $X \subseteq supp(f)$, then $f(X \setminus supp(f)) = f(\emptyset) = \emptyset = X \setminus supp(f)$. The second identity follows because f is strictly monotone; otherwise we could construct an infinite

strictly ascending chain in $\wp_{fin}(A)$, uniformly supported by $supp(f)$, namely $\emptyset \subsetneq f(\emptyset) \subsetneq \ldots \subsetneq f^k(\emptyset) \subsetneq \ldots$, contradicting the fact that $\wp_{fin}(A)$ does not contain an infinite uniformly supported subset.

Now we prove the following intermediate result. Let us consider an arbitrary set $T = \{b_1, \ldots, b_n\}$ such that $b_1, \ldots, b_n \in A \setminus supp(f)$, $n \geq 1$ and $f(T) \setminus supp(f) \neq \emptyset$. We prove that $f(T) = T$ (2). According to (1), $f(T)$ should be $f(T) = \{c_1, \ldots, c_k, b_{i_1}, \ldots, b_{i_l}\}$ with $c_1, \ldots, c_k \in supp(f)$ and $b_{i_1}, \ldots, b_{i_l} \in A \setminus supp(f)$, or $f(T) = \{b_{i_1}, \ldots, b_{i_l}\}$ with $b_{i_1}, \ldots, b_{i_l} \in A \setminus supp(f)$. In both cases we have that $\{b_{i_1}, \ldots, b_{i_l}\}$ is non-empty (i.e. it should contain at least one element, say b_{i_1}, because we assumed that $f(T)$ contains at least one element outside $supp(f)$) and $\{b_{i_1}, \ldots, b_{i_l}\} \subseteq \{b_1, \ldots, b_n\}$. If $n = 1$, then $l = 1$ and $b_{i_1} = b_1$. Now let us consider $n > 1$. Assume by contradiction that there is $j \in \{1, \ldots, n\}$ such that $b_j \notin \{b_{i_1}, \ldots, b_{i_l}\}$. Then $(b_{i_1}\ b_j) \star T = T$ since both $b_{i_1}, b_j \in T$ and T is a finite subset of atoms (b_{i_1} and b_j are interchanged in T under the effect of the transposition $(b_{i_1}\ b_j)$, while the other atoms belonging to T are left unchanged, which means the entire T is left invariant under the effect of the related transposition under the induced action $\star$). Furthermore, since $b_{i_1}, b_j \notin supp(f)$ we have the transposition $(b_{i_1}\ b_j)$ fixes $supp(f)$ pointwise, and by Proposition 3 we get $f(T) = f((b_{i_1}\ b_j) \star T) = (b_{i_1}\ b_j) \star f(T)$ which is a contradiction because $b_{i_1} \in f(T)$ while $b_j \notin f(T)$. Thus, $\{b_{i_1}, \ldots, b_{i_l}\} = \{b_1, \ldots, b_n\}$. Now we prove that $f(T) = T$. Assume, by contradiction, that we are in the case $f(T) = \{c_1, \ldots, c_k, b_1, \ldots, b_n\}$ with $c_1, \ldots, c_k \in supp(f)$. Then $T \subsetneq f(T)$, and since f is strictly monotone we can construct a strictly ascending chain $T \subsetneq f(T) \subsetneq \ldots \subsetneq f^l(T) \subsetneq \ldots$. Since for any $i \in \mathbb{N}$ we have that $f^l(T)$ is supported by $supp(f) \cup supp(T)$ (this follows by induction on l involving Proposition 3) and $\wp_{fin}(A)$ does not contain an infinite uniformly supported subset (the elements of $\wp_{fin}(A)$ supported by $supp(f) \cup supp(T)$ are exactly the subsets of $supp(f) \cup supp(T)$), we get a contradiction. Thus, $f(T) = T$.

We return to the proof of our theorem and we consider the remaining case $X \setminus supp(f) \neq \emptyset$. We should have that $X = \{a_1, \ldots, a_p, d_1, \ldots, d_m\}$ with $a_1, \ldots, a_p \in supp(f)$ and $d_1, \ldots, d_m \in A \setminus supp(f)$, $m \geq 1$, or $X = \{d_1, \ldots, d_m\}$ with $d_1, \ldots, d_m \in A \setminus supp(f)$,

$m \geq 1$. We have that $X \setminus supp(f) = \{d_1, \ldots, d_m\}$. Denote by $U = X \setminus supp(f)$. If $f(U) \setminus supp(f) \neq \emptyset$, then $f(U) = U$ according to (2). Assume, by contradiction, that $f(U) \setminus supp(f) = \emptyset$, that is, $f(U) = \{x_1, \ldots, x_k\}$ with $x_1, \ldots, x_k \in supp(f)$, $k \geq 1$ (we cannot have $f(U) = \emptyset$ because f is strictly monotone $f(\emptyset) = \emptyset$ and $\emptyset \subsetneq U$). Since $supp(f)$ has only finitely many subsets, A is infinite and f is strictly monotone, there should exist $V \in \wp_{fin}(A)$, $V \subsetneq A \setminus supp(f)$ such that $U \subsetneq V$ and $f(V)$ contains at least one element outside $supp(f)$; for example, we can choose finitely many distinct atoms $d_{m+1}, \ldots, d_{m+2^{|supp(f)|}+1} \in A \setminus (supp(f) \cup \{d_1, \ldots, d_m\})$, and consider $V = \{d_1, \ldots, d_m, d_{m+1}, \ldots, d_{m+2^{|supp(f)|}+1}\}$; since $\{d_1, \ldots, d_m\} \subsetneq \{d_1, \ldots, d_m, d_{m+1}\} \subsetneq \ldots \subsetneq \{d_1, \ldots, d_m, \ldots, d_{m+2^{|supp(f)|}+1}\}$ and f is strictly monotone, we get that $f(V)$ should contain at least one element outside the finite set $supp(f)$. However, in this case, $f(V) = V$ according to (2), and since $f(U) \subsetneq f(V) = V$, we get $\{x_1, \ldots, x_k\} \subseteq V$, i.e. $x_1, \ldots, x_k$ are outside $supp(f)$, a contradiction. Therefore, we necessarily have $f(U) \setminus supp(f) \neq \emptyset$, and hence $f(U) = U$, that is $X \setminus supp(f) = f(X \setminus supp(f))$ for all $X \in \wp_{fin}(A)$. □

Theorem 3. *Let $f : \wp_{fin}(A) \to \wp_{fin}(A)$ be a finitely supported progressive function (i.e. f has the property that $X \subseteq f(X)$ for all $X \in \wp_{fin}(A)$). There are infinitely many fixed points of f, namely the finite subsets of A containing all the elements of $supp(f)$.*

Proof. Let $X \in \wp_{fin}(A)$. Since the support of a finite subset of atoms coincides with the related subset (see Proposition 2 and use the trivial remark that any finite set is uniformly supported), we have $supp(X) = X$ and $supp(f(X)) = f(X)$. According to Proposition 3, for any permutation π fixing $supp(f) \cup supp(X) = supp(f) \cup X$ pointwise we have $\pi \star f(X) = f(\pi \star X) = f(X)$ meaning that $supp(f) \cup X$ supports $f(X)$, that is $f(X) = supp(f(X)) \subseteq supp(f) \cup X$ (1). Since we also have $X \subseteq f(X)$, we obtain $X \setminus supp(f) \subseteq f(X) \setminus supp(f) \subseteq X \setminus supp(f)$, that is $X \setminus supp(f) = f(X) \setminus supp(f)$ (2). If $supp(f) = \emptyset$, the result follows immediately. Let us consider the case $supp(f) = \{a_1, \ldots, a_k\}$. According to (1) and to the hypothesis, we have $supp(f) \subseteq f(supp(f)) \subseteq supp(f)$, and so

$f(supp(f)) = supp(f)$. If X has the form $X = \{a_1, \ldots, a_k, b_1, \ldots, b_n\}$ with $b_1, \ldots, b_n \in A \setminus supp(f)$, $n \geq 1$, we should have by hypothesis that $a_1, \ldots, a_k \in f(X)$, and by (2) $f(X) \setminus supp(f) = X \setminus supp(f) = \{b_1, \ldots, b_n\}$. Since no other elements different from $a_1, \ldots, a_k$ are in $supp(f)$, from (1) we obtain $f(X) = \{a_1, \ldots, a_k, b_1, \ldots, b_n\} = X$. □

Theorem 4. *Let $f : \wp_{fin}(A) \to \wp_{fin}(A)$ be a finitely supported function having the properties that $f(X) \subseteq X$ for all $X \in \wp_{fin}(A)$ and $f(X) \neq \emptyset$ for all $X \neq \emptyset$. Then $f(Y) = Y$ for all $Y \in \wp_{fin}(A)$ with $Y \cap supp(f) = \emptyset$.*

Proof. Let $Y \in \wp_{fin}(A)$ with $Y \cap supp(f) = \emptyset$. Thus, Y is either equal to the empty set or Y is of form $Y = \{b_1, \ldots, b_m\}$ with $b_1, \ldots, b_m \in A \setminus supp(f)$, $m \geq 1$. Obviously, $f(\emptyset) = \emptyset$ from our hypothesis. Furthermore, from the hypothesis we should have $f(Y) = \{b_{i_1}, \ldots, b_{i_n}\}$ with $b_{i_1}, \ldots, b_{i_n} \in A \setminus supp(f)$, $n \geq 1$ and $\{b_{i_1}, \ldots, b_{i_n}\} \subseteq \{b_1, \ldots, b_m\}$.

Assume by contradiction that there exists $b_j \in \{b_1, \ldots, b_m\}$ such that $b_j \notin \{b_{i_1}, \ldots, b_{i_n}\}$. Hence $b_j \neq b_{i_1}$ and $(b_{i_1}\ b_j) \star Y = Y$ because we have $b_{i_1}, b_j \in Y$ and $Y \in \wp_{fin}(A)$. Moreover, since $b_{i_1}, b_j \notin supp(f)$, we have that $(b_{i_1}\ b_j) \in Fix(supp(f))$. From Proposition 3, we obtain $\{b_{i_1}, \ldots, b_{i_n}\} = f(\{b_1, \ldots, b_m\}) = f((b_{i_1}\ b_j) \star \{b_1, \ldots, b_m\}) = (b_{i_1}\ b_j) \star f(\{b_1, \ldots, b_m\}) = (b_{i_1}\ b_j) \star \{b_{i_1}, b_{i_2}, \ldots, b_{i_n}\} = \{b_j, b_{i_2}, \ldots, b_{i_n}\}$, which is a contradiction. Thus, $f(Y) = Y$. □

Theorem 5. *Let $f : \wp_{fin}(A) \to \wp_{fin}(A)$ be a finitely supported function and let $X \in \wp_{fin}(A)$ such that $X \subseteq f(X)$. If f is monotone or progressive, then there exists $n \in \mathbb{N}^*$ such that $f^l(X)$ is a fixed point of f for all $l \geq n$.*

Proof. Since $X \subseteq f(X)$ and f is monotone (i.e. order preserving) or progressive, we can define the ascending sequence $X \subseteq f(X) \subseteq f^2(X) \subseteq \ldots \subseteq f^m(X) \subseteq \ldots$.

We prove by induction that the sequence $(f^m(X))_{m \in \mathbb{N}^*}$ is uniformly supported by $supp(f) \cup supp(X)$, that is, $supp(f^m(X)) \subseteq supp(f) \cup supp(X)$ for each $m \in \mathbb{N}^*$. Let $m = 1$. For any permutation π fixing $supp(f) \cup supp(X)$ pointwise, from Proposition 3 we have $\pi \star f(X) = f(\pi \star X) = f(X)$ meaning that $supp(f) \cup supp(X)$

supports $f(X)$, that is $supp(f(X)) \subseteq supp(f) \cup supp(X)$. Let us suppose that $supp(f^k(X)) \subseteq supp(f) \cup supp(X)$ for some $k \in \mathbb{N}^*$. We have to prove that $supp(f^{k+1}(X)) \subseteq supp(f) \cup supp(X)$. Equivalently, we have to prove that each permutation π fixing $supp(f) \cup supp(X)$ pointwise also fixes $f^{k+1}(X)$. Let $\pi \in Fix(supp(f) \cup supp(X))$. From the inductive hypothesis, we have $\pi \in Fix(supp(f^k(X)))$, and hence $\pi \star f^k(X) = f^k(X)$. According to Proposition 3, we have $\pi \star f^{k+1}(X) = \pi \star f(f^k(X)) = f(\pi \star f^k(X)) = f(f^k(X)) = f^{k+1}(X)$. Therefore, $(f^m(X))_{m \in \mathbb{N}^*}$ is uniformly supported by $supp(f) \cup supp(X)$. Therefore, this sequence should be stationary because $\wp_{fin}(A)$ does not contain an infinite uniformly supported subset. Thus, there exists $n \in \mathbb{N}$ such that $f^n(X) = f^l(X)$ for all $l \geq n$. Fix some $l \geq n$. We have $f(f^l(X)) = f^{l+1}(X) = f^n(X) = f^l(X)$, and so $f^l(X)$ is a fixed point of f. □

Corollary 2. *Let $f : \wp_{fin}(A) \to \wp_{fin}(A)$ be a finitely supported monotone function. Then there exists a least $X_0 \in \wp_{fin}(A)$ supported by $supp(f)$ such that $f(X_0) = X_0$.*

Proof. Since $\emptyset \subseteq f(\emptyset)$ and f is monotone (order preserving), from Theorem 5 we have that there exists $m_0 \in \mathbb{N}^*$ such that $f^{m_0}(\emptyset)$ is a fixed point of f. This fixed point is supported by $supp(f) \cup supp(\emptyset)$. However, we prove that $supp(\emptyset) = \emptyset$. Indeed, from the definition of $\emptyset$, we have $\emptyset \subseteq \pi \star \emptyset$ and $\emptyset \subseteq \pi^{-1} \star \emptyset$ for each π, which means $\emptyset = \pi \star \emptyset$ and $supp(\emptyset) = \emptyset$.

If T is another fixed point of f, then from $\emptyset \subseteq T$, we get $f^n(\emptyset) \subseteq f^n(T)$ for all $n \in \mathbb{N}$. Therefore, $f^{m_0}(\emptyset) \subseteq f^{m_0}(T) = T$, and so $f^{m_0}(\emptyset)$ is the least fixed point of f. □

Theorem 6. *Let $f : \wp_{fin}(A) \to \wp_{fin}(A)$ be a finitely supported function.*

1. *We have $f(supp(f)) \subseteq supp(f)$;*
2. *If f is monotone, then there exists $n \in \mathbb{N}^*$ such that $f^l(supp(f))$ is a fixed point of f for all $l \geq n$.*

Proof. According to Proposition 3, for any permutation π fixing $supp(f)$ pointwise we have $\pi \star supp(f) = supp(f)$ and $\pi \star f(supp(f)) = f(\pi \star supp(f)) = f(supp(f))$ meaning that $supp(f)$ supports $f(supp(f))$, that is, $supp(f(supp(f))) \subseteq supp(f)$. Since the support of a finite subset of atoms coincides with the related subset, we obtain $supp(f(supp(f))) = f(supp(f))$, and so $f(supp(f)) \subseteq supp(f)$.

Assume now that f is monotone. According to the previous item, we can construct the sequence $\ldots \subseteq f^m(supp(f)) \subseteq \ldots \subseteq f^2(supp(f)) \subseteq f(supp(f)) \subseteq supp(f)$. Since $supp(f)$ is finite, the related sequence should be finite, and so there exists $n \in \mathbb{N}$ such that $f^n(supp(f)) = f^l(supp(f))$ for all $l \geq n$. Fix some $l \geq n$. We have $f(f^l(supp(f))) = f^{l+1}(supp(f)) = f^n(supp(f)) = f^l(supp(f))$, and so $f^l(supp(f))$ is a fixed point of f. □

Proposition 5. *Let $f : \wp_{fin}(A) \to \wp_{fin}(A)$ be a finitely supported injective and progressive function. Then $f(Y) = Y$ for all $Y \in \wp_{fin}(A)$.*

Proof. Let $Y \in \wp_{fin}(A)$. As in the proof of Theorem 5, the ascending sequence $Y \subseteq f(Y) \subseteq f^2(Y) \subseteq \ldots \subseteq f^m(Y) \subseteq \ldots$ is uniformly supported by $supp(f) \cup supp(Y)$. Therefore, this sequence should be stationary because $\wp_{fin}(A)$ does not contain an infinite uniformly supported subset. Thus, there exists $n \in \mathbb{N}$ such that $f^n(Y) = f^{n+1}(Y) = f^n(f(Y))$. Since f is injective (and so is f^n), we obtain $f(Y) = Y$. □

Corollary 3. *Let $f : \wp_{fin}(A) \to \wp_{fin}(A)$ be a finitely supported surjective and progressive function. Then $f(Y) = Y$ for all $Y \in \wp_{fin}(A)$.*

Proof. According to Theorem 1, f should be injective. The result now follows from Proposition 5. □

Proposition 6. *Let $f : \wp_{fin}(A) \to \wp_{fin}(A)$ be a finitely supported injective function having the property that $f(X) \subseteq X$ for all $X \in \wp_{fin}(A)$. Then $f(Y) = Y$ for all $Y \in \wp_{fin}(A)$.*

Proof. Let $Y \in \wp_{fin}(A)$. The sequence $\ldots \subseteq f^i(Y) \subseteq \ldots \subseteq f^2(Y) \subseteq f(Y) \subseteq Y$ should be finite since Y is finite. Therefore, there exists

$n \in \mathbb{N}$ such that $f^n(f(Y)) = f^{n+1}(Y) = f^n(Y)$. Since f is injective (and so is f^n), we obtain $f(Y) = Y$. □

Corollary 4. *Let $f : \wp_{fin}(A) \to \wp_{fin}(A)$ be a finitely supported surjective function having the property that $f(X) \subseteq X$ for all $X \in \wp_{fin}(A)$. Then $f(Y) = Y$ for all $Y \in \wp_{fin}(A)$.*

Proof. According to Theorem 1, f should be injective. The result now follows from Proposition 6. □

Theorem 7. *Let $f : \wp_{fin}(A) \to \wp_{fin}(A)$ be a finitely supported function having the property that $f(X \cup Y) = f(X) \cup f(Y)$ for all $X, Y \in \wp_{fin}(A)$. If $X_0 \in \wp_{fin}(A)$ and $k \in \mathbb{N}^*$ such that $X_0 \subseteq f^k(X_0)$, then $\bigcup\limits_{n\in\mathbb{N}} f^n(X_0)$ is a finite subset of A and a fixed point of f.*

Proof. As in the proof of Theorem 5, the sequence $(f^n(X_0))_{n\in\mathbb{N}} \subseteq \wp_{fin}(A)$ is uniformly supported by $supp(f) \cup supp(X_0)$. Therefore, this sequence should be finite, and so there exist $\bigcup\limits_{n\in\mathbb{N}} f^n(X_0)$ and $\bigcup\limits_{n\in\mathbb{N}} f(f^n(X_0))$ which are proved to be supported by $supp(f) \cup supp(X_0)$. Clearly, $\{f^{n+1}(X_0) \,|\, n \in \mathbb{N}\} = \{f^n(X_0) \,|\, n \in \mathbb{N}^*\} \subseteq \{f^n(X_0) \,|\, n \in \mathbb{N}\}$, and so $\bigcup\limits_{n\in\mathbb{N}} f^{n+1}(X_0) \subseteq \bigcup\limits_{n\in\mathbb{N}} f^n(X_0)$. Since $f^0(X_0) = X_0 \subseteq f^k(X_0)$ with $k \geq 1$, we have $f^0(X_0) \subseteq \bigcup\limits_{n\in\mathbb{N}} f^{n+1}(X_0)$. However, obviously, $f^i(X_0) \subseteq \bigcup\limits_{n\in\mathbb{N}^*} f^n(X_0) = \bigcup\limits_{n\in\mathbb{N}} f^{n+1}(X_0)$ for all $i \in \mathbb{N}^*$, and so $\bigcup\limits_{n\in\mathbb{N}} f^n(X_0) \subseteq \bigcup\limits_{n\in\mathbb{N}} f^{n+1}(X_0)$. Therefore, $\bigcup\limits_{n\in\mathbb{N}} f^{n+1}(X_0) = \bigcup\limits_{n\in\mathbb{N}} f^n(X_0)$, and so, according to the hypothesis, $f(\bigcup\limits_{n\in\mathbb{N}} f^n(X_0)) = \bigcup\limits_{n\in\mathbb{N}} f(f^n(X_0)) = \bigcup\limits_{n\in\mathbb{N}} f^{n+1}(X_0) = \bigcup\limits_{n\in\mathbb{N}} f^n(X_0)$, which means $\bigcup\limits_{n\in\mathbb{N}} f^n(X_0)$ is a fixed point of f. □

Theorem 8. *Let $f : \wp_{fin}(A) \to \wp_{fin}(A)$ be a finitely supported injective function. Then for any $X \in \wp_{fin}(A)$ there exists $n \in \mathbb{N}^*$ such that X is a fixed point of f^n.*

Proof. Let $X \in \wp_{fin}(A)$. As in the proof of Theorem 5, the sequence $(f^m(X))_{m\in\mathbb{N}} \subseteq \wp_{fin}(A)$ is uniformly supported by $supp(f) \cup supp(X)$.

Therefore, this sequence should be finite, and so there exist $l, k \in \mathbb{N}$, $l > k$, such that $f^l(X) = f^k(X)$. Since f is injective, we get $X = f^{l-k}(X)$ and so the result follows by denoting $n = l - k$. □

Corollary 5. *Let $f : \wp_{fin}(A) \to \wp_{fin}(A)$ be a finitely supported surjective function. Then for any $X \in \wp_{fin}(A)$ there exists $n \in \mathbb{N}^*$ such that X is a fixed point of f^n.*

Proof. From Theorem 1, the surjective function $f : \wp_{fin}(A) \to \wp_{fin}(A)$ should be injective, and the result follows from Theorem 5. □

4 Conclusion

This paper is the extended and revised version of the conference paper [1] presented at MFOI 2020. We are able to prove that for finitely supported self-mappings (self-functions) defined on $\wp_{fin}(A)$ the injectivity is equivalent with the surjectivity. These mappings also satisfy some fixed point properties if some particular requirements (such as injectivity, surjectivity, monotony or progressivity) are introduced.

References

[1] A. Alexandru, "Finitely Supported Mappings Defined on the Finite Powerset of Atoms," in *The Sixth International Conference on Mathematical Foundations of Informatics*, (Kyiv), 13-15 January 2021.

[2] A. Alexandru and G. Ciobanu, *Foundations of Finitely Supported Structures: a set theoretical viewpoint*, Springer, 2020.

[3] A. Alexandru and G. Ciobanu, "Fixed point results for finitely supported algebraic structures," *Fuzzy Sets and Systems*, vol. 397, pp. 1–27, 2020.

[4] A. Alexandru and G. Ciobanu, "Uniformly supported sets and fixed points properties," *Carpathian Journal of Mathematics*, vol. 36, no.3, pp. 351–364, 2020.

[5] A.M. Pitts, *Nominal Sets Names and Symmetry in Computer Science*, Cambridge University Press, 2013.

Andrei Alexandru

Received February 01, 2021
Accepted February 26, 2021

Romanian Academy, Institute of Computer Science,
Iaşi, Romania
E–mail: andrei.alexandru@iit.academiaromana-is.ro

Computer Science Journal of Moldova, vol.**29**, no.1(85), 2021

On the bondage, strong and weak bondage numbers in Complementary Prism Graphs

Aysun Aytaç Tufan Turacı

Abstract

Let $G = (V(G), E(G))$ be a simple undirected graph of order n, and let $S \subseteq V(G)$. If every vertex in $V(G) - S$ is adjacent to at least one vertex in S, then the set S is called a *dominating set*. The *domination number* of G is the minimum cardinality taken over all sets of S, and it is denoted by $\gamma(G)$. Recently, the effect of one or more edges deletion on the domination number has been examined in many papers. Let $F \subseteq E(G)$. The *bondage number* $b(G)$ of G is the minimum cardinality taken over all sets of F such that $\gamma(G-F) > \gamma(G)$. In the literature, a lot of domination and bondage parameters have been defined depending on different properties. In this paper, we investigate the *bondage, strong and weak bondage numbers* of complementary prism graphs of some well-known graph families.

Keywords: Connectivity, Domination number, Strong and weak domination numbers, Bondage number, Strong and weak bondage numbers, Complementary prism graphs.

MSC 2010: 05C40, 05C69.

1 Introduction

Graph theory has become an important mathematical tool in many different sciences. For example, the domination number is an important graph parameter, and it has many different application areas [15]. In the near past, some papers published about how the domination varies when there are changes in edges or vertices by adding or removing. This is important because vertices in the domination set can be considered

transmitters that cover a wide variety of communication links. The loss of certain links may make the transmitter set a non-dominating set. i.e., the communications between some links can be disrupted by a wrecker. Consider that a wrecker does not know which vertices in the network serve as transmitters but knows that those vertices form a minimum domination set in the network. What is the minimum number of connections that the wrecker must disrupt such that at least a new transmitter is needed to connect with all sites? With this in mind, the concept of bondage has begun to be studied in graph theory.

Let G be a simple undirected graph without loops and multiple edges with vertex set $V(G)$ and edge set $E(G)$. The order of G is the number of vertices in G. The degree of a vertex $v \in V(G)$ is the number of edges incident to v and it is denoted by $deg(v)$. Let $S \subseteq V(G)$. If every vertex in $V(G) - S$ is adjacent to at least one vertex in S, then the set S is called a *dominating set*. The *domination number* of G is the minimum cardinality over all domination set of G, and it is denoted by $\gamma(G)$.

The following question about domination number is very important: what is the minimum number of links that must be removed so that the domination number increases? Bauer et al. [7] has given the answer of this question. They have defined the *bondage number* for the vulnerability of a graph. The *bondage number* $b(G)$ of G is defined as the minimum cardinality among all subsets of edges $F \subseteq E(G)$ for which $\gamma(G-F) > \gamma(G)$ [11]. There are different parameters depending upon the domination number such as the reinforcement number [13], the average lower bondage number [18], the average lower reinforcement number [19], the residual domination number [20] and the link residual domination number [21]. Furthermore, different papers about the domination and bondage numbers can be seen in [2],[4],[5],[11],[18],[19].

The concept of a *strong dominating set* (sd-set) has been introduced by Sampathkumar and Pushpalatha [9]. Let $u, v \in V(G)$. A set $S \subseteq V(G)$ is a strong dominating set of G if every vertex u in $V(G) - S$ is adjacent to the vertex v in S such that $deg(v) \geq deg(u)$ and $(u, v) \in E(G)$. The *strong domination number* $\gamma_s(G)$ is the minimum cardinality over all strong dominating set of G. The strong bondage

number $b_s(G)$ of G is defined as the minimum cardinality among all subsets of edges $F \subseteq E(G)$ for which $\gamma_s(G-F) > \gamma_s(G)$. This concept has been introduced by J. Ghoshal et al. [12].

A set $S \subseteq V(G)$ is a *weak dominating set* (wd-set) of G if every vertex u in $V(G)-S$ is adjacent to the vertex v in S such that $deg(v) \leq deg(u)$ and $(u,v) \in E(G)$. The *weak domination number* $\gamma_w(G)$ is the minimum cardinality over all strong dominating set of G. The *weak bondage number* $b_w(G)$ of G is defined as the minimum cardinality among all subsets of edges $F \subseteq E(G)$ for which that $\gamma_w(G-F) > \gamma_w(G)$ [9].

There have been applications of *strong* and *weak domination* in specific practical situations. For example, in a road network, where certain locations are related, the degree of vertex v is the number of roads that meet at v. If $deg(u) \geq deg(v)$, then the traffic at u is more severe than that at v, and vice versa. If traffic between u and v is considered, predilection should be given to the vehicles going from u to v. Thus, *u strongly dominates v* and *v weakly dominates u*.

Complementary prism graphs have been introduced by Haynes et al. [17]. Let $\overline{G}$ be a complemantary graph of a graph G. The complementary prism is denoted by $G\overline{G}$. It is a graph formed from the disjoint union of G and $\overline{G}$ by adding the edges of a perfect matching between the corresponding vertices of G and $\overline{G}$. The vertex $\overline{v}$ denotes the vertex v in the copy of $\overline{G}$, and it is defined for each $v \in V(G)$ [16], [17]. Many well-known graphs may be actualized as complementary prism graphs. For example, the corona $K_n \circ K_1$ is the complementary prism $K_n\overline{K}_n$. Another example, the Petersen graph is the complementary prism $C_5\overline{C}_5$ (see [17]).

Throughout this paper, minimum degree, maximum degree, vertex set and edge set of the graph G are denoted by $\delta(G)$, $\Delta(G)$, V and E, respectively [8]. Similarly, the vertex set and the edge set of the graph $\overline{G}$ are denoted by $\overline{V}$ and $\overline{E}$, respectively [8]. Furthermore, e_{uv} denotes the edges between the vertices u and v, $N(u)$ denotes the neighborhood of the vertex u.

The paper proceeds as follows. In Section 2, basic results of literature on the *strong-weak bondage number* of some special graphs are

presented. Some results of the *bondage*, *strong* and *weak bondage numbers* for complementary prisms are given in Section 3. Finally, the conclusion of paper is given in Section 4.

2 General Bounds on Strong and Weak Bondage Numbers

In [14], sharp bounds were obtained for $b(G)$, $b_s(G)$ and $b_w(G)$. Furthermore, the exact values were determined for several classes of graphs such as K_n, C_n, P_n, $W_{1,n}$, $K_{m,n}$. In this section, we will review some of the known results.

Theorem 1 ([14]). *If G is a nonempty graph with a unique minimum dominating set, then $b(G) = 1$.*

Theorem 2 ([7], [11]). *If G is a nonempty graph, then $b(G) \leq min_{uv \in E(G)}(deg(u) + deg(v) - 1)$.*

Theorem 3 ([6]). *If G has edge connectivity k, then $b(G) \leq \Delta(G) + k - 1$.*

Theorem 4 ([14]). *If T is a nontrivial tree, then $b_s(T) \leq 3$ and $b_w(T) \leq \Delta(T)$.*

Theorem 5 ([14]). *If any vertex of tree T is adjacent with two or more end-vertices, then $b_s(T) = 1$.*

3 Exact Values for $b(G\overline{G})$, $b_s(G\overline{G})$ and $b_w(G\overline{G})$

We begin this subsection by determining the bondage, strong and weak bondage of the complementary prism $G\overline{G}$ when G is a specified family of graphs, such as the star graph $K_{1,n}$, the complete graph K_n, the path graph P_n, the cycle graph C_n, the wheel graph $W_{1,n}$, the complete bipartite graph $K_{m,n}$, the graph tK_2 and the graph K_noK_1. The graph K_noK_1 is obtained by adding a pendant vertex v that is $deg(v) = 1$ to each vertex of the graph K_n. In order to understand the proofs of the

theorems given in this section more easily, the set of vertices belonging to the graph G in $G\overline{G}$ graph is shown as V and the set of vertices belonging to the $\overline{G}$ graph are shown as $\overline{V}$. Furthermore, when the edges set of $G\overline{G}$ is divided into $E(G\overline{G}) = E_1(G\overline{G}) \cup E_2(G\overline{G}) \cup E_3(G\overline{G})$, the edge sets here are respectively expressed as the set of edges of the graph G, the set of edges combining the graph G with the $\overline{G}$ graph, and the set of edges of the $\overline{G}$ graph.

Theorem 6. *If $G = K_{1,n}$, then $b(G\overline{G}) = b_s(G\overline{G}) = b_w(G\overline{G}) = 1$.*

Proof. Let $G = K_{1,n}$ and u be the center vertex of the graph G.
For $b(G\overline{G})$ and $b_s(G\overline{G})$; vertices u and $\overline{u}$ have to be in $\gamma(G\overline{G})$ and $\gamma_s(G\overline{G})$-strong dominating sets, where $\overline{v} \in \overline{V} - \{\overline{u}\}$. $\gamma(G\overline{G} - e_{u\overline{u}}) = \gamma_s(G\overline{G} - e_{u\overline{u}}) = \gamma(G\overline{G}) + |\{\overline{u}\}| = \gamma_s(G\overline{G}) + |\{\overline{u}\}|$ when an edge $e_{u\overline{u}} \in E(G\overline{G})$ is removed from the graph $G\overline{G}$, also it is easy to see that $b(G\overline{G}) = b_s(G\overline{G}) = 1$.

Now let's calculate the $b_w(G\overline{G})$ value of the graph. It is easily seen that the weak domination number of graph G is $\gamma_w(G\overline{G}) = n + 1$ from [1]. There are n vertices of degree 2 and a vertex of degree 1, say $\overline{u}$, which is adjacent to center vertex in $\gamma_w(G\overline{G})$-weak dominating set. This dominating set is unique. If an edge $e_{v\overline{v}} \in E(G\overline{G})$ is removed from the graph $G\overline{G}$, then the vertex $\overline{v}$ is not weakly dominated, where the vertex v is degree of 2 and so the vertex $\overline{v}$ must be in $\gamma_w(G\overline{G})$-weak dominating set. It can be easily seen that, $\gamma_w(G\overline{G} - e_{v\overline{v}}) = \gamma_w(G\overline{G}) + |\{\overline{v}\}|$ and it follows that $b_w(G\overline{G}) = 1$. □

Theorem 7. *If $G = K_n$, then $b(G\overline{G}) = b_s(G\overline{G}) = n$ and $b_w(G\overline{G}) = 1$.*

Proof. Let $G = K_n$. The vertices of graph $G\overline{G}$ are of two kinds: vertices of degree $n+1$ and one, respectively. The vertices of degree one will be referred to as pendant vertices and vertices of degree $n+1$ – as support vertices.

Let's calculate the bondage and strong bondage numbers of the graph. Since each vertex $\overline{v}$ in $\overline{V}$ $deg(\overline{v}) = 1$, $\gamma(G\overline{G})$-dominating sets and $\gamma_s(G\overline{G})$-strong dominating sets have to contain vertex set V. There are two ways for deleting the edges to increase the domination number of the graph $G\overline{G}$:

(i) If the n edges between V and $\overline{V}$ are removed from the graph $G\overline{G}$, then the rest of the graph $G\overline{G}$ is all independent pendant vertices and support vertices which consist of two complete graphs K_n. There are all pendant vertices and one support vertex in $\gamma(G\overline{G})$-dominating set and $\gamma_s(G\overline{G})$-strong dominating set. Therefore, we obtain $b(G\overline{G}) = b_s(G\overline{G}) = n$.
(ii) If one of the support vertices is isolated, then the domination number increases by one. Thus, n edges attached to any support vertex are removed.

From (i) and (ii), we have $b(G\overline{G}) = b_s(G\overline{G}) = n$.

Now let's calculate the weak bondage number of the graph. $\gamma_w(G\overline{G})$-weak dominating set must contain all vertices of $\overline{V}$. If an edge $e_{u\overline{u}} \in E(G\overline{G})$ for $\exists u \in V$ is removed from $G\overline{G}$, then the vertex $\overline{u}$ becomes the isolated vertex. Furthermore, since the degree of the vertex u is less than degrees of all vertices of the $V - \{u\}$, then $V - \{u\}$ set does not weakly dominate the vertex u. So, the vertex u must be in $\gamma_w(G\overline{G})$ -weak dominating set. It can be easily seen that $\gamma_w(G\overline{G} - e_{u\overline{u}}) > \gamma_w(G\overline{G})$ and we have $b_w(G) = 1$. □

Theorem 8. *If $G = P_n$ for $n > 5$ and $k > 1$, then*

i) $b(G\overline{G}) = \begin{cases} 1, & n=3k, \\ 2, & otherwise \end{cases}$

ii) $b_s(G\overline{G}) = \begin{cases} 3, & n=3k, \\ 2, & otherwise \end{cases}$

iii) $b_w(G\overline{G}) = \begin{cases} 2, & n=3k+1, \\ 1, & otherwise. \end{cases}$

Proof. In three different bondage measure proofs, three cases are examined according to n *mod* 3.

Proof of $b(G\overline{G})$ is obtained by three cases.

Case 1. If $n = 3k$, then the graph G consists of k-copies of P_3. The dominance number increases when an edge between two 3-degree vertices of any P_3 graph in the $G\overline{G}$ graph is deleted. Since $n = 3k$, $\gamma(G\overline{G})$-dominating set is unique. Therefore, we have $b(G\overline{G}) = 1$.

Case 2. $n = 3k + 1$, then there are many $\gamma(G\overline{G})$-dominating sets. Furthermore, it is easy to see that $b(G\overline{G}) > 1$. Let the vertex u be any end vertex with degree two. Let $S(u)$ be the set of edges connected to the vertex of u. When $S(u)$ is removed from the graph $G\overline{G}$, the remaining structure contains graph $P_{3k}\overline{P_{3k}}$. So, we have $\gamma(G\overline{G}) = \lceil (n+3)/3 \rceil = k + 1$ from [1]. Thus, $\gamma(G\overline{G} - S(u)) = \gamma(P_{3k}\overline{P_{3k}}) + 1 = \lceil (n+3)/3 \rceil + 1 = k + 2$. Then, we obtain $b(G\overline{G}) = 2$.

Case 3. $n = 3k + 2$, then the proof is made similar to Case 2.

Proof is completed by Case 1, Case 2 and Case 3.

For $\forall u \in V(G)$ and $\forall \bar{u} \in V(\bar{G})$, $\quad d_{G\bar{G}}(\bar{u}) > d_{G\bar{G}}(u)$. Thus, $\gamma_s(G\overline{G})$-strong dominating set contains any two vertices of $\overline{G}$ to strong dominate all vertices of the graph $\overline{G}$. It is easily seen that $b_s(G\overline{G}) > 1$. The graph $G\overline{G}$ has more than one $\gamma_s(G\overline{G})$-strong dominating sets. The proof of $b_s(G\overline{G})$ is obtained by three cases.

Case 1. Let $n = 3k$. Since $\delta(G\overline{G}) = 2$, when any two edges are deleted from the $G\overline{G}$ graph, the strong dominance number of the graph does not change. Therefore, $b_s(G\overline{G}) > 1$. Let vertices u and v be end vertices of the graph G and P_4 induced subgraph of the graph G without $\{u, v\}$-vertices. When the edges of this P_4 graph are removed, it is easy seen that $\gamma_s(G\overline{G} - E(P_4)) > \gamma_s(G\overline{G})$. So, we have $b_s(G\overline{G}) = 3$.

Case 2. Let $n = 3k + 1$. By [1], we have $\gamma_s(G\overline{G}) = k + 2$. Let u and v be end vertices of the graph $G\overline{G}$. Let $S(u)$ be the set of edges connected to the vertex of u. When $S(u)$ is removed from the graph $G\overline{G}$, the remaining structure contains graph $P_{3k}\overline{P_{3k}}$. As the vertex u is an isolated vertex, it is in $\gamma_s(G\overline{G})$-strong dominating set. Furthermore, the strong domination number of remaining graph is equal to strong domination number of graph $P_{3k}\overline{P_{3k}}$. Thus, $\gamma_s(P_{3k}\overline{P_{3k}}) = \lceil (3k+4)/3 \rceil = k + 2$ and vertex $\overline{u}$ is strong dominated by $\gamma_s(P_{3k}\overline{P_{3k}})$-strong dominating set. Finally, $\gamma_s(G\overline{G} - S(u)) > \gamma_s(G\overline{G})$. Now, we have $b_s(G\overline{G}) = 2$, since $\gamma_s(G\overline{G} - S(u)) > \gamma_s(G\overline{G})$.

Case 3. Let $n = 3k + 2$. When the same edges are deleted as in Case 2, the remaining structure contains the graph $P_{3k+1}\overline{P_{3k+1}}$. The rest of the proof is made similar to Case 2. Therefore, we obtain $b_s(G\overline{G}) = 2$.

Proof is completed by Case1, Case2 and Case3.

The proof of $b_w(G\overline{G})$ is obtained by two cases.

Case 1. Let $n = 3k+1$. It can be easily seen that $\gamma_w(G\overline{G}) = \gamma_w(G\overline{G}-e)$, where e is an edge of the graph $G\overline{G}$. Therefore, $b_w(G\overline{G}) > 1$. Let u be an end vertex of the graph G. Similarly, let $v \in N(u)$ and $m \in N(v)$ in G. When the edges e_{uv} and e_{vm} are removed from the graph $G\overline{G}$, the vertices u, v and m must be in $\gamma_w(G\overline{G})$-weak dominating set from definition of weak domination set. Furthermore, the other end vertex of the graph G must be also in $\gamma_w(G\overline{G})$-weak dominating set. Since $\overline{v}$ is weakly dominate to $\overline{V} - \{\overline{u}, \overline{m}\}$, this vertex must be in $\gamma_w(G\overline{G})$-weak dominating set. There are $(n-6)$ vertices that are not weakly dominated such that these vertices are formed as graph P_{n-6}. So, $\gamma_w(G\overline{G}) = \gamma_w(P_{n-6}) + 5$ and $\gamma_w(G\overline{G}) = \lceil (n-6)/3 \rceil + 5 = \lceil (n+9)/3 \rceil$. Therefore, $\lceil (n+9)/3 \rceil > \lceil (n+6)/3 \rceil$, since $n = 3k+1$. Consequently, we have $b_w(G\overline{G}) = 2$.

Case 2. For $n = 3k$ and $n = 3k+2$, let u be end vertex of the graph G and $v \in N(u)$, where $N(u)$ be the neighborhood of vertex u in the graph G. When an edge e_{uv} is removed from the graph $G\overline{G}$, $\gamma_w(G\overline{G})$-weak dominating set must contain vertices u and v and also the other end vertex of the graph G. So, the remaining graph is P_{n-5}. The rest of the proof is similar to Case 1. So, we obtain $\gamma_w(G\overline{G}) = \lceil (n+10)/3 \rceil$. Then we have $b_w(G\overline{G}) = 1$, since $\lceil (n+10)/3 \rceil > \lceil (n+6)/3 \rceil = \gamma_w(G\overline{G})$. The proof is completed. □

Theorem 9. *If $G = C_n$ for $n > 5$ and $k > 1$, then*

i) $b(G\overline{G}) = b_s(G\overline{G}) = \begin{cases} 5, & n=3k, \\ 3, & otherwise. \end{cases}$

ii) $b_w(G\overline{G}) = \begin{cases} 2, & n=3k, \\ 1, & otherwise. \end{cases}$

Proof. In three different bondage measure proofs, three cases are examined according to n *mod* 3.

Proof of $b_s(G\overline{G})$ is obtained by three cases.

When one or two edges are removed from $G\overline{G}$, strong domination number does not increase since $\gamma_s(G\overline{G})$-strong dominating set is more than one set. It is easy to see that $b_s(G\overline{G}) > 2$, since $\delta(G\overline{G}) = 2$.

Case 1. Let $n = 3k + 1$ and u and v be two adjacent vertices in graph G. When the edge e_{uv} is removed from the graph $G\overline{G}$, the remaining graph is $(P_{3k+1}\overline{P_{3k+1}} + \{e_{\overline{u}\,\overline{v}}\})$. Furthermore, it is easy to see that $\gamma_s(P_{3k+1}\overline{P_{3k+1}} + \{e_{\overline{u}\,\overline{v}}\}) = \gamma_s(P_{3k+1}\overline{P_{3k+1}})$. Finding the bondage number of $(P_{3k+1}\overline{P_{3k+1}} + \{e_{\overline{u}\,\overline{v}}\})$ is similar to finding the bondage number of $(P_{3k+1}\overline{P_{3k+1}})$ by Theorem 8. So, we have $b_s(G\overline{G}) = 1 + b_s(P_{3k+1}\overline{P_{3k+1}}) = 3$.

Case 2. Let $n = 3k + 2$. The proof is made similar to Case 1.

Case 3. Let $n = 3k$. The degrees of all vertices of graph G and graph $\overline{G}$ are 3 and $(n - 2)$, respectively. The edges of the graph $\overline{G}$ are not removed, since $n - 2 > 3$. It can be easily seen that $\gamma_s(G\overline{G})$-strong dominating set must contain some vertices from the graph G. In order to increase strong domination number of the graph $G\overline{G}$, we have two sub cases.

Subcase 1. Let's take any subgraph P_6 of the graph G. When all edges of the graph P_6 are removed, the remaining graph includes the graph P_{n-4} and 4-isolated vertices. Let x and y be any two isolated vertices. Then we have $\gamma_s(G\overline{G} - E(P_6)) = \gamma_s(P_{n-4}) + 2\gamma_s(K_1) + |\{\overline{x}, \overline{y}\}|) = \lceil (n - 4)/3 \rceil + 4 = \lceil (n + 8)/3 \rceil$. So, it can be easily seen that $\gamma_s(G\overline{G} - E(P_6)) > \gamma_s(G\overline{G})$, since $\gamma_s(G\overline{G}) = \lceil (n - 4)/3 \rceil$. Then we obtain $b_s(G\overline{G}) = 5$.

Subcase 2. $\gamma_s(G\overline{G})$-strong dominating set must contain any two vertices from the graph $G\overline{G}$. According to this situation, we must remove some edges. These are three edges from $E_1(G\overline{G})$ and two edges from $E_2(G\overline{G})$. Let S be the set of these edges, so $|S| = 5$. The remaining graph is graph P_{n-2} and two isolated vertices, when edges of S are removed. Furthermore, $\gamma_s(G\overline{G})$-strong dominating set must contain any two vertices in $\overline{V}$. These vertices strong dominate all vertices of $\overline{V}$, since the degree of all vertices of graph $\overline{G}$ is $(n - 2)$ [1]. Then, we have $\gamma_s(G\overline{G} - S) = \gamma_s(P_{n-2}) + 2 + \gamma_s(\overline{G})$. So, it can be easily seen that $\gamma_s(G\overline{G} - S) > \gamma_s(G\overline{G})$ since $\gamma_s(G\overline{G}) = \lceil (n + 4)/3 \rceil$. Finally, we obtain

$b_s(G\overline{G}) = 5$.

By Subcase 1 and Subcase 2, we have $b_s(G\overline{G}) = 5$ for $n = 3k$.

Thus, the proof of $b_s(G\overline{G})$ is completed by Case 1, Case 2 and Case 3.

Since the G and $\overline{G}$ graphs are regular graphs, $\gamma(G\overline{G})$-dominating set and $\gamma_s(G\overline{G})$-strong dominating set are the same set from the definition of domination and strong domination. Therefore, $b(G\overline{G}) = b_s(G\overline{G})$.

Proof of $b_w(G\overline{G})$ is obtained by two cases.

Case 1. If $n = 3k + 1$ and $n = 3k + 2$, then degrees of all vertices of the graphs G and $\overline{G}$ are 3 and $(n-2)$, respectively. Let u be any vertex of the graph G. Furthermore, $deg(u) = 2$ and $deg(\overline{u}) = n - 3$. When the edge $e_{u\,\overline{u}} \in E(G\overline{G})$ is removed from the graph $G\overline{G}$, $\delta(G) = deg(u)$ and $\delta(\overline{G}) = deg(\overline{u})$. Thus, $\gamma_w(G\overline{G})$-weak dominating set must contain $\{u, \overline{u}\}$. $|D_u| = \gamma_w(P_{n-3}) = \lceil (n-3)/3 \rceil$, where D_u is weak dominating set of the graph $G - N(u)$. Moreover, the vertex $\overline{u}$ weak dominates all vertices of $\overline{V} - N(\overline{u})$. So, the remaining graph is $P_2 = \overline{G} - N(\overline{u})$ that is not weakly dominated. If $\gamma_w(G\overline{G})$-weak dominating set includes any vertex of the graph P_2, then $\gamma_w(G\overline{G}) = |D_u| + \{u, \overline{u}\} + 1 = \lceil (n - 3)/3 \rceil + 2 + 1 = \lceil (n + 6)/3 \rceil$. Thus, it can be easily seen that we have $b_w(G\overline{G}) = 1$ since $\lceil (n + 6)/3 \rceil > \lceil (n + 4)/3 \rceil$ when $n = 3k + 1$ and $n = 3k + 2$.

Case 2. If $n = 3k$, then the domination number does not increase when any edge is removed from the graph $G\overline{G}$. So, $b_w(G\overline{G}) > 1$. Let vertex v be neighbor of vertex u in the graph G for $\exists u \in V$. $\gamma_w(G\overline{G})$-weak dominating set must contain vertices u, v and $\overline{u}$ when the edges $e_{u\overline{u}}$ and e_{uv} are removed from the graph $G\overline{G}$. Then, there are $(n - 4)$ vertices with degree three in the graph G and one vertex from the graph $\overline{G}$, where these vertices are not weakly dominated. So, there are $\lceil (n-4)/3 \rceil + 4 = \lceil (n+8)/3 \rceil$ vertices in $\gamma_w(G\overline{G})$-weak dominating set. Then we have $b_w(G\overline{G}) = 2$ since $\lceil (n + 8)/3 \rceil > \lceil (n + 4)/3 \rceil$ for $n = 3k$.

The proof is completed. □

Remark 1. *The $\gamma_s(G\overline{G})$-strong dominating sets and the $\gamma(G\overline{G})$-dominating sets are the same since graphs G and $\overline{G}$ are regular. So,*

the value of the bondage number and strong bondage number are the same if $G = C_n$.

Theorem 10. *If* $G = tK_2$ *and* $(t > 2)$, *then* $b(G\overline{G}) = b_s(G\overline{G}) = 2$ *and* $b_w(G\overline{G}) = 1$.

Proof. For $b(G\overline{G})$ and $b_s(G\overline{G})$, it is easy to see that any edge e is removed from the graph $G\overline{G}$, we have $\gamma(G\overline{G}) = \gamma(G\overline{G} - e)$ and $\gamma_s(G\overline{G}) = \gamma_s(G\overline{G} - e)$. So, $b(G\overline{G}) > 1$ and $b_s(G\overline{G}) > 1$. Let vertex u be a vertex of any graph K_2 and $v \in N(u)$ for the graph G. If the edge e_{uv} and $e_{u\overline{u}}$ are removed from the graph $G\overline{G}$, then $\gamma(G\overline{G} - \{e_{uv}, e_{u\overline{u}}\}) = \gamma(G\overline{G} - \{e_{uv}, e_{u\overline{u}}\}) = t + 2$. So, we obtain $b(G\overline{G}) = b_s(G\overline{G}) = 2$.

For $b_w(G\overline{G})$, we have $\gamma(G\overline{G}) = \gamma_s(G\overline{G}) = \gamma_w(G\overline{G}) = t + 1$ by [1]. Let u and v be two vertices of any graph K_2. $\gamma_w(G\overline{G})$-weak dominating set must contain vertices u and v, when the edge e_{uv} is removed from the graph $G\overline{G}$. Furthermore, $\gamma_w(G\overline{G})$-weak dominating set must contain a vertex of every remaining graph K_2. Moreover, all vertices of $\overline{V}$ are weakly dominated, when $\gamma_w(G\overline{G})$-weak dominating set contains vertex $\overline{u}$. So, it can be easily seen that $\gamma_w(G\overline{G} - e_{uv}) > \gamma_w(G\overline{G})$. Then we have $b_w(G\overline{G}) = 1$. □

Corollary 1. *If* $G = tK_2$ *and* $t = 1$, *then* $\gamma(G\overline{G}) = \gamma_s(G\overline{G}) = \gamma_w(G\overline{G}) = 1$.

Theorem 11. *If* $G = tK_n$, *then* $b(G\overline{G}) = b_s(G\overline{G}) = n$ *and* $b_w(G\overline{G}) = 1$.

Proof. For $b(G\overline{G})$ and $b_s(G\overline{G})$, we have $\gamma(G\overline{G}) = \gamma_s(G\overline{G}) = \gamma_w(G\overline{G}) = t+2$ by [1]. Let vertices u and v be two vertices of different two graphs K_n. $\{\overline{u}, \overline{v}\}$-set dominates (strong dominates) $\overline{V} \cup \{u, v\}$. There are two ways to increase domination (strong domination) number, since $\gamma(K_n) = \gamma_s(K_n) = 1$.

Case 1. If any vertex of any graph K_n is an isolated vertex, then $\gamma(G\overline{G}) = \gamma(G\overline{G}) = t + 3$. So, we have $b(G\overline{G}) = b_s(G\overline{G}) = n$.

Case 2. For the graph $G\overline{G}$ the degrees of the all vertices of the graph K_n are n. Let $u \in V(K_n)$. If the degree of all vertices of graph K_n

are $(n-3)$, then all vertices of the graph K_n are dominated by two vertices. The number of the edges of the graph K_n is $(n(n-1)/2)$. When the degree of the vertices of the graph K_n decreases, the number of the edges of the new graph is $(n(n-3)/2)$. Moreover, we obtain $(n(n-1)/2) - (n(n-3)/2) = n$ and $\gamma(G\overline{G}) = \gamma(G\overline{G}) = t+3$. Then we have $b(G\overline{G}) = b_s(G\overline{G}) = n$.

The proof is completed by Case 1 and Case 2.

For $b_w(G\overline{G})$, let u and v be two vertices of different two graphs K_n. $\{\overline{u}, \overline{v}\}$-set weak dominates $\overline{V}$. If an edge is removed from any graph K_n, then $\gamma_w(G\overline{G}) = t+3$. So, we have $b_w(G\overline{G}) = n$. □

Theorem 12. *If $G = K_n o K_1$, then $b(G\overline{G}) = b_s(G\overline{G}) = \lceil n/2 \rceil$ and $b_w(G\overline{G}) = 2$.*

Proof. It is easy to see that the proofs of $b(G\overline{G})$ and $b_s(G\overline{G})$ are similar to $b_s(K_n)$ in [4]. So, $b(G\overline{G}) = b_s(G\overline{G}) = \lceil n/2 \rceil$.

For $b_w(G\overline{G})$, there are n vertices which are degree 2 and n vertices which are degree n in the subgraph G and $\overline{G}$ of $G\overline{G}$, respectively. These vertices are the smallest degree vertices of the graph $G\overline{G}$. Moreover, these vertices are independent from each other. So, $\gamma_w(G\overline{G}) = 2n$ by [1]. If any edge e is removed from the graph $G\overline{G}$, then $\gamma_w(G\overline{G}) = \gamma_w(G\overline{G} - e)$. So, $b_w(G\overline{G}) > 1$. Let u be any vertex whose degree 2 and $v \in N(u)$ and $v \neq \overline{u}$. It can be easily seen that vertices u and $\overline{v}$ weakly dominate vertex u. If the edges $e_{uv} \in E(G\overline{G})$ and $e_{v\overline{v}} \in E(G\overline{G})$ are removed from the graph $G\overline{G}$, then $\gamma_w(G\overline{G} - \{e_{uv}, e_{v\overline{v}}\} = 2n+1$. So, we obtain $b_w(G\overline{G}) = 2$. □

Theorem 13. *If $G = W_{1,n}$ and $(n > 5)$, then*

$$b(G\overline{G}) = b_s(G\overline{G}) = 1 \text{ and } b_w(G\overline{G}) = \begin{cases} 3 & , \ n{=}3k; \\ 2 & , \ otherwise. \end{cases}$$

Proof. For $b(G\overline{G})$ and $b_s(G\overline{G})$, let u be center vertex in the graph G and let $v, z \in V(G) - \{u\}$. $\{u, \overline{v}, \overline{z}\}$-set dominates (strong dominates) $V \cup \overline{V}$. It is easy to see that $\gamma(G\overline{G}) = \gamma_s(G\overline{G}) = 3$. $\gamma(G\overline{G})$-dominating set ($\gamma_s(G\overline{G})$-strong dominating set) must contain the vertex u, when the

edge $e_{u\overline{u}}$ is deleted from the graph $G\overline{G}$. Clearly, $\gamma(G\overline{G} - e_{u\overline{u}}) > \gamma(G\overline{G})$ ($\gamma_s(G\overline{G} - e_{u\overline{u}}) > \gamma_s(G\overline{G})$). So, we have $b(G\overline{G}) = b_s(G\overline{G}) = 1$.

For $b_w(G\overline{G})$, the graph G consists of $K_1 + C_n$. $\gamma_w(G\overline{G})$-weak dominating set includes the vertex $\overline{u}$ and vertices of $\gamma_w(C_n\overline{C}_n)$-weak dominating set. It can be easily seen that the proof of $b_w(G\overline{G})$ is similar to the proof of $b_w(C_n\overline{C}_n)$. □

Theorem 14. *If $G = K_{m,n}$ and $(m \leq n)$, then*

i) $b(G\overline{G}) = \begin{cases} 2, & m=2 \text{ and } m < n, \\ m, & \text{otherwise} \end{cases}$

ii) $b_s(G\overline{G}) = \begin{cases} 1, & m < n, \\ m, & m=n \end{cases}$

iii) $b_w(G\overline{G}) = \begin{cases} 2, & m=n=3, \\ 1, & \text{otherwise.} \end{cases}$

Proof. Let G_1 and G_2 be a partite sets of the graph G, whose cardinality are m and n, respectively. Clearly, m vertices are of order $(n+1)$ and these vertices are independent from each other. Similar to n vertices of order $(m+1)$, these vertices are also independent from each other. In $\overline{G}$, complete graphs K_m and K_n are formed by vertices of G_1 and G_2 [1].

For $b_s(G\overline{G})$, we recall $\gamma_s(G\overline{G}) = m+1$ by [1]. We must examine in two cases for the proof of $b_s(G\overline{G})$. Let $v \in V(G_1)$ and $u \in V(G_2)$.

Case 1. If $m < n$, $\gamma_s(G\overline{G})$-strong dominating set must include the vertex $\overline{u}$ since it is not strong dominated, where the edge $e_{u\overline{u}}$ is removed from the graph $G\overline{G}$. So, we have $\gamma_s(G\overline{G} - e_{u\overline{u}}) > \gamma_s(G\overline{G})$ and $b_s(G\overline{G}) = 1$.

Case 2. If $m = n \geq 3$, the graphs K_m and K_n are m-regular. There are two Sub Cases for this situation.

Subcase 2.1. The proof is similar to the proof of Case 1 of Theorem 11.

Subcase 2.2. The proof is similar to the proof of Case 2 of Theorem 11.

By Case 1 and Case 2, we have $b_s(G\overline{G}) = m$.

For $b(G\overline{G})$, we must examine in two cases for the proof of $b(G\overline{G})$.
Case 1. If $m = 2$ and $m < n$, the domination number of the graph $G\overline{G}$ is 3. $\gamma(G\overline{G})$-dominating set includes two vertices of the G_1 and any vertex of the graph K_n. The domination number does not change, an edge is removed from the graph $G\overline{G}$. So, $b(G\overline{G}) > 1$. The domination number increases by 1, when all edges are removed between G_1 and $V(K_n)$. Then we have $b(G\overline{G}) = 2$.

Case 2. If $m > 2$ and ($m < n$ or $m = n$), $\gamma(G\overline{G})$-dominating set and $\gamma_s(G\overline{G})$-strong dominating set are the same, where ($m = n \geq 3$). Clearly, the proof is similar to proof of the $b_s(G\overline{G})$, where for ($m = n \geq 3$). So, we obtain $b(G\overline{G}) = m$.

By Case 1 and Case 2, we have $b(G\overline{G}) = m$.

For $b_w(G\overline{G})$, we must examine in two cases for the proof of $b_w(G\overline{G})$.
Case 1. If $m \leq n$, we recall $\gamma_w(G\overline{G}) = n + 1$ by [1]. $\gamma_w(G\overline{G})$-weak dominating set must contain the vertex $\overline{u}$ when the edge $e_{u\overline{u}}$ is removed from the graph $G\overline{G}$, where $u \in V(G_2)$. So, $\gamma_w(G\overline{G} - e_{u\overline{u}}) > \gamma(G\overline{G})$, then we have $b_w(G\overline{G}) = 1$.

Case 2. If $m = n$, then let $m = n \neq 3$. The degrees of all vertices of G_1 and G_2 are $(m + 1)$. Similarly, degrees of all vertices of the graphs K_m and K_n are m. Degrees of vertices, which are incident with the edge e, decrease by one when an edge e is removed from the graph K_m. $\gamma_w(G\overline{G})$-weak dominating set must contain these vertices. So, $\gamma_w(G\overline{G} - e) > \gamma(G\overline{G})$, then we have $b_w(G\overline{G}) = 1$. Let $m = n = 3$. The weak domination number does not increase, when any edge e is removed from the graph K_m. Therefore, if any edge between graphs K_m and G_1 is removed, then the weak domination number increases by one. So, we have $b_w(G\overline{G}) = 2$. □

4 Conclusion

The characteristics of strong and weak dominating sets are not exhibited by the ordinary dominating sets and hence the problems of strong and weak bondage numbers for the graph are considerably harder than

bondage number of that. In this paper, the results of the bondage number, strong bondage number and weak bondage number of the complementary prisms of several well-known graphs have been obtained. As a further study, many general results of bondage parameters of complementary prism of any given graph G may be obtained.

Acknowledgement

The authors are grateful to the editors and the anonymous referees for their constructive comments and valuable suggestions which have helped us very much to improve the paper.

References

[1] A. Aytac and T. Turacı, "Strong Weak Domination in Complementary Prisms," *Dyna. of Cont. Disc. Impul. Sys. Series B: App. and Alg.*, vol. 22, no. 2b, pp. 85–96, 2015.

[2] A. Aytac and T. Turacı, "Bondage and Strong-Weak Bondage Numbers of Transformation Graphs G^{xyz}," *Inter. Jour.of Pure and App. Math.*, vol. 106, no. 2, pp. 689–698, 2016.

[3] A. Aytac, T. Turacı and Z.N. Odabas, On The Bondage Number of Middle Graphs, Mathematical Notes, vol. 93, no. 6, pp. 795–801, 2013.

[4] A. Aytac, Z.N. Odabas, and T. Turacı, "The Bondage Number for Some Graphs," *Comptes Rendus de Lacademie Bulgare des Sciences*, vol. 64, no. 7, pp. 925–930, 2011.

[5] A. Aytac and T. Turaci, "On the domination, strong and weak domination in transformation graph G^{xy^-}," *Utilitas Mathematica*, vol. 113, pp. 181–189, 2019.

[6] B.L. Hartnell and D.F. Rall, "Bounds on the bondage number of a graph," *Discrete Math.*, vol.28, pp. 173–177, 1994.

[7] D. Bauer, F. Harary, J. Nieminen, and C.L. Suffel, "Domination alteration sets in graph," *Discrete Math.*, vol. 47, pp. 153–161, 1983.

[8] D.B. West, *Introduction to Graph Theory*, 2nd ed., NJ, USA: Prentice Hall, 2001, xx+588 pages. ISBN: 0-13-014400-2.

[9] E. Sampathkumar and L. Pushpalatha, "Strong (weak) domination and domination balance in graph," *Discrete Math.*, vol. 161, pp. 235–242, 1996.

[10] F. Harary F. and F. Buckley, *Distance in Graphs*, Addison-Wesley Publishing Company, 1989.

[11] J.F. Fink, M.S. Jacobson, L.F. Kinch, and J. Roberts, "The bondage number of a graph," *Discrete Math.*, vol. 86, pp. 47–57, 1990.

[12] J. Ghoshal, R. Laskar, D. Pillone, and C. Wallis, "Strong bondage and strong reinforcement numbers of graphs," *Congressus numerantium* (Print in English), vol. 108, pp. 33–42, 1995.

[13] J. Kok and C.M. Mynhardt, "Reinforcement in Graphs," *Congressus numerantium*, vol.79, pp. 225–231, 1990.

[14] K. Ebadi and L. Pushpalatha, "Smarandachely Bondage number of a graph," *International J. Math. Combin.*, vol. 4, pp. 9–19, 2009.

[15] T.W. Haynes, S.T. Hedetniemi, and P.J. Slater, *Fundamentals of Domination in Graphs*, New York: Marcel Dekker, Inc., 1998.

[16] T.W. Haynes, M.A. Henning, and L.C. Van Der Merwe, "The complementary product of two graphs," *Bull. Instit. Combin. Appl.*, vol. 51, pp. 21–30, 2007.

[17] T.W. Haynes , M.A. Henning, and L.C. Van Der Merwe, "Domination and Total Domination in Complementary Prisms," *J. Comb. Optim.*, vol. 18, pp. 23–37, 2009.

[18] T. Turacı, "On the Average Lower Bondage Number of a Graph," *Rairo-Oper. Res.*, vol. 50, no. 4-5, pp. 1003–1012, 2016.

[19] T. Turacı and E. Aslan, "The Average Lower Reinforcement Number of a Graph," *Rairo-Theor. Inf. Appl.*, vol. 50, no. 2, pp. 135–144, 2016.

[20] T. Turaci and A. Aytac, "Combining the Concepts of Residual and Domination in Graphs," *Fundamenta Informaticae*, vol. 166, no. 4, pp. 379–392, 2019.

[21] T. Turaci, "On Combining the Methods of Link Residual and Dominationin Networks," *Fundamenta Informaticae*, vol. 174, no. 1, pp. 43–59, 2020.

Aysun Aytaç, Tufan Turacı

Received August 10, 2020
Accepted January 8, 2021

Aysun Aytaç
Ege University, Science Faculty, Mathematics Dept.
Bornova-IZMIR-TURKEY
Phone:+90 232 311 17 45
E–mail: aysun.aytac@ege.edu.tr

Tufan Turacı
Department of Computer Engineering, Faculty of Engineering,
Pamukkale University
20160, Denizli/TURKEY
E–mail: tturaci@pau.edu.tr
Phone: +90 258 296 30 62

Computer Science Journal of Moldova, vol.**29**, no.1(85), 2021

Some properties of maximum deficiency energy of a graph

Omendra Singh, Pravin Garg,
Neha Kansal

Abstract

The concept of maximum deficiency matrix $M_{df}(G)$ of a simple graph G is introduced in this paper. Let $G = (V, E)$ be a simple graph of order n and let $df(v_i)$ be the deficiency of a vertex v_i, $i = 1, 2, \ldots, n$, then the maximum deficiency matrix $M_{df}(G) = [f_{ij}]_{n\times n}$ is defined as:

$$f_{ij} = \begin{cases} max\{df(v_i), df(v_j)\}, & \text{if } v_iv_j \in E(G) \\ 0 \qquad\qquad\qquad\quad , & \text{otherwise.} \end{cases}$$

Further, some coefficients of the characteristic polynomial $\phi(G;\gamma)$ of the maximum deficiency matrix of G are obtained. The maximum deficiency energy $EM_{df}(G)$ of a graph G is also introduced. The bounds for $EM_{df}(G)$ are established. Moreover, maximum deficiency energy of some standard graphs is shown, and if the maximum deficiency energy of a graph is rational, then it must be an even integer.

Keywords: Deficiency, maximum deficiency matrix, maximum deficiency eigenvalues, maximum deficiency energy.

MSC 2020: 05C50.

1 Introduction

In this paper, it is assumed that all graphs are simple, finite and undirected. Let $v_1, v_2, \ldots, v_n$ be vertices of graph G, then number of edges incident to vertex v is called the degree and denoted by $d(v)$. The set

of vertices adjacent to v is called neighbourhood of v and denoted by $N(v)$.

Let G be a graph with maximum degree r and vertex set $V(G) = \{v_1, v_2, \ldots, v_n\}$. Then deficiency is $df(v_i) = r - d(v_i)$. We define the following two measures, α_j and β_j, that will be used throughout the paper:

$\alpha_j = |\{u \in N(v_j) : df(u) < df(v_j),\ 1 \leq j \leq n\}|$

$\beta_j = |\{u_k \in N(v_j), k > j : df(u_k) = df(v_j),\ 1 \leq j \leq n\}|$.

I. Gutman [5] proposed, for the first time, that the energy of a graph G is defined as $\mathcal{E}(G) = \sum_{j=1}^{n} |\gamma_j|$, where $\gamma_1, \gamma_2, \ldots, \gamma_n$ are the eigenvalues of the adjacency matrix of G. The theory of energy emerges from chemical sciences. In theoretical chemistry, the π-electron energy E in Huckel theory is the sum of the energies of all electrons in a molecule. Now-a-days the concept of graph energy is much studied in the mathematics literature ([7], [2], [11], [10], [6], [1], [9]).

Let G be a graph with vertices $v_1, v_2, \ldots, v_n$ and let $d(v_i)$ be the degree of v_i, then the maximum degree matrix $M(G) = [d_{ij}]_{n \times n}$ is defined as

$$d_{ij} = \begin{cases} max\{d(v_i), d(v_j)\}, & \text{if } v_i v_j \in E(G) \\ 0 \qquad\qquad\qquad\quad , & \text{otherwise.} \end{cases}$$

Now, the maximum degree energy of a graph G is defined as $EM(G) = \sum_{j=1}^{n} |\gamma_j|$, where $\gamma_1, \gamma_2, \ldots, \gamma_n$ are the eigenvalues of the maximum degree matrix of G.

Let G be a graph with vertices $v_1, v_2, \ldots, v_n$ and let $e(v_i)$ be the eccentricity of v_i, which is the maximum number of edges required to connect v_i to other vertices (or infinity in a disconnected graph), then the maximum eccentricity matrix $M_e(G) = [e_{ij}]_{n \times n}$ is defined as

$$e_{ij} = \begin{cases} max\{e(v_i), e(v_j)\}, & \text{if } v_i v_j \in E(G) \\ 0 \qquad\qquad\qquad\quad , & \text{otherwise.} \end{cases}$$

The maximum eccentricity energy of a graph G is defined as $EM_e(G) = \sum_{j=1}^{n} |\gamma_j|$, where $\gamma_1, \gamma_2, \ldots, \gamma_n$ are the eigenvalues of the maximum eccentricity matrix of G.

Got motivated by the maximum degree energy [1] and maximum eccentricity energy [9], the concept of maximum deficiency energy has been introduced and studied in the following sections.

2 The maximum deficiency energy of graphs

It is assumed that $G = (V, E)$ is a simple graph with vertex set $V = \{v_1, v_2, \ldots, v_n\}$, and let $df(v_i)$ be the deficiency of a vertex v_i. Then, the maximum deficiency matrix $M_{df}(G) = [f_{ij}]_{n\times n}$ is defined as:

$$f_{ij} = \begin{cases} max\{df(v_i), df(v_j)\}, & \text{if } v_iv_j \in E(G) \\ 0 \qquad\qquad\qquad\quad , & \text{otherwise.} \end{cases}$$

The characteristic polynomial of the maximum deficiency matix $M_{df}(G)$ will be

$$\begin{aligned} \phi(G;\gamma) &= det(\gamma I - M_{df}(G)) \\ &= \gamma^n + c_1\gamma^{n-1} + c_2\gamma^{n-2} + \cdots + c_n, \end{aligned}$$

where I is the identity matrix of order n. Suppose $\gamma_1, \gamma_2, \ldots, \gamma_n$ are roots of $\phi(G;\gamma) = 0$ which have been presumed to be in non-increasing order. These roots are the eigenvalues of the given matrix $M_{df}(G)$ and termed as the maximum deficiency eigenvalues of G.

The maximum deficiency energy of a graph G is defined as

$$EM_{df}(G) = \sum_{j=1}^{n} |\gamma_j|.$$

The above formulation suggests that $M_{df}(G)$ is a real and symmetric matrix with trace zero, and its eigenvalues are real numbers with sum equal to zero.

Remark 2.1. *The adjacency energy $\mathcal{E}(G)$, maximum degree energy $EM(G)$, maximum eccentricity energy $EM_e(G)$, and maximum deficiency energy $EM_{df}(G)$ are all well-defined for unlabeled graphs.*

Theorem 2.2. *Let G be a regular graph, then maximum deficiency energy $EM_{df}(G)$ is zero.*

Proof. Let G be a r-regular graph. We know that deficiency of every vertex in regular graph is always zero. Therefore, the maximum deficiency matrix is zero matrix and each eigen value of the matrix $M_{df}(G)$ is zero. Hence, maximum deficiency energy $EM_{df}(G)$ is zero. □

Example 2.3. *If G_1 is a graph in Figure 1,*

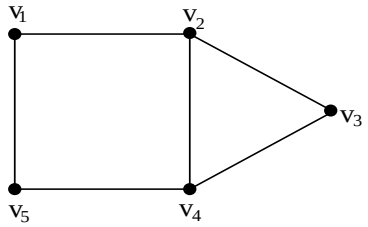

Figure 1. Graph G_1

then the maximum deficiency matrix of G_1 is

$$M_{df}(G_1) = \begin{pmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \end{pmatrix}.$$

The characteristic polynomial of $M_{df}(G_1)$ is

$$\phi(G_1;\gamma) = det(\gamma I - M_{df}(G_1))$$

$$= \begin{vmatrix} \gamma & -1 & 0 & 0 & -1 \\ -1 & \gamma & -1 & 0 & 0 \\ 0 & -1 & \gamma & -1 & 0 \\ 0 & 0 & -1 & \gamma & -1 \\ -1 & 0 & 0 & -1 & \gamma \end{vmatrix}$$

$$= \gamma^5 - 5\gamma^3 + 5\gamma - 2.$$

Then the maximum deficiency eigenvalues of G_1 are

$\gamma_1 = \gamma_2 = -1.618034,\ \gamma_3 = \gamma_4 = 0.618034, \gamma_5 = 2.$
Therefore,
Maximum deficiency energy $EM_{df}(G_1) = 6.472136.$
Adjacency energy $\mathcal{E}(G_1) = 6.340172.$
Maximum degree energy $EM(G_1) = 18.288101.$
Maximum eccentricity energy $EM_e(G_1) = 12.68034.$

Example 2.4. *Suppose the graph in Figure* 2 *is* G_2,

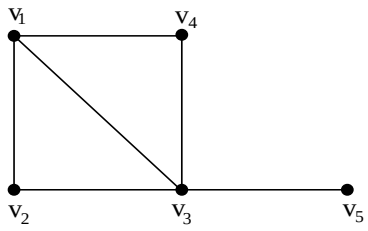

Figure 2. Graph G_2

then the maximum deficiency matrix of the graph G_2 *is*

$$M_{df}(G_2) = \begin{pmatrix} 0 & 2 & 1 & 2 & 0 \\ 2 & 0 & 2 & 0 & 0 \\ 1 & 2 & 0 & 2 & 3 \\ 2 & 0 & 2 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 \end{pmatrix}.$$

The characteristic polynomial of $M_{df}(G_2)$ *is*

$$\phi(G_2;\gamma) = det(\gamma I - M_{df}(G_2))$$

$$= \begin{vmatrix} \gamma & -2 & -1 & -2 & 0 \\ -2 & \gamma & -2 & 0 & 0 \\ -1 & -2 & \gamma & -2 & -3 \\ -2 & 0 & -2 & \gamma & 0 \\ 0 & 0 & -3 & 0 & \gamma \end{vmatrix}$$

$$= \gamma^5 - 26\gamma^3 - 16\gamma^2 + 72\gamma.$$

Then the maximum deficiency eigenvalues of G_2 *are*
$\gamma_1 = -4.2821879,\ \gamma_2 = -2.2862586, \gamma_3 = 0, \gamma_4 = 1.4317057,$
$\gamma_5 = 5.1367409.$
Therefore,
Maximum deficiency energy $EM_{df}(G_2) = 13.1368931.$
Adjacency energy $\mathcal{E}(G_2) = 6.040894.$
Maximum degree energy $EM(G_2) = 21.92652.$
Maximum eccentricity energy $EM_e(G_2) = 12.081788.$

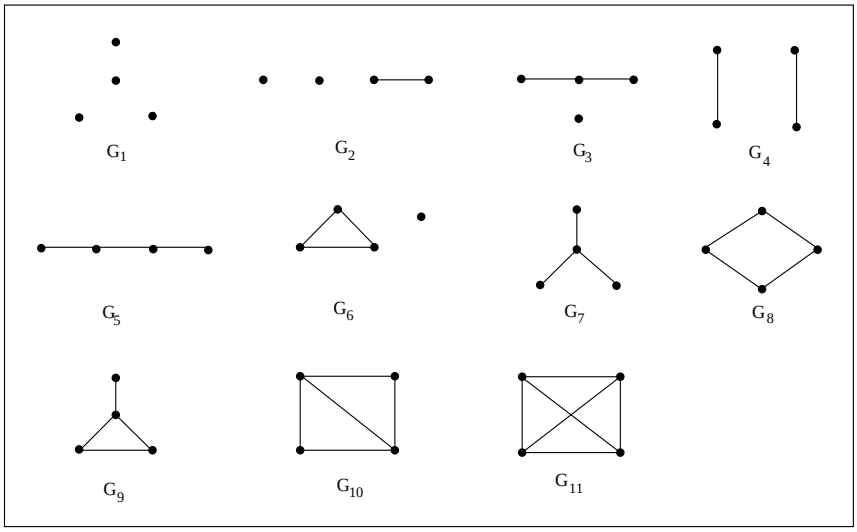

Figure 3. 11 Isomorphism classes of graphs of order 4

Table 1. Energies of isomorphism classes of graphs of order 4 as shown in Figure 3

	$\mathcal{E}(G)$	$EM(G)$	$EM_e(G)$	$EM_{df}(G)$
G_1	0	0	Not defined	0
G_2	2	2	Not defined	0
G_3	2.828427	5.656854	Not defined	2.828427
G_4	4	4	Not defined	0
G_5	4.472398	8.944796	12.649111	4
G_6	4	8	Not defined	0
G_7	3.464101	10.392303	6.928202	6.928202
G_8	4	8	8	0
G_9	4.962388	13.2915026	9.92477	6.646804
G_{10}	5.123105	15.369315	9.06226	4
G_{11}	6	18	6	0

We analyze from Table 1, in the connected graphs of order 4, the complete graph has the largest adjacency energy and the maximum degree energy, the path graph has the largest maximum eccentricity energy, while the star graph has the largest maximum deficiency energy. The maximum deficiency energy is not just complementary to maximum degree energy, since the non-regular graphs with smallest non-zero maximum deficiency energy do not always have large maximum degree energy.

3 Properties of maximum deficiency energy

In this section, the explicit expression for the coefficient c_i of γ^{n-i} $(i = 0, 1, 2, 3)$ in the characteristic polynomial $\phi(G; \gamma)$ of the maximum deficiency matrix $M_{df}(G)$ has been rendered. In addition to this, some properties of maximum deficiency eigenvalues of a graph G have been investigated.

Theorem 3.1. *If G is a simple graph of order n and $\phi(G; \gamma) =$*

$c_0\gamma^n + c_1\gamma^{n-1} + c_2\gamma^{n-2} + \ldots + c_n$ is the characteristic polynomial of the maximum deficiency matrix of G, then,

(i) $c_0 = 1$.

(ii) $c_1 = 0$.

(iii) $c_2 = -\sum_{j=1}^{n}(\alpha_j + \beta_j)df^2(v_j)$.

(iv) $c_3 = -2\sum_{\substack{\Delta v_i v_j v_k \\ df(v_i)\leq df(v_j)\leq df(v_k)}} df^2(v_k)df(v_j)$.

Proof. The proof is similar to Theorem 2.1 in [1]. □

Remark 3.2. (a) *The sum $\sum_{j=1}^{n}(\alpha_j + \beta_j)$ represents the total number of edges in the graph G.*

(b) *The total number of terms in the sum*

$$\sum_{\substack{\Delta v_i v_j v_k \\ df(v_i)\leq df(v_j)\leq df(v_k)}} df^2(v_k)df(v_j)$$

is equal to the number of triangles in the graph G.

(c) *$c_3 = 0$ if and only if the graph is a triangle-free graph.*

(d) *$c_n = 0$ if and only if $M_{df}(G)$ is singular.*

Example 3.3. *For the graph G_1 in Figure 1, the coefficient c_2 of γ^3 in $\phi(G_1;\gamma)$ is equal to*

$$-\sum_{j=1}^{5}(\alpha_j + \beta_j)df^2(v_j)$$

$$= -[(1+1)1^2 + (0+1)0^2 + (2+0)1^2 + (0+0)0^2 + (1+0)1^2]$$

$$= -5.$$

Example 3.4. *For the graph G_2 in Figure 2, the coefficient c_2 of γ^3 in $\phi(G_2;\gamma)$ is equal to*

$$-\sum_{j=1}^{5}(\alpha_j+\beta_j)df^2(v_j)$$

$$=-[(1+0)1^2+(2+0)2^2+(0+0)0^2+(2+0)2^2+(1+0)3^2]$$

$$=-[1+8+8+9]$$

$$=-26.$$

.

Theorem 3.5. *Let G be a graph of order n and $\gamma_1,\gamma_2,\gamma_3,\ldots,\gamma_n$ be the maximum deficiency eigenvalues of $M_{df}(G)$. Then*

(i) $\sum_{j=1}^{n}\gamma_j=0$.

(ii) $\sum_{j=1}^{n}\gamma_j^2=-2c_2$.

(iii) $\sum_{j=1}^{n}\gamma_j^3=-3c_3$.

Proof. The proof is based on the outcomes of Newton's identity [8] and Theorem 3.1. □

Remark 3.6. *The sum of the cubes of maximum deficiency eigenvalues is zero if and only if the graph is triangle free.*

Theorem 3.7. *If G is the complete bipartite graph $K_{m,n}$, $\gamma_1,\gamma_2,\gamma_3,\ldots,\gamma_{m+n}$ are its maximum deficiency eigenvalues and $m\leq n$, then*

(i) $\sum_{j=1}^{m+n}\gamma_j^2=2mn(n-m)^2$

(ii) $\sum_{j=1}^{m+n}\gamma_j^3=0$.

Proof. (i) Consider $K_{m,n}$ be the complete bipartite graph with vertices $u_1,u_2,\ldots,u_m,v_1,v_2,\ldots v_n$.

By Theorem 3.5,

$$\sum_{j=1}^{n} \gamma_j^2 = 2\sum_{j=1}^{n} (\alpha_j + \beta_j) df^2(v_j).$$

For the graph $K_{m,n}$

$$\alpha_j = 0, \forall u_j, j = 1, 2, \ldots, m,$$

$$\alpha_j = m, \forall v_j, j = 1, 2, \ldots, n,$$

$$\beta_j = 0, \forall u_j, j = 1, 2, \ldots, m,$$

$$\beta_j = 0, \forall v_j, j = 1, 2, \ldots, n,$$

$$df(u_j) = 0, \forall u_j, j = 1, 2, \ldots, m,$$

$$df(v_j) = n - m, \forall v_j, j = 1, 2, \ldots, n,$$

Then,

$$\sum_{j=1}^{m+n} \gamma_j^2 = 2mn(n-m)^2.$$

(ii) Since $K_{m,n}$ is a triangle free graph, then

$$\sum_{j=1}^{m+n} \gamma_j^3 = 0.$$

□

Theorem 3.8. *Let G be a graph of order n, in which $df(v) = df(u) = f$ and $N(v) - u = N(u) - v$. Then λ is a maximum deficiency eigenvalue of G, where*

$$\lambda = \begin{cases} -f, & \text{if } uv \in E(G) \\ 0, & \text{otherwise.} \end{cases}$$

Proof. The proof is similar to Theorem 2.5 in [1]. □

Theorem 3.9. *For a graph G, the maximum deficiency energy of G must be an even integer, if it is rational.*

Proof. We know that the characteristic polynomial of maximum deficiency matrix of a graph G is monic polynomial with integer coefficients. Therefore, roots of the characteristic polynomial are either integers or irrational numbers. Thus, the maximum deficiency energy of G is also either integer or an irrational number. Now, we know that the maximum deficiency energy of G is two times sum of positive maximum deficiency eigenvalues of G. Hence, if the maximum deficiency energy of G is rational, then it is always an even integer. □

4 The maximum deficiency energy of some classes of graphs

This section briefs about how the exact values of maximum deficiency eigenvalues and maximum deficiency energies of some well known graphs were obtained.

Theorem 4.1. *Let G be the star graph $K_{1,r-1}$ of order $r \geq 3$, then maximum deficiency eigenvalues of G are 0, $(r-2)\sqrt{(r-1)}$ and $-(r-2)\sqrt{(r-1)}$ with multiplicity $(r-2)$, 1 and 1, respectively and $EM_{df}(K_{1,r-1}) = 2(r-2)\sqrt{(r-1)}$.*

Proof. Let $K_{1,r-1}$ be the star graph with vertices $u_0, u_1, \ldots, u_{n-1}$,

where u_0 is the central vertex. Then we have

$$M_{df}(K_{1,r-1}) = \begin{pmatrix} 0 & r-2 & r-2 & \dots & r-2 \\ r-2 & 0 & 0 & \dots & 0 \\ r-2 & 0 & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ r-2 & 0 & 0 & \dots & 0 \end{pmatrix}.$$

The characteristic polynomial of $M_{df}(K_{1,r-1})$ is

$$\phi(K_{1,r-1};\gamma) = \begin{vmatrix} \gamma & -(r-2) & -(r-2) & \dots & -(r-2) \\ -(r-2) & \gamma & 0 & \dots & 0 \\ -(r-2) & 0 & \gamma & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ -(r-2) & 0 & 0 & \dots & \gamma \end{vmatrix}$$

$$= \gamma^r - (r-1)(r-2)^2\gamma^{r-2}$$

$$= \gamma^{r-2}(\gamma^2 - (r-1)(r-2)^2).$$

Hence, the maximum deficiency spectrum of $K_{1,r-1}$ is

$$M_{df}SP(K_{1,r-1}) = \begin{pmatrix} (r-2)\sqrt{r-1} & 0 & -(r-2)\sqrt{r-1} \\ 1 & r-2 & 1 \end{pmatrix}.$$

Hence, the maximum deficiency energy of $K_{1,r-1}$ is

$$EM_{df}(K_{1,r-1}) = 2(r-2)\sqrt{r-1}.$$

□

Remark 4.2. *In the above theorem, for $r = 1$ and $r = 2$, the maximum deficiency eigenvalues and maximum deficiency energy is zero.*

Theorem 4.3. *Let P_n be the path graph of order n, then the maximum deficiency eigenvalues of P_n are 0, 1 and -1 with multiplicity $(n-4)$, 2 and 2, respectively and $EM_{df}(P_n) = 4$.*

Proof. Let P_n be the path graph with vertices $v_1, v_2, \ldots, v_n$, then we have

$$M_{df}(P_n) = \begin{pmatrix} 0 & 1 & 0 & \ldots & 0 & 0 \\ 1 & 0 & 0 & \ldots & 0 & 0 \\ 0 & 0 & 0 & \ldots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & 0 & 1 \\ 0 & 0 & 0 & \ldots & 1 & 0 \end{pmatrix}.$$

The characteristic polynomial of $M_{df}(P_n)$ is

$$\phi(P_n;\gamma) = \begin{vmatrix} \gamma & -1 & 0 & \ldots & 0 & 0 \\ 1 & \gamma & 0 & \ldots & 0 & 0 \\ 0 & 0 & \gamma & \ldots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & \gamma & -1 \\ 0 & 0 & 0 & \ldots & -1 & \gamma \end{vmatrix}$$

$$= \gamma^n - 2\gamma^{n-2} - \gamma^{n-4}$$

$$= \gamma^{n-4}(\gamma^2 - 1)^2.$$

Hence, the maximum deficiency spectrum of P_n is

$$M_{df}SP(P_n) = \begin{pmatrix} 0 & 1 & -1 \\ n-4 & 2 & 2 \end{pmatrix}.$$

Hence, the maximum deficiency energy of P_n is

$$EM_{df}(P_n) = 4.$$

□

5 Bounds for maximum deficiency energy

The formulation of the lower bound and upper bound for the maximum deficiency energy of a graph has been detailed in this section.

Theorem 5.1. *Let G be a simple graph of order $n \geq 2$. Then*

$$\sqrt{-2c_2 + n(n-1)|det(M_{df}(G))|^{\frac{2}{n}}} \leq EM_{df}(G) \leq \sqrt{-2nc_2}.$$

Proof. The similar theorem is proved in [1] for maximum degree energy. We know that

$$(EM_{df}(G))^2 = \left(\sum_{j=1}^{n}|\gamma_j|\right)^2$$

$$= \left(\sum_{j=1}^{n}|\gamma_j|^2\right) + \left(\sum_{i\neq j}|\gamma_i||\gamma_j|\right).$$

From the arithemetic and geometric mean inequality, we get

$$\frac{1}{n(n-1)}\sum_{i\neq j}|\gamma_i||\gamma_j| \geq \left(\prod_{i\neq j}|\gamma_i||\gamma_j|\right)^{1/n(n-1)}.$$

By Theorem 3.5 and this inequality, we get

$$(EM_{df}(G))^2 \geq \left(\sum_{j=1}^{n}|\gamma_j|^2\right) + n(n-1)\left(\prod_{i\neq j}|\gamma_i||\gamma_j|\right)^{1/n(n-1)}$$

$$\geq \left(\sum_{j=1}^{n}|\gamma_j|^2\right) + n(n-1)|\prod_{j=1}\gamma_j|^{2/n}$$

$$\geq -2c_2 + n(n-1)|det(M_{df}(G))|^{\frac{2}{n}}.$$

For the upper bound, consider the cauchy-schwartz inequality

$$\left(\sum_{j=1}^{n}|x_jy_j|\right)^2 \leq \left(\sum_{j=1}^{n}|x_j|^2\right)\left(\sum_{j=1}^{n}|y_j|^2\right).$$

By choosing $x_j = 1$ and $y_j = |\gamma_j|$, we get

$$EM_{df}(G) \leq \sqrt{n\sum_{j=1}^{n}|\gamma_j|^2}$$
$$= \sqrt{-2nc_2}.$$

Hence,

$$\sqrt{-2c_2 + n(n-1)|det(M_{df}(G))|^{\frac{2}{n}}} \leq EM_{df}(G) \leq \sqrt{-2nc_2}.$$

□

Remark 5.2. *For the graph shown in Figure* 2, $c_2 = -26$. *Thus, from Theorem 5.1,*

$$\sqrt{52} \leq EM_{df}(G_2) \leq \sqrt{260}$$
$$7.211 \leq 13.136 \leq 16.1245.$$

Lemma 5.3. [3].

Suppose that x_j and y_j, $1 \leq j \leq n$, are non-negative real numbers. Then

$$\left| n\sum_{j=1}^{n} x_j y_j - \sum_{j=1}^{n} x_j \sum_{j=1}^{n} y_j \right| \leq \alpha(n)(X-x)(Y-y),$$

where x, y, X and Y are real constants, such that for each j, $1 \leq j \leq n$, the conditions $x \leq x_j \leq X$ and $y \leq y_j \leq Y$ are satisfied. Further, $\alpha(n) = n\left[\frac{n}{2}\right]\left(1 - \frac{1}{n}\left[\frac{n}{2}\right]\right)$, while [x] denotes integer part of a real number x.

Lemma 5.4. [12]. *Suppose that x_j and y_j, $1 \leq j \leq n$, are non-negative real numbers. Then*

$$\sum_{j=1}^{n} x_j^2 \sum_{j=1}^{n} y_j^2 \leq \frac{1}{4}\left(\sqrt{\frac{M_1M_2}{m_1m_2}} + \sqrt{\frac{m_1m_2}{M_1M_2}}\right)^2 \left(\sum_{j=1}^{n} x_j y_j\right)^2,$$

where $M_1 = \max_{1\leq j\leq n}(x_j), M_2 = \max_{1\leq j\leq n}(y_j), m_1 = \min_{1\leq j\leq n}(x_j)$ and $m_2 = \min_{1\leq j\leq n}(y_j)$.

Lemma 5.5. [4].

Suppose that x_j and y_j, $1 \leq j \leq n$, are non-negative real numbers. Then

$$\sum_{j=1}^{n} y_j^2 + rR\sum_{j=1}^{n} x_j^2 \leq (r+R)\left(\sum_{j=1}^{n} x_j y_j\right),$$

where r, R are real constants, such that for each j, $1 \leq j \leq n$, the conditions $rx_j \leq y_j \leq Rx_j$ are satisfied.

Theorem 5.6. *Suppose G is a graph with n vertices. Let γ_j, $j = 1, 2, \ldots, n$ be the maximum deficiency eigenvalues of G. Let $\gamma_{min} = \min\limits_{1\leq j\leq n}(|\gamma_j|)$ and $\gamma_{max} = \max\limits_{1\leq j\leq n}(|\gamma_j|)$ and $\alpha(n) = n\left[\frac{n}{2}\right]\left(1 - \frac{1}{n}\left[\frac{n}{2}\right]\right)$, while [x] denotes integer part of a real number x.*

Then

$$EM_{df}(G) \geq \sqrt{-2nc_2 - \alpha(n)(\gamma_{max} - \gamma_{min})^2}. \tag{1}$$

Proof. Applying Lemma 5.3 and putting $x_j = |\gamma_j| = y_j$, $x = \gamma_{min} = y$ and $X = \gamma_{max} = Y$ imply that

$$\left| n\sum_{j=1}^{n} |\gamma_j|^2 - \left(\sum_{j=1}^{n} |\gamma_j\right)^2 \right| \leq \alpha(n)(\gamma_{max} - \gamma_{min})^2.$$

By Theorem 3.5, we get

$$-2nc_2 - EM_{df}(G)^2 \leq \alpha(n)(\gamma_{max} - \gamma_{min})^2.$$

Hence

$$EM_{df}(G) \geq \sqrt{-2nc_2 - \alpha(n)(\gamma_{max} - \gamma_{min})^2}.$$

□

Corollary 5.7. *Since $\alpha(n) \leq \frac{n^2}{4}$, then by Theorem 5.6, we get*

$$EM_{df}(G) \geq \sqrt{-2nc_2 - \frac{n^2}{4}(\gamma_{max} - \gamma_{min})^2}. \tag{2}$$

Remark 5.8.

$$EM_{df}(G) \geq \sqrt{-2nc_2 - \alpha(n)(\gamma_{max} - \gamma_{min})^2} \geq$$
$$\geq \sqrt{-2nc_2 - \frac{n^2}{4}(\gamma_{max} - \gamma_{min})^2}.$$

Thus, inequality (1) is stronger than inequality (2).

Theorem 5.9. *Suppose G is a graph with n vertices. Let γ_j, $j = 1, 2, \ldots, n$ be the maximum deficiency eigenvalues of G. If zero is not an eigenvalue of $M_{df}(G)$, then*

$$EM_{df}(G) \geq \frac{2\sqrt{-2nc_2\gamma_{max}\gamma_{min}}}{\gamma_{max} + \gamma_{min}}, \tag{3}$$

where $\gamma_{min} = \min\limits_{1\leq j\leq n}(|\gamma_j|)$ and $\gamma_{max} = \max\limits_{1\leq j\leq n}(|\gamma_j|)$.

Proof. Using Lemma 5.4 for $x_i = |\gamma_j|$ and $y_j = 1$, we get

$$\sum_{j=1}^{n} |\gamma_j|^2 \sum_{j=1}^{n} 1^2 \leq \frac{1}{4}\left(\sqrt{\frac{\gamma_{max}}{\gamma_{min}}} + \sqrt{\frac{\gamma_{min}}{\gamma_{max}}}\right)^2 \left(\sum_{j=1}^{n} |\gamma_j|\right)^2$$

$$\Rightarrow -2nc_2 \leq \frac{1}{4}\left(\sqrt{\frac{\gamma_{max}}{\gamma_{min}}} + \sqrt{\frac{\gamma_{min}}{\gamma_{max}}}\right)^2 EM_{df}(G)^2.$$

Hence

$$EM_{df}(G) \geq \frac{2\sqrt{-2nc_2\gamma_{max}\gamma_{min}}}{\gamma_{max} + \gamma_{min}},$$

where $\gamma_{min} = \min\limits_{1\leq j\leq n}(|\gamma_j|)$ and $\gamma_{max} = \max\limits_{1\leq j\leq n}(|\gamma_j|)$.

□

Theorem 5.10. *Suppose G is a graph of order n. Let $\gamma_j, i = 1, 2, \ldots, n$ be the maximum deficiency eigenvalues of G. Then*

$$EM_{df}(G) \geq \frac{-2c_2 + n\gamma_{max}\gamma_{min}}{\gamma_{max} + \gamma_{min}}, \tag{4}$$

where $\gamma_{min} = \min\limits_{1\leq j\leq n}(|\gamma_j|)$ and $\gamma_{max} = \max\limits_{1\leq j\leq n}(|\gamma_j|)$.

Proof. Using Lemma 5.5 and setting $x_j = 1, y_j = |\gamma_j|, r = \gamma_{min}$ and $R = \gamma_{max}$, we get

$$\sum_{j=1}^{n} |\gamma_j|^2 + \gamma_{min}\gamma_{max} \sum_{j=1}^{n} 1 \leq (\gamma_{min} + \gamma_{max}) \sum_{j=1}^{n} |\gamma_j|$$

$$\Rightarrow -2c_2 + n\gamma_{min}\gamma_{max} \leq (\gamma_{min} + \gamma_{max}) EM_{df}(G).$$

Hence

$$EM_{df}(G) \geq \frac{-2c_2 + n\gamma_{max}\gamma_{min}}{\gamma_{max} + \gamma_{min}},$$

where $\gamma_{min} = \min_{1 \leq j \leq n} (|\gamma_j|)$ and $\gamma_{max} = \max_{1 \leq j \leq n} (|\gamma_j|)$. □

Remark 5.11. Using inequality between arithmetic and geometric means,

$$EM_{df}(G) \geq \frac{-2c_2 + n\gamma_{max}\gamma_{min}}{\gamma_{max} + \gamma_{min}} \geq \frac{2\sqrt{-2nc_2\gamma_{max}\gamma_{min}}}{\gamma_{max} + \gamma_{min}}.$$

Thus, inequality (4) is stronger than inequality (3).

Remark 5.12. *There are analogous bounds for the adjacency energy $\mathcal{E}(G)$, the maximum degree energy $EM(G)$ and the maximum eccentricity energy $EM_e(G)$*

6 Conclusion

A newly developed matrix of a graph G called maximum deficiency matrix $M_{df}(G)$ has been introduced in this paper. The underlying graph along with the deficiency on its vertices are found to be the influencing factors of it. Some coefficients of the characteristic polynomial of the maximum deficiency matrix are found. For a graph, the maximum deficiency energy $EM_{df}(G)$ has been formulated and the upper and lower bounds have been obtained. It is possible that the maximum deficiency energy that we are considering in this paper may have some applications in chemistry as well as in other areas.

Acknowledgement

We are greatly indebted to the referee for his critical suggestions that led us to refine our results.

References

[1] C. Adiga and M. Smitha, "On maximum degree energy of a graph," *Int. J. Contemp. Math. Science*, vol. 4, no. 8, pp. 385–396, 2009.

[2] R. Balakrishanan, "The energy of graph," *Linear Algebra Appl.*, vol. 387, pp. 287–295, 2004.

[3] M. Biernacki, H. Pidek, and C. R. Nardzewski, "Sur une inegalite entre des integrales definies," *Ann. Univ. Mariae Curie-Skolodowska*, A4, pp. 1–4, 1950. (in French)

[4] J. B. Diaz and F. T. Metcalf, "Stronger forms of a class of inequalities of G.Polya-G.Szego and L.V. Kantorovich," *Bull. Am. Math. Soc.*, vol. 69, pp. 415–418, 1963.

[5] I. Gutman, "The energy of a graph," *Ber. Math-Stat. Sekt. Forschungszent Graz*, vol. 103, pp. 1–22, 1978.

[6] A. Jahanbani, "Some new lower bounds for energy of graphs," *Appl. Math. Comput.*, vol. 296, pp. 233–238, 2017.

[7] X. Li, Y. Shi, and I. Gutman, *Graph Energy*, New York: Springer, 2012.

[8] D. Kalaman, "A matrix proof of newton's identities," *Math. Magz.*, vol. 73, no. 4, pp. 313–315, 2000.

[9] A. M. Naji and N. D. Soner, "The maximum eccentricity energy of a graph," *Int. J. Sci. Eng. Research*, vol. 7, no. 5, pp. 5–13, 2016.

[10] V. Nikiforov, "Graphs and matrices with maximal energy," *J. Math. Anal. Appl.*, vol. 327, no. 1, pp. 735–738, 2007.

[11] M. R. Oboudi, "A new lower bound for the energy of graphs," *Linear Algebra appl.*, vol. 580, pp. 384–395, 2019.

[12] G. Polya and G. Szego, *Problems and theorems in analysis, Series, Integral Calculus, Theory of Functions*, Berlin: Springer, 1972.

Omendra Singh, Pravin Garg,
Neha Kansal

Received March 18, 2020
Revised December 23, 2020

Omendra Singh
Department of Mathematics,
University of Rajasthan,
Jaipur - 302004, India
Phone: +919521531401
E–mail: omendrasingh3003@gmail.com

Pravin Garg
Department of Mathematics,
University of Rajasthan,
Jaipur - 302004, India
Phone: +919982123280
E–mail: garg.pravin@gmail.com

Neha Kansal
Department of Mathematics,
University of Rajasthan,
Jaipur - 302004, India
Phone: +918952836930
E–mail: kansalneha.ngr@gmail.com

Computer Science Journal of Moldova, vol.**29**, no.1(85), 2021

Connected Domination Number and a New Invariant in Graphs with Independence Number Three

Vladimir Bercov

Abstract

Adding a connected dominating set of vertices to a graph G increases its number of Hadwiger $h(G)$. Based on this obvious property in [2] we introduced a new invariant $\eta(G)$ for which $\eta(G) \leq h(G)$. We continue to study its property. For a graph G with independence number three without induced chordless cycles C_7 and with $n(G)$ vertices, $\eta(G) \geq n(G)/4$.

Keywords: dominating set, number of Hadwiger, clique number, independence number.

1 Introduction

All graphs considered in this paper are undirected, simple and finite. Let G be a graph with vertex set $V(G)$. We denote $|V(G)|$ by $n(G)$. Let $X \subseteq V(G)$, X is connected if the subgraph $G[X]$ induced by X is connected. Further, $G-X = G[V(G)-X]$. X is dominating in a graph G if every vertex of G is in X or has a neighbor in X. We will write $v \sim u$ ($v \nsim u$) when vertices v and u are (are not) adjacent. If every pair of vertices in X are adjacent, then $G[X]$ is a complete subgraph or a clique K_n, where $n = |X|$. The clique number $\omega(G)$ of a graph G is the number of vertices in a maximum clique in G. The degree of a vertex v is $deg(v)$, the number of edges that are incident to the vertex. The maximum degree $\Delta(G)$ and the minimum degree $\delta(G)$ of a graph G are the maximum and the minimum degree of its vertices. A k-colouring of G is a function that assigns one of k colours to each vertex of G such

that adjacent vertices receive distinct colours. The chromatic number $\chi(G)$ is the minimum integer k such that G is k-colourable. A graph H is a minor of the graph G if H can be formed from G by deleting edges and vertices and by contacting edges. The Hadwiger number $h(G)$ is the maximum integer n such that the complete graph K_n is a minor of G. Further, a cycle $C_n = (v_1, v_2, \ldots, v_l)$ is a chordless cycle of length n, and any dominating set is a connected dominating set.

In [2] we introduced a new invariant $\eta = \eta(G)$ of a graph G as a maximum length of a sequence of subsets of its vertices $V_1, V_2, V_3, \ldots$, where $V_i \cap V_j = \varnothing$ $(i \neq j)$ and V_k is a dominating set in a graph $G[V_1 \cup V_2 \cup \ldots \cup V_k]$, $k = 1, 2, \ldots, \eta$. In the mentioned paper we have proved some properties of $\eta(G)$:

(i) $\omega(G) \leq \eta(G) \leq h(G)$,

(ii) If D is any dominating set in G, then $\eta(G) \geq \eta(G - D) + 1$,

(iii) $\eta(G) \leq \Delta(G) + 1$,

(iv) $\eta(G) \geq \chi(G)$ if $\chi(G) \leq 4$,

and we have posed the stronger than Hadwiger's conjecture:

Conjecture 1. *For all graphs G, $\chi(G) \leq \eta(G)$.*

2 Vertex cut sets and one more property of the new invariant

We say that a graph G is η-critical if $\eta(G) = \eta$ and $\eta(H) < \eta$ for every proper subgraph H of G. It is obvious that any η-critical graph is connected. 2-critical graph is an edge, 3-critical graph is a cycle. It's clear that if graph G is η-critical, $\eta(G - D) = \eta - 1$ and D is a dominating set in G, then $G - D$ is $(\eta - 1)$-critical.

A vertex cut set of a connected graph G is a subset $S \subseteq V(G))$ such that $G - S$ has more than one connected component. A subgraph induced by a vertex cut set is a cut subgraph.

Let S be a vertex cut set of a connected graph G, and the components of $G - S$ have vertex sets $U_1, U_2, \ldots, U_m$, $m \geq 2$. Denote $G_i = G[U_i \cup S]$.

Theorem 1. *Let $G[S]$ be complete, then for all G_i and for each its induced subgraph G'_i, the following is true: if $D \subseteq V(G) - V(G'_i)$ and D is a dominating set in a graph $G[D \cup V(G'_i)]$, then there exists D_i such that $D_i \subseteq D$, $D_i \subseteq V(G_i)$ and D_i is a dominating set in a graph $G[D_i \cup V(G'_i)]$.*

Proof. $D = A \cup S_i \cup B$, where $A \subseteq U_i$ $S_i \subseteq S$, $B \cap V(G_i) = \varnothing$. Since $G[S_i]$ is complete, then $G[A \cup S_i]$ is connected and all vertices of subgraph G'_i are joined with at least one vertex of the set $A \cup S_i$. So we can take $D_i = A \cup S_i$. ■

Theorem 2. *If S is a vertex cut set of a connected graph G and $G[S]$ is complete, then $\eta(G) = max_{1 \le i \le m} \eta(G_i)$.*

Proof. From definition $\eta(G) \ge max_{1 \le i \le m} \eta(G_i)$. Let $\eta = \eta(G)$ and $V_1, V_2, \ldots, V_\eta$ be a longest sequence of the sets of vertices in the definition of $\eta(G)$. If $\eta = |S|$, then $\eta(G_i) \ge \eta$ for all i, and the theorem is true. If $\eta > |S|$, then at least one set V_j contains vertices from set $V(G) - S$. Let V_{j_0} be the first such set from the sequence and let V_{j_0} contains vertices from U_{i_0}. By Theorem 1, there exists a dominating set $V'_{j_0}, V'_{j_0} \subseteq V_{j_0}$ containing only vertices of the set V_{i_0}. Therefore, there exists a sequence $V_1, \ldots, V_{j_0-1}, V'_{j_0}, V'_{j_0+1}, \ldots, V'_\eta$ of dominating sets of vertices of the graph $G_{i_0} = G[U_{i_0} \cup S]$ and therefore, $\eta(G_{i_0}) \ge \eta = \eta(G)$. ■

Corollary 1. *If G is η-critical, then G does not contain complete cut subgraphs.*

Proof. Let G be η-critical, $G[S]$ is complete. If we suppose that graph $G - S$ has components with not empty vertex sets U_1, U_2 and $\eta(G_1) \ge \eta(G_2)$, where $G_1 = G[U_1 \cup S]$ and $G_2 = G[U_2 \cup S]$, then, by Theopem 2, $\eta(G) = \eta(G_!)$ and therefore, G is not η-critical.

In [2] we proved that $\eta(G) \ge 4$ for graphs G with $\delta(G) \ge 3$. Using Theorem 2, we can slightly strengthen this result.

Corollary 2. *If degrees of all vertices of a graph G are at least three, except maybe one case from three: (a) one vertex of degree one, (b) one vertex of degree two, (c) two adjacent vertices of degree two, then $\eta(G) \ge 4$.*

Proof. (a) Let in graph G $deg(v) = 1$, $v \sim u$ and graph G' is isomorphic to G. Let in graph G' $deg(v') = 1$, $v' \sim u'$. From disjoint union of G and G' we form a new graph H by identifying v and u', u and v'. Since $\delta(H) \geq 3$, $\eta(H) \geq 4$ and by Theorem 2, $\eta(G) = \eta(H)$.

(b) Let in graph G $deg(v) = 2$, G' is isomorphic to G and in graph G' $deg(v') = 2$. We form a new graph H by identifying v and v'. Since $\delta(H) \geq 3$, $\eta(G) = \eta(H) \geq 4$.

(c) Let in graph G $deg(v) = deg(u) = 2$, $v \sim u$ and G' is isomorphic to G. Let in graph G' $deg(v') = deg(u') = 2$ and $v' \sim u'$. We form a new graph H by identifying v and v', u and u'. Since $\delta(H) \geq 3$, $\eta(G) = \eta(H) \geq 4$.

3 Domination and independence number

We need to introduce more notations. In a graph G, the independence number $\alpha(G)$ is the maximum cardinality of an independent set. The connected domination number $\gamma_c(G)$ is the number of vertices in the minimum connected dominating set. The neighborhood of vertex $v \in V(G)$, denoted by $N(v)$, is a set of all vertices adjacent to v. The closed neighborhood of v is $N[v] = N(v) \cup v$. A simplicial vertex of a graph G is a vertex v for which $G[N(v)]$ is complete.

Duchet and Meyniel [3] proved that $\gamma_c(G) \leq 2\alpha(G) - 1$ for any graph G. It is clear that $\alpha(C_{2l+1}) = l$, $\gamma_c(C_{2l+1}) = 2l - 1$, and for these graphs $\gamma_c(C_{2l+1}) = 2\alpha(C_{2l+1}) - 1$. Other upper bounds include additional graph parameters or conditions (see [1, 4, 5]). Plummer, Stiebitz and Toft [6] proved for any connected graph G that if $\alpha(G) = 2$ and G does not contain C_5, then for any non-simplicial vertex v graph G contains a dominating edge vu.

A claw $K_{1,3}$ is a graph with four vertices for which one vertex has three pairwise nonadjacent neighbors. It is clear that if $\alpha(G) = 3$ and graph G has a claw $K_{1,3}$ as an induced subgraph, then $V(K_{1,3})$ is a dominating set.

Theorem 3. *If graph G is connected, claw-free, $\alpha(G) = 3$, and G does not contain an induced C_7, then for any non-simplicial vertex v there*

exists connected dominating set D, such that $v \in D$ and $n(D) \leq 4$.

Proof. Let $v \sim v_1, v \sim v_2, v_1 \nsim v_2$. Denote by V_1 the set of neighbors of v_1 which are not neighbors of v_2, by V_2 – the set of neighbors of v_2 which are not neighbors of v_1 and by V_{12} – the set of vertices adjacent to both v_1 and v_2. Denote by G' subgraph $G - N[v]$ and by V'_{12} – the set of vertices $V(G') - (V_1 \cup V_{12} \cup V_2)$. Since $\alpha(G) = 3$, $\alpha(G') \leq 2$, and subgraph $G[V'_{12}]$ is complete or $V'_{12} = \varnothing$.

If $V'_{12} = \varnothing$, then $D = v, v_1, v_2$. If $V'_{12} \neq \varnothing$ and $V_1 \cup V_2 \cup V_{12} = \varnothing$, then $D = v, a, b$, where a is any vertex adjacent to v and to vertex $b \in V'_{12}$. Let $V'_{12} \neq \varnothing$ and $V_1 \cup V_2 \cup V_{12} \neq \varnothing$. Since G is claw-free, induced subgraphs $G[V_1 \cup V_{12}]$ and $G[V_2 \cup V_{12}]$ are complete (one of them can be null graph), and if vertex $u \in V_{12}$, then u does not have neighbors in $G[V'_{12}]$.

Case 1. G' is connected.

1.1 G' contains $C_5 = (u_1, u_2, u_3, u_4, u_5)$.

Sets $V_1 \cup V_{12}$, $V_2 \cup V_{12}$, V'_{12} contain at most two consecutive vertices of C_5, and V_{12} contains at most one. Since G is claw-free, if V_{12} does not contain vertex of C_5, then V_1 and V_2 contain two vertices each. There are two possible cases:

1.1.1 $u_1 \in V_1$, $u_2 \in V_{12}$, $u_3 \in V_2$, $\{u_4, u_5\} \subseteq V'_{12}$. See Figure 1.

1.1.2 $\{u_1, u_2\} \subseteq V_1$, $\{u_3, u_4\} \in V_2$, $u_5 \subseteq V'_{12}$.

Since $\alpha(G') = 2$, any pair of nonadjacent vertices from V_1 and V_2 is adjacent to all vertices from V'_{12}. If u_4 is connected to all vertices V'_{12}, then $D = \{v, v_1, v_2, u_4\}$. If $u \in V'_{12}$ and $u_4 \nsim u$, then $u_2 \sim u$. See Figure 2.

1.2 G' does not contain C_5.

1.2.1 G' is complete.

In this case there exists a dominating set $D = \{v, a, b\}$, where a is any vertex such that $a \sim v$, $a \sim b$, where b is any vertex from $V(G')$.

1.2.2 G' is not complete.

In subgraph G' for any non-simplicial vertex $b \in V(G')$ there exists an edge bc dominating in G' (see [6]). If b is adjacent to $a \in N(v)$, then $D = \{v, a, b, c\}$. Now let all vertices of G', which are connected to $N(v)$, be simplicial.

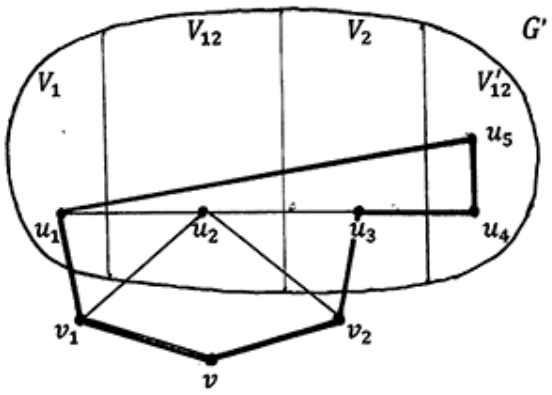

Figure 1. $C_7 = (v, v_2, u_3, u_4, u_5, u_1, v_1)$

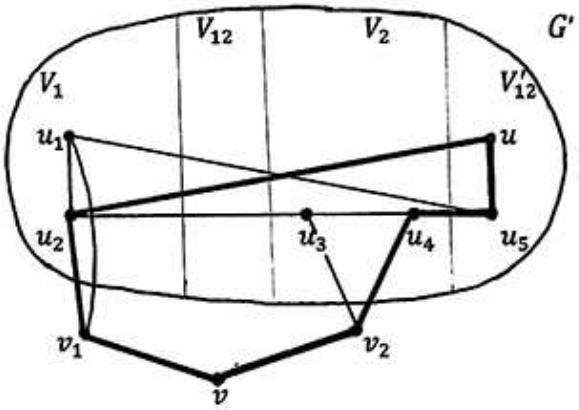

Figure 2. $C_7 = (v, v_2, u_4, u_5, u, u_2, v_1)$

1.2.2.1 $V_{12} \neq \varnothing$.

In this case subgraph $G[V_1 \cup V_2 \cup V_{12}]$ is complete and, since G' is connected, there exists an edge ab, where $b \in V'_{12}$ and $a \in V_1$ (or $a \in V_2$). Therefore, $D = \{v, v_1, a, b\}$ (or $D = \{v, v_2, a, b\}$).

1.2.2.2 $V_{12} = \varnothing$.

Since all vertices of the sets V_1 and V_2 are simplicial, $G[V_1 \cup V_2]$ is complete (and we have the same dominating set as in case 1.2.2.1) or V_1 does not have neighbors in V_2. In the last case any two vertices $u_1 \in V_1$ and $u_2 \in V_2$ are adjacent to all vertices V'_{12}. If one of these two vertices is adjacent to all V'_{12}, then $D = \{v, v_1, v_2, u_1\}$ or $D = \{v, v_1, v_2, u_2\}$. Otherwise, there exist two vertices $u_3, u_4 \in V'_{12}$ such that $u_1 \nsim u_3$ and $u_2 \nsim u_4$. See Figure 3.

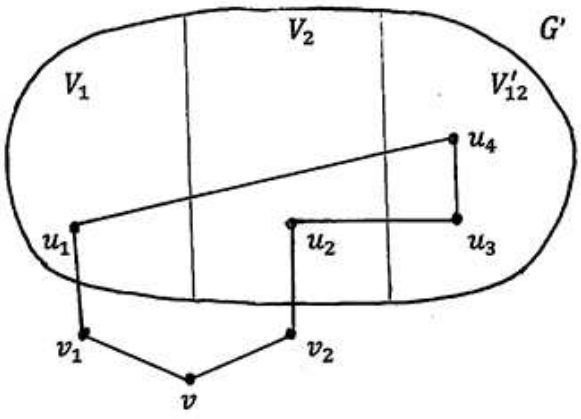

Figure 3. $C_7 = (v, v_1, u_1, u_4, u_3, u_2, v_2)$

Case 2. G' is not connected.

Since $V_1 \cup V_2 \cup V_{12} \neq \varnothing$, $V'_{12} \neq \varnothing$, $\alpha(G') = 2$, subgraph G' is a disjoint union of two complete subgraphs $G'_1 = G[V_1 \cup V_2 \cup V_{12}]$ and $G'_2 = G[V'_{12}]$. Since G is claw-free, any vertex $a \in N(v)$ is adjacent to at least one vertex v_1 or v_2, and if a has neighbors in G'_2, then a is adjacent to exactly one. Since $\alpha(G) = 3$, if a has neighbors in G'_2, $a \sim v_1$, $a \nsim v_2$, then a is adjacent to all vertices V'_{12} or $V_1 = \varnothing$. If a is adjacent to all

vertices V'_{12}, then we can take $D = \{a, v, v_2, b\}$, where $b \in V_2 \cup V_{12}$ or $D = \{a, v, v_1\}$ if $V_2 \cup V_{12} = \varnothing$. If $V_1 = \varnothing$, then $D = \{v_2, v, a, c\}$, where $c \sim a$ and $c \in V'_{12}$. ■

Remark 1. *The graph G shown in Figure 4 does not contain C_7, $\alpha(G) = 3$, and the set of vertices $\{a, v_1, v_2, b\}$ induced a claw. In this graph any connected dominating set with vertex v, contains at least five vertices.*

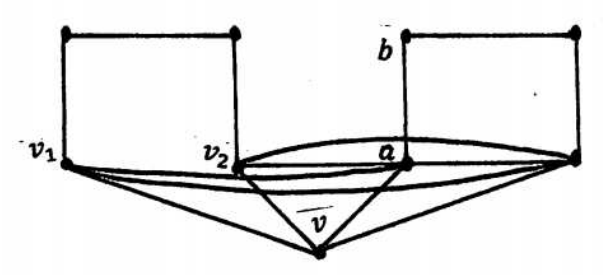

Figure 4. v is a non-simplicial vertex, $\{a, v_1, v_2, b\}$ is a vertex set of a claw

Corollary 3. *Let G be a graph with $\alpha(G) = 3$. If G does not contain an induced C_7, then $h(G) \geq n(G)/4$.*

Proof. We proceed by induction on $n = n(G)$. For $n \leq 4$, the result is clear. Suppose $n \geq 5$ and suppose the result is true for all graphs with fewer than n vertices and let G be a graph with n vertices. If G contains a claw, then the set D of vertices of this claw is dominating in G, if not, by Theorem 3, we can build a dominating set D with $n(D) \leq 4$. In both cases

$$h(G) \geq h(G - D) + 1 \geq \frac{n(G - D)}{4} + 1 \geq \frac{n(G) - 4}{4} + 1 = n(G)/4.$$

References

[1] G.Bacso and Z.Tuza, "Graphs without induced P_5 and C_5," *Discuss. Math. Graph Theory*, vol. 24, pp. 503–507, 2004.

[2] V. Bercov, "Connected Dominating Sets and a New Graph Invariant," *Computer Science Journal of Moldova*, vol. 27, no. 1(79), pp. 23–33, 2019.

[3] P. Duchet and H. Meyniel, "On Hadwiger's Number and the Stability Number," in *Graph Theory* (North-Holland Mathematical Studies, vol. 62), Bela Bollobas, Ed. Amsterdam: North-Holland, 1982, pp. 71–73.

[4] P. Kaemawichanurat and T. Jiarasuksakun, "Some results on the independence number of connected domination critical graphs," *AKCE International Journal of Graphs and Combinatorics*, vol. 15, pp. 190–196, 2018.

[5] S. Mukwembi, "Size, order, and connected domination," *Canad. Math. Bull.*, vol. 57(1), pp. 141–144, 2014.

[6] M. D. Plummer, M. Stiebitz, and B. Toft, "On a special case of Hadwiger's Conjecture," *Discuss. Math. Graph Theory*, vol. 23, no. 2, pp. 333–363, 2003.

Vladimir Bercov

Department of Mathematics
CUNY Borough of Manhattan Community College
199 Chambers St, New York, NY 10007, USA
E–mail: vbercov@bmcc.cuny.edu

Received May 4, 2020
Accepted February 22, 2021

Computer Science Journal of Moldova, vol.**29**, no.1(85), 2021

Total energy of signed digraphs

Sumaira Hafeez Mehtab Khan

Abstract

The energy of a sidigraph is defined as the sum of absolute values of real parts of its eigenvalues. The iota energy of a sidigraph is defined as the sum of absolute values of imaginary parts of its eigenvalues. Recently a new notion of energy of digraphs is introduced which is called the total energy of digraphs. In this paper, we extend this concept of total energy to sidigraphs. We compute total energy formulas for negative directed cycles and show that the total energy of negative directed cycles with fixed order increases monotonically. We introduce complex adjacency matrix to give the integral representation for total energy of sidigraphs. We discuss the increasing property of total energy over some particular subfamilies of $S_{n,h}$, where $S_{n,h}$ contains n-vertex sidigraphs with each cycle having length h. Using the Cauchy-Schwarz inequality, we find upper bound for the total energy of sidigraphs. Finally, we find the class of noncospectral equienergetic sidigraphs.

Keywords: Signed Digraphs, Total energy, Increasing property, T-equienergetic sidigraphs.

MSC 2010: 05C35, 05C50.

1 Introduction

A signed digraph (or sidigraph, for short) is a pair $S = (D, \alpha)$, where $D = (\mathcal{V}, \mathcal{A})$ is the underlying digraph of S and $\alpha : \mathcal{A} \longrightarrow \{-1, 1\}$ is the signing function. Elements of $\mathcal{V}$ are called vertices and elements of $\mathcal{A}$ are called arcs. An arc from a vertex u to a vertex v is denoted by uv. A positive (respectively, a negative) arc is an arc with a $+1$ (respectively, a -1) sign. The product of signs of all arcs of a sidigraph

S is called the sign of S. The sets $\mathcal{A}^+(S)$ and $\mathcal{A}^-(S)$ are the sets of positive and negative arcs of S, respectively. Thus, the set of all signed arcs of S is $\mathcal{A}(S) = \mathcal{A}^+(S) \cup \mathcal{A}^-(S)$. If the direction of all arcs of the underlying digraph D is removed, then $S = (D, \alpha)$ is called a sigraph.

A signed directed path of length $n \geq 1$ is a sidigraph on n vertices $v_1, v_2, \ldots v_n$ with $n-1$ signed arcs $v_i v_{i+1}$, $i = 1, 2, \ldots, n-1$. A signed directed cycle of length $n \geq 2$ is a sidigraph with vertices $v_1, v_2, \ldots, v_n$ and signed arcs $v_i v_{i+1}$, $i = 1, 2, \ldots, n-1$ and $v_n v_1$. If each directed cycle of a sidigraph S has positive sign, then it is said to be cycle-balanced; otherwise non cycle-balanced. If for each pair of vertices u and v of a sidigraph S, there is a path from u to v and a path from v to u, then S is called a strongly connected sidigraph. The strong components of a sidigraph are maximally connected subsidigraphs. The indegree and outdegree of a vertex u of S, denoted by $d^+(u)$ and $d^-(u)$, respectively, is the number of arcs with tail u and the number of arcs with head u, respectively. A linear sidigraph is a sidigraph in which $d^+(u) = 1 = d^-(u)$ for every vertex $u \in \mathcal{V}(S)$. A sidigraph is positive (respectively, negative) if its sign is positive (respectively, negative). A sidigraph in which the number of vertices equals the number of arcs and has a unique directed cycle is called a unicyclic sidigraph. Throughout this paper, we denote a positive directed cycle by C_n and a negative directed cycle by $\boldsymbol{C}_n$, where n is the length of cycle.

The adjacency matrix $A(S) = [a_{ij}]_{n \times n}$ of an n-vertex sidigraph S is defined as

$$a_{ij} = \begin{cases} \alpha(v_i, v_j) & \text{if there is an arc from } v_i \text{ to } v_j, \\ 0 & \text{otherwise.} \end{cases}$$

The characteristic polynomial of a sidigraph S is the polynomial

$$\phi_S(\lambda) = \det(A(S) - \lambda I_n),$$

where I_n is the identity matrix of order n. The eigenvalues of a sidigraph S are the eigenvalues of its adjacency matrix $A(S)$. The multiset of eigenvalues of a sidigraph S is called the spectrum of S. It is denoted by spec(S).

A sidigraph is said to be symmetric, if for an arc $uv \in \mathcal{A}(S)$ an arc $vu \in \mathcal{A}(S)$ also holds, where $u, v \in \mathcal{V}(S)$. A one to one correspondence between sigraphs and symmetric sidigraphs is given by $S \rightsquigarrow \overleftrightarrow{S}$, where $\overleftrightarrow{S}$ is the sidigraph with the same vertex set as that of sigraph S and each signed edge is replaced by a pair of symmetric arcs, uv and vu, both with the same sign as that of edge uv. Under this correspondence, a sigraph can be identified with a symmetric sidigraph.

Let $S_1 = (\mathcal{V}_1, \mathcal{A}_1, \alpha_1)$ and $S_2 = (\mathcal{V}_2, \mathcal{A}_2, \alpha_2)$ be two sidigraphs. The cartesian product $S_1 \times S_2$ is the sidigraph $S = (\mathcal{V}, \mathcal{A}, \alpha)$, where $\mathcal{V} = \mathcal{V}_1 \times \mathcal{V}_2$, the arc set $\mathcal{A}$ is that of the Cartesian product of underlying unsigned digraphs, and the signing function is defined by:

$$\alpha((x_1, x_2)\,(y_1, y_2)) = \begin{cases} \alpha_1(x_1y_1) & \text{if } x_2 = y_2 \\ \alpha_2(x_2y_2) & \text{if } x_1 = y_1. \end{cases}$$

In 1978, Gutman [7] introduced the concept of energy of a simple graph. He defined the energy of a graph as the sum of the absolute values of its eigenvalues. The concept of energy in signed graphs was introduced by Germina et al. [6] in 2010. Bhat and Pirzada [3] finds the unicyclic sigraphs with minimal energy. The authors show that even and odd coefficients of the characteristic polynomial of a unicyclic sigraph respectively alternate in sign. The concept of energy was extended to digraphs by Peña and Rada [12] in 2008. Since the adjacency matrix of a digraph is not necessarily symmetric, its eigenvalues may be complex. Pirzada and Bhat [13] extended the concept of energy of digraphs to sidigraphs. Khan et al. [10] extend the concept of energy of digraphs to iota energy of digraphs and defined iota energy as the sum of absolute values of the imaginary parts of its eigenvalues. Khan et al. [9] introduced the notion of total energy of digraphs. The authors find the uncyclic digraphs with minimal and maximal total energy. Motivated by Khan et al. [9], we extend this concept of total energy to sidigraphs. Among all n-vertex unicyclic sidigraphs, we find unicyclic sidigraphs with minimal and maximal total energy. We show that total energy increases over the sets $S^1_{n,h}$ and $S^2_{n,h}$ with respect to the quasi-order relation when $h \equiv 4 (\mathrm{mod}\ 8)$ and $h \equiv 0 (\mathrm{mod}\ 8)$, respectively. The classes $S^1_{n,h}$ and $S^2_{n,h}$ are defined in section 4. We find upper bound

for the total energy of sidigraphs using the Cauchy-Schwarz inequality. Finally, we find the class of noncospectral equienergetic sidigraphs.

2 Total energy of sidigraphs

The total energy of a digraph is defined by Khan et al. [9] as the sum of absolute values of both real and imaginary part of its eigenvalues. In this section, we extend the concept of total energy to sidigraphs.

The following theorem gives the coefficients of the characteristic polynomial of the adjacency matrix of sidigraphs.

Theorem 1 (Acharya et al. [1])**.** *Let S be an n-vertex sidigraph with characteristic polynomial given by*

$$\phi_S(\lambda) = \lambda^n + \sum_{k=1}^{n} c_k \, \lambda^{n-k}.$$

Then

$$c_k = \sum_{L \in \mathcal{L}_k} (-1)^{p(L)} \prod_{Z \in c(L)} s(Z)$$

for all $k = 1, 2, \ldots, n$, where $\mathcal{L}_k$ is the set of all linear subdigraphs L of S of order k, $p(L)$ denotes the number of components of L, $c(L)$ denotes the set of all cycles and $s(Z)$ the sign of cycle Z.

Let $\lambda_1, \ldots, \lambda_n$ be the eigenvalues of an n-vertex sidigraph S. Then energy and iota energy is defined as

$$E(S) = \sum_{k=1}^{n} |\mathrm{Re}(\lambda_k)|, \tag{1}$$

$$E_c(S) = \sum_{k=1}^{n} |\mathrm{Im}(\lambda_k)|. \tag{2}$$

Now we define total energy of S as follows:

$$E_t(S) = \sum_{k=1}^{n} |\mathrm{Re}(\lambda_k) + \mathrm{Im}(\lambda_k)|, \tag{3}$$

where $\mathrm{Re}(\lambda_k)$ and $\mathrm{Im}(\lambda_k)$ are the real part and imaginary part of eigenvalue λ_k, respectively.

Example 1. *By Theorem 1, the characteristic polynomial of an acyclic sidigraph of order n is given as $\phi_S(\lambda) = \lambda^n$. So its total energy is $E_t(S) = 0$.*

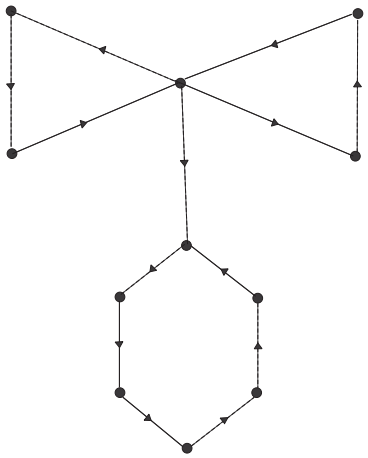

Figure 1. A sidigraph with three signed directed cycles

Example 2. *Consider the sidigraph S shown in Figure* 1. *Negative arcs and positive arcs are represented by dotted lines and solid lines, respectively. By Theorem 1, the characteristic polynomial of S is given by*

$$\phi_S(\lambda) = \lambda^{11} + \lambda^5 = \lambda^5(\lambda^6 + 1).$$

The spectrum of S is given by

$$spec(S) = \left\{ 0, 0, 0, 0, 0, \pm i, \frac{-i \pm \sqrt{3}}{2}, \frac{i \pm \sqrt{3}}{2} \right\}.$$

Thus,

$$E_t(S) = 2 + 2\sqrt{3}.$$

The following theorem gives the relation between total energy of sidigraph and its strong components.

Theorem 2. *Total energy of a sidigraph is the sum of total energies of its strong components.*

Proof. Proof is similar to the proof of Theorem 2.7 [13]. □

The next two lemmas will be useful in proving several results.

Lemma 1. (Khan et al. [8]) *Let x, a, b be real numbers such that $x \geq a > 0$ and $b > 0$. Then we have*

$$\frac{\pi\, x}{b\, x^2 - \pi^2} \leq \frac{\pi\, a}{b\, a^2 - \pi^2}.$$

Lemma 2. (Farooq et al. [4]) *For $x \in (0,\ \frac{\pi}{2}]$, the following inequality holds:*

$$\frac{1}{x} - 0.429\, x \leq \cot\, x \leq \frac{1}{x} - \frac{x}{3}.$$

For any real number x with $0 < x < \frac{\pi}{2}$, sine function satisfies the following:

$$\sin x \leq x, \quad \sin x \geq x - \frac{x^3}{3!}. \tag{4}$$

3 Computation of total energy of signed directed cycles

In this section, we calculate the total energy formulae for negative directed cycles. If S is an n-vertex sidigraph with a unique signed directed cycle of length m, where $2 \leq m \leq n$, then by Theorem 1, $\phi_S(\lambda) = \lambda^n + (-1)^p \lambda^{n-m} = \lambda^{n-m}(\lambda^m + (-1)^p)$, where $p = 1$ or $p = 0$ according to whether S is cycle-balanced or non cycle-balanced. Clearly, total energy equals to the total energy of the unique cycle.

Using Theorem 1, the characteristic polynomial of C_n is given by:

$$\phi_{C_n}(x) = x^n - 1.$$

Thus, the eigenvalues of C_n are $\exp^{\frac{2k\pi i}{n}}$, where $k = 1, \ldots, n-1$. Therefore, the energy and total energy of C_n are given by:

$$E(C_n) = \sum_{k=1}^{n} \left| \cos \frac{2k\pi}{n} \right|, \tag{5}$$

$$E_t(C_n) = \sum_{k=1}^{n} \left| \cos \frac{2k\pi}{n} + \sin \frac{2k\pi}{n} \right|. \tag{6}$$

Using (5), Pirzada and Bhat [13] calculated the following energy formulas for cycle C_n, $n \geq 2$.

$$E(C_n) = \begin{cases} 2\cot \frac{\pi}{n} & \text{if } n \equiv 0(\text{mod}4) \\ 2\csc \frac{\pi}{n} & \text{if } n \equiv 2(\text{mod}4) \\ \csc \frac{\pi}{2n} & \text{if } n \equiv 1(\text{mod}2). \end{cases} \tag{7}$$

For a directed cycle C_n, $n \geq 2$, Khan et al. [9] gave the following total energy formulae,

$$E_t(C_n) = \begin{cases} 2\sqrt{2}\csc \frac{\pi}{n} & \text{if } n \equiv 4(\text{mod } 8) \\ 2\sqrt{2}\cot \frac{\pi}{n} & \text{if } n \equiv 0(\text{mod } 8) \\ \sqrt{2}\csc \frac{\pi}{2n} & \text{if } n \equiv 2(\text{mod } 4) \\ \frac{1}{\sqrt{2}}\csc \frac{\pi}{4n} & \text{if } n \equiv 1(\text{mod } 2). \end{cases} \tag{8}$$

Again, using Theorem 1, the characteristic polynomial of $\boldsymbol{C}_n$ is given by:

$$\phi_{C_n}(x) = x^n + 1.$$

Thus, the eigenvalues of $\boldsymbol{C}_n$ are $\exp^{\frac{(2k+1)\pi i}{n}}$, where $k = 1, \ldots, n-1$. Therefore, the energy and total energy of $\boldsymbol{C}_n$ are computed by:

$$E(\boldsymbol{C}_n) = \sum_{k=1}^{n} \left| \cos \frac{(2k+1)\pi}{n} \right|, \tag{9}$$

$$E_t(\boldsymbol{C}_n) = \sum_{k=1}^{n} \left| \cos \frac{(2k+1)\pi}{n} + \sin \frac{(2k+1)\pi}{n} \right|. \tag{10}$$

Using (9), Pirzada and Bhat [13] calculated the energy of a cycle $\boldsymbol{C}_n$, $n \geq 2$ as follows:

$$E(\boldsymbol{C}_n) = \begin{cases} 2\csc\frac{\pi}{n} & \text{if } n \equiv 0(\text{mod}4) \\ 2\cot\frac{\pi}{n} & \text{if } n \equiv 2(\text{mod}4) \\ \csc\frac{\pi}{2n} & \text{if } n \equiv 1(\text{mod}2). \end{cases} \tag{11}$$

Next, we calculate the total energy formulae of $\boldsymbol{C}_n$, by considering the following four cases:
Case 1. If $n \equiv 4(\text{mod } 8)$, then

$$\begin{aligned} E_t(\boldsymbol{C}_n) &= \sum_{k=0}^{n-1}\left|\cos\frac{(2k+1)\pi}{n} + \sin\frac{(2k+1)\pi}{n}\right| \\ &= 2\sum_{k=0}^{\frac{n}{2}-1}\left|\cos\frac{(2k+1)\pi}{n} + \sin\frac{(2k+1)\pi}{n}\right| \\ &= 2\Bigg(\sum_{k=0}^{\frac{n}{4}-1}\left|\cos\frac{(2k+1)\pi}{n} + \sin\frac{(2k+1)\pi}{n}\right| \\ &\quad + \sum_{k=0}^{\frac{n}{4}-1}\left|\cos\frac{(2k+1)\pi}{n} - \sin\frac{(2k+1)\pi}{n}\right|\Bigg) \\ &= 2\Bigg(\sum_{k=0}^{\frac{n}{4}-1}\left(\cos\frac{(2k+1)\pi}{n} + \sin\frac{(2k+1)\pi}{n}\right) \\ &\quad + \sum_{k=0}^{\frac{n-12}{8}}\left(\cos\frac{(2k+1)\pi}{n} - \sin\frac{(2k+1)\pi}{n}\right) \\ &\quad \sum_{k=\frac{n+4}{8}}^{\frac{n}{4}-1}\left(\sin\frac{(2k+1)\pi}{n} - \cos\frac{(2k+1)\pi}{n}\right)\Bigg). \end{aligned} \tag{12}$$

Using geometric series sum formula and some basic trigonometric iden-

tities, we obtain

$$\begin{aligned}&\sum_{k=0}^{\frac{n}{4}-1}\left(\cos\frac{(2k+1)\pi}{n}+\sin\frac{(2k+1)\pi}{n}\right)\\&=\left(\cos\frac{\pi}{4}+\sin\frac{\pi}{4}\right)\sin\frac{\pi}{4}\csc\frac{\pi}{n}=\csc\frac{\pi}{n}.\end{aligned}\tag{13}$$

$$\begin{aligned}&\sum_{k=0}^{\frac{n-12}{8}}\left(\cos\frac{(2k+1)\pi}{n}-\sin\frac{(2k+1)\pi}{n}\right)\\&=\frac{1}{2}\left(\sqrt{2}\cos\frac{\pi}{n}-1\right)\csc\frac{\pi}{n}.\end{aligned}\tag{14}$$

$$\begin{aligned}&\sum_{k=\frac{n+4}{8}}^{\frac{n}{4}-1}\left(\sin\frac{(2k+1)\pi}{n}-\cos\frac{(2k+1)\pi}{n}\right)\\&=\frac{1}{2}\left(\sqrt{2}\cos\frac{\pi}{n}-1\right)\csc\frac{\pi}{n}.\end{aligned}\tag{15}$$

Using (13)$\sim$(15), equation (12) can be written as:

$$E_t(\boldsymbol{C}_n)=\sum_{k=0}^{n-1}\left|\cos\frac{(2k+1)\pi}{n}+\sin\frac{(2k+1)\pi}{n}\right|=2\sqrt{2}\cot\frac{\pi}{n}.$$

Case 2. If $n \equiv 0 (\mathrm{mod}\ 8)$, then

$$\begin{aligned} E_t(\boldsymbol{C}_n) &= \sum_{k=0}^{n-1} \left| \cos \frac{(2k+1)\pi}{n} + \sin \frac{(2k+1)\pi}{n} \right| \\ &= 2 \sum_{k=0}^{\frac{n}{2}-1} \left| \cos \frac{(2k+1)\pi}{n} + \sin \frac{(2k+1)\pi}{n} \right| \\ &= 2 \Bigg(\sum_{k=0}^{\frac{n}{4}-1} \left| \cos \frac{(2k+1)\pi}{n} + \sin \frac{(2k+1)\pi}{n} \right| \\ &\quad + \sum_{k=0}^{\frac{n}{4}-1} \left| \cos \frac{(2k+1)\pi}{n} - \sin \frac{(2k+1)\pi}{n} \right| \Bigg) \\ &= 2 \Bigg(\sum_{k=0}^{\frac{n}{4}-1} \left(\cos \frac{(2k+1)\pi}{n} + \sin \frac{(2k+1)\pi}{n} \right) \\ &\quad + \sum_{k=0}^{\frac{n-8}{8}} \left(\cos \frac{(2k+1)\pi}{n} - \sin \frac{(2k+1)\pi}{n} \right) \\ &\quad \sum_{k=\frac{n}{8}}^{\frac{n}{4}-1} \left(\sin \frac{(2k+1)\pi}{n} - \cos \frac{(2k+1)\pi}{n} \right) \Bigg). \end{aligned} \tag{16}$$

Using geometric series sum formulas and some basic trigonometric identities, we get

$$\sum_{k=0}^{\frac{n-8}{8}} \left(\cos \frac{(2k+1)\pi}{n} - \sin \frac{(2k+1)\pi}{n} \right) = \frac{1}{2} \left(\sqrt{2} - 1 \right) \csc \frac{\pi}{n}. \tag{17}$$

$$\sum_{k=\frac{n}{8}}^{\frac{n}{4}-1} \left(\sin \frac{(2k+1)\pi}{n} - \cos \frac{(2k+1)\pi}{n} \right) = \frac{1}{2} \left(\sqrt{2} - 1 \right) \csc \frac{\pi}{n}. \tag{18}$$

Using (13), (17) and (18), equality (16) becomes

$$E_t(\boldsymbol{C}_n) = \sum_{k=0}^{n-1} \left| \cos \frac{(2k+1)\pi}{n} + \sin \frac{(2k+1)\pi}{n} \right| = 2\sqrt{2} \csc \frac{\pi}{n}.$$

Similary, one can prove that $E_t(\boldsymbol{C}_n) = \sqrt{2}\csc\frac{\pi}{2n}$ when $n \equiv 2(\text{mod } 4)$ and $E_t(\boldsymbol{C}_n) = \frac{1}{\sqrt{2}}\csc\frac{\pi}{4n}$ when $n \equiv 1(\text{mod } 2)$.

In brief, we get

$$E_t(\boldsymbol{C}_n) = \begin{cases} 2\sqrt{2}\cot\frac{\pi}{n} & \text{if } n \equiv 4(\text{mod } 8) \\ 2\sqrt{2}\csc\frac{\pi}{n} & \text{if } n \equiv 0(\text{mod } 8) \\ \sqrt{2}\csc\frac{\pi}{2n} & \text{if } n \equiv 2(\text{mod } 4) \\ \frac{1}{\sqrt{2}}\csc\frac{\pi}{4n} & \text{if } n \equiv 1(\text{mod } 2). \end{cases} \tag{19}$$

To find minimal and maximal total energy among all non cycle-balanced unicyclic sidigraphs, the following lemma will be useful. We would like to mention that the idea of proof is taken from the proof of Lemma 3.5 [2].

Lemma 3. *For $n \geq 2$, the sequence $\langle a_n \rangle$ given by*

$$a_n = \begin{cases} 2\sqrt{2}\cot\frac{\pi}{n} & \textit{if } n \equiv 4(\text{mod } 8) \\ 2\sqrt{2}\csc\frac{\pi}{n} & \textit{if } n \equiv 0(\text{mod } 8) \\ \sqrt{2}\csc\frac{\pi}{2n} & \textit{if } n \equiv 2(\text{mod } 4) \\ \frac{1}{\sqrt{2}}\csc\frac{\pi}{4n} & \textit{if } n \equiv 1(\text{mod } 2) \end{cases}$$

is strictly increasing sequence.

Proof. To prove that $\langle a_n \rangle$ is strictly increasing sequence, we need to show that for $k \geq 1$, the following inequalities hold true.
$2\csc\frac{\pi}{2(4k-2)} < \csc\frac{\pi}{4(4k-1)} < 4\cot\frac{\pi}{4k} < \csc\frac{\pi}{4(4k+1)} < 2\csc\frac{\pi}{2(4k+2)} < \csc\frac{\pi}{4(4k+3)} < 4\csc\frac{\pi}{4k+4}$.

Using (4), we get

$$\begin{aligned} 2\csc\frac{\pi}{8k-4} &\leq 2\left(\frac{1}{\frac{\pi}{8k-4}(1-\frac{\pi^2}{6(8k-4)^2})}\right) \\ &= \frac{2(8k-4)}{\pi}\left(1+\frac{\pi^2}{6(8k-4)^2-\pi^2}\right) \\ &= \frac{2(8k-4)}{\pi}+\frac{2(8k-4)\pi}{6(8k-4)^2-\pi^2}. \end{aligned}$$

By Lemma 1, we obtain

$$\begin{aligned}\frac{2(8k-4)}{\pi}+\frac{2(8k-4)\pi}{6(8k-4)^2-\pi^2} &\leq \frac{2(8k-4)}{\pi}+\frac{2(4)\pi}{6(4)^2-\pi^2}\\ &\leq \frac{16k}{\pi}-2.2547.\end{aligned}$$

Thus

$$2\csc\frac{\pi}{8k-4}\leq\frac{16k}{\pi}-2.2547. \tag{20}$$

On the other hand, using (4), we have

$$\csc\frac{\pi}{4(4k-1)}\geq\frac{4(4k-1)}{\pi}\geq\frac{16k}{\pi}-1.2732. \tag{21}$$

(20) and (21) give $2\csc\frac{\pi}{2(4k-2)}<\csc\frac{\pi}{4(4k-1)}$, thereby proving the first inequality.

Next, we need to show that $\csc\frac{\pi}{4(4k-1)}<4\cot\frac{\pi}{4k}$. It is equivalent to show

$4\cos\frac{\pi}{n}\sin\frac{\pi}{4n-4}-\sin\frac{\pi}{n}>0$, where $n=4k$.

We use Taylor series expansion for $\sin x$ and $\cos x$ to prove this. Now

$$\begin{aligned}&4\cos\frac{\pi}{n}\sin\frac{\pi}{4n-4}-\sin\frac{\pi}{n}\\ =\;&4\,[1-\frac{1}{2!}(\frac{\pi}{n})^2+\dots]\,[\frac{\pi}{4n-4}-\frac{1}{3!}(\frac{\pi}{4n-4})^2+\dots]\\ &-[\,\frac{\pi}{n}-\frac{1}{3!}(\frac{\pi}{n})^3+\dots]\\ =\;&\frac{\pi}{n-1}-\frac{\pi}{n}+o(n^{-3})>0,\end{aligned}$$

where $f(n)\in o(g(n))$ if $\frac{f(n)}{g(n)}\longrightarrow 0$ as $n\longrightarrow\infty$. This proves the second inequality.

Now we will prove that $4\cot\frac{\pi}{4k}<\csc\frac{\pi}{4(4k+1)}$. Using Lemma 2, we

get

$$\begin{aligned} 4\cot\frac{\pi}{4k} &\leq 4\left(\frac{4k}{\pi}-\frac{\pi}{3(4k)}\right) \\ &= \frac{16k}{\pi}-\frac{\pi}{3k}. \end{aligned} \tag{22}$$

On the other hand, using $\csc x \geq \frac{1}{x}$, we obtain

$$\begin{aligned} \csc\frac{\pi}{4(4k+1)} &\geq \frac{4(4k+1)}{\pi} \\ &= \frac{16k}{\pi}+\frac{4}{\pi}. \end{aligned} \tag{23}$$

Using (22) and (23), we get the third inequality.

Since $\sin x$ is an increasing function in the interval $[0, \frac{\pi}{2}]$, we have $\sin\frac{\pi}{16k+8} < \sin\frac{\pi}{16k+4}$. Also we know that $\cos\frac{\pi}{16k+8} < 1$. This gives $2\sin\frac{\pi}{16k+8}\cos\frac{\pi}{16k+8} < 2\sin\frac{\pi}{16k+4}$. Using $\sin 2x = 2\sin x\cos x$, we get $\sin\frac{\pi}{8k+4} < 2\sin\frac{\pi}{4(4k+1)}$, that is, $\csc\frac{\pi}{4(4k+1)} < 2\csc\frac{\pi}{2(4k+2)}$. This proves the fourth inequality.

Next, we will prove that $2\csc\frac{\pi}{2(4k+2)} < \csc\frac{\pi}{4(4k+3)}$. Using (4), we have

$$\begin{aligned} 2\csc\frac{\pi}{2(4k+2)} &\leq 2\left(\frac{1}{\frac{\pi}{2(4k+2)}(1-\frac{\pi^2}{24(4k+2)^2})}\right) \\ &= \frac{4(4k+2)}{\pi}\left(1+\frac{\pi^2}{24(4k+2)^2-\pi^2}\right) \\ &= \frac{4(4k+2)}{\pi}+\frac{4(4k+2)\pi}{24(4k+2)^2-\pi^2}. \end{aligned}$$

Using Lemma 2, we get

$$\begin{aligned} \frac{4(4k+2)}{\pi}+\frac{4(4k+2)\pi}{24(4k+2)^2-\pi^2} &\leq \frac{4(4k+2)}{\pi}+\frac{4(6)\pi}{24(6)^2-\pi^2} \\ &\leq \frac{16k}{\pi}+2.63475. \end{aligned}$$

Thus

$$2\csc\frac{\pi}{2(4k+2)} \leq \frac{16k}{\pi} + 2.63475. \tag{24}$$

On the other hand, using (4), we obtain

$$\begin{aligned}\csc\frac{\pi}{4(4k+3)} &\geq \frac{4(4k+3)}{\pi} \\ &= \frac{16k}{\pi} + 3.81972.\end{aligned} \tag{25}$$

Using (24) and (25), we get the desired result.

Finally, we will show that $\csc\frac{\pi}{4(4k+3)} < 4\csc\frac{\pi}{4k+4}$. Since $\sin x$ is an increasing function in the interval $[0, \frac{\pi}{2}]$. This gives $\sin\frac{\pi}{16k+16} < \sin\frac{\pi}{16+12}$. As we know that $\cos\frac{\pi}{8k+8} < 1$ and $\cos\frac{\pi}{16k+16} < 1$, this gives $4\sin\frac{\pi}{16k+16}\cos\frac{\pi}{16k+16}\cos\frac{\pi}{8k+8} < 4\sin\frac{\pi}{16k+12}$. Using some of the basic trigonometric identities, we get $\sin\frac{\pi}{4k+4} < 4\sin\frac{\pi}{4(4k+3)}$, that is $\csc\frac{\pi}{4(4k+3)} < 4\csc\frac{\pi}{4k+4}$, and we are done. □

The next two theorems give extremal energy among n-vertex cycle-balanced and non cycle-balanced unicyclic sidigraphs, $n \geq 2$.

Theorem 3 (Khan et al. [9])**.** *The unicyclic n-vertex digraphs which contain a directed cycle C_2 have minimal total energy among all n-vertex unicyclic digraphs. The directed cycle C_n has the maximal total energy among all n-vertex unicyclic digraphs.*

Theorem 4. *Among n-vertex unicyclic sidigraphs with negative cycle, minimal total energy is attained in a sidigraph which contains $\boldsymbol{C}_2$. Moreover, among all non cycle-balanced unicyclic sidigraphs on n vertices, the cycle $\boldsymbol{C}_n$ has the largest total energy.*

The next theorem gives a few characteristics of positive and negative directed cycles.

Theorem 5. *Total energy of positive and negative directed cycles satisfies the following:*

(*i*) $E_t(C_n) = E_t(\boldsymbol{C}_n)$ *if and only if* $n \equiv 1(\mathrm{mod}2)$ *or* $n \equiv 2(\mathrm{mod}4)$.

(*ii*) $E_t(C_n) > E_t(\boldsymbol{C}_n)$ *if and only if* $n \equiv 4(\mathrm{mod}8)$.

(*iii*) $E_t(C_n) < E_t(\boldsymbol{C}_n)$ *if and only if* $n \equiv 0(\mathrm{mod}8)$.

Proof. Proof follows from Lemma 3. □

In the following theorem, we compare the total energy of positive directed cycles with their energy and iota energy.

Theorem 6. *Let* $n \geq 2$ *be an integer. Then* $E_t(C_n) > E(C_n)$ *and* $E_t(C_n) > E_c(C_n)$.

Proof. Proof follows from (7), (8) and Theorem 2.4 [10]. □

The next corollary is an immediate consequence of Theorem 6.

Corollary 1. *Let* S *be an* n*-vertex unicylic sidigraph with unique directed cycle* C_m, $2 \leq m \leq n$. *Then* $E_t(S) > E(S)$ *and* $E_t(S) > E_c(S)$.

In the next theorem, we give comparison between energy, iota energy and the total energy of negative directed cycles.

Theorem 7. *Let* $n \geq 2$ *be an integer. Then* $E_t(\boldsymbol{C}_n) > E(\boldsymbol{C}_n)$ *and* $E_t(\boldsymbol{C}_n) > E_c(\boldsymbol{C}_n)$.

Proof. Using (11), (19) and Theorem 2.11 [5], we can easily get the desired result. □

An immediate consequence of Theorem 7 is the following corollary.

Corollary 2. *Let* S *be an* n*-vertex unicylic sidigraph with unique directed cycle* $\boldsymbol{C}_m$, $2 \leq m \leq n$. *Then* $E_t(S) > E(S)$ *and* $E_t(S) > E_c(S)$.

4 Complex adjacency matrix and increasing property

In this section, we introduce complex adjacency matrix for sidigraphs in order to define total energy of sidigraph as the sum of absolute values of real parts of its eigenvalues. Let $S = (D, \alpha)$, where $D = (\mathcal{V}, \mathcal{A})$ is the underlying digraph of S and $\alpha : \mathcal{A} \longrightarrow \{-1, 1\}$ is the signing function. We define the complex adjacency matrix $A_t(S) = b_{jk}$ of S by:

$$b_{jk} = \begin{cases} (1-i)\,\alpha(v_j, v_k) & \text{if } v_j v_k \in \mathcal{A}, \\ 0 & \text{otherwise.} \end{cases}$$

The characteristic polynomial of the sidigraph S with respect to complex adjacency matrix $A_t(S)$ is $\phi_S(\lambda) = \det(\lambda I_n - A_t(S))$. The zeros of $\phi_S(\lambda)$ are called the A_t-eigenvalues of S. The spectrum of $A_t(S)$ is denoted by $\mathrm{Spec_t(S)}$. If $A(S)$ is the adjacency matrix of S, then note that $A_t(S) = (1-i)A(S)$. Therefore, $\mathrm{Spec_t(S)} = (1-i)\ \mathrm{Spec(S)}$. Thus, $\mathrm{Re}(\lambda) + \mathrm{Im}(\lambda) = \mathrm{Re}((1-i)\lambda)$, where λ is the eigenvalue of S. Let $\lambda_1, \ldots, \lambda_n$ be the A_t-eigenvalues of a sidigraph S. Then, the total energy of S can also be defined as:

$$E_t(S) = \sum_{k=1}^{n} |\mathrm{Re(\lambda_k)}|,$$

where $\mathrm{Re}(\lambda_k)$ denote the real part of the A_t-eigenvalue λ_k. The coefficient theorem for sidigraphs with complex adjacency matrix $A_t(S)$ is given by:

Theorem 8. *Let S be an n-vertex sidigraph with characteristic polynomial*

$$\Phi_S^t(\lambda) = \lambda^n + \sum_{k=1}^{n} a_k(S)(1-i)^k \lambda^{n-k},$$

then

$$a_k(S) = \sum_{L \in \mathcal{L}_k} (-1)^{p(L)} \prod_{Z \in c(L)} s(Z),$$

for all $k = 1, 2, \ldots, n$, where $\mathcal{L}_k$ is the set of all linear subsidigraphs L of S of order k, $p(L)$ denotes the number of components of L. Moreover $c(L)$ and $s(Z)$ represent the set of all cycles of L and the sign of cycle Z, respectively.

Let S be an n-vertex sidigraph and let $S_1, \ldots, S_r$ be its strong components. Then, from the proof of Theorem 2, the characteristic polynomial of S is given by:

$$\phi_S(\lambda) = \phi_{S_1}(\lambda) \ \ldots \ \phi_{S_r}(\lambda). \tag{26}$$

The following lemma is about the characteristic polynomial of sidigraph S such that all of its cycles are vertex-disjoint.

Lemma 4. *Let S be an n-vertex sidigraph containing cycles $C_1, \ldots, C_r$ of lengths m_1, m_2, ..., m_r, respectively. Assume that C_1, ..., C_r are pairwise vertex-disjoint. Then*

$$\phi_S(\lambda) = \lambda^{n-m} \, \phi_{C_1}(\lambda) \, . \, \ldots \, \phi_{C_r}(\lambda),$$

where $m = \sum_{i=1}^{r} m_i$.

Proof. Proof follows from equation (26). □

The concept of energy of polynomials is introduced by Mateljevic et al. [11] in 2010. The authors defined the energy of a complex polynomial $P(\lambda)$ of degree n as

$$E(P(\lambda)) = \sum_{k=1}^{n} sgn(\mathrm{Re}(\lambda_k))\lambda_k,$$

where $\lambda_1, \ldots, \lambda_n$ are the zeros of $P(\lambda)$. In general, the zeros of the complex polynomial do not satisfy the conjugate pairing property. Therefore, the energy of $P(\lambda)$ is not a real number. Moreover, by complex conjugate root theorem, the roots of the polynomial with real coefficients satisfy conjugate pairing property. Thus

$$E(P(\lambda)) = \sum_{k=1}^{n} sgn(\mathrm{Re}(\lambda_k))\lambda_k = \sum_{k=1}^{n} |\mathrm{Re}(\lambda_k)|. \tag{27}$$

Throughout the paper, we denote by $p.v \int_{-\infty}^{\infty} F(\lambda)\, d\lambda$, the principal value of an integral $\int_{-\infty}^{\infty} F(\lambda)\, d\lambda$. The following theorem is well known.

Theorem 9 (Mateljevic et al. [11]). *Let $\phi(\lambda)$ be a monic real polynomial of degree n and $\lambda_1, \ldots, \lambda_n$ be its zeros. Then*

$$\sum_{k=1}^{n} |\mathrm{Re}(\lambda_k)| = \sum_{k=1}^{n} sgn(\mathrm{Re}(\lambda_k))\lambda_k = \frac{1}{\pi}\, p.v \int_{-\infty}^{\infty} \left(n - \frac{i\lambda\phi'(i\lambda)}{\phi(i\lambda)}\right) d\lambda,$$

where $\mathrm{Re}(\lambda_k)$ *is the real part of* λ_k.

If S is an n-vertex sidigraph which has cycles of length $h \equiv 0 (\mathrm{mod}\ 4)$, then all coefficients of $\Phi_S^t(\lambda)$ in Theorem 8 are real. Thus, in this case we can represent the total energy of sidigraphs in integral form.

Theorem 10. *Let S be an n-vertex sidigraph, which has cycles of length $h \equiv 0 (\mathrm{mod}\ 4)$ and $\Phi_S^t(\lambda)$ be the characteristic polynomial of the complex adjacency matrix $A_t(S)$. Then*

$$\sum_{k=1}^{n} |\mathrm{Re}(\lambda_k)| = \frac{1}{\pi}\, p.v \int_{-\infty}^{\infty} \left(n - \frac{i\lambda\Phi_S'^t(i\lambda)}{\Phi_S^t(i\lambda)}\right) d\lambda,$$

where $\lambda_1, \ldots, \lambda_n$ are A_t-eigenvalues of S and $\mathrm{Re}(\lambda_k)$ *is the real part of* λ_k.

The following corollary is an immediate consequence of Theorem 10.

Corollary 3. *Let ϕ be a monic polynomial of degree n. Let $\lambda_1, \ldots, \lambda_n$ be its roots. Then*

$$\sum_{k=1}^{n} |\mathrm{Re}(\lambda_k)| = \frac{1}{\pi}\, p.v \int_{-\infty}^{\infty} \log|\gamma(t)| \frac{dt}{t^2},$$

where $\gamma(t) = t^n \phi(\frac{i}{t})$ and $\mathrm{Re}(\lambda_k)$ *is the real part of* λ_k.

Let $S_{n,h}$ be the set of sidigraphs with n vertices such that each sidigraph in $S_{n,h}$ has signed directed cycles of length h. Now we will study the increasing property of total energy of sidigraphs in $S_{n,h}$. The following theorem gives the characteristic polynomial of the sidigraphs in $S_{n,h}$.

Theorem 11 (Pirzada and Bhat [13]). *If $S \in S_{n,h}$, then $\Phi_S(\lambda) = \lambda^n + \sum_{k=1}^{\lfloor \frac{n}{h} \rfloor} (-1)^k c^*(S, kh)\lambda^{n-kh}$, $c^*(S, kh)$ is the number of positive linear subsidigraphs of order kh − number of negative linear subsidigraphs of order kh, $k = 1, \ldots, \lfloor \frac{n}{h} \rfloor$.*

Let $S^1_{n,h} = \left\{ S \in S_{n,h} \mid c^*(S, kh) \geq 0, k = 1, \ldots, \lfloor \frac{n}{h} \rfloor \right\}$. Pirzada and Bhat [13] define the quasi-order relation over $S^1_{n,h}$ as follows:

Let $S_1, S_2 \in S^1_{n,h}$. If for all $k = 1, \ldots, \lfloor \frac{n}{h} \rfloor$, $c^*(S_1, kh) \leq c^*(S_2, kh)$, then $S_1 \preceq S_2$ and if there exists k such that $c^*(S_1, kh) < c^*(S_2, kh)$, then $S_1 \prec S_2$. Clearly it is symmetric and transitive relation over $S^1_{n,h}$.

The next theorem is an analogue of Theorem 11.

Theorem 12. *If $S \in S^1_{n,h}$, then $\Phi^t_S(\lambda) = \lambda^n + \sum_{k=1}^{\lfloor \frac{n}{h} \rfloor} (-1)^k \ (1 - i)^k \ c^*(S, kh)\lambda^{n-kh}$, $c^*(S, kh)$ is the number of positive linear subsidigraphs of order kh − number of negative linear subsidigraphs of order kh, $k = 1, \ldots, \lfloor \frac{n}{h} \rfloor$.*

Farooq et al. [5] define a new subfamily $S^2_{n,h}$ of $S_{n,h}$, which consists of those sidigraphs $S \in S_{n,h}$ which have negative cycles of length h. The following theorem gives the characteristic polynomial of the sidigraphs in $S^2_{n,h}$.

Theorem 13 (Farooq et al. [5]). *Let $S \in S^2_{n,h}$. Then the characteristic polynomial of S according to adjacency matrix $A(S)$ is given by $\Phi_S(\lambda) = \lambda^n + \sum_{k=1}^{\lfloor \frac{n}{h} \rfloor} c(S, kh)\lambda^{n-kh}$, where $c(S, kh)$ is the number of linear subsidigraphs of S containing k negative cycles of order h, $k = 1, \ldots, \lfloor \frac{n}{h} \rfloor$.*

Farooq et al. [5] define a quasi-order relation over $S^2_{n,h}$ as follows: Let $S_1, S_2 \in S^2_{n,h}$. If for all $k = 1, \ldots, \lfloor \frac{n}{h} \rfloor$, $c(S_1, kh) \leq c(S_2, kh)$, then $S_1 \preceq S_2$ and if there exists k such that $c(S_1, kh) < c(S_2, kh)$, then $S_1 \prec S_2$. This relation is symmetric and transitive relation over $S^2_{n,h}$.

The next theorem is an analogue of Theorem 13.

Theorem 14. *Let $S \in S^2_{n,h}$. Then the characteristic polynomial of S according to adjacency matrix $A_t(S)$ is given by $\Phi^t_S(\lambda) = \lambda^n + \sum_{k=1}^{\lfloor \frac{n}{h} \rfloor} (1 - i)^k \, c(S, kh)\lambda^{n-kh}$, where $c(S, kh)$ is the number of linear subsidigraphs of S containing k negative cycles of order h, $k = 1, \ldots, \lfloor \frac{n}{h} \rfloor$.*

The following theorems give the increasing property of total energy over $S^1_{n,h}$ when $h \equiv 4 (\text{mod } 8)$ and over $S^2_{n,h}$ when $h \equiv 0 (\text{mod } 8)$. The proofs are similar to the proof of Theorem 5.3 [12]. However, for the sake of self-containment, we only include the proof of Theorem 16.

Theorem 15. *Let h be an integer of the form $h \equiv 4 (\text{mod } 8)$. Then total energy of a sidigraph $S \in S^1_{n,h}$ increases with respect to the quasi-order relation $\preceq$ defined over $S^1_{n,h}$. That is, if $S_1 \preceq S_2$, then $E_t(S_1) \leq E_t(S_2)$, where $S_1, S_2 \in S^1_{n,h}$.*

Theorem 16. *Let h be an integer of the form $h \equiv 0 (\text{mod } 8)$. Then total energy of a sidigraph $S \in S^2_{n,h}$ increases with respect to the quasi-order relation $\preceq$ defined over $S^1_{n,h}$. That is, if $S_1 \preceq S_2$, then $E_t(S_1) \leq E_t(S_2)$, where $S_1, S_2 \in S^2_{n,h}$.*

Proof. Let $S \in S^2_{n,h}$ be an n-vertex sidigraph. Since $h \equiv 0 (\text{mod } 8)$, there is a positive integer l such that $h = 8l$. Then by Theorem 13, the characteristic polynomial of $A_t(S)$ is given by

$$\Phi^t_S(\lambda) = \lambda^n + \sum_{k=1}^{\lfloor \frac{n}{h} \rfloor} (1 - i)^{kh} c(S, kh)\lambda^{n-kh}. \tag{28}$$

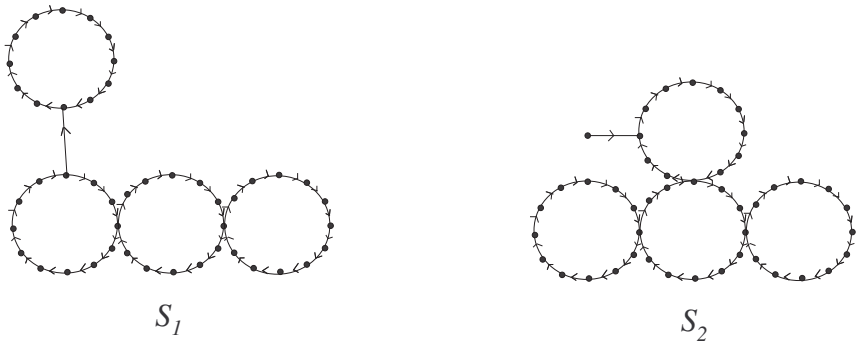

Figure 2. $S_1, S_2 \in S^2_{46,12}$

This gives

$$\begin{aligned}\Phi^t_S(\frac{\dot{\imath}}{\lambda}) &= \left(\frac{\dot{\imath}}{\lambda}\right)^n + \sum_{k=1}^{\lfloor \frac{n}{h} \rfloor}(1-\dot{\imath})^{kh} c(S,kh)\left(\frac{\dot{\imath}}{\lambda}\right)^{n-kh} \\ &= \left(\frac{\dot{\imath}}{\lambda}\right)^n \left(1 + \sum_{k=1}^{\lfloor \frac{n}{h} \rfloor}(-2\dot{\imath})^{4kl} c(S,kh)\lambda^{kh}\right) \\ &= \left(\frac{\dot{\imath}}{\lambda}\right)^n \left(1 + \sum_{k=1}^{\lfloor \frac{n}{h} \rfloor} 2^{4kl} c(S,kh)\lambda^{kh}\right).\end{aligned}$$

By applying Corollary 3, we get

$$\begin{aligned}E_t(S) &= \frac{1}{\pi}\, p.v \int_{-\infty}^{\infty} \frac{1}{\lambda^2} \log \left| \lambda^n \frac{\dot{\imath}^n}{\lambda^n}\left(1 + \sum_{k=1}^{\lfloor \frac{n}{h} \rfloor} 2^{4kl} c(S,kh)\lambda^{kh}\right)\right| d\lambda \\ &= \frac{1}{\pi}\, p.v \int_{-\infty}^{\infty} \frac{1}{\lambda^2} \log \left(1 + \sum_{k=1}^{\lfloor \frac{n}{h} \rfloor} 2^{4kl} c(S,kh)\lambda^{kh}\right) d\lambda.\end{aligned}$$

The last equality is obtained by using $\frac{1}{\pi} p.v. \int_{-\infty}^{\infty} \log(\dot{\imath}^n) \frac{d\lambda}{\lambda^2} = 0$. It is easy to see from the above total energy expression that total energy increases with respect to quasi-order relation $\preceq$ over $S^2_{n,h}$. □

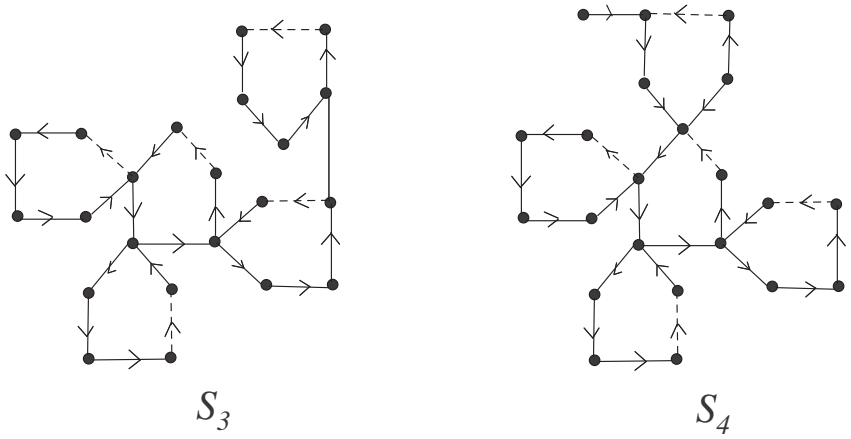

Figure 3. $S_3, S_4 \in S^2_{22,5}$

Example 3. *Consider the sidigraphs S_1 and S_2 as shown in Figure* 2*, where $h \equiv 4(\mathrm{mod}\,8)$ and each cycle has negative sign. By Theorem 13, the characteristics polynomials of S_1 and S_2 are, respectively, given by*

$$\begin{aligned} \phi_{S_1} &= \lambda^{46} + 4\lambda^{34} + 4\lambda^{22} + \lambda^{10}, \\ \phi_{S_2} &= \lambda^{46} + 4\lambda^{34} + 3\lambda^{22} + \lambda^{10}. \end{aligned}$$

Here $c(S_1, kh) \geq c(S_2, kh)$, $k = 1, 2, 3$. However, $E_t(S_1) = 31.7354$ and $E_t(S_2) = 32.0260$. Clearly, $E_t(S_2) \geq E_t(S_1)$. Thus, the total energy may not increase with respect to the quasi-order relation over the set $S^2_{n,h}$ when $h \equiv 4(\mathrm{mod}\ 8)$.

Example 4. *Consider the sidigraphs S_3 and S_4 as shown in Figure* 3*, where $h \equiv 1(\mathrm{mod}2)$. By Theorem 13, the characteristics polynomials of S_3 and S_4 are, respectively, given by*

$$\begin{aligned} \phi_{S_3} &= \lambda^{22} + 5\lambda^{17} + 7\lambda^{12} + 4\lambda^{7} + \lambda^{2}, \\ \phi_{S_4} &= \lambda^{22} + 5\lambda^{17} + 6\lambda^{12} + 4\lambda^{7} + \lambda^{2}. \end{aligned}$$

Here $c(S_3, kh) \geq c(S_4, kh)$, $k = 1, 2, 3, 4$. *However,* $E_t(S_3) = 18.1843$ *and* $E_t(S_4) = 18.2420$. *Clearly,* $E_t(S_4) \geq E_t(S_3)$. *Thus, the total energy may not increase with respect to the quasi-order relation over the set* $S^2_{n,h}$ *when* $h \equiv 1(\text{mod } 2)$.

For counter examples of increasing property of total energy in $S^1_{n,h}$ when $h \equiv 0(\text{mod } 8)$ or $h \equiv 1(\text{mod } 2)$, see Example 5.4 and 5.5 [9].

5 Upper bound for total energy of sidigraphs

In this section, we obtain upper bound for the total energy of sidigraphs. An alternating sequence of vertices and directed arcs is called a directed walk. Let $c^+(2)$ and $c^-(2)$, respectively, denote the number of positive and negative closed directed walks of length 2. Let K_2^+(respectively, K_2^-) be the sidigraph whose sign of an arc is $+1$(respectively, -1). The following Lemma is used to find upper bound.

Lemma 5 (Pirzada and Bhat [13])**.** *Let* S *be a sidigraph having* n *vertices and* a *arcs and let* $\lambda_1, \ldots, \lambda_n$ *be its eigenvalues. Then*

(1). $\sum_{k=1}^{n}(\text{Re}(\lambda_k))^2 - \sum_{k=1}^{n}(\text{Im}(\lambda_k))^2 = c^+(2) - c^-(2)$,

(2). $\sum_{k=1}^{n}(\text{Re}(\lambda_k))^2 + \sum_{k=1}^{n}(\text{Im}(\lambda_k))^2 \leq a$.

It is easy to see from the proof of Lemma 5 that

$$\sum_{k=1}^{n} \text{Re}(\lambda_k)\, \text{Im}(\lambda_k) = 0. \tag{29}$$

Let S be any sidigraph. We denote by $q \oplus S$, the direct sum of q copies of S. Now we will give upper bound for the total energy of sidigraphs. We remark that the idea of the proof is taken from Theorem 2.3 [14].

Theorem 17. *Let* S *be an* n*-vertex sidigraph and* a *be the total number of arcs of* S. *Let* $\lambda_1, \ldots, \lambda_n$ *be the eigenvalues of* S. *Then* $E_t(S) \leq \sqrt{na}$. *Moreover, if* S *is balanced, the equality holds if* $S = \frac{n}{2} \oplus \overleftrightarrow{K}_2^+$ *or*

$S = \frac{n}{2} \oplus \overleftrightarrow{K}_2^-$ *or* $S = \frac{n}{4} \oplus C_4$. *If* S *is unbalanced, the equality holds if* $S = \frac{n}{2} \oplus \boldsymbol{C}_2$.

Proof. Using (29) and part 2 of Lemma 5 and applying the Cauchy-Schwarz inequality to vectors $(|\mathrm{Re}(\lambda_1)+\mathrm{Im}(\lambda_1)|, \ldots, |\mathrm{Re}(\lambda_n)+\mathrm{Im}(\lambda_n)|)$ and $(1, 1, \ldots, 1) \in \mathbb{R}^n$, we have

$$\begin{aligned} E_t(S) &= \sum_{k=1}^{n} |\mathrm{Re}(\lambda_k) + \mathrm{Im}(\lambda_k)| \\ &\leq \sqrt{n}\sqrt{\sum_{k=1}^{n} |\mathrm{Re}(\lambda_k) + \mathrm{Im}(\lambda_k)|^2} \\ &= \sqrt{n}\sqrt{\sum_{k=1}^{n} (\mathrm{Re}(\lambda_k))^2 + \sum_{k=1}^{n} (\mathrm{Im}(\lambda_k))^2} \leq \sqrt{na}. \end{aligned}$$

For the second part, we consider two cases: (1). S is balanced, (2). S is unbalanced.

(1). Note that the eigenvalues of sidigraphs $\frac{n}{2} \oplus \overleftrightarrow{K}_2^+$ and $\frac{n}{2} \oplus \overleftrightarrow{K}_2^-$ are ± 1 and ± 1, respectively, each repeated $\frac{n}{2}$ times and the eigenvalues of sidigraph $\frac{n}{4} \oplus C_4$ are $\{\pm 1, \pm i\}$, each repeated $\frac{n}{4}$ times. Thus, $E_t(\frac{n}{2} \oplus \overleftrightarrow{K}_2^+) = \frac{n}{2}(|1| + |-1|) = n$, $E_t(\frac{n}{2} \oplus \overleftrightarrow{K}_2^-) = \frac{n}{2}(|1| + |-1|) = n$ and $E_t(\frac{n}{4} \oplus C_4) = \frac{n}{4}(2\,(\,|1| + |-1|\,)) = n$. On the other hand, clearly, the number of vertices and number of arcs in all these three sidigraphs are n and n. Therefore, for these sidigraphs, $E_t(S) = \sqrt{nn} = n$.

(2). The sidigraph $S = \frac{n}{2} \oplus \boldsymbol{C}_2$ has n vertices and n arcs. Thus, $E_t(S) = \sqrt{nn} = n$. However, from total energy formula (3), we have $E_t(\frac{n}{2} \oplus \boldsymbol{C}_2) = \frac{n}{2}(|1| + |-1|) = n$, since the eigenvalues of a sidigraph $\frac{n}{2} \oplus \boldsymbol{C}_2$ are $\pm i$, each repeated $\frac{n}{2}$ times. This completes the proof. □

Theorem 18. *Let* S *be a sidigraph on* n *vertices and* a *arcs and let* $S_1, \ldots, S_m$ *be its strong components with* $n_i > 1$ *vertices and* a_i *arcs, where* $i = 1, 2, \ldots, m$, *respectively. Then* $E_t(S) \leq a$. *Moreover, if* S *is balanced, the equality holds if* $S = \frac{a}{2} \oplus \overleftrightarrow{K}_2^+$ *plus some isolated vertices or* $S = \frac{a}{2} \oplus \overleftrightarrow{K}_2^-$ *plus some isolated vertices or* $S = \frac{a}{4} \oplus C_4$ *plus some*

isolated vertices, and if S is unbalanced, the equality holds if $S = \frac{a}{2} \oplus \boldsymbol{C}_2$ plus some isolated vertices.

Proof. We assume that S is a strongly connected sidigraph with n vertices and a arcs. Then $n \leq a$. By Theorem 17, we have

$$E_t(S) \leq \sqrt{na} \leq a.$$

As $S_1 \ldots, S_m$ are strong components of S with n_i vertices and a_i arcs, where $i = 1, 2, \ldots, m$, therefore $\sum_{i=1}^{m} = n$ and $\sum_{i=1}^{m} a_i \leq a$. By Theorem 2, we have

$$E_t(S) = \sum_{i=1}^{r} E_t(S_i) \leq \sum_{i=1}^{r} \leq a.$$

The proof of the second part is similar to the proof of the second part of Theorem 17 and is, thus, omitted. □

6 T-Equienergetic sidigraphs

Two sidigraphs with the same spectrum are said to be cospectral, otherwise non-cospectral. Two T-equienergetic sidigraphs of the same order are the sidigraphs which have the same total energy. Two isomorphic sidigraphs are always cospectral and, thus, are T-equienergetic. In this section, we are interested to construct a few classes of non-cospectral T-equienergetic sidigraphs.

Example 5. *Consider the sidigraphs S and H as shown in Figure 2. The dotted lines represent negative arcs and solid ones represent positive arcs. The characteristic polynomials of S and H are, respectively, given by*

$$\begin{aligned} \phi_S(\lambda) &= (\lambda^6 - 1) \\ \phi_H(\lambda) &= (\lambda^6 + 1). \end{aligned}$$

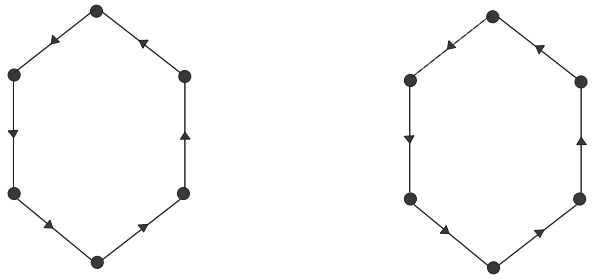

Figure 4. Equienergetic sidigraphs

Thus, the spectrums of S and H are, respectively, given by

$$\text{spec(S)} = \left\{ \pm 1, \frac{1 \pm \sqrt{3}i}{2}, \frac{-1 \pm \sqrt{3}i}{2} \right\}, \tag{30}$$

$$\text{spec(H)} = \left\{ \pm i, \frac{\sqrt{3} \pm i}{2}, \frac{-\sqrt{3} \pm i}{2} \right\}. \tag{31}$$

From (30) *and* (31), *S and H are non-cospectral T-equienergetic sidigraphs*

Theorem 19. *Let $S_1 = (2 \oplus C_2) \times S$, $S_2 = (2 \oplus \boldsymbol{C}_2) \times S$ and $S_3 = C_4 \times S$, where S is an n-vertex sidigraph. Then $E_t(S_1) = E_t(S_3)$ and $E_t(S_2) = E_t(S_3)$. Moreover, S_1, S_3 and S_2, S_3 are non-cospectral sidgraphs. Similarly, S_1 and S_2 are non-cospectral T-equienergetic sidigraphs.*

Proof. Let $\lambda_1, \ldots, \lambda_n$ be the eigenvalues of S. Then spectrums of S_1, S_2 and S_3 are, respectively, given by

$$\begin{aligned}
\text{spec}(S_1) &= \{\lambda_1 \pm 1, \lambda_1 \pm 1, \ldots, \lambda_n \pm 1, \lambda_n \pm 1\}, \\
\text{spec}(S_2) &= \{\lambda_1 \pm i, \lambda_1 \pm i, \ldots, \lambda_n \pm i, \lambda_n \pm i\}, \\
\text{spec}(S_3) &= \{\lambda_1 \pm 1, \lambda_1 \pm i, \ldots, \lambda_n \pm 1, \lambda_n \pm i\}.
\end{aligned}$$

It is clear that S_1, S_3 and S_2, S_3 are non-cospectral T-equienergetic sidigraphs. Also S_1 and S_2 are non-cospectral T-equienergetic sidigraphs. □

Example 6. *For each odd n, C_n and $\boldsymbol{C}_n$ are non-cospectral T-equienergetic sidigraphs, because* $\mathrm{spec(C_n)} = -\mathrm{spec}(\boldsymbol{C}_\mathrm{n})$ *and* $1 \in \mathrm{spec(C_n)}$ *but* $1 \notin \mathrm{spec}(\boldsymbol{C}_\mathrm{n})$.

Using Example 6, we have the following corollary of Theorem 19.

Corollary 4. *If $n \equiv 1(\mathrm{mod}\ 2)$, then $(2 \oplus C_2) \times C_n$ and $(2 \oplus C_2) \times \boldsymbol{C}_n$ are non-cospectral T-equienergetic sidgraphs. Similarly, $(2 \oplus \boldsymbol{C}_2) \times C_n$ and $(2 \oplus \boldsymbol{C}_2) \times \boldsymbol{C}_n$ are non-cospectral T-equienergetic sidigraphs.*

The following lemma will be useful to find pair of non-cospectral T-equienergetic sidigraphs.

Lemma 6. *Let $n \geq 6$ be a positive integer. Then*

(1). $E_t(C_n) = 2E_t(\boldsymbol{C}_{\frac{n}{2}})$ *if and only if* $n \equiv 2(\mathrm{mod}\ 4)$.

(2). $E_t(C_n) = 4E_t(\boldsymbol{C}_{\frac{n}{4}})$ *if and only if* $n \equiv 4(\mathrm{mod}\ 8)$.

(3). $E_t(C_n) = E_t(\boldsymbol{C}_n)$ *if and ony if* $n \equiv 2(\mathrm{mod}\ 4)$ *or* $n \equiv 1(\mathrm{mod}\ 2)$.

Proof. Proof is similar to the proof of Lemma 5.2 [10]. □

The proofs of the next Theorems are similar to the proof of Theorem 5.3 [10] and are, thus, omitted.

Theorem 20. *Let S_1 and S_2 be an n-vertex sidigraphs, $n \geq 6$, with k vertex-disjoint positive directed cycles and k vertex-disjoint negative directed cycles of lengths $m_1, \ldots, m_k$, where $m_j \equiv 1(\mathrm{mod}\ 2)$ or $m_j \equiv 2(\mathrm{mod}\ 4)$, $j = 1, 2, \ldots, k$. Then $E_t(S_1) = E_t(S_2)$. Moreover, S_1 and S_2 are non-cospectral.*

Theorem 21. *Let S_1 and S_2 be two n-vertex sidigraphs, $n \geq 6$, with k vertex-disjoint positive directed cycles and $2k$ vertex-disjoint negative directed cycles, respectively, of lengths $m_1, \ldots, m_k$ and $\frac{m_1}{2}, \frac{m_1}{2}, \ldots, \frac{m_k}{2}, \frac{m_k}{2}$, respectively, where $m_j \equiv 2(\mathrm{mod}\ 4)$, $j = 1, 2, \ldots, k$. Then $E_t(S_1) = E_t(S_2)$. Moreover, S_1 and S_2 are non-cospectral.*

Theorem 22. *Let S_1 and S_2 be two n-vertex sidigraphs, $n \geq 6$, with k vertex-disjoint positive directed cycles and $4k$ vertex-disjoint negative directed cycles, respectively, of lengths $m_1, \ldots, m_k$ and $\frac{m_1}{4}, \frac{m_1}{4}, \frac{m_1}{4}, \frac{m_1}{4}, \ldots, \frac{m_k}{4}, \frac{m_k}{4}, \frac{m_k}{4}, \frac{m_k}{4}$, respectively, where $m_j \equiv 4(\text{mod } 8)$, $j = 1, 2, \ldots, k$. Then $E_t(S_1) = E_t(S_2)$. Moreover, S_1 and S_2 are non-cospectral.*

The next two theorems give a class of non-cospectral T-equienergetic digraphs.

Theorem 23. (Khan et al. [9]) *Let S_1 and S_2 be two n-vertex sidigraphs, $n \geq 6$, with k and $2k$ vertex-disjoint directed cycles, respectively, of lengths $m_1, \ldots, m_k$ and $\frac{m_1}{2}, \frac{m_1}{2}, \ldots, \frac{m_k}{2}, \frac{m_k}{2}$, respectively, where $m_j \equiv 2(\text{mod } 4)$, $j = 1, 2, \ldots, k$. Then S_1 and S_2 are non-cospectral T-equienergetic digraphs.*

Theorem 24. (Khan et al. [9]) *Let S_1 and S_2 be two n-vertex sidigraphs, $n \geq 6$, with k and $2k$ vertex-disjoint directed cycles, respectively, of lengths $m_1, \ldots, m_k$ and $\frac{m_1}{2}, \frac{m_1}{2}, \ldots, \frac{m_k}{2}, \frac{m_k}{2}$, respectively, where $m_j \equiv 4(\text{mod } 8)$, $j = 1, 2, \ldots, k$. Then S_1 and S_2 are non-cospectral T-equienergetic digraphs.*

7 Conclusion

In this paper, we extend the concept of total energy of digraphs to sidigraphs and introduced complex adjacency matrix of sidigraphs. We calculated the total energy formulas for negative directed cycles and show that the total energy of negative directed cycles increases monotonically. We discuss the increasing property of total energy over the set $S^1_{n,h}$ and $S^2_{n,h}$ with respect to the quasi-order relation when $h \equiv 4(\text{mod } 8)$ and $h \equiv 0(\text{mod } 8)$, respectively. We find upper bound for total energy of sidigraphs. Finally, we find non-cospectral T-equienergetic sidigraphs. However, it is unclear whether the total energy of sidigraphs increases or not, with respect to the quasi-order relation over $S^1_{n,h}$ and $S^2_{n,h}$ when $h \equiv 2(\text{ mod } 4)$. We leave this problem for future work.

References

[1] B. D. Acharya, "Spectral criterion for the cycle balanced in networks," *J. Graph Theor.*, vol. 4, pp. 1–11, 1980.

[2] M. A. Bhatt, "Energy of weighted digraphs," *Discrete Applied Math.*, vol. 223, pp. 1–14, 2017.

[3] M. A. Bhat and S. Pirzada, "Unicyclic signed graphs with minimal energy," *Discrete Applied Math.*, vol. 226, pp 32–39, 2017. DOI: 10.1016/j.dam.2017.03.015.

[4] R. Farooq, M. Khan, and F. Ahmad, "Extremal iota energy of bicyclic digraphs," *Appl. Math. Comput.*, vol. 303, pp. 24–33, 2017.

[5] R. Farooq, M. Khan, and S. Chand, "On iota energy of signed digraphs," *Linear Multilinear A.*, vol. 67, no. 4, pp.705–724, 2019.

[6] K. A. Germina, K. S. Hameed, and T. Zaslavsky, "On products and line graphs of signed graphs, their eigenvalues and energy," *Linear Algebra Appl.*, vol. 435, pp. 2432–2450, 2010.

[7] I. Gutman, "The energy of a graph," *Ber. Math. statist. Sekt. Forschungszentrum. Graz.*, vol. 103, pp. 1–22, 1978.

[8] M. Khan, R. Farooq, and A. A. Siddiqui, "On the extremal energy of bicyclic digraphs," *J. Math. Inequal.*, vol. 9, no. 3, pp. 799–810, 2015.

[9] M. Khan and S. Hafeez, Total energy of digraphs, submitted.

[10] M. Khan, R. Farooq, and J. Rada, "Complex adjacency matrix and energy of digraphs," *Linear Multilinear A.*, vol. 65, no. 11, pp. 2170–2186, 2017.

[11] M. Mateljevic, V. Bozin, and I. Gutman, "Energy of a polynomial and the Coulson integral formula," *J. Math. Chem.*, vol. 48, pp. 1062–1068, 2010.

[12] I. Peña and J. Rada, "Energy of digraphs," *Linear Multilinear Algebra.*, vol. 56, no. 5, pp. 565–579, 2008.

[13] S. Pirzada and M. A. Bhat, "Energy of signed digraphs," *Discrete Applied Math.*, vol. 169, pp. 195–205, 2014.

[14] J. Rada, "The McClelland inequality for the energy of digraphs," *Linear Algebra Appl.*, vol. 430, pp. 800–804, 2009.

Sumaira Hafeez, Mehtab Khan,

Received May 14, 2020
Accepted February 24, 2021

Sumaira Hafeez
School of Natural Sciences,
National University of Sciences and Technology,
H-12 Islamabad, Pakistan
E–mail: sumaira.hafeez123@gmail.com

Mehtab Khan
School Mathematical Sciences, Anhui University,
Hefei, China
E–mail: mehtabkhan85@gmail.com

Computer Science Journal of Moldova, vol.**29**, no.1(85), 2021

Backtracking algorithm for lexicon generation*

Constantin Ciubotaru

Abstract

This paper is dedicated to generating process of the Romanian Cyrillic lexicon used between 1967 and 1989. The rules for transliteration of words from the modern Romanian lexicon to their equivalents written in Cyrillic were established and argued.

A backtracking algorithm has been developed and implemented that generates the Cyrillic lexicon using the transliteration rules. This algorithm actually is a tool to facilitate the work of the expert. The work of the expert is reduced to checking the transliterated variants and changing the transliteration rules.

Keywords: lexicon, transliteration, backtracking algorithm, decyrillization, morpho-syntactic descriptions (MSD).

1 Introduction

The problem of digitizing and preserving the historical-linguistic heritage is a priority domain of the digital agenda for Europe. The digitization process requires solving a series of problems related to the recognition, editing, translation, and interpretation of printed texts. The solving of these problems for the Romanian historical-linguistic heritage faces difficulties and specific aspects: a large number of periods in the evolution of the language, a small volume of stored resources that are also scattered, a great diversity of alphabets.

The presence of a digitized Romanian Cyrillic lexicon will contribute to the regeneration, revitalization and preservation of the heritage related to this period. Various aspects of the problem have been exposed in [1]–[3].

*This work was supported by the project Nr. 20.80009.5007.22

The paper addresses the issues related to the digitization and transliteration of the historical-linguistic heritage printed in Cyrillic script during 1967–1989 on the territory of the Moldovan Soviet Socialist Republic (MSSR), in accordance with the linguistic norms of the modern Romanian language.

During that period the Moldovan Cyrillic alphabet (AlphaCYR) was used which actually represents the Russian alphabet without the letters "ё", "щ" and "ъ" and extended by adding the letter "ж" in 1967. Complete lack of resources in electronic format and presence of fragmentary grammatical descriptions that admit ambiguous interpretations represent the main difficulties specific to the period.

According to the dexonline definition, *transliteration* is the "transcription of a text from one alphabet to another, rendering the letters by their equivalents, regardless of the phonetic value of the signs" [4].

The process of transliterating Romanian words into their written equivalents with the characters of the AlphaCYR alphabet is called *cyrillization*. For instance, *"puiului"*⇒"пуюлуй", *"fiului"*⇒"фиулуй", *"cenușiu"*⇒"ченушиу", *"viermi"*⇒"вермь", *"vierii"*⇒"виерий".

The inverse procedure for cyrillization is called decyrillization, e.g. "пуюлуй"⇒*"puiului"*, "бьет"⇒*"biet"*, "боер"⇒*"boier"*, "пепт"⇒*"piept"*.

If the digitization of the text is relatively simple, the problem of recognizing the digitized text is quite complicated, especially considering the total lack of Romanian Cyrillic resources for that period. This paper extends the results presented in [5] and aims to develop a tool for generating the lexicon corresponding to that period (noted by LexCYR), starting from the lexicon of the modern Romanian language (noted by LexROM).

The general scheme of the Romanian Cyrillic lexicon generator is presented in Figure 1.

Figure 1. The general scheme of the Cyrillic lexicon generator

2 Selection of the modern Romanian lexicon

For the choice of the modern Romanian lexicon, the following three resources were examined:

1. ***Dexonline*** [4]. It contains over 900000 entries, with a convenient interface for online use. The dictionary structure is less adaptable for processing because it does not contain explicitly the inflected forms, does not contain morpho-syntactic descriptions (MSD), and includes both forms of spelling "î" from "i" and "â" from "a" ("fân"-"fîn", "pârî"-"pîrî").
2. ***The lexicon developed at the "Al.I.Cuza" University, Iași*** [6] with over 1000000 entries. The lexicon is well structured, contains MSD labels in accordance with the tagset proposed in the project MULTEXT-East [7]. But, as in dexonline, we find both spellings "î" / "â", also many proper names and words of foreign origin, to which the rules of transliteration cannot be applied.
3. ***Reusable linguistic resources*** developed at the Institute of Mathematics and Computer Science "Vladimir Andrunachievici" [8] with over 677000 entries, including inflected forms. The formalization (packaging) of resources is quite complicated, the morpho-syntactic descriptions are incomplete.

Finally, the lexicon developed at the "Al.I.Cuza" University (LexROM) was selected with minor modifications, as follows:

1. Proper nouns and words of foreign origin were removed;
2. All words were transliterated using the spelling "â" from "a" according to the provisions of the Romanian Academy. Duplications of spellings "î" / "â" were avoided by applying an algorithm specially developed for this purpose.

The problem of spelling "î"/"â" does not affect the cyrillization process, because in both cases there is the same result at transliteration: *"â"* ⇒ "ы", *"î"* ⇒ "ы". Difficulties arise in the decyrillization process: should we apply the rule "ы" ⇒ *"â"* or rule "ы" ⇒ *"î"*?

We denote by AlphaROM the Romanian language alphabet, and by LexROM(α) – all the words from LexROM that start with the letter α, $\alpha \in$ AlphaROM.

3 Used tools

To formalize the transliteration rules and program the lexicon processing algorithms there was selected the Common LISP functional programming language [9], [10].

The Notepad++ editor was used for word processing [11], which offers advanced editing capabilities, such as:

- select text both horizontally and vertically,
- store search results in separate files,
- mark lines and operations with these lines,
- allow the use of regular expressions,
- support UTF-8 encoding for Romanian letters with diacritics and Russian, for example: Ă, ț, Â, â, Ș, Э, ц, Ы, Ш, ж,
- rich set of plugins: exporting files in various formats (RTF, HTML), the ability to launch applications (files with the extension .exe), sorting and comparing files, etc.

4 Backtracking method

The backtracking method proposes to build the solution(s) of a problem incrementally by applying iterative and/or recursive algorithms. It is assumed that there is a finite set of candidates for solutions and some internal criteria for verifying candidates. The method can be applied to generate the lexicon, as all the necessary conditions are met:

- the modern LexROM lexicon is given,
- sets of rules for transliteration are defined,
- there is a finite set of intermediate transliterated words that represent candidates for solutions,
- there are internal criteria for verifying the variants: the order of application of the rules, context-sensitive dependencies, prefixing and suffixing, the involvement of the expert,
- the set of all solutions meets the LexCYR lexicon,
- iterative and recursive algorithms are applied.

5 Algorithm of switching to the spelling "â" from "a"

The transition to the spelling "â" from "a" also will be done by transliteration. According to the provisions of the Romanian Academy, the letter "î" will always be written at the beginning and end of the word ("început", "înger", "în", "întoarce", "a coborî", "a urî"). Inside the word, it is usually written "â" ("cuvânt", "a mârâi", "român", "fân"). There are, however, a few exceptions to this rule. Words formed by prefixing words that begin with the letter "î" will keep this "î" inside. For example, "neîmpăcat", "neîngrijit", "preîntâmpinat", "dezîntors", "reînarma". The same rule will be applied to compound words: "bineînțeles", "semiînchis", "altîncotro". There are also a few exceptions, for example, the word "altînghie" will be transliterated as "altânghie", because it is not a compound word, this is the name of a flower, also called "lady's slipper". On the other hand, the word "capîntortură" (the name of a bird) will be transliterated, together with its derivatives, as "capîntortură". It is taken into account that the word comes from "cap întors" ("turned head"). The specificity of the LexROM lexicon will also be taken into account, that includes, along with the lemma words and inflected forms, phrases and word combinations, which can be spelled with "î" from "i". These words inside the construction are separated by "~". For example, "pe~înserate", "de~jur~împrejur" etc. All words w that contain at least one letter "î" can be represented as $w = w_0 \bullet$ "î" $\bullet w_1 \bullet$ "î" $\bullet \ldots \bullet w_{n-1} \bullet$ "î" $\bullet w_n$. If the word starts with "î", then $w_0 =$ "". We will mark by "" the empty string. For words ending with "î" we will have $w_n =$ "". Thus, for the letter "î" we get "î"$=w_0 \bullet$ "î" $\bullet w_1$, $w_0 = w_1 =$ "". For the word "coborî" we obtain: "coborî"$=w_0 \bullet$ "î" $\bullet w_1$, with w_0="cobor", $w_1 =$ "". For the combination of words w="din~cînd~în~cînd" we have: $w = w_0$ •"î"• w_1 •"î"• w_2• "î" • w_3="din~c" •"î"• "nd~" • "î"• "n~c" •"î"•"nd". Note that w_1="nd~" ends with "~", which means that the next word will start with "î", analogous to the prefix situation. As a result of the conversion we get "din~când~în~când".

Performing a statistical analysis of the LexROM lexicon leads to

selection of the set of all prefixes that can be inserted in front of words that start with the letter "î". This set is denoted by PREFIXES.

Algorithm of switching to the spelling "â" from "a"

0. ***Start***

1. The lexicon of the modern Romanian language LexROM is given.
* We will modify this lexicon by substituting all words with their written equivalents with "â" from "a" applying the transliteration method *\

2. We modify the LexROM by applying transliteration rules for exceptional situations. For example, "altîngie" ⇒ "altângie" (in other cases "alt" will be a prefix).

3. We build the set of prefixes that can be placed in front of words which start with the letter "î". PREFIXES={"alt" "arhi" "auto" "bine" "bio" "de" "dez" "din" "ex" "ne" "nemai" "ori" "piți" "pre" "prea" "pro" "re" "semi" "sub" "subt" "super" "supra" "tele"}.

4. ***loop for all*** $w \in$ LexROM ***do***

4.1. ***if*** w does not contain "î" ***then return***(w).

4.2. We represent $w = w_0 \bullet$ "î" $\bullet\, w_1 \bullet$ "î" $\bullet \ldots \bullet\, w_{n-1} \bullet$ "î" $\bullet\, w_n$, where $w_0, w_1, \ldots, w_n$ are words which do not contain "î", $n \geq 1$.

4.3. ***if*** $(w_0 =$ "") ***or*** $(w_0 \in$ PREFIXES) ***or*** $(w_0 = w_0' \bullet$"~") ***or*** $(w_1 =$ "") ***then*** $w_r := w_0 \bullet$"î" ***else*** $w_r := w_0 \bullet$"â".

4.4. ***loop for*** i ***from*** 1 ***to*** $(n-1)$ ***do***

4.4.1. $w_r := w_r \bullet w_i$

4.4.2. ***if*** $(w_{i+1} =$ "") ***or*** $(w_i = w_i' \bullet$"~") ***then*** $w_r := w_r \bullet$"î" ***else*** $w_r := w_r \bullet$"â".

4.5. ***end loop***

4.6. ***return***$(w_r \bullet w_n)$

5. ***end loop***

6. ***Stop***

6 The structure of the lexicons

The LexROM lexicon is represented as a list in Common LISP, each element of the list being composed of three components: (word, MSD-

label, word-lemma). For each element of the LexCYR lexicon, the fourth component – the cyrillized word (Figura 2) – is included.

(ghiocei "Ncmprn" "ghiocel")	(гиочей "ghiocei" "Ncmprn" "ghiocel")
(ridic "Vmsp1s" "ridica")	(ридик "ridic" "Vmsp1s" "ridica")
(ridica "Vmn" "ridica")	(ридика "ridica" "Vmn" "ridica")
(ridicam "Vmii1p" "ridica")	(ридикам "ridicam" "Vmii1p" "ridica")
(ridicăm "Vmsp1p" "ridica")	(ридикэм "ridicăm" "Vmsp1p" "ridica")
(ridicare "Ncfsrn" "ridicare")	(ридикаре "ridicare" "Ncfsrn" "ridicare")
(ridicat "Ncmson" "ridicat")	(ридикат "ridicat" "Ncmson" "ridicat")
(ridicat "Afpmson" "ridicat")	(ридикат "ridicat" "Afpmson" "ridicat")
(ridicat "Vmp" "ridica")	(ридикат "ridicat" "Vmp" "ridica")
(ridicat "Rg" "ridicat")	(ридикат "ridicat" "Rg" "ridicat")
(ridicatele "Ncfpry" "ridicat")	(ридикателе "ridicatele" "Ncfpry" "ridicat")
(ridicatule "Ncmsvy" "ridicat")	(ридикатуле "ridicatule" "Ncmsvy" "ridicat")
(a) LexROM structure	(b) LexCYR structure

Figure 2. The lexicons structure

The MSD label is a set of characteristics of the word viewed as part of speech. The label represents a sequence of symbols, the first symbol specifying the part of speech (for example, N - noun, V - verb, A - adjective, Rg - adverb, etc). The rest of the symbols will specify the morphological characteristics of the word, such as number, gender, person, time, case, mode, etc. The scheme of the MSD label for the noun is shown in Figure 3.

$$N\begin{bmatrix}\text{c-common}\\ \text{p-proper}\end{bmatrix}\begin{bmatrix}\text{m-masculine}\\ \text{f-feminine}\end{bmatrix}\begin{bmatrix}\text{s-singular}\\ \text{p-plural}\end{bmatrix}\begin{bmatrix}\text{r}\\ \text{o}\\ \text{v}\end{bmatrix}\begin{bmatrix}\text{y-definiteness}\\ \text{n-indefiniteness}\end{bmatrix}$$

r-direct case (nominative-accusative),
o-oblique case (genitive-dative),
v-vocative

Figure 3. The MSD label structure for noun

The following algorithm based on the backtracking strategy is proposed for generating the Cyrillic lexicon LexCYR. Sets of translitera-

tion rules are defined, cyrillization and decyrillization algorithms are constructed. The cyrillization algorithm is applied on the Romanian lexicon LexROM. A variant of the LexCYR lexicon will be obtained, which can be subjected to decyrillization, thus a new variant for the Romanian lexicon is obtained. The ideal situation would be to match these two lexicons. If inconsistencies occur, the expert intervenes, who can change the rules of cyrillization\decyrillization, can repeat the whole process or can intervene with corrections on the constructed Cyrillic lexicon.

7 Cyrillization

Unlike the problem of digitizing and recognizing printed text, which is solved relatively simply, the problem of cyrillization is more difficult. To solve this problem we will apply the transliteration method. By definition, the transliteration process consists in the consecutive application of a set of substitutions (rewriting rules). For example, *brad* ⇒ б*rad* ⇒ бр*ad* ⇒ бр*a*д ⇒ брад. Here the following rules have been applied consecutively "*b*" ⇒ "б", "*r*" ⇒ "р", "*d*" ⇒ "д", "*a*" ⇒ "а". We will call these rules general rules. For them the order of application is irrelevant.

For the letter "*i*" we have the general transliteration rule "*i*" ⇒ "и", but the following rules are also possible: "*i*" ⇒ "й" and "*i*" ⇒ "ь". Examples: *fuior* ⇒ фуйор, *fior* ⇒ фиор, *miere* ⇒ мьере.

In other cases the rules may be more complicated. For example, two rules can be applied to the letter "*g*": "*g*" ⇒ "г", "*g*" ⇒ "ӂ" – *gigant* ⇒ ӂигант. Here comes the context-sensitive rule that requires substitutions: "*gi*" ⇒ "ӂи", "*ge*" ⇒ "ӂе", "*ghi*" ⇒ "ги", "*ghe*" ⇒ "ге".

Thus, it becomes obvious that these rules must be applied before applying the general rule "*g*" ⇒ "г". Moreover, substitutions "*giu*" ⇒ "ӂиу", "*giu*" ⇒ "ӂю" are also possible. For example, *giulgiu*⇒ ӂюлӂиу, *giugiuli* ⇒ ӂюжюли.

Randomly applying the transliteration rules for the word "ghiocei", the following variants are obtained: {"гхиокеи", "гхиокеь", "гхиокей", "гхиочеи", "гхиочеь", "гхиочей", "гиокеи", "гиокеь", "гиокей", "гиочеи", "гиочеь", "гиочей", "ӂхиокеи", "ӂхиокеь", "ӂхиокей",

"жхиочеи", "жхиочеь", "жхиочей"}.

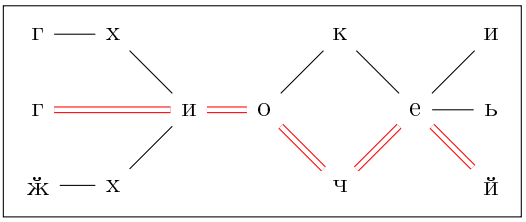

Figure 4. The transliteration scheme for "ghiocei"

In Figure 4 we show the transliteration scheme of the word "ghiocei". The scheme highlights the correct variant – "гиочей". The backtracking method allows eliminating wrong options step by step and selecting the correct one. This is done by changing the transliteration rule set, establishing some contextual dependencies, changing the order of the rules application and examining the MSD labels. In some situations the correct option can only be selected by the expert.

To fix the situations with multiple variants, we will use a list of options denoted by $[w_1][w_2]\dots[w_n]$, finally being selected only one. For example, for the words "ghiocei" and "preaiubiți" we get:

[гх][г][жх] • ио • [к][ч] • е • [и][ь][й] $\Longrightarrow$"гиочей",

пр • [еа][я]• [иу][ю] • биц • [и][ь][й] $\Longrightarrow$"пряюбиць".

We denoted by "•" the concatenation operation.

8 Classification of transliteration rules

8.1 General rules

The general transliteration rules are presented in Table 1.

Table 1. General rules of transliteration

latin	a	ă	â	b	c	d	e	f	g	h	i	î	j	k
cyrillic	а	э	ы	б	к	д	е	ф	г	х	и	ы	ж	к
latin	l	m	n	o	p	r	s	ș	t	ț	u	v	x	z
cyrillic	л	м	н	о	п	р	с	ш	т	ц	у	в	кс	з

To formalize (program) these substitutions, we will introduce the function *replace-all (w lat cyr)*, which will modify the word *w* substituting all occurrences *lat* with *cyr*. This is possible because the order of application of these substitutions is not relevant. E.g, *replace-all* ("dividend" "d" "д") = "дividенд".

Depending on the filtering stage, it is possible to enter some new general transliteration rules, for example, *replace-all* (w "gh" "г"), *replace-all* (w "ch" "к").

Usually, these substitutions are the last filter in the process of cyrillization.

8.2 Rules for prefixes

Because the words are interpreted as strings, we have to use the notions of prefix and suffix defined to process strings, as opposed to the grammatical notions of suffix and prefix. Thus, by prefix (suffix) of the string *w* we will define any substring w_1 (w_2) for which $w = w_1 \bullet w_2$. Substrings w_1 and w_2 can also be empty, i.e. "". Often the transliteration rules for prefixes differ from general rules. Thus, the prefixes "ia" and "iu", with small exceptions, will be transliterated as "я" and "ю", as opposed to their appearance inside the word when in most cases they will be transliterated as options "[я][иа]" and "[ю][иу]". Another example: in the LexROM there are about 650 words that start with the prefix "crea". Only for 6 situations it will be transliterated by "кря". All other occurrences of the prefix will be transliterated by креа". These 6 situations can be easily highlighted and formalized. This observation suggests the need to introduce a special set of transliteration rules for prefixes.

We note these rules by *replace-prefix (w prefixlat prefixcyr)*. For example, *replace-prefix (w "creang" "крянг")*, *replace-prefix (w "crea" "креа")*. The order of application is important for this type of substitution, which is simple: prefixes that are prefixes of other prefixes will be transliterated last. Thus, we first will try the transliteration "creang" ⇒ "крянг", then the transliteration "crea" ⇒ креа". Prefix rules are defined separately for all words in the LexROM that begin with the

same letter. Thus, for all letters there will be defined sets of rules for prefixes that will be applied first in the transliteration process.

8.3 Rules for suffixes

Analogously to the situation with the transliteration of prefixes, also there are defined transliteration rules for suffixes involving some specific conditions. For example, for the termination "ci" transliterations are possible: "ci" ⇒ "ч", "ci" ⇒ "чь", "ci" ⇒ "чи". To make the correct decision, MSD labels are checked. The rule "ci" ⇒ "ч" is applied, for example, for masculine nouns to the singular, nominative-accusative case (MSD = "Ncmsrn", "arici" ⇒ "арич", "cinci"⇒" чинч").

The "ci" ⇒ "чи" rule is applied to infinitive verbs and 3rd person verbs (MSD = "Vmis3s" and MSD = "Vmn", for example, "a munci" ⇒ "а мунчи"), and the rules "ci" ⇒ "чь", "ti" ⇒ "ть", "ți" ⇒ "ць", "și"⇒" шь", etc. – for nouns and adjectives in the plural dative-genitive case, and also for second-person present and past tense verbs (MSD ∈ {"Vmii2p", "Vmis2s", "Vmis2p", "Vmil2s", "Vmil2p", "Vmip2s", "Vmip2p", "Vmsp2s", "Vmsp2p", "Vmmp2p" }). Some examples are presented in Table 2.

Table 2. Transliteration of verb terminations

MSD	lat	cyr
Vmii2p	citeați	читяць
Vmis2s	citiși	читишь
Vmis2p	citirăți	читирэць
Vmil2s	citiseși	читисешь
Vmil2p	citirăți	читирэць

MSD	lat	cyr
Vmip2s	citești	читешть
Vmip2p	citiți	читиць
Vmsp2s	citești	читешть
Vmsp2p	citiți	читиць
Vmmp2p	citiți	читиць

Based on the above, unconditioned and conditioned rules are defined for the transliteration of suffixes. Respectively, the functions are defined *replace-suffix (w lat* cyr) and *replace-suffix-if (w label lat* cyr *msd*). The "label" argument of the *replace-suffix-if* function represents the label MSD of the processed word *w*, and the argument "msd" – a set of valid

MSD labels for this rule. Unlike the rules for prefixes that are defined separately for each letter, the rules for suffixes are universal and can be applied to all words.

8.4 Context sensitive rules (for diphthongs and triphthongs)

As in the case of prefixes (suffixes), along with the usual grammatical notions diphthong/triphthong, we examine other combinations consisting of two, three or more letters. We mentioned above the behavior of the diphthongs "ia" and "iu" as prefixes, but also as occurrences within the word. Other examples are presented in Table 3.

Table 3. Transliteration of diphthongs/triphthongs

Diphthong/ triphthong	Transli- teration	Examples
"ioa"	[оа]	cioară⇒ "чоарэ"
	[ьоа]	chioară⇒ "кьоарэ"
	[иоа]	mioară⇒ "миоарэ"
"ii"	[ий]	fiicele⇒ "фийчеле"
	[ии]	viile⇒ "вииле"
"eie"	[ее]	creier⇒ "креер"
	[ей]	conveier⇒ "конвейер"

Diphthong/ triphthong	Transli- teration	Examples
"ea"	[а]	ceață⇒ "чацэ"
	[я]	rea⇒ "ря"
	[еа]	ocean⇒ "очеан"
"ch"	[к]	ochi⇒ "окь"
"gh"	[г]	ghid⇒ "гид"
"ge"	[же]	ger⇒ "жер"
"ci"	[чи]	circ⇒ "чирк"

Namely the transliteration of these constructions generates the most ambiguities. More information on this topic can be found in [12]. To make right decisions, sometimes contextual rules can be supplemented with morpho-syntactic information (MSD labels). The order of application of the rules is very important. Contextual dependencies always have priority over general rules.

9 Cyrillization algorithm

The cyrillization algorithm applies consecutively the transliteration rules, previously defined, to all the words in the LexROM. It is important to follow the order of application of the rules. Of course, optional combinations will be generated, which correspond to the ambiguities. This means that later it is necessary to modify the transliteration rules or to request the intervention of the linguistic expert.

Below we present the formalized algorithm.

CYRILLIZATION ALGORITHM

0. ***Start***

1. The lexicon of the modern Romanian language LexROM and transliteration rules are given.
* We will build the Romanian Cyrillic lexicon LexCYR for the period 1967-1989 *\

2. Initial LexCYR $= \varnothing$, $\text{LexROM}_1 = \text{LexROM}$.

3. ***loop for all*** letters $\alpha \in$ AlphaROM ***do***

3.1. ***loop for all*** words $w \in \text{LexROM}_1(\alpha)$ ***do***

3.1.1. Transliteration rules for prefixes are applied.

3.1.2. Transliteration rules for suffixes are applied.

3.1.3. Context-sensitive rules for transliteration are applied.

3.1.4. General rules for transliteration are applied. The obtained result is denoted by *wcyr*.

3.1.5. *wcyr* is included in LexCYR.

3.2. ***end loop***

4. ***end loop***

5. ***Stop***

10 Decyrillization

Decyirillization faces the same problems as cyrillization. General and contextual rules are also defined. The general rules are relatively simple, for example, а $\Rightarrow$ *a*, р $\Rightarrow$ *r*, ю $\Rightarrow$ *iu*, ь $\Rightarrow$ *i*. If only the general rules are applied to transliteration, we obtain, for example, пуюлуй $\Rightarrow$ *puiului*,

бьет ⇒ *biet*, боер ⇒ *boer*, пепт ⇒ *pept*. The last two transliterations are incorrect. Correct would be боер ⇒ *boier*, пепт ⇒ *piept*. In this case, as for cyrillization, contextual rules are required (for prefixes/suffixes, diphthongs/triphthongs). E.g, г•β ⇒ *gh*•β, if $\beta \in$ {е,и,я,ю,ь} and г•β ⇒ *g*•β, if $\beta \notin$ {е,и,я,ю,ь} (георгинэ ⇒ *gheorghină*, гогоашэ ⇒ *gogoașă*).

Rules for the letter я: я ⇒ *ia* (usually at the beginning of the word), ия ⇒ *ia* (usually at the end of the word). If it is difficult to make the right decision to transliterate the letter я inside the word, then the algorithm will use the rule я ⇒ [*ia*][*ea*], leaving the right decision to the expert. More information on this topic can be found in [13].

Another difficult problem is the transliteration of the letter ы, which can be replaced by either *î* or *â*. The algorithm follows exactly the recommendations of the Romanian Academy regarding this spelling.

DECYRILLIZATION ALGORITHM

0. ***Start***

1. The Romanian Cyrillic lexicon for the period 1967–1989 LexCYR and the decyrillization rules are given.
* We will build the lexicon of the modern Romanian language (noted by LexROM_2) applying the transliteration method *\

2. Initial $\text{LexROM}_2 = \varnothing$

3. ***loop for all*** letters $\beta \in$ AlphaCYR ***do***

3.1. ***loop for all*** words $w \in \text{LexCYR}(\beta)$ ***do***

3.1.1. Transliteration rules for prefixes are applied.

3.1.2. Transliteration rules for suffixes are applied.

3.1.3. Context-sensitive rules for transliteration are applied.

3.1.4. Transliteration rules for the letter kyr ы are applied.

3.1.5. General rules for transliteration are applied. The obtained result is denoted by *wrom*.

3.1.6. *wrom* is included in LexROM_2.

3.2. ***end loop***

4. ***end loop***

5. ***Stop***

11 Lexicon generation technology

As it was mentioned above, there is a total lack of electronic resources for the period 1967-1989, a complete exposition of the grammar used is missing, and many of interpretations of the transliterated words are ambiguous. Therefore, a major role in the process of generating the lexicon belongs to expert. The proposed technology aims to automate this process. Having the cyrillization and decyrillization algorithms and the formalized sets of transliteration rules, the lexicon generation process can be realized as an backtracking algorithm. The process runs in several iterations, at each iteration the expert intervenes to modify the set of rules and, possibly, directly the built Cyrillic lexicon. This scheme is described in detail in Figure 5.

12 Conclusion

The paper proposes a backtracking technology for the generation of the Romanian Cyrillic lexicon for the period 1967–1989 applying the transliteration method. Starting from the lexicon of the modern Romanian language [6] the cyrillization and decyrillization algorithms are applied consecutively.

The intermediate results are made available to the experts, who can modify\extend the set of rules applied to transliteration, and to directly correct the built Cyrillic lexicon. The final lexicon is obtained as a result of performing several such iterations. The main problems to be solved by the experts are the ambiguities that appear as a result of cyrillization\decyrillization.

For all words in the LexROM(c) (171846 words), 6381 ambiguities were detected at the first iteration, which represents 3.7%. To overcome these ambiguities there were required two iterations. Of course, the degree of accuracy depends considerably on the qualification of the expert. The proposed technology allows the return to the previous intermediate variants, thus revising the lexicon.

In order to become aware of the role of the expert and that of con-

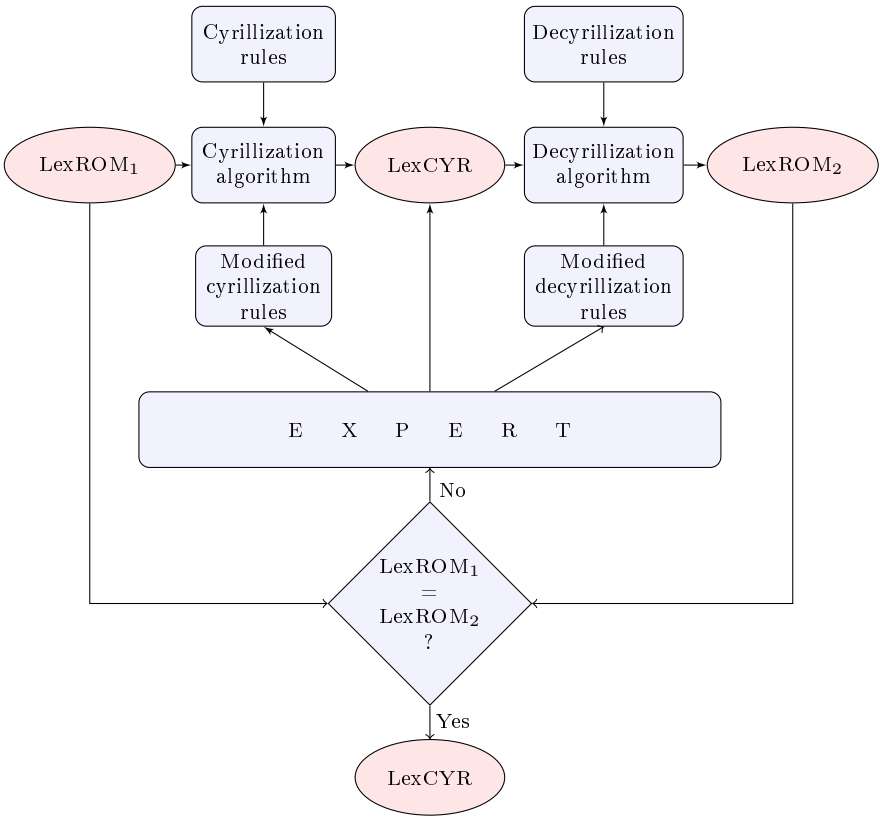

Figure 5. The scheme for generating the Romanian Cyrillic lexicon

textual dependencies, a test was performed applying to the LexROM(c) only the general rules of transliteration (paragraph 8.1). As a result, 42.2% of the correct words are obtained.

References

[1] S. Cojocaru, E. Boian, C. Ciubotaru, A. Colesnicov, V. Demidova, and L. Malahov, "Regeneration of printed cultural heritage : challenges ang technolologies," in *The Third Conference of Matematical*

Society of the republic of Moldova, (19-23 August, Chişinău), 2014, pp. 481–489.

[2] C. Ciubotaru, A. Colesnicov, and L. Malahov, "Vitalization of Moldavian Printings (1967–1989)", in *Proceedings of the 4th Conference of Mathematical Society of Moldova, CMSM4'2017"*, (June 28-July 2, 2017, Chisinau, Rep. of Moldova), pp. 491–494, ISBN 978-9975-71-915-5, Available: `http://cmsm4.math.md/Proceedings_CMSM4.pdf`.

[3] C. Ciubotaru, S. Cojocaru, A. Colesnicov, V. Demidova, and L. Malahov, "Regeneration of cultural heritage: problems related to moldavian cyrillic alphabet", in *Proceedings of the 11th International Conference "Linguistics Resources and Tools for Processing the Romanian Language (ConsILR-2015)"*, (Alexandru Ioan Cuza University, Iași, Romania, 26-27 November 2015), pp.177–184, ISSN: 1843-911X, Available: `http://consilr.info.uaic.ro/2015/Consilr_2015.pdf`.

[4] *Dexonline. Online Dex. Romanian language dictionaries.* [Online]. Available: `https://dexonline.ro/definitie/translitera\%C8\%9Bie`.

[5] C. Ciubotaru, V. Demidova, and T. Bumbu, "Generation of the Romanian Cyrillic lexicon for the period 1967 – 1989," in *Proceedings of the Fifth Conference of Mathematical Society of Moldova IMCS-55*, (September 28 - October 1, Chisinau, Republic of Moldova), 2019, pp. 309–316.

[6] The UAIC Natural Language Processing Group, *WebPosRo/resources/*, Alexandru Ioan Cuza University, Faculty of Computer Science. [Online]. Available: `http://nlptools.info.uaic.ro/WebPosRo/resources/posDictRoDiacr.txt`.

[7] Tomaž Erjavec, ed., *MULTEXT-East Morphosyntactic Specifications, Version 3.0*, May 10th, 2004. [Online]. Available: `http://nl.ijs.si/ME/Vault/V3/msd/html`.

[8] *Reusable Resources for Romanian Language Technology*, Institute of Mathematics and Computer Sciences, Moldova. [Online]. Available: `http://www.math.md/elrr/res_main.php`.

[9] Guy L. Steele, *Common Lisp the Language*, 2nd ed., USA: Thinking Machines, Inc. Digital Press, 1990, 1029 p. ISBN:1-55558-041-6.

[10] *CLISP – an ANSI Common Lisp*, Slashdot Media. [Online]. Available: `http://sourceforge.net/projects/clisp/files/clisp/2.49/`.

[11] *Notepad++. Downloads.* [Online]. Available: `https://notepad-plus-plus.org/download/v7.7.1.html`.

[12] V. Demidova, "Particular Aspects of the Cyrillization Problem," in *The Third Conference of Matematical Society of the Republic of Moldova*, (Chişinău, 19-23 August), 2014, pp. 493–498.

[13] V. Demidova, "Peculiarities of decyrillization of the Romanian language," *Studia universitatis Moldaviae. Seria "Științe exacte și economice"*, no. 2(82), pp. 16–20, 2015. (in Romanian).

Constantin Ciubotaru

Received March 2, 2021
Revised March 20, 2021

Vladimir Andrunachievici Institute of
Mathematics and Computer Science
Republic of Moldova
E–mail: `chebotar@gmail.com`