Comenius University in Bratislava
Faculty of Mathematics, Physics and Informatics
Department of Applied Informatics

Anonymous Leader Election in One- and Two-Dimensional Cellular Automata

Dissertation

RNDr. Peter Banda

Study programme: Informatics
Field of study: 9.2.1 Informatics
Supervisor: prof. RNDr. Jiří Pospíchal, DrSc.

Bratislava, 2014

Comenius University in Bratislava
Faculty of Mathematics, Physics and Informatics

THESIS ASSIGNMENT

Name and Surname:	RNDr. Peter Banda
Study programme:	Computer Science (Single degree study, Ph.D. III. deg., external form)
Field of Study:	9.2.1. Computer Science, Informatics
Type of Thesis:	Dissertation thesis
Language of Thesis:	English
Secondary language:	Slovak

Title: Anonymous Leader Election in One- and Two-Dimensional Cellular Automata

Aim: Solve the leader election problem in one- and two-dimensional cellular automata (CA) by applying genetic algorithms. Evaluate performance and analyze the dynamics of the most successful CA. Find an upped bound on performance for leader election in CA.

Keywords: leader election, cellular automata, genetic algorithms, computational mechanics, perturbation stability, performance upper bound

Tutor:	prof. RNDr. Jiří Pospíchal, DrSc.
Department:	FMFI.KI - Department of Computer Science
Head of department:	doc. RNDr. Daniel Olejár, PhD.
Assigned:	21.10.2010

Approved: 22.10.2010 prof. RNDr. Branislav Rovan, PhD.
Guarantor of Study Programme

.....
Student

.....
Tutor

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: RNDr. Peter Banda
Študijný program: informatika (Jednoodborové štúdium, doktorandské III. st., externá forma)
Študijný odbor: 9.2.1. informatika
Typ záverečnej práce: dizertačná
Jazyk záverečnej práce: anglický
Sekundárny jazyk: slovenský

Názov: Anonymous Leader Election in One- and Two-Dimensional Cellular Automata
Anonymná Voľba Šéfa v Jedno a Dvojrozmerných Celulárnych Automatoch

Cieľ: Vyriešiť problém voľby šéfa v jedno a dvojrozmerných celulárnych automatoch (CA) aplikovaním genetických algoritmov. Stanoviť úspešnosť a analyzovať dynamiku najlepších CA. Nájsť horný limit uspešnosti pre voľbu šéfa v CA.

Kľúčové slová: voľba šéfa, celulárne automaty, genetické algoritmy, komputačná mechanika, perturbačná stabilita, horný limit úspešnosti

Školiteľ: prof. RNDr. Jiří Pospíchal, DrSc.
Katedra: FMFI.KI - Katedra informatiky
Vedúci katedry: doc. RNDr. Daniel Olejár, PhD.

Spôsob sprístupnenia elektronickej verzie práce:
bez obmedzenia

Dátum zadania: 21.10.2010

Dátum schválenia: 22.10.2010 prof. RNDr. Branislav Rovan, PhD.
garant študijného programu

.....
študent

.....
školiteľ

"*I think the next (21st) century will be the century of complexity.*"
Stephen Hawking (1942-)

To my beloved wife, son, and parents.

Statutory declaration:

I hereby formally declare that I am the sole author of this thesis and I have not used any source or aid apart from the stated ones. This thesis was neither presented in equal nor in similar form to another examining board at any other university. I cited all used references observing actual academic rules.

In Bratislava, April 15, 2014 Peter Banda

Abstract

Leader election plays a crucial role in numerous distributed protocols, multi-agent systems and biological societies. Yet there is a fundamental gap in understanding its simplest variant, such as, in cellular automata, employing components that are fully uniform, deterministic, and anonymous.

In this thesis, we investigate various one- and two-dimensional binary state cellular automata that elect a leader by transforming a random initial configuration to a state where exactly one arbitrary cell is active (leader). A cellular automaton (CA) is a distributed system with a spatial topology where each processor (cell) is locally connected to its neighbors. A transition rule is represented by a look-up table, which is uniform, i.e., shared among all cells. We show that leader election is possible even in the minimal, anonymous, and uniform architecture of a binary CA. Despite being one of the structurally simplest distributed systems, a CA can exhibit various types of behavior, including complex dynamics and self-organization.

Our methodology leverages evolution of CAs by employing genetic algorithms, where chromosomes encoding candidate look-up tables undergo selection, cross-over and mutation. Even though CA's transition rules are just local and uniform, the leader election task is global and requires coordination of all cells. The findings show that the emergent dynamics of the best binary CAs are characterized by sophisticated coordination and global computation of cells, a product of spatio-temporal structures or events, namely regular domains, particles and particle interactions, known from the theory of computational mechanics. This collision-based computing enables CA to carry and exchange information over distances, eventually eliminating all but one candidate for leader. In two dimensions, slowly-contracting regions connected by lines of active cells propagate throughout the lattice and sweep any remaining active cells, before shrinking to a single cell (leader). The best strategies for both instances show a remarkably high performance rate of 0.99. In one-dimensional case the number of cells N is often modulo-restricted, such as in the best-performing CA called the strategy of mirror particles, where N is restricted to 5 mod 6. We also analyze the dynamics of two-dimensional CAs by stability measures: the Derrida measure, and the damage spreading with a discrete version of Lyapunov stability. In general, the more complex the dynamics, the better-performing the CA.

Furthermore, we identify fundamental limitations of leader election for one- and two-dimensional CAs. More precisely, we show that configurations that are symmetric or loosely-coupled are principally unsolvable. The proportion of these configurations, however, decreases dramatically with the system size. We enumerate such unsolvable configurations using linear algebra and group theory and formulate a universal upper bound on performance for the anonymous leader election problem in CA.

Our results pave the way to new distributed algorithms that are more robust and efficient than state-of-the-art systems. Our cellular automata consist of only binary components, without any extra memory or communication capabilities, and therefore use minimal resources possible. Our findings are also relevant for better understanding leader election in nature, in order to model biological processes such as morphogenesis of cell differentiation.

Abstrakt

Voľba šéfa má významnú úlohu v mnohých distribuovaných protokoloch, multiagentových systémoch, a biologických spoločenstvách. Napriek tomu bol jej najzákladnejší variant založený na plne uniformných, deterministických, a anonymných komponentoch, ktorý je stelesnený v celulárnych automatoch, doposiaľ neznámy.

Cieľom tejto práce je skúmať rôznorodé jedno a dvojrozmerné binárne celulárne automaty, ktoré si volia šéfa transformovaním náhodne vygenerovanej iniciálnej konfigurácie do stavu, kde je práve jedna bunka aktívna (šéf). Celulárny automat (CA) je distribuovaný systém s priestorovou topológiou, kde je každá bunka (procesor) spojená lokálne so susedmi. Prechodové pravidlo, reprezentované asociačnou tabuľkou, je uniformné, t.z. zdieľané všetkými bunkami. V práci ukážeme, že minimálna, anonymná, a uniformná architektúra binárneho CA umožňuje úspešnú voľbu šéfa. Napriek tomu, že sa jedná o štrukturálne najjednoduchší distribuovaný systém, CA vykazuje širokú škálu správaní, včetne komplexnej dynamiky a samoorganizácie.

Naša metodológia je založená na evolvovaní CA pomocou genetických algoritmov, v ktorých sú chromozómy kódujúce prechodové tabuľky (možné riešenia) vystavené selekcii, kríženiu, a mutácii. Prechodová tabuľka každej bunky je lokálna a uniformná, ale voľba šéfa je globálna a vyžaduje koordináciu všetkých buniek. Naše zistenia ukazujú, že emergentná dynamika najlepších binárnych CA je charakterizovaná sofistikovanou koordináciou a globálnym výpočtom buniek, ktoré sú produktom priestorovo-časových štruktúr teórie komputačnej mechaniky, menovite regulárnych domén, častíc, a interakcie častíc. Tento kolízny výpočtový model umožňuje CA reprezentovať informáciu a prenášať ju na vzdialenosti, a tým eventuálne eliminovať všetkých okrem jedného kandidáta na šéfa. V dvoch rozmeroch, sa pomaly kontrahujúce oblasti, spojené líniami aktívnych buniek, propagajú cez mriežku (toroid) a pohlcujú zvyšné aktívne bunky, až sa napokon zmenšia do jedinej aktívnej bunky (šéfa). Najlepšie stratégie pre obidve inštancie problému vykazujú pozoruhodne vysokú úspešnosť až okolo 0.99. V jednorozmernom prípade je počet buniek N často modulo-limitovaný, ako napríklad v najúspešnejšom jednorozmernom CA, stratégii zrkadlových častíc, kde sa N musí rovnať 5 mod 6. V práci tiež analyzujeme dynamiku dvojrozmerných CA pomocou mier perturbačnej stability, konkrétne Derridovou mierou (Derrida measure), a šírením poškodenia (damage spreading) s diskrétnou variantou Lyapunovej stability. Ukážeme že, čím komplexnejšia dynamika, tým vyššia úspešnosť CA.

Taktiež identifikujeme fundamentálne limitácie voľby šéfa v jedno a dvojrozmerných CA. Konkrétne, konfigurácie, ktoré sú symetrické alebo slabo prepojené sú principiálne neriešiteľné. Pomer týchto konfigurácií však rýchlo klesá s veľkosťou systému. Tieto neriešiteľné konfigurácie vyčíslime pomocou lineárnej algebry a teórie grúp, a sformulujeme univerzálny horný limit úspešnosti pre anonymnú voľbu šéfa v CA.

Naše výsledky dláždia cestu k novým, robustnejším a efektívnejším distribuovaným algoritmom. Naše celulárne automaty pozostávajú iba z binárnych komponentov, bez prídavnej pamäte alebo komunikačných možností, a preto sú ich zdroje a predpoklady minimálne. Naše zistenia sú taktiež relevantné pre lepšie pochopenie voľby šéfa v prírode, ako napríklad pri biologických procesoch zodpovedných za morfogenézu bunkovej diferenciácie.

Acknowledgments

First of all, I would like to thank my advisor prof. Jiří Pospíchal for his continuous support and guidance. I would like to acknowledge the EvCA group (especially Wim Hordijk) for their pioneering work that triggered my interest in CA computation, and John Caughman for consultations on the theoretical part of my dissertation. Also, I appreciate a help of Drew Blount, who did a good job proof-reading the manuscript, Kristina Rebrova for handling all administrative tasks, and Vojto Slovik for printing this dissertation. Finally, this journey would not have been possible without the support of my family, especially my wife, who believed in me and encouraged me to follow my dreams.

Contents

1 Introduction **1**

2 Leader Election Problem **5**
2.1 Distributed Algorithms 6
2.1.1 Definitions 7
2.1.2 UID-Based Protocols 10
2.1.3 Anonymous Protocols 12
2.1.3.1 Deterministic Model 12
2.1.3.2 Randomized Model 13
2.1.4 Self-Stabilizing Protocols 14
2.1.4.1 Self-Stabilizing Leader-Electing Protocols . . . 16
2.1.5 Our Distributed Model 17
2.2 Biological Background 18

3 Cellular Automaton **21**
3.1 Definition 21
3.1.1 One-Dimensional CA 23
3.1.2 Two-Dimensional CA 23
3.2 Overview and Applications 25
3.3 CA Computation 28
3.4 Computational Mechanics 30
3.4.1 Regular Domain 30
3.4.2 Particle 32
3.4.3 Particle Interaction 33
3.4.4 Particle Catalog 34

4 Cellular Automata Evolution **36**
4.1 Introduction to Genetic Algorithms 36
4.2 Related Work 38

5 Leader Election in Cellular Automata **40**
5.1 Leader Election as CA Computational Task 42
5.2 Model of Cellular Automata Evolution 43
5.2.1 Fitness and Performance 44
5.3 Leader Election in One-Dimensional Cellular Automata 45
5.3.1 Results of Evolving Cellular Automata 46
5.3.2 Analysis 48
5.3.2.1 Strategy of Mandatory Function 49
5.3.2.2 Density Reduction 49

5.3.2.3 Divide and Eliminate 51
5.3.2.4 First Particle-Based Strategy 52
5.3.2.5 Strategy of Mirror Particles 54
5.3.2.6 Improved Strategy of Mirror Particles 58
5.3.2.7 Transition Table Density 61
5.3.2.8 Overall N-Modulo Dependence 61
5.4 Leader Election in Two-Dimensional Cellular Automata 65
5.4.1 Results of Evolving Cellular Automata 66
5.4.2 Analysis 69
5.4.2.1 Performance and Strategies 70
5.4.2.2 Density Minimization 73
5.4.2.3 Leader Election Targeting $N = 19^2$ 74
5.4.2.4 Leader Election Targeting $N = 29^2$ 77
5.4.2.5 Transition Table Density 80
5.4.2.6 Derrida Measure 81
5.4.2.7 Damage Spreading 83
5.4.3 Asynchronous Leader Election 85

6 Limitations and Performance Upper Bound 89
6.1 Symmetric Configurations 91
6.1.1 One-Dimensional Symmetric Configurations 91
6.1.2 Two-Dimensional Symmetric Configurations 100
6.2 Loosely-Coupled Configurations 117
6.2.1 One-Dimensional Loosely-Coupled Configurations 118
6.2.2 Two-Dimensional Loosely-Coupled Configurations 123
6.3 Upper Bound on Performance 126
6.3.1 One-Dimensional Upper Bound on Performance 127
6.3.2 Two-Dimensional Upper Bound on Performance 131

7 Conclusion 135

Bibliography 139

List of Figures

2.1 Example of leader election on non-anonymous (UID-based) ring. . 11
2.2 Morphogenesis of fruit fly *Drosophila* as an example of leader election (symmetry breaking) in nature [66]. 19

3.1 Schematic of the configuration update for a binary one-dimensional cellular automaton. The transition table represents ECA 110. . . . 24
3.2 An example space-time diagram of ECA 110 starting at random initial configuration. ECA 110 exhibits complex dynamics at the edge between stability and chaos. 25
3.3 Schematic of the configuration update for a binary two-dimensional cellular automaton, namely Woltz and deOliveira's CA performing density classification. 26
3.4 Example of space-time diagrams of 2D CA. Figures show CA at 6 stages starting from a random initial configuration at time t_0 reaching a final configuration with all inactive cells at time t_{125}. The transition table represents Wolz-deOliveira's CA [98] for the density classification task. 27
3.5 Regular domains of ECA 55: (left) a two-phase domain Λ^0 and (right) a one-phase domain Λ^1. 31
3.6 A domain filter construction and an example of space-time-diagram filtration (partially reproduced from [48]. 33
3.7 (a) Space-time diagram and (b) filtered version with particle identification [28]. 34

4.1 Genetic algorithm as a cyclic process. 38

5.1 The strategy of mandatory transition function - space-time diagrams of (a) successful and (b) unsuccessful leader election. . . . 48
5.2 The density reduction strategy - space-time diagrams of (a) successful and (b) unsuccessful leader election. 50
5.3 The divide and eliminate strategy - space-time diagrams of (a) successful and (b) unsuccessful leader election. 51
5.4 The first particle-based strategy - space-time diagrams of (a) successful and (b) unsuccessful leader election. 54
5.5 The strategy of mirror particles - space-time diagrams of (a) successful and (b) unsuccessful leader election. 56
5.6 The strategy of mirror particles - space-time diagrams annotated with particles showing typical behavior for all N modulo 6 classes. The number of cells N goes from 149 (top left) to 154 (bottom right). 59

5.7 The improved strategy of mirror particles - space-time diagrams of (a) successful and (b) unsuccessful leader election. 60
5.8 Relation between the fitness and λ (transition table density) for one-dimensional leader election showing all chromosomes from both evolutionary sets. Note a critical high-performing region around $\lambda = 0.46$. 62
5.9 The N-modulo universal strategy - space-time diagrams of (a) successful and (b) unsuccessful leader election. 63
5.10 Relation between performance $P^{100}_{unif\,149}(\phi)$ and the ratio of cumulative performance for all N modulo 6 classes (representatives): $\frac{\sum_{N=149}^{154} P^{100}_{unif\,N}(\phi)}{P^{100}_{unif\,149}(\phi)}$. N-modulo universality (y-axis) represents the number of modulo classes (out of 6 defined) that produce similar performance as the training number of cells $N = 149$. 64
5.11 Fitness of the best chromosomes from each population of the three evolutionary sets: DM, LE 19^2, and LE 29^2. 68
5.12 Performance of the last best chromosomes from three evolutionary sets (DM, LE 19^2, and LE 29^2) for the square sizes $N = 1^2, \ldots, 40^2$ calculated as an average over 10^4 runs. The maximal time t_{MAX} allowed for leader election is set to 300 steps in the left and 1000 in the right column. 71
5.13 Example space-time diagrams of the best-performing density-minimizing CA on lattice size $N = 40^2$. Figures show a CA computation starting from an initial configuration with active cells distributed uniformly (time t_0), followed by 7 state snapshots. The CA optimized for density minimization fails to elect a leader and reaches a final configuration with 7 active cells at time t_{94}. 72
5.14 Performance of the best density-minimizing CA for the square sizes $N = 1^2, \ldots, 40^2$ calculated as an average over 10^4 runs using uniform and density-uniform initial distributions. The maximal time t_{MAX} allowed for leader election in (a) and (b) is 300. Figures (a) and (b) show the ratio of runs that end in a fixed point with a single active cell (1 L FP), two active cells (2 Ls FP), three or more active cells (3+ Ls FP), or no fixed point (No FP). Figure (c) plots the number of ones in a final configuration. 74
5.15 Example space-time diagrams of the best-performing leader-electing CA targeting $N = 19^2$ on lattice size $N = 40^2$. Figures show a CA computation starting from an initial configuration generated by a uniform distribution (time t_0), followed by 7 state snapshots. The CA reaches a final configuration with a single active cells at time t_{215}. 75

5.16 Performance of the best leader-electing CA targeting 19^2 for the square sizes $N = 1^2, \ldots, 40^2$ calculated as an average over 10^4 runs using uniform and density-uniform initial distributions. The maximal time t_{MAX} allowed for leader election in (a) and (b) is 300. Figures (a) and (b) show the ratio of runs that end in a fixed point with a single active cells (1 L FP), two active cells (2 Ls FP), three or more active cells (3+ Ls FP), or no fixed point (No FP). Figure (c) plots the number of ones in a final configuration. 76
5.17 Example space-time diagrams of the best-performing leader-electing CA targeting $N = 29^2$ on lattice size $N = 40^2$. Figures show a CA computation starting with a uniform initial distribution (time t_0), followed by 7 state snapshots. The CA reaches a final configuration with a single active cells at time t_{212}. 77
5.18 Performance of the best leader electing CA targeting 29^2 for the square sizes $N = 1^2, \ldots, 40^2$ calculated as an average over 10^4 runs using uniform and density-uniform initial distributions. The maximal time t_{MAX} allowed for leader election is 300 in (a) and (b), and 1000 in (c) and (d). Figures (a-d) show the ratio of runs that end in a fixed point with a single active cell (1 L FP), two active cells (2 Ls FP), three or more active cells (3+ Ls FP), or no fixed point (No FP). Figure (e) plots the number of ones in a final configuration. 79
5.19 Relation between the fitness and λ (transition table density) of CAs from the three evolutionary sets: DM, LE 19^2, and LE 29^2. Note that a critical high-performing region for leader election (b-c) correlates with $\lambda \in (0.48, 0.6)$ excluding 0.5. 80
5.20 Derrida curves for 100 randomly generated two-dimensional CAs with Moore neighborhood, and the last best CAs from all evolutions for the density minimization task (DM), and the leader election task with $N = 19^2$ (LE 19^2) and $N = 29^2$ (LE 29^2). Averages over all CAs and 100 configurations per each Hamming distance d_t are plotted. Note that the identity line represents the critical dynamical regime, i.e., the closer to the line, the more complex the dynamics. The portion shown in the plot is limited to $d_t \leq 0.5$. . . 82
5.21 Damage spreading for 100 randomly generated two-dimensional CAs with Moore neighborhood, and the last best CAs from all evolutions for the density minimization task (DM) and the leader election task with $N = 19^2$ (LE 19^2) and $N = 29^2$ (LE 29^2). Averages for all CAs over 1000 runs are plotted. 84

5.22 Relation between the fitness and λ (transition table density) of asynchronous CA from the density minimization (DM Async)and leader election (LE Async) evolutionary sets. Note that a critical high-performing region for leader election (b) correlates with $\lambda \in (0.61, 0.64)$. 86
5.23 Example space-time diagrams of the best-performing asynchronous leader-electing CA on lattice size $N = 40^2$. Figures show a CA computation starting from an initial configuration generated by using uniform distribution (time t_0), followed by 7 state snapshots. The CA fails to elect a leader and reaches a final configuration with 2 active cells at time t_{116}. 87
5.24 Performance of the best asynchronous leader-electing CA for the square sizes $N = 1^2, \ldots, 40^2$ calculated as an average over 10^4 runs using uniform and density-uniform initial distributions. The maximal time t_{MAX} allowed for leader election is 300 in (a) and (b). Figures (a) and (b) show the ratio of runs that end in a fixed point with a single active cells (1 L FP), two active cells (2 Ls FP), three or more active cells (3+ Ls FP), or no fixed point (No FP). Figure (c) plots the number of ones in a final configuration. 88

6.1 A space-time diagram of CA computation on a one-dimensional symmetric configuration. Note that leader election from a symmetric configuration is impossible. 92
6.2 Space-time diagrams of CA computation on a two-dimensional symmetric configuration showing a lattice at three consecutive time steps. Note that leader election from a symmetric configuration is impossible. 100
6.3 A space-time diagram of leader-electing CA on a one-dimensional loosely-coupled configuration, which is a configuration where distance of active cells $\geq 2r + 1$. Note that leader election from a loosely-coupled configuration (a fixed point) is impossible. 118
6.4 Space-time diagrams of CA computation on a two-dimensional loosely-coupled configuration, which is a configuration where distance of active cells $\geq 2r + 1$. Note that leader election from loosely-coupled configurations (fixed points) is impossible. 122
6.5 Probability of selecting insolvable binary configurations for one-dimensional symmetric and/or loosely-coupled configurations using uniform and density-uniform distributions and radius $r = 3$. . . 129

6.6 Probability of selecting insolvable binary configurations for two-dimensional symmetric and/or loosely-coupled configurations using uniform and density-uniform distributions and Moore neighborhood, i.e., a square neighborhood with radius $r = 1$. 133

List of Tables

5.1 Performance of the strategy of mandatory transition function using uniform and density-uniform distributions. 49
5.2 Performance of the density reduction strategy using uniform and density-uniform distributions. 50
5.3 Performance of the divide and eliminate strategy using uniform and density-uniform distributions. 52
5.4 The particle catalog of the first particle-based strategy. 53
5.5 Performance of the first particle-based strategy using uniform and density-uniform distributions. 54
5.6 The particle catalog of the strategy of mirror-particles. 55
5.7 Performance of the strategy of mirror particles using uniform and density-uniform distributions. 57
5.8 The strategy of mirror particles - typical results of the crucial leader-electing interactions with respect to N modulo 6 classes. . . 58
5.9 Performance of the improved strategy of mirror particles using uniform and density-uniform distributions. 61
5.10 Performance of the N-modulo universal strategy using uniform and density-uniform distributions. 65
5.11 Performance of the best density-minimizing CA using uniform and density-uniform initial distributions. 73
5.12 Performance of the best leader-electing CA targeting $N = 19^2$ using uniform and density-uniform initial distributions. 76
5.13 Performance of the best leader-electing CA targeting $N = 29^2$ using uniform and density-uniform initial distributions and the maximal time $t_{MAX} = 300$ and $t_{MAX} = 1000$. 78

6.1 The mandatory rows for any leader preserving ϕ of a binary CA with $r = 3$. 119
6.2 The mandatory rows for any leader preserving ϕ of a binary two-dimensional CA with Moore neighborhood. 123
6.3 Performance of the improved strategy of mirror particles, the best one-dimensional binary cellular automaton with radius $r = 3$, compared to theoretical upper bound performance. Both uniform and density-uniform distributions are considered. 130
6.4 Maximal performance of the best two-dimensional binary cellular automata with Moore neighborhood from Sections 5.4.2.3 and 5.4.2.4 compared to theoretical upper bound performance. Both uniform and density-uniform distributions are considered. 134

1

Introduction

It is surprising that structurally homogeneous and simple systems, natural as well as artificial, can generate highly complex dynamics. Currently we are at the beginning of a new era, in which the limits of prevalent reductionism in science are questioned. Consequently, the complexity approach, emphasizing holistic and emergent properties of systems, is ambitiously becoming a new paradigm. For me, it is very thrilling to be a part of this scientific exploration. For a long time, I have been wondering about the universality of self-organization and the often striking transition from order to complexity and chaos. The mystery of life, its autopoetic nature and an aim to see beyond boundaries which formally detach individual agents are the most crucial motivational factors, stimulating my imagination and pushing me towards new challenges.

In the last decades, natural and social scientists alike are increasingly facing problems related to the principal concepts of complexity and self-organization [37, 75]. These phenomena are usually closely tied to distributed, decentral-

ized, systems demonstrating (to some extent) unpredictable dynamics. There is a tremendous discrepancy between conventionally designed logic-based computation and the robust continuously-adapting behavior of the systems observed in nature. A better understanding of underlying global organizational principles and limitations is, therefore, critical. The primary goal of this thesis is to explain how global coordination, information processing and information exchange emerge from local interactions.

In order to do that, we tackle a core distributed problem, the leader election problem; introduced by Smith [89]. Given a net of processors, the problem is to design a distributed algorithm that elects a single processor, a leader, starting from an initial configuration where all the processors are in the same state. The purpose of leader election is to choose a processor that will coordinate activities of the system. This is an important prerequisite to many distributed algorithms for such tasks as finding maximal cliques, exploring graphs, and broadcasting. The most common approach is to consider leader election of distinguishable unique processors by applying a search for a processor with particular minimal or maximal ID [41]. The probabilistic Las Vegas algorithms, which assign IDs to processors randomly, may run forever but they terminate within finite time on average [56].

Even though the leader election problem originates in the theory of distributed algorithms, our model and methodology are applied in the unconventional context of multi-agent systems and complexity research. To be able to analyze the system dynamics, we opt for a distributed system with the simplest structure, binary cellular automaton, which has been extensively used to study various aspects of dynamical systems and artificial life. Cellular automata ware introduced by a pioneer of the computational age, von Neumann [77], as an alternative to a con-

ventional computer architecture that was mostly serial. Despite being one of the structurally simplest distributed systems, a CA can exhibit various types of behavior, including complex dynamics and self-organization [62, 96, 97]. We tackle the most difficult, however most fundamental, variant of leader election, in the minimal, anonymous, and uniform architecture of a binary one- and two-dimensional CAs.

Our methodology leverages evolution of CAs [74, 28] by employing genetic algorithms [73], a standard optimization technique—an advanced gradient-descent climb—to find (sub)optimal solutions, i.e., CA's transition tables, for leader election. The findings [8, 9, 11] show that the emergent dynamics of the best CAs is characterized by sophisticated coordination and global computation of cells, a product of spatial-temporal structures or events, namely regular domains, particles and particle interactions, known from the theory of computational mechanics [27, 55, 48, 49]. These approaches have been applied at the Santa Fe Institute by former EvCA (Evolutionary Cellular Automata) and CM (Computational Mechanics) groups [54] led by J. Crutchfield and M. Mitchell to understand natural systems and also to engineer decentralized artificial systems which can give rise to emergent computation. The best-performing CAs for leader election show a remarkably high performance of $0.94 - 0.99$. Our CA model has $O(N)$ time complexity, and each processor (cell) uses just $O(1)$ memory. We also analyze the dynamics of two-dimensional CAs by stability measures: the Derrida measure [34, 86], and the damage spreading [83, 68] with a discrete version of Lyapunov stability [17]. A perturbation of CA's configuration, whose dynamics are closer to the critical complex regime (also called the edge of chaos), will not die out nor spread out. The critical regime has been shown optimal for information pro-

cessing and computing. Our findings agree with the general properties of the complex regime, namely, more complex the CA's dynamics, more successful the leader election. Further, we briefly introduce and analyze two-dimensional asynchronous leader election.

In this thesis we ask whether a better-performing CA than we found could exist. We identify general limitations that no one- or two-dimensional CA can overcome [10, 12]. We show that a minimal, fully uniform and anonymous (no identifications) architecture of CA cannot produce a correct output from all input configurations. We enumerate such *unsolvable configurations*, both symmetric and loosely-coupled configurations, using linear algebra and group theory and formulate a universal upper bound on performance for the anonymous one-dimensional leader election problem. Since a transition rule is synchronous and uniform, configuration symmetry is maintained through computation, and therefore a desired state with a single active cell (leader), which is inherently non-symmetric, could not be reached. Loosely-coupled configurations are configurations where active cells are too far from each other, and therefore by implications of leader election task must be fixed points. That again prevents a successful leader election. Despite these problems, we show that the proportion of insolvable configurations decreases dramatically with the system size.

Our findings are directly applicable for design of more effective and robust distributed protocols and networks. We also suggest that by CA-based leader election we could better understand and model biological processes such as morphogenesis of cell differentiation [66, 76], where leader election breaks symmetry in a newly formed organism.

A leader is best when people barely know he exists.
Witter Bynner (1881-1968)

2

Leader Election Problem

Leader election [89] addresses a very rudimentary issue. How could the components of a distributed system reach a general, global agreement on a single individual leader? Leader election, in contrast with its simple and intuitive definition, is due to its applications one of the most extensively studied and complicated problems. As a basic routine for global coordination in artificial and natural systems, it is widely used in the theory of distributed algorithms [35, 64, 40], sociobiology [90, 61], development biology [66, 76], and multi-agent systems [59].

Even though the CA model and methodology we employ is not usually related to distributed algorithms, to understand why the problem is non-trivial we discuss various (standard) distributed algorithms for leader election. Moreover, we present a brief (socio)biological background and illustrate the functions of leader election on selected examples.

2.1 Distributed Algorithms

The leader election problem, also known as distributive choice problem, was originally introduced by Smith in 1971 [89] and soon after became one of the most fundamental problem in distributed computing. Informally, the processors in a distributed network, each executing the same algorithm, are required to elect a unique processor (a leader).

The goal of leader election is to break the system's symmetry by choosing a single processor that will coordinate global activities. It is also a prerequisite of other distributed algorithms such as finding maximal cliques, exploring graphs, and broadcasting. By choosing a leader it is possible to execute centralized protocols in a decentralized environment. There also exist numerous other leader election algorithms depending on the chosen topology (e.g., ring, complete graph, and matrix), on the amount of information processors can handle (e.g., global orientation, net size), and on self-stability etc.

In this thesis we deal exclusively with anonymous leader election in one- and two-dimensional toroidal topologies. Considering uniform and simple topologies allows dynamics analysis, such as that using the theory of computational mechanics. Moreover, the ring and toroid are the *drosophila* of distributed computing, as many interesting challenges already reveal the root of the problem in these fundamental topologies.

This section gives the reader a general overview of leader electing distributed algorithms, with special focus on memory and time complexity, model assumptions and limitations. Since our cellular automaton model can be understood without much knowledge about distributed algorithms, and our approach is not

prevalent in this field, we omit the construction details of most algorithms. First, we start with the core definitions and then we briefly show the most fundamental findings for UID-based and anonymous, deterministic and randomized, and self-stabilizing instances of the problem. For more information about distributed algorithms for leader election, we refer the reader to the many excellent textbooks that already cover these areas, e.g., [70, 91].

2.1.1 Definitions

Definition 2.1.1. (Distributed System) *A distributed system is a set of n communicating processors (state machines)* $P_1, P_2, \ldots, P_n$. *Each processor is represented as a distinct node of the directed graph* $G = (V, E)$ *where* V *is a set of nodes and* E *is a set of communication links (edges) such that* $\forall p, q \in V, (p, q) \in E$ iff *processor* p *can communicate by sending a message to or sharing memory with processor* q. *Symbol* Σ_i *denotes the set of states of processor* P_i. *A configuration is a vector* $c \in \Gamma = \Sigma_1 \times \ldots \Sigma_n$ *of the processors' states. A transition relation* δ_i *defines an event (e.g. send message, read from/write to registry) and a change of state of processor* P_i *according to the chosen communication model.*

Definition 2.1.2. (Computation) *A computation* e *of the system is a finite or infinite sequence of configurations* $c_0, c_1, \ldots$ *where* c_{j+1} *is reached from* c_j *by a step in which transition relations* $\delta_1, \ldots, \delta_n$ *are applied, starting from initial configuration* c_0. *A computation is said to be* maximal *if the sequence is either infinite or it is finite and no processor is enabled in the final configuration. The set of maximal computations starting from initial configurations* $\mathcal{B} \subset \Gamma$ *is denoted* $\varepsilon_{\mathcal{B}}$, *where* ε *is the set of all possible maximal computations (i.e.* $\varepsilon = \varepsilon_{\Gamma}$*)*

Definition 2.1.3. (Ring) *A distributed system consisting of n-processors arranged in a cycle, where each processor has exactly two neighbors.*
(Unidirectional Ring) *A ring where edges (links) are oriented in one direction, so that each processor can retrieve information only from one processor, called its predecessor.*
(Bidirectional Ring) *A ring where edges (links) are not oriented, thus each processor can retrieve information from both neighboring processors.*

Definition 2.1.4. (Message Passing Communication Model) *Processors communicate by sending messages (taken from some alphabet M) to each other. The state of each processor contains a special component,* buff_i*, in which incoming messages are buffered. The possible events include state change, sending of a message on some edge* $e \in E$ *and delivering a message.*

Definition 2.1.5. (Shared Memory Communication Model) *Processors communicate via a common memory area that contains a set of shared variables (registers). There exist various types of registers differing in supported atomic operations (e.g. read/write, read-modify-write) and access (e.g. single-writer single-reader, single-writer multi-reader). Depending on type, processors can read, write or modify the content of register. Let* $R_1, \ldots, R_m$ *be m registers associated with processors* $P_1, \ldots, P_n$*. A configuration is then a vector* $c \in \Sigma_1 \times \ldots \Sigma_n \times \mathrm{T}_1 \times \ldots \mathrm{T}_m$ *where* T_j *is the set of possible values of register* R_j*.*

Remark. *Shared memory models for leader election on rings usually presume that each processor has its own register, thus* $m = n$*. Each register is only writable by its associated processor, but neighboring processors are also allowed to read its content.*

Definition 2.1.6. (Uniformity) *A distributed system (algorithm) is called* uniform *if the number of processors n is not known to the processors. If n is known, the algorithm is called* non-uniform.

Remark. *In some cases, expression* uniformity *has much wider scope. Namely, it also refers to the attribute of distributed systems in which the processors' state sets and transition relations are the same, thus* $\Sigma_1 = \Sigma_2 = \ldots = \Sigma_n = \Sigma$ *and* $\delta_1 = \delta_2 = \ldots = \delta_n = \delta$. *This type of uniformity is considered to be a natural requirement for most distributed systems.*

Definition 2.1.7. (Anonymity) *A distributed system is* anonymous *if processors are identical (not distinguishable), i.e. they do not have any form of identity. Otherwise, each processor has unique identity (UID) and system is* non-anonymous.

Definition 2.1.8. (Timing) *In a* synchronous *system an algorithm proceeds in rounds, so the processors operate simultaneously in lockstep by using the same clock. In each round a processor receives messages that were sent to it in that round, performs a local computation and then sends messages. The shared memory version of this model is called PRAM. In an* asynchronous *system processors take steps in arbitrary order and each processor has its own independent clock.*

Definition 2.1.9. (Self-Stabilization) *Following Dijkstra's original notion, a protocol* $\mathcal{P}$ *is self-stabilizing if, starting from any initial configuration, every execution of* $\mathcal{P}$ *eventually reaches a point from which its behavior is correct. Formally [84], the protocol* $\mathcal{P}$ *is self-stabilizing for the problem* $\mathcal{PR}$ *if and only if there exists a predicate* $\mathcal{L}$ *defined on configurations such that:*

- Convergence - *All computations reach a configurations that satisfies* $\mathcal{L}$. $\forall e = (c_1, c_2, \ldots) \in \varepsilon, \exists n \geq 1, c_n \vdash \mathcal{L}$
- Correctness - *All computations, from* $\mathcal{L}$*, satisfy the problem* $\mathcal{PR}$*. Formally:* $\forall e \in \varepsilon_{\mathcal{L}}$, $e \vdash \mathcal{PR}$

Here $x \vdash \mathcal{P}$ *means that x satisfies the predicate* $\mathcal{P}$*.*

Definition 2.1.10. (Leader Election) *We assume that there is a distinguished subset of possible processor states in which a processor is considered to be a leader. To solve the leader election problem, a distributed system must reach a configuration where the state of exactly one processor is, and then remains, within a defined subset, while the states of all other processors remain outside that subset. Naturally, it is presumed that all processors must have the same transition relation (function), otherwise the problem would be trivial. Initial configuration in the classic definition are expected to be homogenous configurations with all processors in the same non-leader state, whereas in self-stabilizing versions the initial configuration is arbitrary.*

Remark. *Besides this general definition, the leader election problem has several other variants. For example, in a system with distinct UIDs one may require that leader must be the processor with the maximal (or minimal) UID. Also, one may require that all processors will know the UID of the elected leader.*

2.1.2 UID-Based Protocols

UID-based protocols, in which processors are required to be distiguishable by (comparable) unique identifiers (UIDS), were historically the first distributed algorithms for leader election. In this case, the processor with a particular (minimal

or maximal) UID is elected. Therefore, this approach is often referred to as a decentralized extrema-finding problem (Figure 2.1).

The basic algorithm for unidirectional rings was given by Le Lann in 1977 [64]. The idea is that each processor sends a message with its UID around the ring. When a processor receives a message with UID, it always stores the UID and passes it further. This routine continuous until each processor receives back its own identity, which indicates that all messages have passed the full cycle. A processor becomes leader if and only if it has the smallest identifier among all identifiers collected during execution. The algorithm requires $O(N^2)$ messages in the worst and average case. Chang and Roberts [22] adapted Le Lann's algorithm by reducing the number of transferred messages. When processor receives a message it compares the UID in this message with its own. If the delivered UID is greater then it keeps passing it, otherwise discards the incoming message. By this improvement the average-case message complexity decreases to $O(N \log N)$.

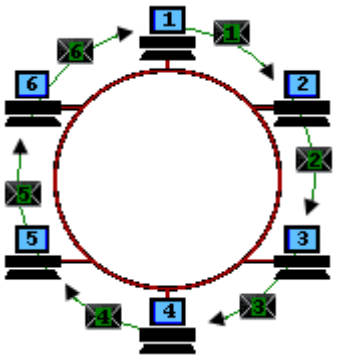


Figure 2.1: Example of leader election on non-anonymous (UID-based) ring.

Hirschberg and Sinclair [51] developed a leader election algorithm for bidirectional rings with a worst-case message complexity of $O(N \log N)$, which is optimal in O-notation. The idea is that rather than sending messages all the way around

the ring, each processor sends messages in both directions that turn around and come back. Such "boomerang" method repeats to successively greater distances of power 2. Similar algorithms for unidirectional rings were proposed soon after by Peterson [80], and Dolev, Klawe and Rodeh [36].

In a UID-based system, each processor keeps the UID-value of its candidate for leadership in regular memory, therefore, at least N states per processor is required. Obviously all mentioned algorithms have $O(N)$ time complexity.

Another approach to distinguish processors is based on orientation (*compass*). Each processor is assigned to some coordinates within the space of Z^d and through the distance measurement by sending of signals, the lexicographically lowest processor is elected leader [78].

2.1.3 Anonymous Protocols

Now we ask, what kind of a distributed system could be topology-robust, such that processors can be freely added or removed during runtime? Intuitively, without a central entity that assign unique IDs to the components, keeping all processors distinct would be challenging. Requiring each (new) component to register violates the very essence of distributed control. In this section we present so-called anonymous protocols, which consist of processors that do not carry an identity and are in fact indistinguishable (see Definition 2.1.7).

2.1.3.1 Deterministic Model

We start this section with one of the most crucial impossibility results that clearly shows how symmetry and leader election are interrelated. In fact, leader election

is often referred to as symmetry breaking.

Theorem 2.1.1. *(Angluin[3]) There does not exist a terminating deterministic algorithm for the leader election problem in an anonymous network due to the fact that full symmetry of the system cannot be broken without allowing either an infinite computation or an erroneous result.*
Remark. *This holds for both synchronous and asynchronous, as well for uniform and non-uniform protocols.*

Imagine a deterministic, anonymous system starting from the initial configuration in which all processors are in the same state. At first glance it is obvious that such a system is fully symmetric. By simple induction it can be shown that in each computational step, full uniformity of states is maintained, thus, the processors are always in the same state and no leader can be elected.

2.1.3.2 Randomized Model

Randomized algorithms use random assignment of pseudo-IDs to processors, breaking symmetry in anonymous networks. After this initial step, a deterministic leader election algorithm is employed. Unlike UID-based algorithms presented in section 2.1.2, randomized algorithms must be able to handle situations where multiple processors generate the same pseudo-ID, and also detect if no leader is elected.

The pioneering Itai and Rodeh [56] algorithm, which is probabilistic and non-uniform, terminates with probability one, and all its terminal states are correct, meaning that exactly one leader is elected. It operates in $O(N)$ memory and average-time complexity and has $O(N \log N)$ average-case message complexity.

More recently, a constant space, probabilistic, anonymous algorithm based on UID-based Franklin's algorithm was proposed [7].

We would like to stress that in general, randomized algorithms require weakened a definition of the distributed problem to embrace uncertainty about the execution time or accuracy of a solution. For instance, the Itai-Rodeh algorithm is always accurate, however, hypothetically it may run forever.

2.1.4 Self-Stabilizing Protocols

The notion of self-stabilization was originally introduced by Dijkstra in 1974 and has quite an interesting story. As defined in Definition 2.1.9 self-stabilizing algorithms eventually achieve a legitimate state in a finite number of steps regardless of initial configuration. In other words, the system does not presume any specific initial configuration. The system can recover from deviations or failures and return to the desired global state. A specific IC required by non-stabilizing protocols is often unrealistic in real life scenarios, since addition or removal of a processor in a network would require a reset of all processors to a defined initial state. This means that the system would have to be temporarily stopped and restarted. Currently, design of self-stabilizing algorithms is very popular but notoriously difficult. Self-stabilization embraced terms of fault tolerance and reliability and, beyond a doubt, proved to be a beneficial attribute of distributed systems. It can resolve several types of failures, such as inconsistent initialization, memory crash, and transmission errors.

Dijkstra coined the term self-stabilization in conjunction with the token circulation problem on a bidirectional ring. While speculating on the existence of

self-stabilizing algorithm, he realized the first barrier, which is a direct implication of Angluin's theorem.

Theorem 2.1.2. *(Non trivial) deterministic, anonymous self-stabilization is not possible.*

Remark. *This is valid for all distributed algorithms, naturally including the leader election.*

As a matter of fact, Dijkstra had to presume something that would break a symmetry of his model. He introduced a composite atomicity model with a centralized daemon, the scheduler, which determines a single processor that makes a step when several processors are enabled. Even in this model, considering an adversary daemon, symmetry might be maintained for a ring of composite size when processors of the same modulo class are selected after each other. Therefore, in order to solve his problem he concluded that processors must be distinguished (asymmetric), and he assumed the system consists of a single distinguished processor and the rest are identical.

Theorem 2.1.3. *(Dijkstra [35]) There is no anonymous, uniform n-processor self-stabilizing ring if n is composite.*

Remark: The term uniform here means that all processors are the same in terms of state space and transition relation (see Remark in Definition 2.1.6), and cannot be distinguished.

After about 20 years, Burns and Pachl demonstrated that the distinguished processor is actually not necessary for a ring of prime size.

Theorem 2.1.4. *(Burns, Pachl [16]) Anonymous, self-stabilizing, uniform token circulation is possible if n is prime.*

Burns has also shown that a central daemon (centralized control) is not required for a self-stabilizing system if the so-called noninterference property is presumed (more in [15]).

A successful technique for symmetry breaking in self-stabilizing algorithms is randomization. As mentioned in Section 2.1.3.2, randomized algorithms in general require a weakened definition of the problem (in this case including self-stabilization) to incorporate a probabilistic outcome.

2.1.4.1 Self-Stabilizing Leader-Electing Protocols

The first deterministic leader-electing protocol on bidirectional prime-size rings was presented by Itkins, Lin and Simon [57]. It uses just constant space per processor, however, all possible values of its 9 variables form 6272 different states, time complexity is $O(N^2)$. In addition, the model requires a central daemon that every step enables a processor. In [13] a lower bound of N states per processor for any deterministic or randomized, self-stabilizing, unidirectional ring for leader election was proved. Fish and Johnen [39] presented a space-optimal, deterministic algorithm with a central daemon for prime-sized unidirectional rings with $O(N)$ space and $O(N^3)$ time complexity. Fischer and Jiang's approach [40] leveraged so-called eventual *leader detector ?*, an oracle that eventually detects the presence or absence of a leader. There also exist various UID-based, deterministic, self-stabilizing leader-electing protocols that are basically very similar to those discussed in section 2.1.2.

Higham and Myers [50] developed a randomized, self-stabilizing, non-uniform algorithm solving token circulation and leader election on anonymous, synchronous, and unidirectional rings of arbitrary size. Awerbuch and Ostrovsky [5] presented a randomized protocol requiring just $\lg^* N$ states and $O(N \log N)$ time on bidirectional rings. Furthermore, the leader election problem on anonymous ring has been shown equivalent (transformable) to round-robin token management [72].

2.1.5 Our Distributed Model

We use the model of cellular automata, which is anonymous, deterministic, uniform, synchronous, self-stabilizing and shared-memory-based. It operates in linear time $O(N)$ and uses constant (binary state) memory on bidirectional rings or two-dimensional toroids. We would like to emphasize three crucial aspects of our CA model with relation to the distributed algorithms presented in previous sections.

First of all, our CA model tackles one of the fully symmetric instances of the problem that are principally insolvable. Second, because of this limitation, CAs could not reach 100% performance and the self-stabilizing property could not hold for all configurations. Still, it is one of the most fundamental instances of leader election. Third, each processor operates with minimal possible memory—just a binary state—and compared to other distributed models mentioned in previous sections do not require any additional prerequisites, such as a centralized demon, oracle, randomization etc. A detail comparison of our CA model with distributed algorithms is provided in Chapter 5.

2.2 Biological Background

Leader election has an important role for global coordination, decision making and spatial orientation of a variety of social and biological systems. Animal groups have to collectively decide about communal movements, activities, nesting sites and cooperative hunting, that crucially affect their survival and reproduction [24].

Sociobiology distinguishes two decision making procedures—shared consensus, in which the group does what the majority or defined threshold number (quorum) of its members vote for, and unshared consensus also referred to despotic decision, in which one individual (leader) makes the decision that the rest of the group follows. For example, penguins *Columbia livia* decide mostly by shared consensus, however, in certain situations experienced birds contribute more to the decision [25]. Despotic decision of one individual can be found in the population of bottlenose dolphins where behavioral signals of one individual precipitate shifts in the behavior of the entire group [69]. A consensus decision in the group of *Macaca tonkeana* involves nearly all group members. On the other hand, just a few dominant and old individuals take a prominent role in populations of *Macaca mulatta* [90]. Biologically-inspired computational models of cell differentiation elect a leader by combining local inhibition and competition [76]. Within a homogeneous region, some cells have to be elected to take on special roles, such as in the development of a wing during morphogenesis of a fruit fly *Drosophila* (Figure 2.2) [66]. Evolution of leader election in the system of self-replicating digital organisms (AVIDA) is described in [59].

From the functional point of view, shared consensus is more balanced and

Figure 2.2: Morphogenesis of fruit fly *Drosophila* as an example of leader election (symmetry breaking) in nature [66].

less error-prone than despotic decision, even though the group stability is paid for in higher time consumption. On the other hand, despotic decision is prompt, which is beneficial especially in danger and prevents any kind of unproductive stalemates in society. A despotic decision inherently involves a higher risk and extremal tendencies, due to absolute disregard of opinions of the rest of group. An interesting analogy can be found in the political organization of ancient Rome. In the time of peace two consuls together with senate ruled the country and made decisions, whereas single (elected) individual, dictator, was in command of the whole empire for the period of six months during a war.

In animal societies, a leader is usually elected or determined by attributes such as age, knowledge and/or dominance; however, variable leadership with no correlation to dominance has been observed in bird and mammal species [61]. From

the perspective of distributed algorithms, a biological leader election based on attribute comparison (where the individual with "highest score" becomes a leader) is an instance of the extrema-finding problem, whereas variable leadership is anonymous or probabilistic. The leader election problem is also referred to as a queen bee problem that underlines its biological plausibility.

The leader election is widely used, but what processes hidden behind the scene are responsible for that? Let us make a fundamental abstraction and consider a fully uniform society with anonymous agents without any memory. Would it be possible to elect a leader in such a case? In this work we demonstrate that the model required for leader election does not have to be complicated at all, and no comparable attributes of individuals, such as, size and age, are needed. In particular, we present the minimal biologically-inspired distributed system of cellular automaton with the aspiration to explain or at least give some insights to leader election at the elementary levels of cells and societies, both biological and artificial.

Things don't change. You change your way of looking, that's all.

Carlos Castaneda
(1931-1998)

3

Cellular Automaton

A cellular automaton (CA) is a distributed system with spatial topology where each processor (cell) is locally connected to its neighbors. As one of the structurally simplest distributed systems, the CA model is fundamental for studying complexity in its purest form [95, 28]. CAs have been successfully used in numerous research fields and applications, such as modeling artificial life [62], physical equations [42, 93], social and biological simulations [38], etc.

3.1 Definition

A CA [23] consists of a lattice of N components, called *cells*, with cycled boundaries (toroid topology) and a *state set*, Σ. A state of the cell with index i is denoted $s_i \in \Sigma$, where $k = |\Sigma|$. A *configuration* is then a sequence of cell states:

$$\mathbf{s} = (s_0, s_1, \ldots, s_{N-1}).$$

Given topology and the number of neighbors n, a *neighborhood* function $\eta : \mathbb{N} \rightarrow \Sigma^n$ defines the set of cells whose state is accessible (visible) to cell i. Note that usually each cell is its own neighbor.

The transition rule $\phi : \Sigma^n \rightarrow \Sigma$ is applied in parallel to each cell's neighborhood resulting in the synchronous update of all of the cells' states $s_i^{t+1} = \phi(\eta_i^t)$. The update repeats over time starting from an initial configuration (IC), usually generated at random. The transition rule is represented either by a transition table, also called a look-up table, or a finite state transducer [53]. In this thesis we focus exclusively on uniform binary CAs, where all cells share the same transition function and each cell can be in one of two possible states.

The global transition rule $\Phi : \Sigma^N \rightarrow \Sigma^N$ is defined as the transition rule with scope over the configurations

$$\mathbf{s}^{t+1} = \Phi(\mathbf{s}^t)$$

Because CAs are deterministic, their state evolution is fully determined by the initial configuration and the global transition rule governing the cell updates. For all possible configurations $\Omega = \Sigma^N$ of N cells, an *ensemble operator* $\mathbf{\Phi} : 2^\Omega \rightarrow 2^\Omega$ is defined as a function mapping a set of possible lattice configurations Ω^t at time t to another set of configurations Ω^{t+1} at the next time step $t + 1$.

A CA's configuration $\mathbf{s}^t$ can be seen as a finite string, or word, over the alphabet Σ, and a set of configurations Ω^t as a regular language [49]. Therefore, $\mathbf{\Phi}$, which maps a regular language Ω^t to another regular language Ω^{t+1}, is in fact a *finite state transducer* and CA a *regular language processor*.

In this thesis we deal with two CA topologies: one-dimensional (ring) and two-dimensional (toroid), discussed in Section 3.1.1 and 3.1.2 respectively.

3.1.1 One-Dimensional CA

The neighborhood function $\eta : N \rightarrow \Sigma^n$ for a one-dimensional CA is defined by radius r ($n = 2r + 1$) as

$$\eta_i = (s_{i-r}, \ldots, s_i, \ldots, s_{i+r})$$

A one-dimensional CA with k states and radius r is often denoted as a (k, r)-CA. An elementary cellular automaton (ECA) is a specific kind of one-dimensional CA, where $(k, r) = (2, 1)$. Since the transition table of an ECA has $2^3 = 8$ rows there are $2^8 = 256$ possible ECAs. An ECA is encoded by the decimal representation of the binary transition table's outputs ordered from the highest to the lowest neighborhood configurations. For instance, ECA 110's table outputs are $0, 1, 1, 0, 1, 1, 1, 0$, which correspond to 8 possible neighborhood configurations going from $1, 1, 1$ to $0, 0, 0$. Figure 3.1 shows the update mechanism of a one-dimensional binary CA exemplified on ECA 110.

Dynamics of a one-dimensional CA are often illustrated by using a space-time diagram (Figure 3.2), where the lattice of cells is displayed horizontally with an active cell marked as black (state 1) and an inactive cell as white (state 0). Time goes vertically from the top to the bottom.

3.1.2 Two-Dimensional CA

The neighborhood function of a two-dimensional CA is defined by radius r, however, unlike the one-dimensional case, the shape of neighborhood could vary depending on chosen metrics or distance formula. The most commonly used is a

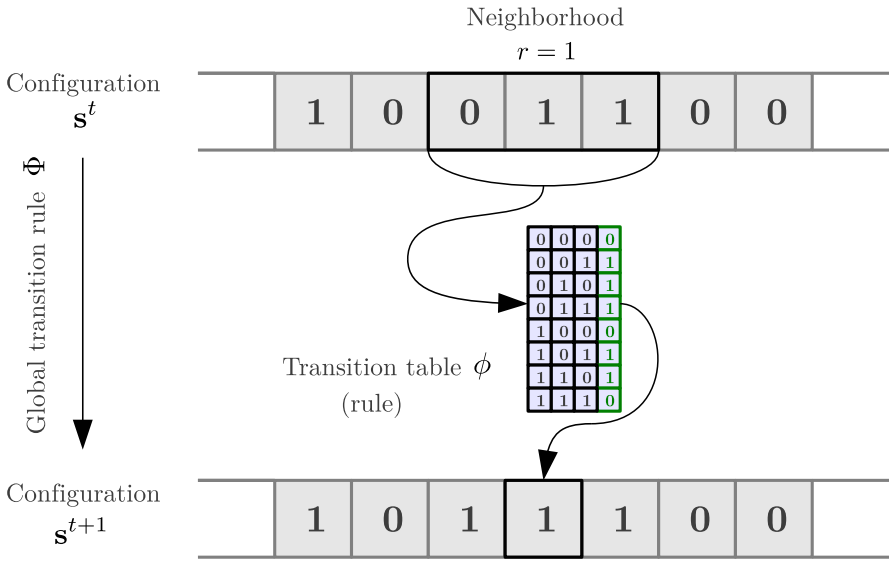


Figure 3.1: Schematic of the configuration update for a binary one-dimensional cellular automaton. The transition table represents ECA 110.

square neighborhood, where distance is defined as maximum of coordinate differences in each direction, thus, neighborhood has a square shape of length $2r + 1$ containing $(2r + 1)^2$ cells. The so-called Moore neighborhood is a square neighborhood with radius $r = 1$ containing 9 cells. The square neighborhood function $\eta : N \to \Sigma^n$ is therefore defined as

$$\eta_{i,j} = (s_{i-r,j-r}, \ldots, s_{i+r,j-r}, \ldots, s_{i-r,j}, \ldots, s_{i+r,j}, \ldots, s_{i-r,j+r}, \ldots, s_{i+r,j+r})$$

Similarly to the one-dimensional case, two-dimensional CA's boundaries are cyclic, i.e., we treat them as tori. Figure 3.3 shows the update mechanism for a two-dimensional binary CA with a Moore neighborhood.

The dynamics of two-dimensional CAs are illustrated as a series of configura-

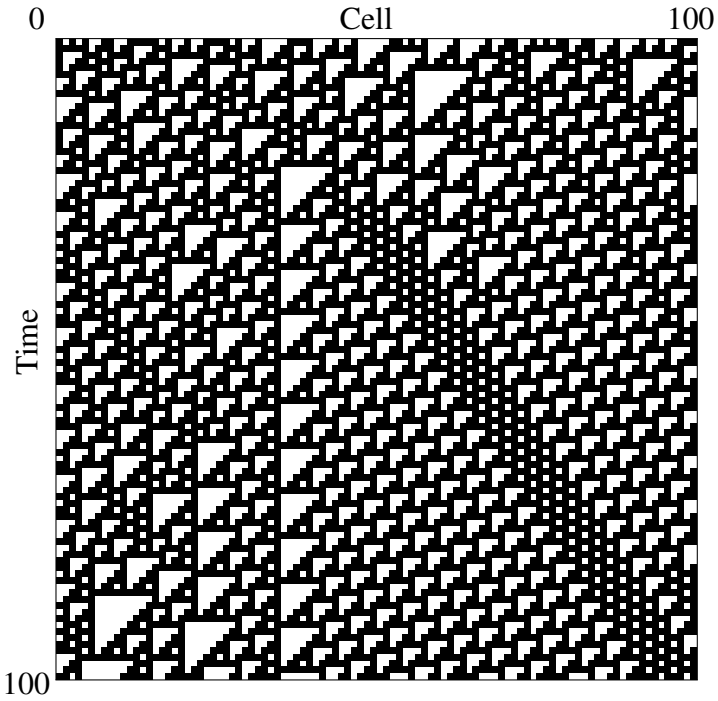


Figure 3.2: An example space-time diagram of ECA 110 starting at random initial configuration. ECA 110 exhibits complex dynamics at the edge between stability and chaos.

tion snapshots, where again an active cell is black and inactive cell white (Figure 3.4).

3.2 Overview and Applications

John von Neumman [77] introduced the concept of a cellular automaton (CA) to explore the logical requirements for machine self-replication and information processing in nature. Despite having no central control and limited communication among components, CAs are capable of universal computation, i.e., are Turing-machine equivalent, and can exhibit various dynamical regimes.

One of the most famous cellular automata is the Game of Life [44], a two-dimensional CA with Moore neighborhood, introduced by Conway in 70's. The

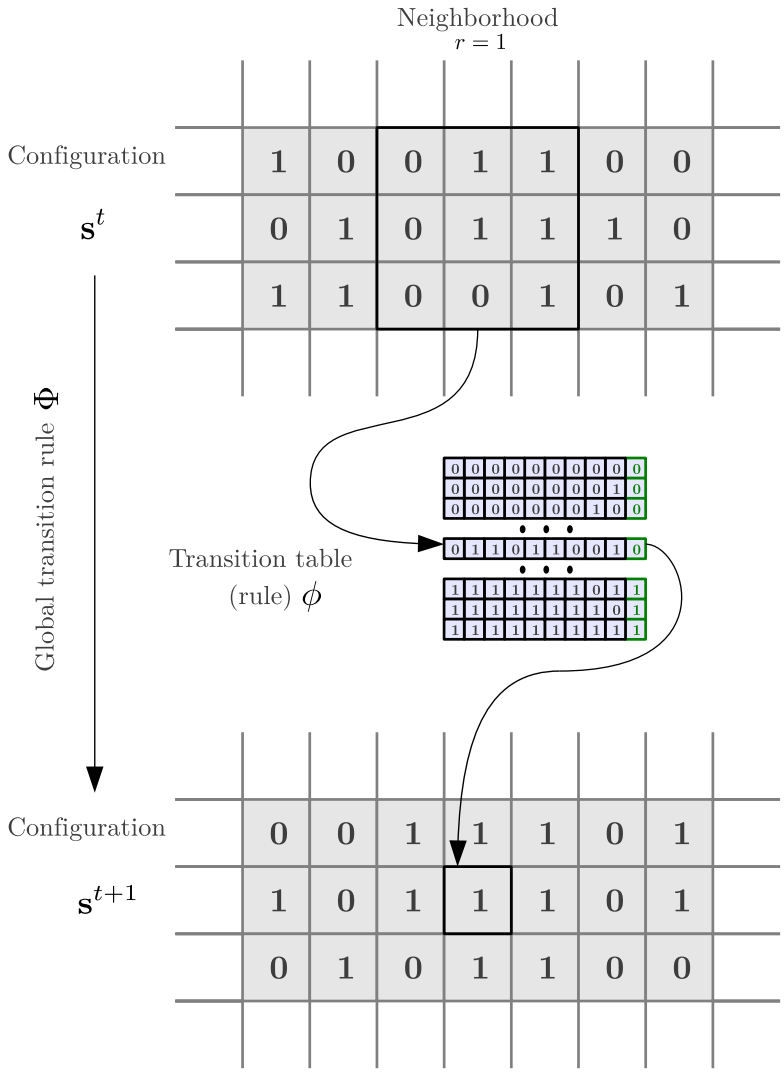
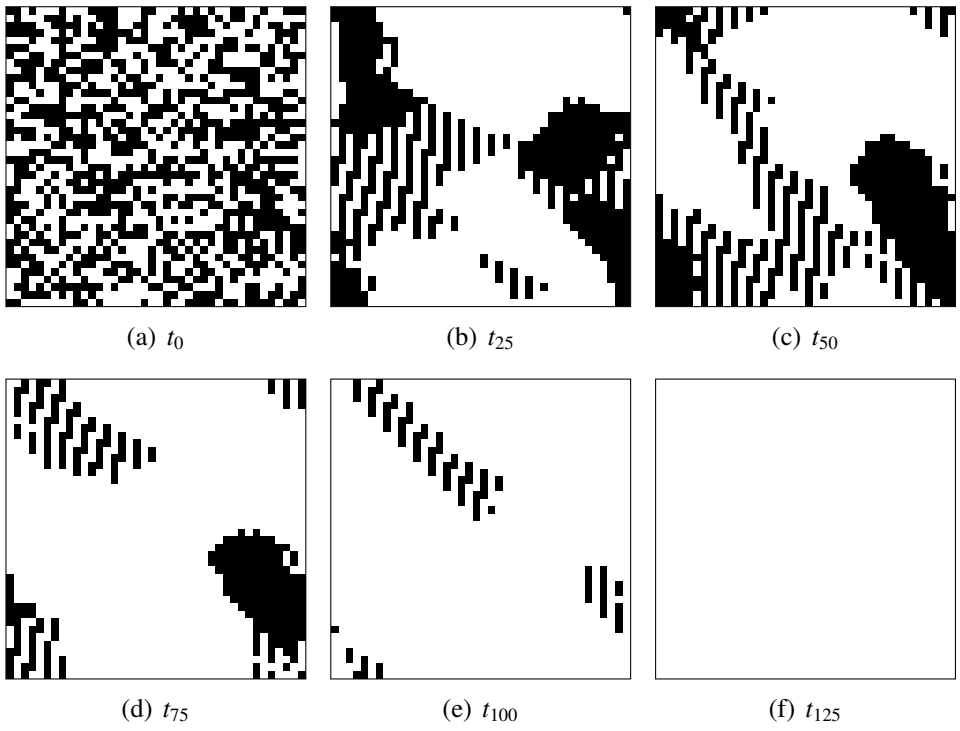


Figure 3.3: Schematic of the configuration update for a binary two-dimensional cellular automaton, namely Woltz and deOliveira's CA performing density classification.

Game of Life CA has shown universal computability. Its dynamics are capable of generating several dozen spatial-temporal propagating patters, such as so-called gliders and spaceships. It was used to study emergent behavior, complexity, and to some extent biological features. The Game of Life transition rule is defined as follows—a cell becomes active if it is surrounded by exactly three active cells



(a) t_0 (b) t_{25} (c) t_{50}

(d) t_{75} (e) t_{100} (f) t_{125}

Figure 3.4: Example of space-time diagrams of 2D CA. Figures show CA at 6 stages starting from a random initial configuration at time t_0 reaching a final configuration with all inactive cells at time t_{125}. The transition table represents Wolz-deOliveira's CA [98] for the density classification task.

(reproduction); an active cell stays active if it neighbors two or three active cells (life), and turns inactive otherwise (death). It means that cells die if they are too densely or too loosely-coupled.

Smith [88] showed that one-dimensional CAs are computationally universal. Toffoli and Fredkin [42, 92] employed CAs to model physics, namely the equations of heat and waves, and the Navier-Stoke equation. Their CA computational model assumed reversibility and information storing (fundamental laws of particles physics). Wolfram [96, 97] systematically explored dynamics of one-dimensional CAs, especially ECAs. He identified four qualitative classes of

CA behavior, which apply to all dynamical systems—fixed points, limit cycles, chaotic and complex dynamics. Vichniac and Bennett [42, 93, 14] demonstrated that CAs are ideal, sufficiently simple candidates for studying various emergent properties of dynamical systems, such as turbulence, chaos, symmetry breaking, and fractality. Vichniac [94] also investigated one-dimensional nonuniform CAs that chose probabilistically one of two rules each step. He showed that this nonuniform model creates different complex structures. Langton [63] postulated self-reproducing CA, so-called Q-loops or SR-loops. He also coined the term ”edge of chaos”, a special narrow complex regime, promoting information transfer and processing. Langton classified CA dynamics [62] by λ parameter, density of ones in the transition table. Garzon [45] presented two generalizations of CA—discrete neuron networks and automota networks. Ermentrout [38] used CA to model various biological phenomena. CAs are also relevant for new computer distributed architectures. Thousands or millions of processors organized into regular lattice could in fact harness several key CA properties [100]. Other applications include market simulation (voting model), design and simulation of new biomolecules, and simulation of reversible logic in quantum computational models [47]. Solving computational tasks by evolutionary optimization of CA transition tables is discussed separately in Section 4.2.

3.3 CA Computation

CA computation is a product of local interactions of cells, which share the same transition rule. Designing cellular automata to perform a task that requires global coordination of cells is notoriously difficult. Because of massive parallelism, it

is often nontrivial to predict the behavior of a CA from its transition rule. For instance, the density (or majority) classification task is to decide whether the initial configuration contains more ones or zeros. If there are more ones in the IC, a final configuration is expected to be a homogeneous configuration of all ones, otherwise all zeros. This task is very easy for an architecture with central processing, which could straightforwardly calculate the ratio of ones in a configuration by a simple iteration through all cells. This task is, however, quite difficult for CA. All greedy approaches such as the local voting or region expanding do not sufficiently work. In fact because this task requires a global information exchange it is often used a CA benchmark task [28, 20].

Another issue with CA computation is the interpretation of CA dynamics. Even if we find a CA that handles a given task, such as, density classification, we often cannot properly describe *how* it solved the task. Most common approaches rely on statistical information-theoretical measures of CA dynamics, e.g., entropy [99], mutual information, and perturbation stability [67]. Note that these measures are often used for random Boolean networks [58], which are generalized CAs with non-spatial (non-uniform) topology and non-uniform transition tables. Based on these properties CA dynamics could be categorized in one of the four dynamical regimes, known also as Wolfram's classes. Wolfram's classes are stable point, limited cycle, complex and chaotic behavior.

The statistical analysis and categorization of CA's dynamical regime is beneficial, but not detailed enough. In the next section we provide an alternative, more *narrative* description of CA dynamics using the theory of computational mechanics.

3.4 Computational Mechanics

Since CAs are completely discrete, it was quite difficult to analyze their behaviors with instruments known from the theory of conventional dynamical systems. The methodology of computational mechanics [27, 48] finally bridged this gap by synthesising the concepts from both computational and dynamical system theories. The global, collective dynamics of CA can be, therefore, understood and described in terms of space-time structures: domains, which form the regular background of computation, particles acting as carriers of information, and particle interactions, which perform the CA's information processing.

3.4.1 Regular Domain

A regular domain is a homogeneous space-time region containing the same set of (sub)configurations that appear invariantly over and over again, both in time and in space. A regular domain forms a regular, "nicely shaped" region—it is a spatio-temporal region in which "things are basically the same" [48].

Formally, a *regular domain* Λ^j is a process language consisting of a set of spatial configurations with following two properties:

1. *Temporal invariance* - CA dynamics represented by $\mathbf{\Phi}$, maps Λ^j to itself, i.e. $\mathbf{\Phi}^p(\Lambda^j) = \Lambda^j$ (minimal p defines temporal periodicity of domain). Alternatively, CA map a configuration in Λ^j to another configuration in Λ^j, i.e. $\Phi(\mathbf{s}) = \mathbf{s}'$ and $s, s' \in \Lambda^j$.

2. *Spatial homogeneity* - Λ^j is spatial translation invariant, i.e. it is position independent and can occur at any site. Thus the process language of a Λ^j is

strongly connected.

Regular domains can be discovered either by a visual inspection of space-time diagrams or a method called *ϵ-machine reconstruction* [30, 26, 48]. The ϵ-machine reconstruction automatically identifies deterministic finite state automata (DFAs) as candidates for regular domains. After finding potential regular domains, one must verify whether the candidate domains satisfy the given two conditions. The FME algorithm [48] could be used for validation of temporal invariance. For spatial homogeneity, we need to verify whether any pair of states is path-accessible by a simple exploration of the DFA process graph. Each regular domain Λ^j has a temporal and a spatial periodicity. As mentioned in the definition of temporal invariance, the temporal periodicity of a domain Λ^j is the smallest value p for which $\mathbf{\Phi}^p(\Lambda^j) = \Lambda^j$. A domain Λ^j then cycles through its phases $p^t_{\Lambda^j}$

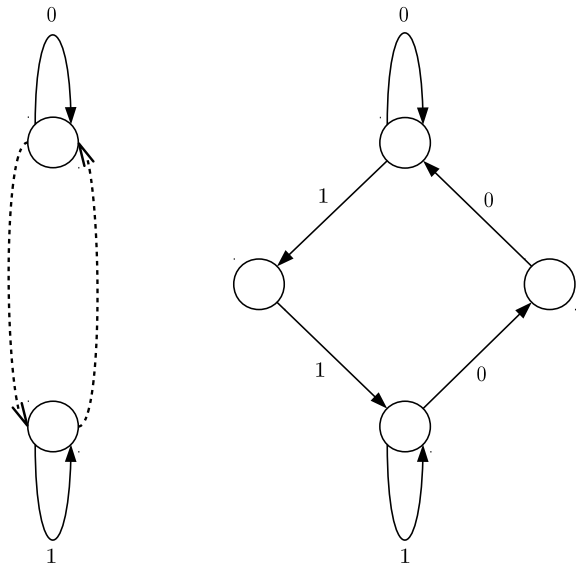


Figure 3.5: Regular domains of ECA 55: (left) a two-phase domain Λ^0 and (right) a one-phase domain Λ^1.

in a fixed order (numbered $1, \ldots, p^t_{\Lambda^j}$). There is a corresponding regular expression (and minimal DFA) for each one of these phases. The spatial periodicity $p^s_{\Lambda^j}$ of a domain Λ^j is the number of cells in the CA lattice after which each temporal phase of the domain repeats itself. Expressed in other words, it is equal to the number of states of associated minimal DFAs. The spatial periodicities of all temporal phases of a domain Λ^j are equal.

For instance ECA 55 has two regular domains $\Lambda^0_{55} = \{0^*, 1^*\}$ and $\Lambda^1_{55} = \{(000^*111^*)^*\}$. The temporal periodicity p^t reflecting the number of domain phases is 2 for Λ^0_{55} and 1 for Λ^1_{55}. Spatial periodicity is determined by the number of states of each phase DFA, therefore $p^s_{\Lambda^0_{55}} = 1$ and $p^s_{\Lambda^1_{55}} = 4$ (Figure 3.5).

For given domains $\mathbf{\Lambda} = \{\Lambda^0, \Lambda^1, \ldots\}$ with associated DFAs $M_{\Lambda^0}, M_{\Lambda^1}, \ldots$ a so-called domain filter or domain transducer filters all lattice configurations that belong to some domain as illustrated in Figure 3.6. For construction details see , e.g., [27].

3.4.2 Particle

A filtered space-time diagram reveals space-time structures that are not part of domains but are regular propagating objects formed at a boundary between domains. These structures, called *particles*, encode and transfer information among distinct regions, and therefore enable global communication. As described by Mitchell [75], particles are similar to *signals* that are created and propagated by local interactions among cells.

Formally, a particle, usually marked by a Greek letter, is a spatially localized and temporally periodic structure at the boundary of two domains with limited

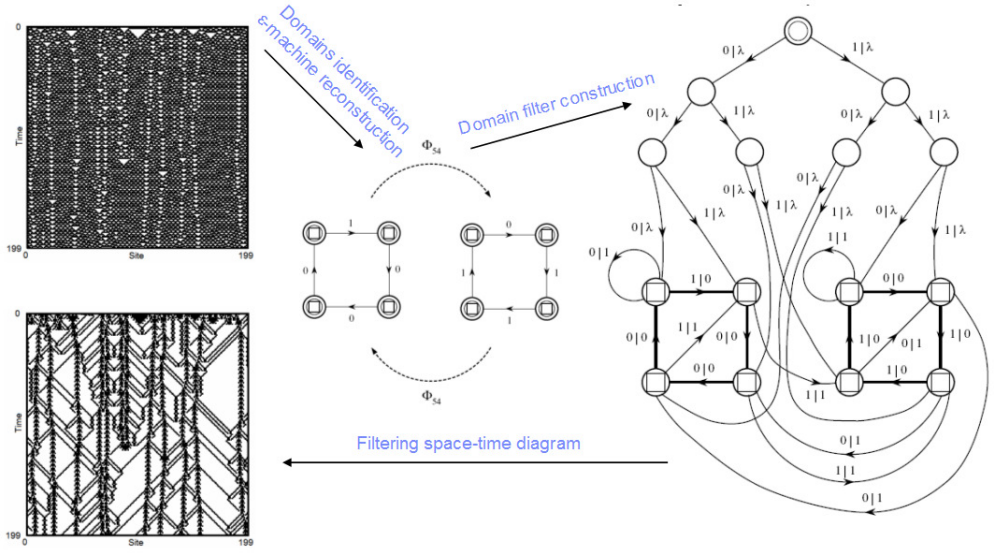


Figure 3.6: A domain filter construction and an example of space-time-diagram filtration (partially reproduced from [48].

width (Figure 3.7). As opposed to a domain, a particle is bounded in space, thus, it does not have a spatial periodicity. The temporal periodicity of a particle α is denoted as p_α and set of particles as $\mathbf{P} = \{\alpha, \beta, \dots\}$. The displacement d_α of a particle α is defined as the number of cells that particle is shifted during one period (the left displacement is negative, the right one is positive). Velocity v_α is then $v_\alpha = d_\alpha / p_\alpha$. Note that the term "particle" was coined to emphasize the analogy with physical particles, which are however much more complicated.

3.4.3 Particle Interaction

Since the particle velocities often vary, their trajectories are not collinear and they can collide. A *particle interaction* processes information encoded in colliding particles and represents a high-level form of decision making.

For instance, a particle interaction denoted as $\alpha + \beta \rightarrow \gamma$ (Figure 3.7) represents

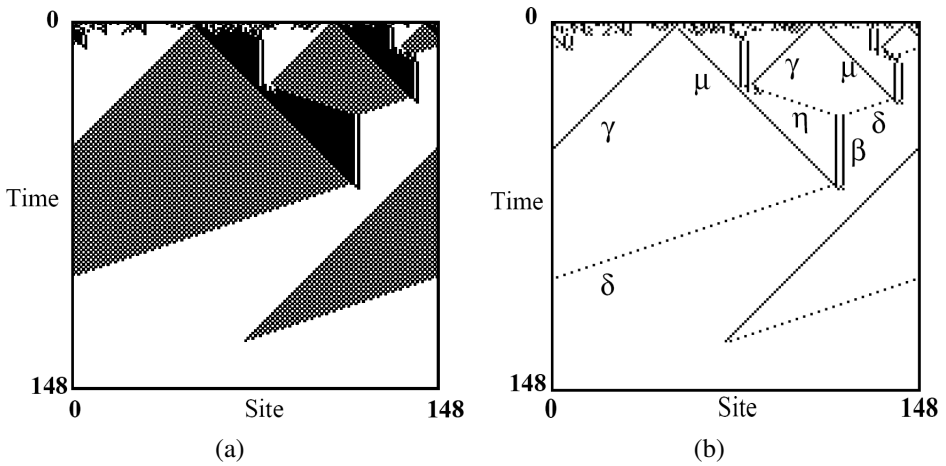


Figure 3.7: (a) Space-time diagram and (b) filtered version with particle identification [28].

a collision of particles α and β that creates a new particle γ. Besides the production of new particle(s), the possible result of interaction is an annihilation of colliding particles (result $\emptyset$), in which domains separated by particles merge into one.

The result of particle interaction is determined by the phase of the colliding particles. If a particle collision is phase-dependent, all possible results are written at the right side of the interaction, e.g., $\alpha + \beta \rightarrow \gamma \,|\, \beta + \omega$. The set $\mathbf{I}$ is a set containing all possible interactions of a given CA.

3.4.4 Particle Catalog

The *particle catalog* contains all domains, particles and particle interactions $\{\mathbf{\Lambda}, \mathbf{P}, \mathbf{I}\}$. It provides a descriptive tool to understand the processes underlying CA dynamics. It is a relevant abstraction of CA computation capable of predicting CA behavior without having to run the CA itself.

The quote stated in the title of this chapter applies very much to the particle-

based description of CA dynamics. The same space-state diagram looks different, if we use the optics of particles transferring and processing information.

Domains and particles usually cannot be identified from the very beginning of a CA computation. The *condensation time* t_C defines the first moment, when there are only domains and particles present in a space-time diagram and all pre-condensational fluctuations disappeared. The pre-condensational phase is not relevant for the whole dynamics of CA and thus can be omitted. Computational mechanics also omits the size of the particle (zero size), interaction time (appears instantly), etc.

The theory of evolution by cumulative natural selection is the only theory we know of that is in principle capable of explaining the existence of organized complexity.

Richard Dawkins (1961-)

4

Cellular Automata Evolution

In this chapter we introduce genetic algorithms and briefly present the selected work on evolutionary CA optimization for several computational tasks.

4.1 Introduction to Genetic Algorithms

Genetic algorithm (GA) [73] was originally introduced by Holland [52] as a stochastic optimization tool inspired by the Darwinian evolution. GA is an iterative process that intelligently searches through a space of possible solutions. GAs are popular and widely applied for many scientific or technological problems [46, 6, 2, 4].

GA operates on a population of chromosomes, which encode possible solutions for a given problem and are represented by vectors. Initial population is usually generated at random or using a heuristic. Every generation (evolutionary step) GA calculates the fitness of each chromosome, which reflects how well the chromosome solves a given problem. For instance, the fitness could be a fraction of correctly classified instances to the number of trials.

The best chromosomes in terms of fitness act as parents of new individuals in the next generation. Chromosomes can be selected to reproduce either by elite or roulette method. The elite method selects deterministically the certain number of fittest chromosomes. The roulette method selects chromosome with probability proportional to their fitness. In the roulette method selection probability could be adjusted by fitness renormalization.

Reproduction can be sexual or asexual. In the former case, crossover between two selected parent chromosomes can be either one-point (i.e., in chromosomes of length n, the child's first $p \leq n$ genes are from one parent and the last $n - p$ are from the other), or a probabilistic shuffle. Also, crossover could be conditional, hence it occurs with a probability $p_{cross} \leq 1$, otherwise new off-springs are exact copies of their parents.

The operation of mutation alters certain bits in newly created chromosomes. The bit alternation can be produced either by a full replacement with a newly generated bit, or for Integer or Real numbers a new bit can be generated by a perturbation. The number of bits the mutation changes depends on a mutation type: one-bit, two-bit, exchange and per-bit. Similarly to crossover the mutation could be conditional.

As presented in Figure 4.1 an evolutionary cycle consists of fitness calculation, selection, crossover and mutation, and it repeats until the stop-criterion, such as, the target (maximum) fitness, or alternatively a fixed (maximal) number of generations is reached. For more information about GA and evolutionary dynamics we advise the reader to refer to the many excellent textbooks that already cover these areas, e.g., [73, 29].

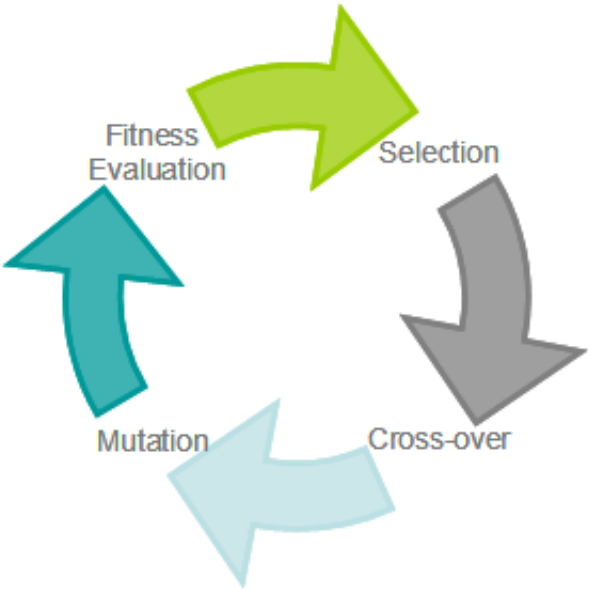


Figure 4.1: Genetic algorithm as a cyclic process.

4.2 Related Work

The idea of applying evolutionary optimization to CAs is not new. Several models and techniques have been proposed. Koza [60] applied genetic programming on CAs to generate simple random numbers. Sipper evolved non-uniform CAs for the iterated prisoner's dilemma and other problems [87]. Note that nonuniform CAs where each cell has a different transition rule make solving a computational task easier, however, the search space is much larger.

In his pioneering work Packard [79, 82] employed genetic algorithms to evolve uniform CAs to perform the density classification task. Note that the population of chromosomes, which encode candidate CAs as bit vectors, consists of the transition (look-up) tables' outputs using the standard ordering. In the density classification task a final configuration is expected to consist of all ones if the initial con-

figuration contains more ones, all zeros otherwise. Since Packard, density classification has been widely used as a benchmark task for one- and two-dimensional CAs.

Ganguly and colleagues [43] modelled an associative memory (pattern recognition) by so-called generalized multiple attractor cellular automata. They have developed constrained genetic algorithms, with the help of which the evolutionary process can be guided through a special class of additive or linear CA.

The largest contribution to the field of evolutionary cellular automata is, beyond a doubt, due to the former EvCA [54] (Evolutionary Cellular Automata) and CM (Computational Mechanics) groups at Santa Fe Institute led by M. Mitchell and J. Crutchfield. The EvCA and CM groups evolved one-dimensional binary CAs to perform the density classification [74, 31, 28] and synchronization [32, 53] tasks.

More recently, a two-dimensional variant of the density classification has been successfully tackled by several authors [21, 81, 71]. Other tasks for two-dimensional CAs include synchronization, spatial density niching, and rectangle image bounding [20].

An organized product of nature is that in which all the parts are mutually ends and means.

Immanuel Kant (1724-1804)

5

Leader Election in Cellular Automata

Starting in this chapter, we present our original contributions. Here we describe a GA optimization technique employed to search the space of possible CA transition tables, and the solutions found for the leader election problem in one- and two-dimensional binary cellular automata. Further, we discuss the performance results of our solutions, and analyze them with computational mechanics, transition table density, and perturbation stability.

In Chapter 2 we introduced the leader election problem in the context of distributed algorithms, showing that the anonymous, deterministic instance we are about to address is insolvable. We argue that leader election should not expect the components (cells) to be distinct, because that would require a central entity (a sequencer) assigning unique IDs, which would break the very essence of distributed control. Moreover, we opt for the lowest memory usage per processor (cell) possible—a constant (binary) memory. Non-constant memory distributed routines would need extra memory each time a new processor is added. Also,

we aim to incorporate the self-stabilization property. Self-stabilization promotes robustness to external interference and allows recovery from failures without any (or minimal) assumption about IC. We suggest that these attributes would greatly contribute to efficient and robust leader election in both artificial and natural distributed protocols.

It is surprising that the anonymous, deterministic, self-stabilizing, uniform leader election which we are about to investigate has not been addressed in distributed algorithms nor multi-agent systems research. Even though, as a result of Angluin's theorem, this instance of the problem is not solvable to an accuracy of 100%, we decided to explore leader election in its simplest form. Our CA architecture is visibly simpler than the distributed algorithms presented in Section 2.1. All state-of-the-art self-stabilizing distributed models assume additional aids, such as a daemon or oracle, extra memory, randomization, and more advanced computational or communication capabilities. Also, our target memory of $O(1)$ (binary state) and $O(N)$ time complexity is clearly superior to any of known distributed algorithms.

CAs are massively parallel and spatially extended pattern-forming systems. Our goal is to use machine learning procedures, such as a GA stochastic search, to automatically design CAs that harness parallelism and form propagating patterns that transcend individual cells. Since cells operate just with a binary state—they cannot store any additional data. Further, cells do not send information through explicit messages but rather transfer information collectively using recurring state patterns spreading through several cells. This is the main difference between our approach and "classical" distributed algorithms.

We assert that our CA is the simplest resource-efficient model capable of

leader election. As we mentioned, this instance of the problem is not fully solvable. We show that for any CA there exist configurations that prevent successful leader election as a result of configuration symmetry and loose-coupling of active cells, discussed in Chapter 6. This situation occurs, however, very sporadically. We present several successful CAs performing leader election. The best one-dimensional CA, called the improved strategy of mirror particles, reaches performance of 99.7% for uniform and 94.5% for density-uniform distributions. The success rate of the best two-dimensional CA is 98.7% for uniform and 80% for density-uniform distributions.

5.1 Leader Election as CA Computational Task

Before we take a closer look at the GA model of CA evolutions we need to formally define a CA computational task [53]. The computational task T expresses a functional relation between an input, *question set* $C_i \subseteq \Omega = \Sigma^N$, and the desired output of CA, *Answer set* $A_i \subseteq \Omega$:

$$T : \Sigma^N \rightarrow \cup_i A_i$$

$$\mathbf{s_0} \in C_i \Rightarrow T(\mathbf{s_0}) \in A_i$$

There always exists an answer, i.e., configuration(s) of one particular answer set A_i, for a question (IC) from C_i ($C_i \cap C_j = \emptyset$, $i \neq j$ and $\cup_{i=1}^{n} C_i = \Omega$). Thus, a computational task is the set of question-answer pairs. If an answer set contains more than one configuration, sometimes it is required that the CA must cycle through all of them.

Now we define the leader election computational task T reflecting the relationship between an input (IC) and desired output (final configuration(s)) as

$$T : \Sigma^N \rightarrow \{\mathbf{s} \in \Sigma^N |_{\#_1} \mathbf{s} = 1, \Phi(\mathbf{s}) = \mathbf{s}\} \tag{5.1}$$

The function $\#_1$ denotes the number of ones in a configuration (word). The goal is to transform each IC to a fixed point configuration containing exactly one cell in the state 1 and the rest in the state 0.

The leader election task T, as just formulated, differs slightly from the general definition of a computational task. Specifically, a CA is not required to cycle through all configurations in the answer set (configurations with exactly one active cells), but it is supposed to settle down to one of the legal stable-point configurations.

5.2 Model of Cellular Automata Evolution

To search the space of possible CAs for leader election for both, one- and two-dimensional configurations, we employed a standard GA [52, 73] introduced in Section 4.1, inspired by Darwin's classical theory of natural selection.

Each chromosome represents a transition function ϕ shared globally by all cells. More specifically, a chromosome is a linear vector of length $|\Sigma|^n$ consisting of the output bits of a transition table, where Σ is the state set and n is the number of cells in a neighborhood. Since we deal exclusively with binary CA, $|\Sigma| = 2$ and a transition table (chromosome) consists of 2^n rows. The size of the chromosome space in which the GA searches is, therefore, 2^{2^n}. Since one or two-dimensional topology as well neighborhood size (shape) is implied we will use the terms CA,

transition table or function, and chromosome interchangeably.

Now we present the common GA settings for both one- and two-dimensional instances. The specifics will be discussed separately in Sections 5.3 and 5.4. The GA uses an elite selection, hence we choose the E fittest chromosomes in each generation and copy them directly to the next without any change. The remaining $M-E$ chromosomes, where M is population size, are discarded. $M-E$ offsprings are then generated by crossover of randomly chosen elitist chromosomes. The crossover is one-point and not conditional. All $M-E$ chromosomes are subject to mutation with probability p_{mut}. If mutation occurs, each bit is flipped with probability p_m, so the average number of inverted bits is $k^{2r+1}p_m$. Fitness renormalization is absent, since chromosomes are selected solely according to their rank, not a numerical value of their fitness. The stop criterion is the maximal number of generations G.

5.2.1 Fitness and Performance

By $P^I_N(\phi)$ we denote the performance of CA with transition rule ϕ and lattice size N on the leader election problem. Performance is calculated as the fraction of correctly computed ICs to the total number of randomly drawn test ICs I. Following Definition (Equation) 5.1 we assign the tested CA a point only if its final configuration is a fixed point with exactly one active cell. Additionally, we impose a maximal time (the number of steps) t_{MAX} allowed for CA computation. Thus, if a CA does not reach a correct answer within t_{MAX} steps, the output is considered incorrect. Due to the huge number of possible configurations 2^N, performance $P^I_N(\phi)$ with transition rule ϕ and lattice size N reflects only a reasonable

number of randomly generated ICs I. Therefore, the calculated performance is an approximation of "real performance". Fitness $F_N^I(\phi)$ is defined the same way as performance but usually for a smaller number of test ICs to balance the execution time and accuracy.

To generate test ICs and chromosomes in the initial population we use two standard probability distributions: uniform and density-uniform. *Uniform* distribution chooses each bit independently with equal probability for all states from Σ. On the other hand, in *density-uniform* distribution each density, the fraction of ones in generated strings has uniform probability. This means that density is uniform over $\lambda \in \langle 0, 1\rangle$ or over $\rho \in \langle 0, 1\rangle$, where λ is the fraction of ones in ϕ's output bits and ρ is the fraction of ones in the IC.

5.3 Leader Election in One-Dimensional Cellular Automata

Let us recall that the transition table ϕ of a one-dimensional binary CA with radius r consists of 2^{2r+1} rows. As stated in Section 5.2, each chromosome represents a transition table shared globally by all cells. To generate initial chromosomes as well as test ICs we use density-uniform distributions. To impose a linear execution time we allow $t_{MAX} = 2N$ steps for CA computation. We calculated fitness $F_N^{I_1}(\phi)$ for the training number of cells $N = 149$ and $I_1 = 100$ ICs. We generated the test ICs with density-uniform distribution, because, as we will show later, this distribution type prefers ICs with lower densities, which are generally more difficult. After each evolution we determined performance $P_N^{I_2}(\phi)$ of selected chromosomes more accurately for both density-uniform and uniform

distributions, $N \in \{149, 599, 999\}$ and $I_2 = 10^4$.

The GA settings employed for one-dimensional leader election are as follows: population size $M = 100$, number of test ICs $I = 100$, number of elite chromosomes $E = 20$, crossover probability $p_{cross} = 1$, mutation probability $p_{mut} = 1$, per-bit mutation probability $p_m = 0.0016$, and maximum generation $G = 100$.

Initially we aimed to find CA for the leader election task with the minimal radius. Since there are just 256 CAs for radius $r = 1$ we could directly calculate their performance without a help of GA. Not surprisingly the results were just marginal. The CA evolutions with $r = 2$ resulted in a noticeably higher performance reaching maximum of 0.586, however, the performance is not very stable and decreases rapidly with respect to N.

In following sections we present the results of evolutions, various one-dimensional strategies, their performance, and computational mechanics and transition table density analysis.

5.3.1 Results of Evolving Cellular Automata

The minimal radius that allows CA to solve the leader election task with satisfactory results is $r = 3$. Interestingly, EvCA group, which performed CA evolutions for the density classification and synchronization tasks came to the same conclusion, thus we could speculate that radius $r = 3$ could be a general threshold for a complex computation. Note that the size of the chromosome space in which the GA searches is 2^{128}, far too large for exhaustive enumeration and performance evaluation.

We performed two sets of evolutionary runs. The first set with 98 evolutions

used randomly generated initial populations, the second one containing 104 evolutions started with the identical initial population consisting of several mutated copies of the first chromosome from the first evolutionary set that reached fitness > 0.9.

The first evolutionary set targeted an exploration of the fitness landscape and identification of underlying evolutionary strategies. Only 20% of evolutions reached particle-based strategies, identified by fitness > 0.8, and just a small fraction (7%) reached a high-performing region of > 0.9. Leader election strategies had to overcome the major issues of CA architecture, the local scope of individual cells $r << N$ and the lack of memory. The GAs pushed the transition tables towards efficient and reliable ways of global information exchange between distinct regions. The evolutionary dynamics of leader election is not smooth and shows some very dramatic leaps, or innovations. A sudden jump from the fitness 0.4 to 0.8 is the most crucial innovation from the computational perspective, since it separates localistic from global particle-based strategies.

The second evolutionary set aimed to find the best-performing CA for leader election. The most successful strategy of mirror particles reached a performance of 0.944 for density-uniform and 0.992 for uniform distribution and $N = 149$. We further improved these results in additional evolutions that started with an initial population consisting of randomly perturbed chromosomes which use the strategy of mirror particles.

5.3.2 Analysis

The evolutionary search uncovered various strategies which we further examined and analyzed. They can be roughly categorized according to their characteristic computational aspects into 5 representative types: the strategy of mandatory function, density reduction, the divide and eliminate strategy, the first particle-based strategy, and the (improved) strategy of mirror particles. This categorization grasps the most prevalent computational approaches, but many strategies (transition functions) that occurred during evolutions defy this high-level characterization and are often border-line.

The first three strategies are localistic and often exploit the statistical properties of generated ICs, such as mean and variance. There is no long-distance information exchange, so a leader is elected only on short subsequences of cells. That results in a sharp decrease of performance with respect to N. The last two strategies leverage a particle collision-based model of the computational mechanics, allowing a coordination of cells at the global scale.

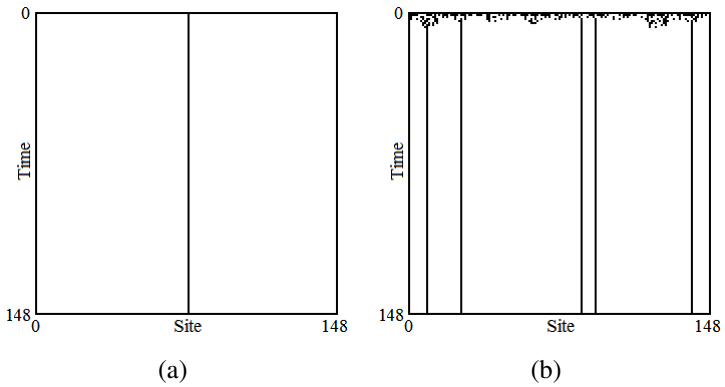


Figure 5.1: The strategy of mandatory transition function - space-time diagrams of (a) successful and (b) unsuccessful leader election.

5.3.2.1 Strategy of Mandatory Function

Each transition rule with non-zero chance to solve the leader election problem must include $2r + 2$ mandatory transitions (see Section 6.2.1) securing the final leader configuration to be a stable point (e.g., $\phi(0001000) = 1$ and $\phi(1000000) = 0$). It is not surprising that the first strategy with a non-zero fitness implements mandatory transitions by setting $2r + 2$ bits in the chromosome. Naturally, this strategy is usable only on ICs with exactly one active cell (respectively on a few more cases). Performance (Figure 5.1) roughly approaches the probability that IC generated as either density-uniform or uniform has exactly one active cell.

N	Uniform	Density-Uniform
149	0.002	0.018
599	0.000	0.003
999	0.000	0.002

Table 5.1: Performance of the strategy of mandatory transition function using uniform and density-uniform distributions.

5.3.2.2 Density Reduction

The density reduction strategy with fitness 0.2 is a greedy and localistic strategy that produces a leader by reducing the number of active cells (density) to minimum. This naive approach would correspond to a transition rule designed by hand, rather than any automatized optimization technique. All bits in the chromosome, that encode this transition table, contain zeros at the position of almost all bits (except $\phi(0001000)$ and $\phi(1111111)$), i.e., the transition table density $\lambda \approx 0$. Correctness of this approach is determined by the number of occurrences of exactly

one string 0001000 or continuous parts of 1s with suitable length for shrinking. The density reduction strategy (Figure 5.2) performs significantly better on higher densities of $0.45 - 0.99$. Lower densities lead to higher frequencies of more than one 0001000 string, thus, more than one active cell in a stable point configuration. Since density-uniform distributions prefer ICs with lower densities, which are generally difficult to solve, performance for uniform distribution of ICs (with density clustered around 0.5) is higher.

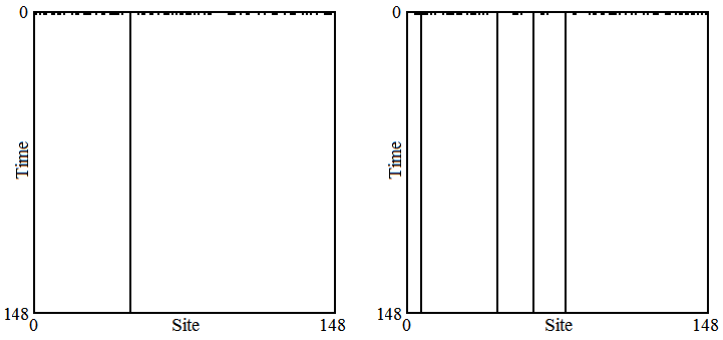


Figure 5.2: The density reduction strategy - space-time diagrams of (a) successful and (b) unsuccessful leader election.

N	Uniform	Density-Uniform
149	0.330	0.214
599	0.020	0.024
999	0.001	0.008

Table 5.2: Performance of the density reduction strategy using uniform and density-uniform distributions.

5.3.2.3 Divide and Eliminate

The divide and eliminate strategy (Figure 5.3) is the last localistic and non-particle strategy in the evolutionary process. The core mechanism of this strategy consists of two operations, we call them division and elimination. The division operation splits the continuous regions of active or inactive cells into configuration sequences $(10)^+1$ and $(100)^+1$ where the distance between active cells is 2 or 3 respectively. Consequently, the elimination operates on these continuous sequences and reduces them from one or both sides. There is a sign of emerging domains and particles, however, this strategy still cannot be considered particle-based. The division and elimination processes run simultaneously producing non-trivial dynamics. Fitness of this strategy is about 0.4, performance is 0.32 for density-uniform and 0.38 for uniform distribution (Figure 5.3).

Since this strategy is still localistic it lacks global information exchange, and so it tackles the leader election problem only on limited subsequences of cells. Hence, performance decreases rapidly with respect to N.

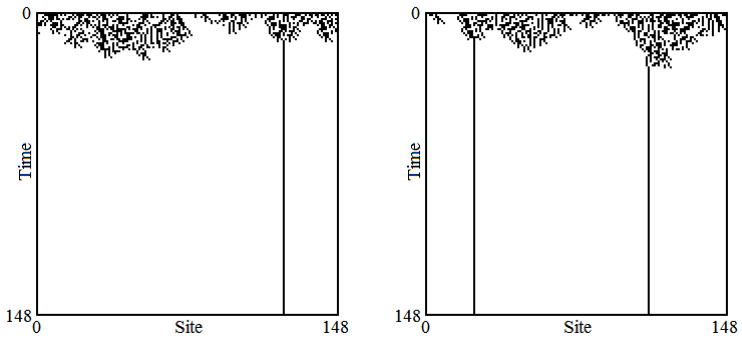


Figure 5.3: The divide and eliminate strategy - space-time diagrams of (a) successful and (b) unsuccessful leader election.

N	Uniform	Density-Uniform
149	0.383	0.321
599	0.100	0.068
999	0.018	0.011

Table 5.3: Performance of the divide and eliminate strategy using uniform and density-uniform distributions.

5.3.2.4 First Particle-Based Strategy

At this point of the evolutionary process, a first particle-based strategy was found. To describe the dynamics of this strategy we use the framework of computational mechanics. We identified several domains, particles, and their interactions responsible for leader election (Figure 5.4). Let us recall that regular domains are homogeneous space-time regions that are temporally invariant, forming a background for computation (Section 3.4). Particles are propagating objects formed at a boundary between domains, which transfer information on distances, and finally particle interactions carry information processing.

From the computational mechanics perspective a leader itself can be considered as a particle separating two (white) domains $\Lambda^0 = 0^*$. We denote this particle ω. The strategy produces a leader ω by an interaction of particles α and β. The particle with most occurrences in space-time diagrams and a crucial function in a collision-based computation is β. The particle β is responsible for leader election by interaction $\alpha + \beta \rightarrow \omega$ and sweeping of intermediate particles in configuration by interactions $\omega + \beta \rightarrow \beta, \beta + \gamma \rightarrow \delta$, and $\delta + \beta \rightarrow \beta$. Each β interaction generates either a leader particle ω or β itself. The only exception is an interaction with γ producing δ, that is, however, eliminated rapidly by interaction $\delta + \beta \rightarrow \beta$. Table

Domains $\mathbf{\Lambda}$	
Label	Regular language
Λ^0	0^*
Λ^1	$(1001000101)^*$

Particles $\mathbf{P}$		
Label	Boundary	*Velocity*
α	$\Lambda^0\Lambda^1$	3
β	$\Lambda^1\Lambda^0$	1
γ	$\Lambda^0\Lambda^1$	$-1/3$
δ	$\Lambda^1\Lambda^1$	3
ω	$\Lambda^0\Lambda^0$	0

Interactions $\mathbf{I}$	
$\alpha + \beta \rightarrow \omega$	$\omega + \gamma \rightarrow \gamma + \beta + \alpha$
$\omega + \beta \rightarrow \beta$	$\delta + \beta \rightarrow \beta$
$\beta + \gamma \rightarrow \delta \mid \alpha + \beta$	

Table 5.4: The particle catalog of the first particle-based strategy.

5.4 presents the full particle catalog of this strategy.

The main drawback is the less-than-ideal directions of the particles, since most of them have positive velocities and the only particle with a negative velocity, γ, is very slow (its velocity is $-1/3$). In addition, the particles α and β separated by a domain Λ^1 do not have enough time to move through the whole configuration before they collide. As a matter of fact, a final configuration as shown in Figure 5.4(b) might end up having more than one leader cell (a particle ω).

Fitness and performance is, compared to any localistic strategies, fairly acceptable. Fitness reaches values around 0.8 and performance is about 0.725 for density-uniform and 0.759 for uniform distribution as shown in Table 5.5. Performance decreases very slowly with respect to N, and as opposed to following, otherwise best-performing, strategies is fairly universal, i.e., it does not show signs of measurable N-modulo restrictions (discussed in Section 5.3.2.8).

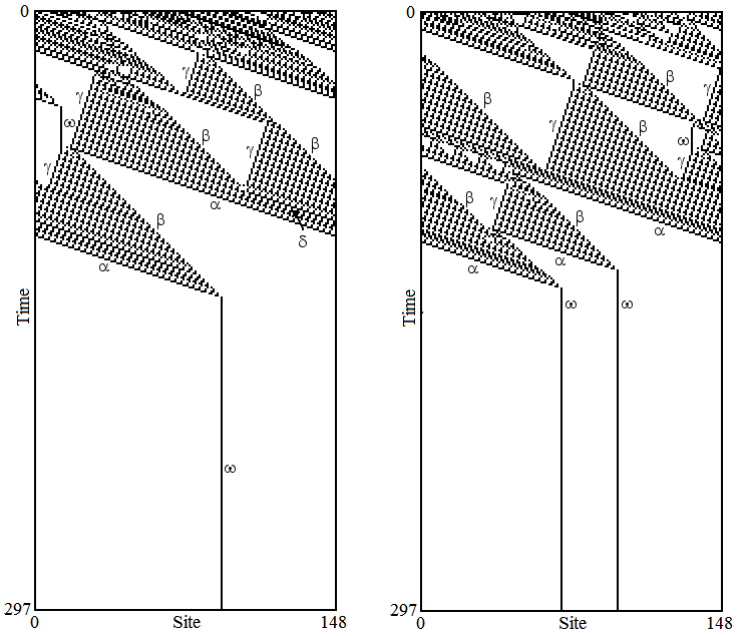


Figure 5.4: The first particle-based strategy - space-time diagrams of (a) successful and (b) unsuccessful leader election.

N	Uniform	Density-Uniform
149	0.759	0.725
599	0.715	0.680
999	0.677	0.659

Table 5.5: Performance of the first particle-based strategy using uniform and density-uniform distributions.

5.3.2.5 Strategy of Mirror Particles

The strategy of mirror particles is characterized by the occurrence of pair-like particles moving with the same speed but in opposite directions. We call these *mirror particles*. These include particle pairs α and β, γ and δ, ε and ζ. They all lie at the border of domain 0^* and the zig-zag domain $(01)^*$. Table 5.6 presents

the full particle catalog of this strategy. As opposed to previous strategies, the strategy of mirror particles resolved the problem of two or more active cells in a final configuration by a two-phase sweep consisting of $\alpha+\beta$ and $\gamma+\delta$ interactions as shown in Figure 5.4(b). More precisely, the main rule responsible for leader (particle ω) election is the interaction $\alpha + \beta \rightarrow \gamma + \delta$ followed by $\gamma + \delta \rightarrow \omega$ visible in Figure 5.6(a). A collision of α and β indicates that the final stage of leader election could start. Particles γ and δ are emitted to verify if there are any particles left. They turn around the whole configuration with high opposite velocities. In case they do not collide on their routes, they meet in the middle, and finally produce a global leader ω.

Domains $\mathbf{\Lambda}$	
Label	Regular language
Λ^0	0^*
Λ^1	$(01)^*$

Particles $\mathbf{P}$		
Label	Boundary	Velocity
α	$\Lambda^1\Lambda^0$	1
β	$\Lambda^0\Lambda^1$	-1
γ	$\Lambda^0\Lambda^1$	3
δ	$\Lambda^1\Lambda^0$	-3
ε	$\Lambda^0\Lambda^1$	3
ζ	$\Lambda^1\Lambda^0$	-3
ω	$\Lambda^0\Lambda^0$	0

Interactions $\mathbf{I}$	
$\alpha+\beta \rightarrow \gamma+\delta \,\vert\, \emptyset(\Lambda^1)$	$\alpha+\omega \rightarrow \alpha$
$\gamma+\delta \rightarrow \omega \,\vert\, \alpha+\beta$	$\beta+\omega \rightarrow \beta$
$\alpha+\gamma \rightarrow \alpha+\beta$	$\varepsilon \rightarrow \varepsilon+\omega$
$\beta+\delta \rightarrow \omega$	$\zeta \rightarrow \zeta+\alpha+\beta$
	$\varepsilon+\zeta \rightarrow \alpha+\beta+\varepsilon$

Table 5.6: The particle catalog of the strategy of mirror-particles.

Further particles detected in space-time diagrams are long-periodic particles ε and ζ. Their unique attribute is a spontaneous emission of particles during each period. A particle ε generates a leader particle ω, whereas ζ generates mirror particles α and β. Also, the particles α and β clean ω particles on their routes, which in case of the interaction $\alpha + \omega \rightarrow \alpha$ causes the phase shift of α.

The number of particles with positive velocities equals those with negative ones. All particles are fairly fast. Their absolute velocities range from 1 to 3 excluding a non-moving particle ω. Also, the differences of colliding particles' velocities are high. That has a positive impact on the overall performance.

Fitness of this evolutionary strategy is 0.99; performance reaches 0.944 and 0.992 for $N = 149$ depending on the IC distribution type, and stays high for much

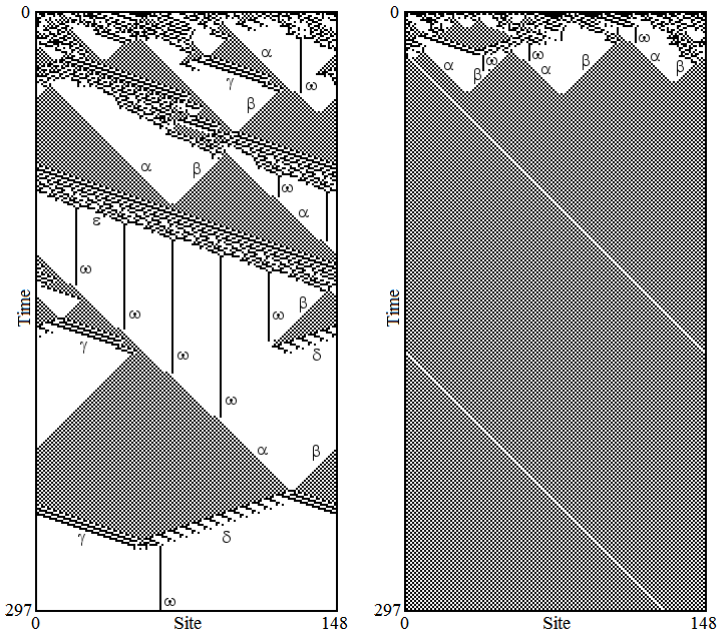


Figure 5.5: The strategy of mirror particles - space-time diagrams of (a) successful and (b) unsuccessful leader election.

N	Uniform	Density-Uniform
149	0.992	0.944
599	0.954	0.932
999	0.009	0.011
1001	0.989	0.971
1301	0.987	0.972

Table 5.7: Performance of the strategy of mirror particles using uniform and density-uniform distributions.

larger N, such as, $N = 599$ with performance 0.932 and 0.954 respectively (Table 5.7). For $N = 999$ cells performance is extremely low; only about 0.01, yet for even larger N, such as, 1001 and 1301, it rebounces to 0.97 for density-uniform and 0.99 for uniform distribution. An explanation of this anomaly is the principal restriction on the number of cells N. The strategy of mirror particles produces satisfactory results for the leader election problem only for $N \equiv 5 \bmod 6$, $N \geq 23$. Our experiments show that the sequence of solvable number of cells is 15, 23, 29, etc.

The fundamental leader-electing interactions $\alpha+\beta \rightarrow \gamma+\delta$ and $\gamma+\delta \rightarrow \omega$ are phase dependent, i.e., their outcomes depend on the phase of colliding particles. Since α and β are two-phase particles, their interactions have two possible results $\alpha+\beta \rightarrow \gamma+\delta \,|\, \emptyset$. Similarly, γ and δ are three-phase particles, hence the interactions $\gamma+\delta \rightarrow \omega \,|\, \alpha+\beta \,|\, \alpha+\beta$ are possible. Note that the result of the second and the third interaction of $\gamma + \delta$ is the same, but the interaction process is slightly different.

In the last phase of leader election, when $\gamma + \delta$ and partially $\alpha + \beta$ have to turn around the whole configuration, the number of cells N becomes essential in determining their phases in a moment of interaction. We identified typical results of $\alpha + \beta$ and $\gamma + \delta$ interactions with respect to the modulo classes of N shown

in Table 5.8. A goal to produce a single leader ω in a final configuration can be achieved by the interaction $\gamma + \delta \rightarrow \omega$ occurring for $N \equiv x \bmod 6$, $x \in \{2, 5\}$. Further, a pair of particles γ and δ needed for this interaction are produced from α and β only for $N \equiv x \bmod 6$, $x \in \{1, 3, 5\}$. As a result, the only acceptable number of cells allowing leader election (with satisfactory result) is 5 mod 6. Further, we found by experimentation the constraint ($N \geq 23$). CA dynamics for the modulo classes 0, 2 and 4 produce a global zig-zag domain Λ^1 (Figures 5.6(b), 5.6(d), 5.6(f)), the remainder 5 leads to a stable point of the leader particle ω (Figure 5.6(a)), and the remainders 1 and 3 result in a cyclic behavior $\alpha + \beta \rightarrow \gamma + \delta \rightarrow \alpha + \beta \rightarrow \ldots$ (Figures 5.6(c), 5.6(e)). Let us recall that the predecessor of this strategy, called the first-particle based strategy, is not $N \equiv 5 \bmod 6$ restricted and for 999 cells reaches performance of about 0.7.

$N\ mod\ 6$	$\alpha + \beta \rightarrow$	$\gamma + \delta \rightarrow$
0	$\emptyset$	$\alpha + \beta$
1	$\gamma + \delta$	$\alpha + \beta*$
2	$\emptyset$	ω
3	$\gamma + \delta$	$\alpha + \beta$
4	$\emptyset$	$\alpha + \beta*$
5	$\gamma + \delta$	ω

Table 5.8: The strategy of mirror particles - typical results of the crucial leader-electing interactions with respect to N modulo 6 classes.

5.3.2.6 Improved Strategy of Mirror Particles

The improved strategy of mirror particles is a successor of the previous strategy. We evolved this strategy from an initial population consisting of randomly perturbed chromosomes representing the strategy of mirror particles. The basic

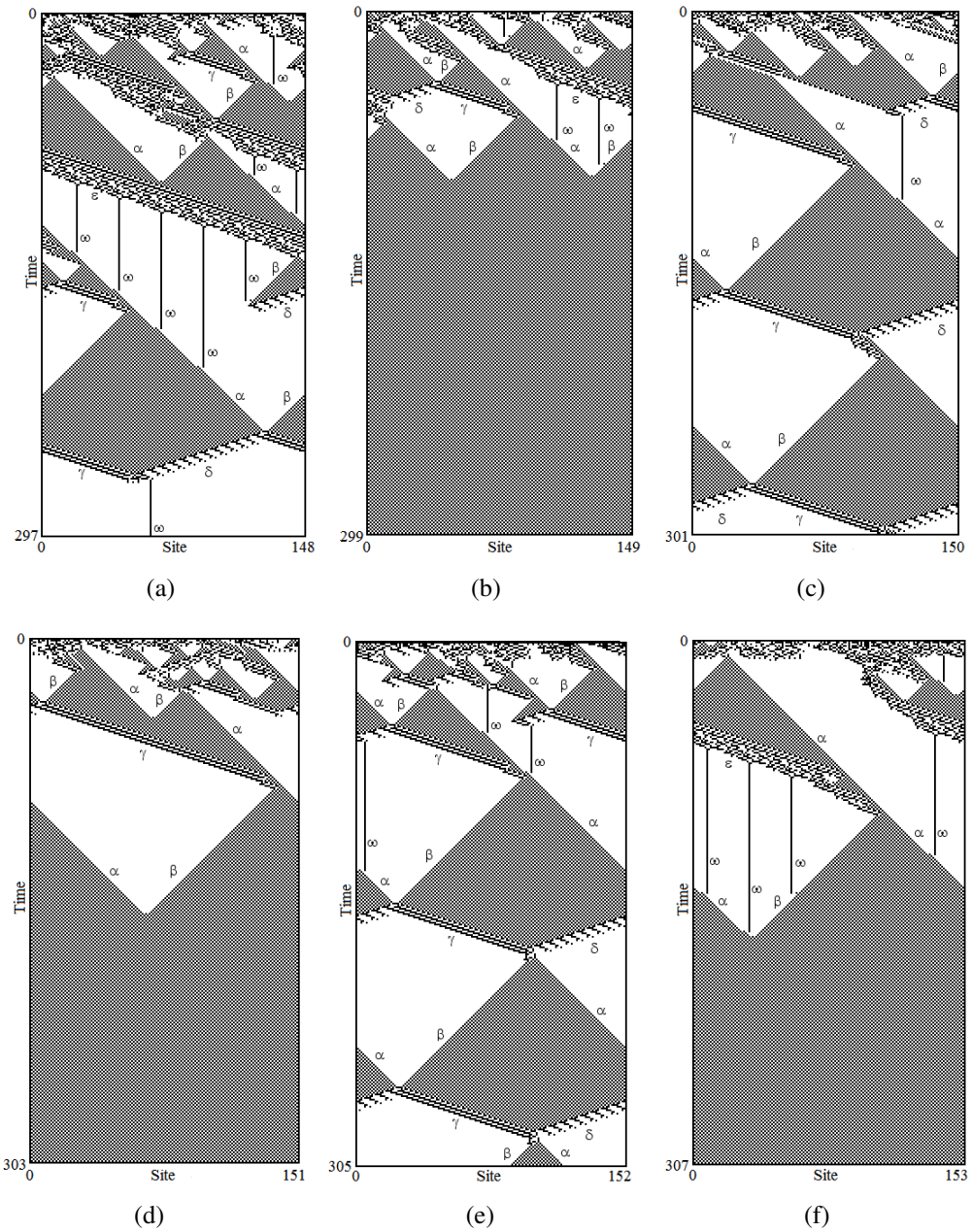


Figure 5.6: The strategy of mirror particles - space-time diagrams annotated with particles showing typical behavior for all N modulo 6 classes. The number of cells N goes from 149 (top left) to 154 (bottom right).

particle mechanism stays the same, i.e., a leader is elected by a chain of interactions $\alpha+\beta \rightarrow \gamma+\delta \rightarrow \omega$ as shown in Figure 5.7. Only minor changes are present, e.g., emission of particle ε disappeared.

Table 5.9 shows that performance has improved and for uniform distribution reaches 0.997. Nevertheless, higher performance is offset by an additional N-modulo restriction. Namely, this strategy is usable only for the $N \equiv 5$ modulo 12 number of cells.

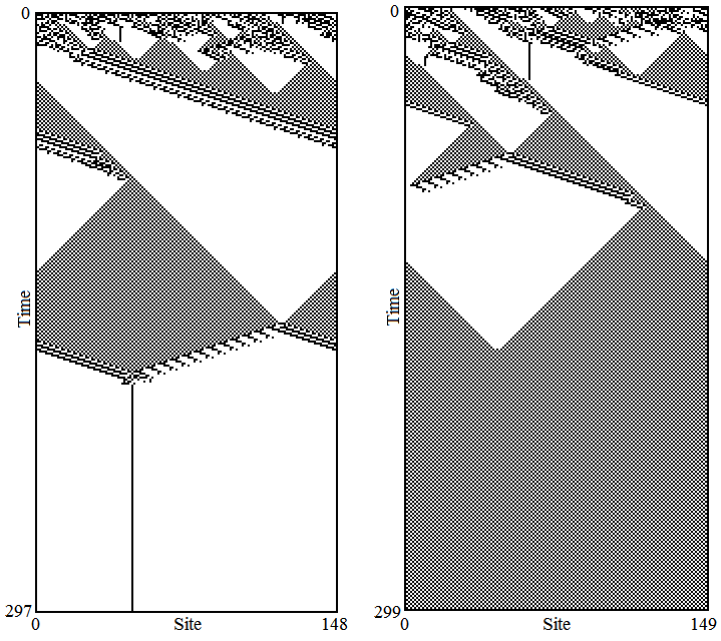


Figure 5.7: The improved strategy of mirror particles - space-time diagrams of (a) successful and (b) unsuccessful leader election.

N	Uniform	Density-Uniform
149	0.997	0.945
593	0.997	0.973
599	0.002	0.003
999	0.009	0.011
1001	0.996	0.979
1301	0.995	0.979

Table 5.9: Performance of the improved strategy of mirror particles using uniform and density-uniform distributions.

5.3.2.7 Transition Table Density

The transition table density (λ) is the fraction of ones in transition table's outputs. Figure 5.8 shows the fitness-λ correlation of the best chromosomes of each population for all evolutionary runs we performed.

Unlike the density classification and synchronization tasks tackled by the EvCA group [74, 32], where λ for the best-performing CAs is spread throughtout the whole spectrum, the leader election task requires a very narrow λ range of 0.46. Note that the critical region avoids 0.5-density, which could result in a symmetric transition table and make leader election more difficult.

5.3.2.8 Overall N-Modulo Dependence

Because of the N-modulo restrictions we found for the best-performing CAs, we asked whether there is a correlation between N-modulo dependence and performance. More precisely, we investigated the relation between performance for the training number of cells, 149, and universality of CA performance measured by the number of N modulo 6 classes ($N = 149, \ldots, 154$) that produce similar performance as the $N = 149$ case.

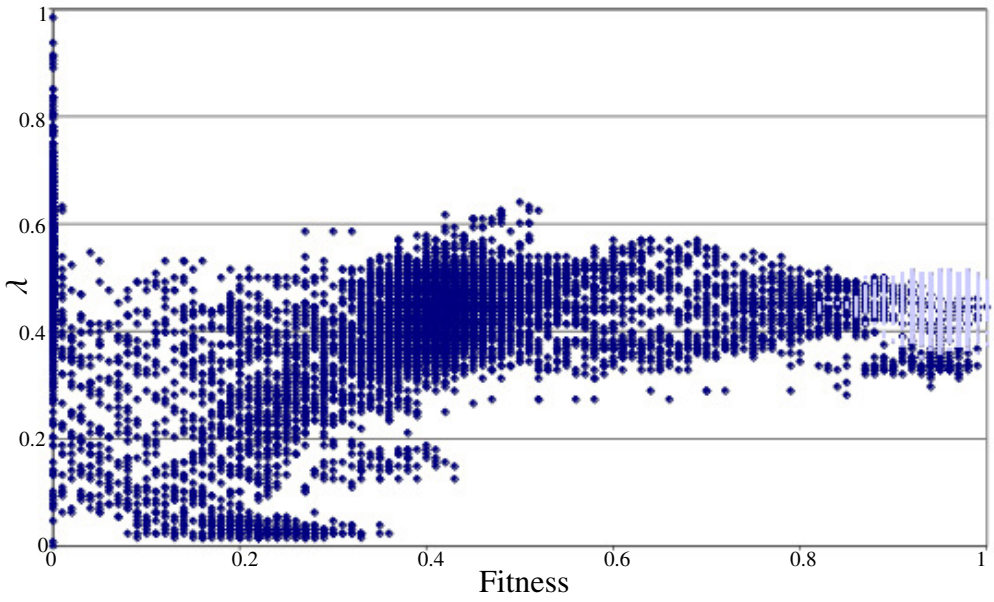


Figure 5.8: Relation between the fitness and λ (transition table density) for one-dimensional leader election showing all chromosomes from both evolutionary sets. Note a critical high-performing region around $\lambda = 0.46$.

As shown in Figure 5.10, poor strategies rank around the number 6, meaning their performance is fairly uniform across different numbers of cells. As performance increases, CA become more and more specialized with respect to modularity of N. Our results suggest that the N-modulo specialization has a positive impact on performance.

Recall that the best strategies achieve high performance by emission of particles moving through the whole configuration in opposite directions. The number of cells N is therefore important in determining their phases at the moment of eventual collision (assuming the particles do not colide with other particles before making it all the way around the 1d ring). Also, particles with higher number of phases allow more options for information processing and encoding, which

benefits performance. As reported in [53], the number of different interaction outcomes is directly proportional to a disparity of velocities of colliding particles. Thus, particles with high opposite velocities have generally a higher probability of more interaction results. As a matter of fact, the faster the particles, the faster information can spread, and the sooner the final leader-electing sweeping can proceed. However, fast interactions have increasingly more alternative results, which implies stronger N-modulo specialization.

Furthermore, we evaluated CAs from all evolutions to identify the one with the most universal performance: the CA with least N-modulo restriction. Our search for a N-modulo universal strategy with high performance was partially successful. The strategy with best cumulative performance on $N = 149, \ldots, 154$

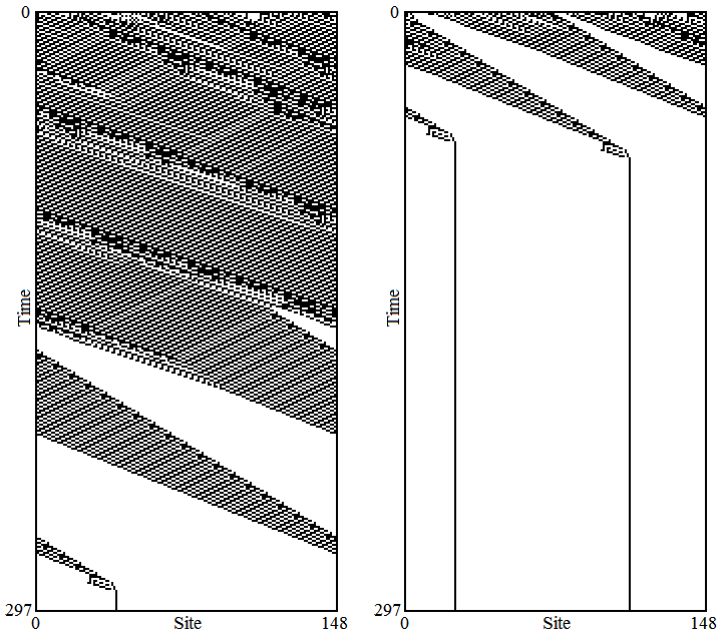


Figure 5.9: The N-modulo universal strategy - space-time diagrams of (a) successful and (b) unsuccessful leader election.

presented in following diagrams (Figure 5.9) does not scale well and fails for larger N (Table 5.10). These findings suggest that for one-dimensional leader election there is a tradeoff between performance and scalability on one side and N-modulo universality on the other.

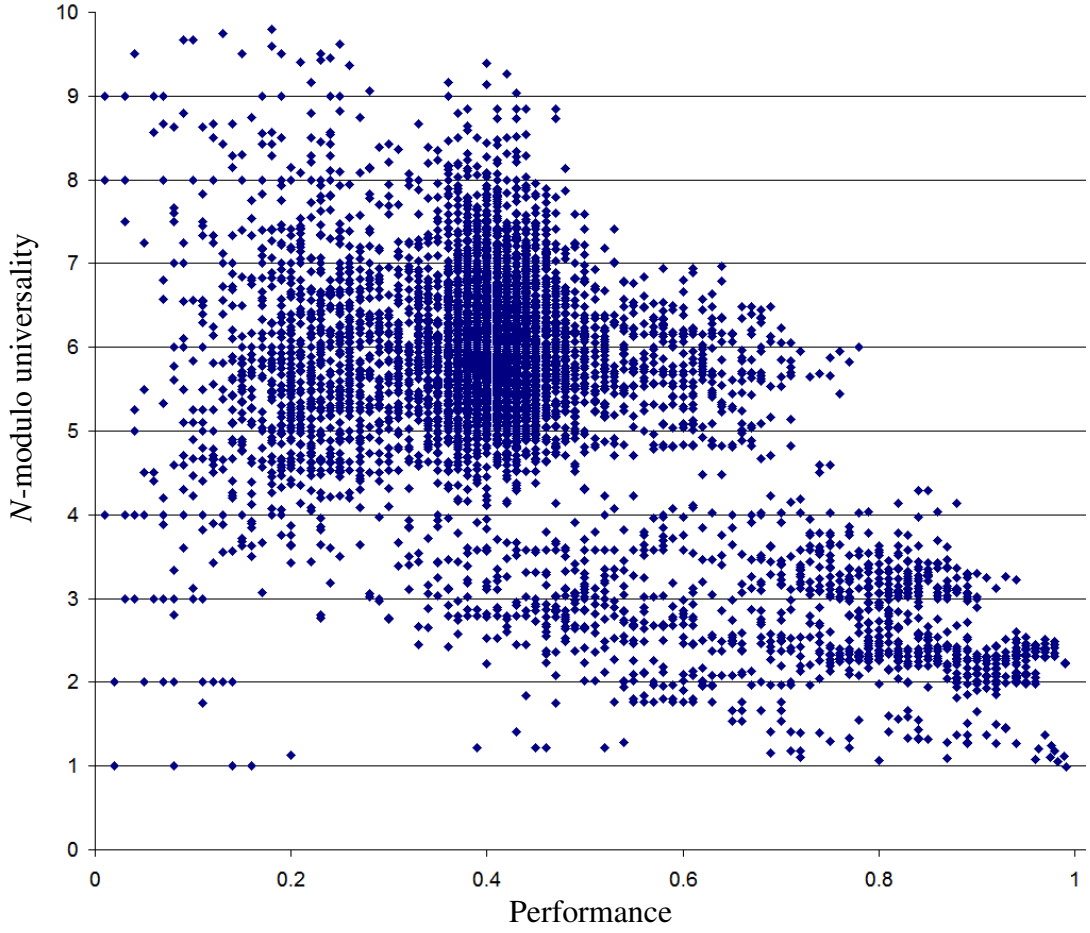


Figure 5.10: Relation between performance $P^{100}_{unif\,149}(\phi)$ and the ratio of cumulative performance for all N modulo 6 classes (representatives): $\frac{\sum_{N=149}^{154} P^{100}_{unif\,N}(\phi)}{P^{100}_{unif\,149}(\phi)}$. N-modulo universality (y-axis) represents the number of modulo classes (out of 6 defined) that produce similar performance as the training number of cells $N = 149$.

N	Uniform	Density-Uniform
149	0.821	0.625
599	0.135	0.115
999	0.019	0.010

Table 5.10: Performance of the N-modulo universal strategy using uniform and density-uniform distributions.

5.4 Leader Election in Two-Dimensional Cellular Automata

As presented in Section 3.1.2, we tackle leader election in two-dimensional CA with square lattices, and Moore neighborhoods, i.e., square neighborhoods with $r = 1$ and 9 cells. Since the transition table ϕ consists of 2^9 rows and the space of possible solutions 2^{2^9} is even larger than in the one-dimensional case, we again search the solution space by GA. To generate initial chromosomes as well as test ICs we use a uniform random distribution. To limit execution time we set $t_{MAX} = 300$ steps of CA computation. We calculated fitness $F_N^{I_1}(\phi)$ for the training number of cells $N = 19^2$. Note that we opted for $N = 19^2$, since it is a standardly used benchmark lattice size [71, 20], large enough to require a global coordination of cells.

Because the evolution of two-dimensional leader election is generally more difficult than the one-dimensional case, we improved several key attributes of the GA. We increased the population size M from 100 to 400, the number of test ICs I from 100 to 500, as well as the number of generations from 100 to 200. The selection is elitist with elite size (the number of chromosomes from one generation which live and reproduce in the next) of 80 (20% of the total population). We also

substituted a simple one-point crossover with more advanced shuffle crossover, where each bit of the offspring is copied from either of the parents with even probability. The remaining settings, such as per-bit mutation probability, are the same as before.

After each evolution we determined performance $P_N^{I_2}(\phi)$ of selected chromosomes more accurately than fitness for both density-uniform and uniform distributions, $1 \leq N \leq 40$ and $I_2 = 10^4$ test ICs.

5.4.1 Results of Evolving Cellular Automata

Compared to the one-dimensional case, evolution of two-dimensional CAs with Moore neighborhood and size $N = 19^2$ take substantially longer due to the larger neighborhood and lattice size. Further, synchronous updates in one-dimensional CA benefited from a finite-state transducer representation of the transition table, a technique impossible in two-dimensional CA. Even utilizing fixed-point detection, which halts simulations when a fixed point is reached before the max time, the execution time of a single evolution is around 40 days on a single core. That limited the number of evolutions we could perform.

For two-dimensional leader election we executed four sets of evolutionary runs: the leader election with $N = 19^2$ (LE), the density minimization task (DM), and two leader election tasks starting from an initial population generated from the last best chromosomes of DM, one with with $N = 19^2$ (LE 19^2) and another with $N = 29^2$ (LE 29^2).

We performed 10 evolutionary runs for the LE task with initial populations generated at random. In all cases, the fitness stayed flat and did not rise above zero,

hence, unlike the one-dimensional case, non-zero performance for randomly-generated transition tables (chromosomes) are statistically very rare. Note that for leader election a final configuration with a single active cell is supposed to be fixed. Compared to the one-dimensional case the number of mandatory transitions implementing the fixed point requirement is larger (discussed in Section 6.2.2). More precisely, any transition table for leader election must contain 10 bits at specific locations.

To aid the evolutionary search we introduced an intermediate task, the density minimization (DM), which continuously reduces the density (the ratio of ones) in a final fixed point configuration. Note that as opposed to the leader election task, a maximization problem, the DM's output (an error) needs to be minimized. Formally, the minimum square non-zero density task DM is defined as

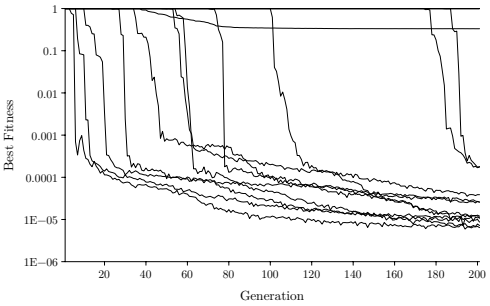
$$\mathrm{DM}(\mathbf{s}) = \left\{ \begin{array}{lll} (\frac{\#_1 \mathbf{s}}{N}) & \Longleftrightarrow & {}_{\#_1}\mathbf{s} > 0, \Phi(\mathbf{s}) = \mathbf{s} \\ 0.25 & \Longleftrightarrow & {}_{\#_1}\mathbf{s} = 0, \Phi(\mathbf{s}) = \mathbf{s} \\ 0 & \Longleftrightarrow & \Phi(\mathbf{s}) \neq \mathbf{s} \end{array} \right\},$$

where N is the number of cells and $\mathbf{s}$ is a final configuration. To prevent CA from a trivial elimination of all active cells, we penalize configurations with no active cells by 0.25. We square the error to impose a stronger evolutionary pressure for lower densities. This is important especially because the evaluation is repeated multiple times, and fitness is calculated as an average. Also note that we expect a final configuration to be a fixed point to make the transition to an actual leader election smoother.

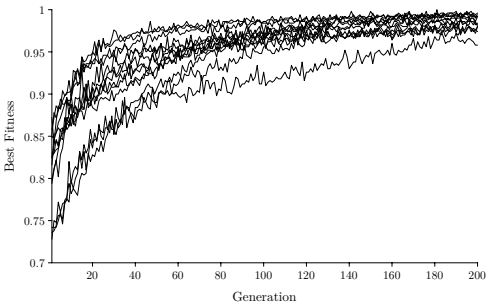
We ran 20 evolutions, of which 12 reached fitness (error) smaller than 1. It

means that 8 evolutions could not find a single chromosome resulting in a fixed point configuration. Figure 5.11(a) presents the best (minimal) fitness of all evolutionary runs. Once a fixed point solution is found the square error quickly drops to the range $(10^{-5}, 10^{-4})$, with the overall last best fitness of $6.4\,10^{-6}$.

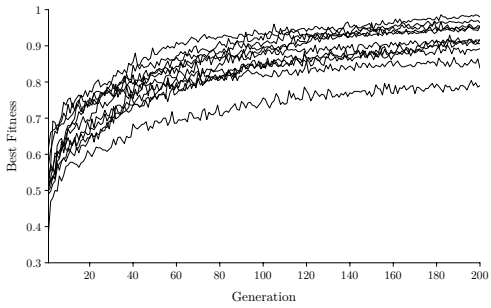
The higher the performance for the original leader election task, the lower the error for the DM task. Yet the opposite implication does not hold, since the DM task might assign a low error to transition tables with configurations containing two or more active cells, whose performance for the leader election task could be only marginal. As a matter of fact, the DM task is not intended to solve the leader election completely. The best chromosome of the DM task reached performance



(a) DM



(b) LE 19^2



(c) LE 29^2

Figure 5.11: Fitness of the best chromosomes from each population of the three evolutionary sets: DM, LE 19^2, and LE 29^2.

of around 0.7 when run against the leader election task.

To further improve this result we switched back to the original leader election, and we used the last best chromosomes of the DM task with < 1 error, which were randomly perturbed to fill the full population of 200 chromosomes. In the third set of evolutions, with the fitness determined by the actual leader election evaluation, we ran 13 evolutions with the lattice size $N = 19^2$. As shown in Figure 5.11(b) the best fitness starts at the range $0.7 - 0.8$ and after 40 generations climbs above 0.9. The overall last best fitness is 0.996.

To investigate scalability of the system with respect to N we performed an additional set of evolutions for leader election, again using the chromosomes from the DM task, but with an increased lattice size $N = 29^2$. Since the system size is substantially larger, the task is more difficult and fitness converges at slower pace. Figure 5.11(c) shows all 13 evolutions starting from fitness $0.4 - 0.6$ reaching maximum at 0.986.

5.4.2 Analysis

The computational mechanics analysis employed in Section 5.3 has been introduced by Crutchfield and Hanson [27, 49, 48] to describe the information processing in one-dimensional CAs. An extension of computational mechanics to higher dimensions faces major challenges. Due to non-linear spatial connections, the identification of domains and particles based on ϵ-machine reconstruction and finite state transducer representation could not be applied. Recently, Cenek [20] proposed several methods that automatically identify spatio-temporal patterns and describe their kinematic models using several filters based on local sensitivity

[33], local statistical complexity [85], local information storage [85], and local information transfer. These filters, however, fail to identify all spatio-temporal structures in general, and the area of computational mechanics in higher dimensions is currently being investigated. Since the goal of this thesis was not to propose new means for CA analysis, but rather dwell on existing mature techniques, we opted for statistical methods classifying overall dynamics of two-dimensional CAs through perturbation stability. The measures are the Derrida measure [34, 86] (Section 5.4.2.6) and the damage spreading measure[83, 68], related to a discrete version of Lyapunov exponent [17] (Section 5.4.2.7). We also investigated λ, the transition table density, of the evolved CAs (Section 5.4.2.5).

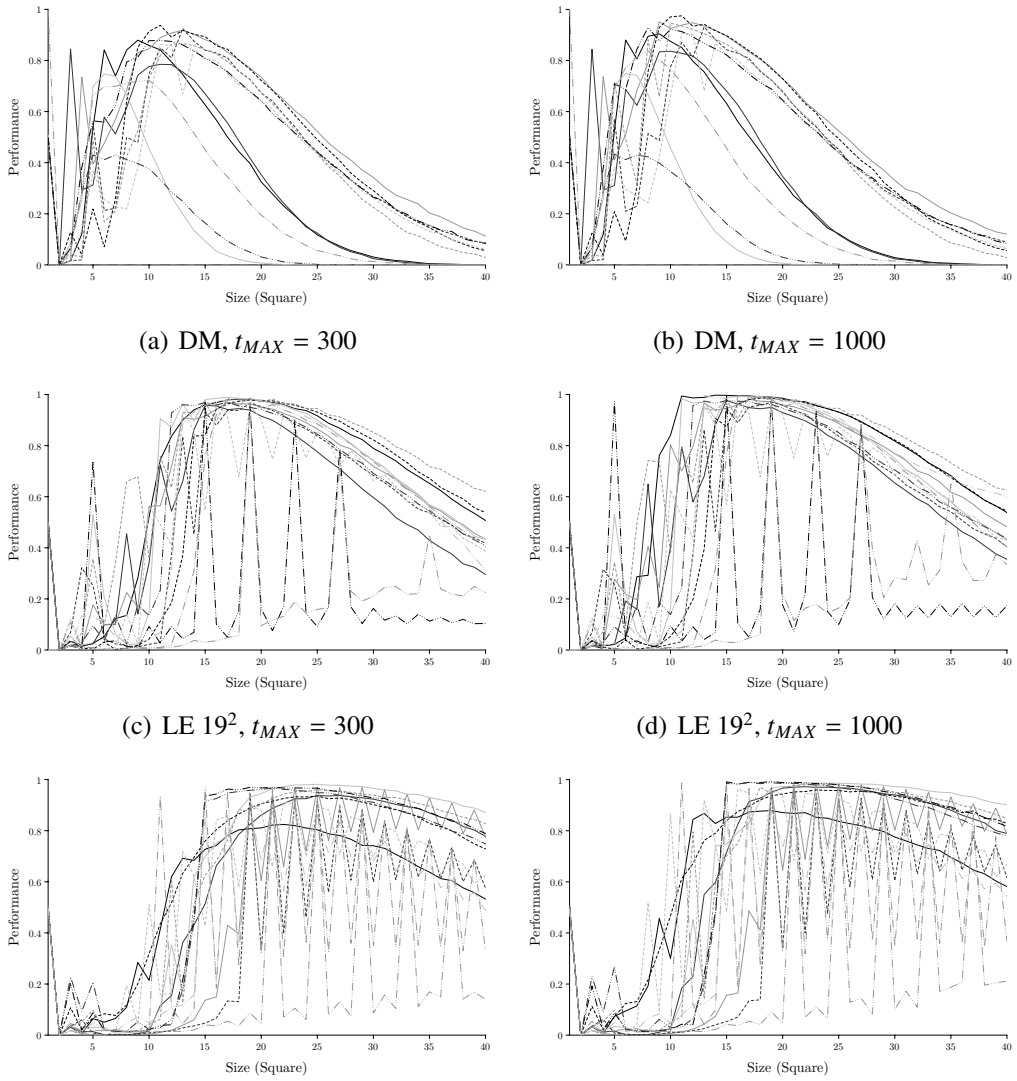
5.4.2.1 Performance and Strategies

We evaluated leader election performance of the last best chromosomes from all three evolutionary sets for square sizes $N = 1^2$ to 40^2. Figure 5.12 shows averaged performance for all CAs for $t_{MAX} = 300$ and 1000, and uniform and density-uniform distributions of initial configurations.

As we will prove in Chapter 6, small lattices are generally very difficult to solve due to the configuration symmetry and loose-coupling of active cells. As a matter of fact, all CAs perform poorly for $N < 5^2$.

In general, performance peaks occur for low N for the density minimization task, and move to larger sizes for leader election with $N = 19^2$. Finally CAs for leader election targeting $N = 29^2$ are the most scalable and their performance peaks are rather wide starting from $N = 15$. This shows that for targeted applications we can provide a CA that performs well on a specific range of system sizes. Still, the best CA for LE 29^2 could reach > 0.9 performance even for a large lattice

size of $N = 40^2$, and so its scalability allows it to cover the largest set of system sizes. In Sections 5.4.2.6 and 5.4.2.7 we demonstrate that the CA's performance



(a) DM, $t_{MAX} = 300$

(b) DM, $t_{MAX} = 1000$

(c) LE 19^2, $t_{MAX} = 300$

(d) LE 19^2, $t_{MAX} = 1000$

(e) LE 29^2, $t_{MAX} = 300$

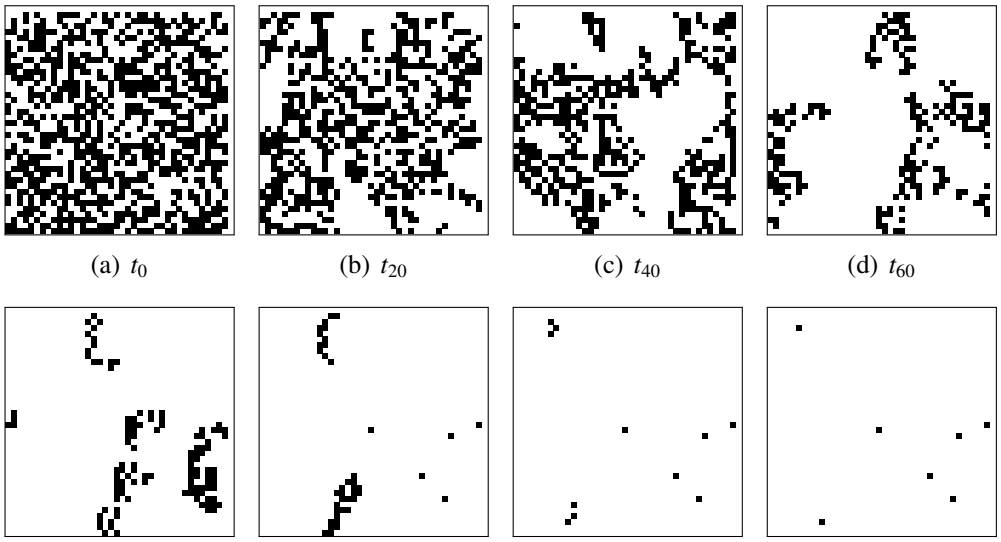
(f) LE 29^2, $t_{MAX} = 1000$

Figure 5.12: Performance of the last best chromosomes from three evolutionary sets (DM, LE 19^2, and LE 29^2) for the square sizes $N = 1^2, \ldots, 40^2$ calculated as an average over 10^4 runs. The maximal time t_{MAX} allowed for leader election is set to 300 steps in the left and 1000 in the right column.

and the task difficulty correlates highly with complexity and perturbation stability calculated by the Derrida measure and damage spreading (Lyapunov exponent).

For most of the CAs the maximal time $t_{MAX} = 300$ is sufficient to reach a fixed point (if there is one) on any test size. Also, the execution time increases slowly. In fact, time complexity is sublinear, i.e., $< O(N)$. That is due to two-dimensionality of a toroidal lattice, which allows to spread information from a single cell to all other cells in $0.5\sqrt{N}$ steps with radius $r = 1$.

In the next sections, we present performance results as well as the best-performing CAs for the DM, LE 19^2, and LE 29^2 tasks in more detail.



(a) t_0 (b) t_{20} (c) t_{40} (d) t_{60}

(e) t_{70} (f) t_{80} (g) t_{90} (h) t_{94}

Figure 5.13: Example space-time diagrams of the best-performing density-minimizing CA on lattice size $N = 40^2$. Figures show a CA computation starting from an initial configuration with active cells distributed uniformly (time t_0), followed by 7 state snapshots. The CA optimized for density minimization fails to elect a leader and reaches a final configuration with 7 active cells at time t_{94}.

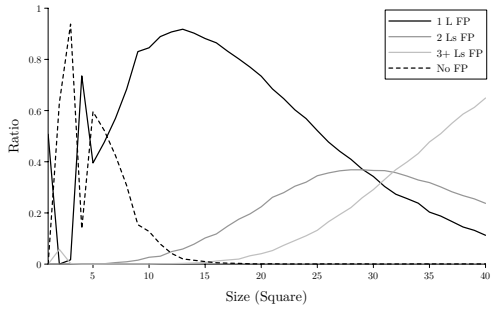
5.4.2.2 Density Minimization

The best CA evolved for the DM task has a peak performance of 0.92 for uniform initial distributions and $N = 14^2$. For the lattice sizes $N > 18^2$ performance quickly drops, since the DM approach is localistic. The performance for $N = 39^2$ is only 0.132 (Table 5.11).

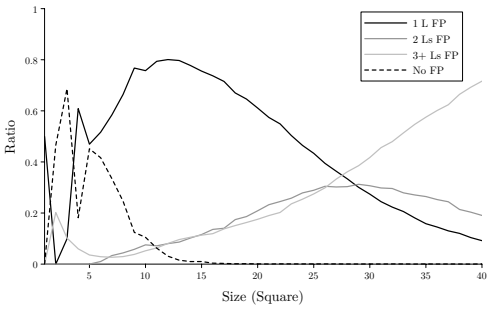
The time needed to halt is around 100 steps for the benchmark size $N = 19^2$. Density minimization is fast because there is only limited coordination between leader-electing regions, which are fairly static and dense (shown in Figure 5.13). This results in a quick disconnection of the regions, and therefore the number of active cells in a final configuration is for $N > 20^2$ generally equal or larger than 2 (Figure 5.14). On the other hand, the number of active cells in a final fixed point configuration is on average very low: $1.5 - 3$ for $N > 12^2$ and uniform distribution. In fact, it is lower than in evolved leader-electing CAs. Note that minimizing density to a non-zero number does not necessarily imply a successful leader election.

N	Uniform	Density-Uniform
19^2	0.770	0.646
29^2	0.371	0.301
39^2	0.132	0.104

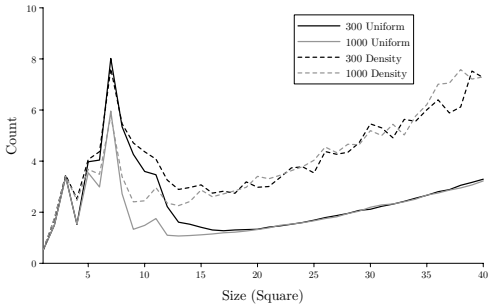
Table 5.11: Performance of the best density-minimizing CA using uniform and density-uniform initial distributions.



(a) uniform



(b) density-uniform

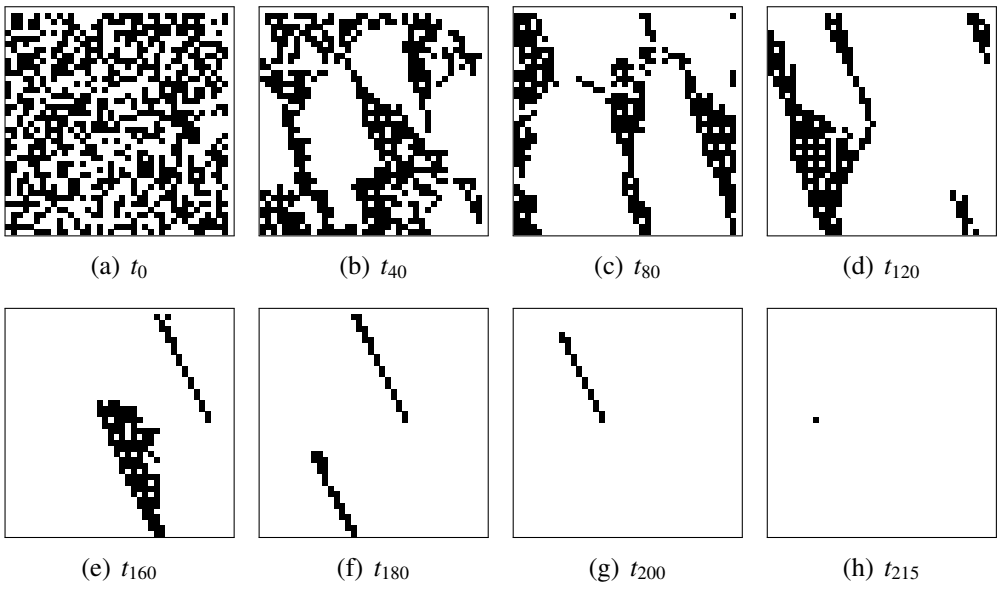


(c) final one-count

Figure 5.14: Performance of the best density-minimizing CA for the square sizes $N = 1^2, \ldots, 40^2$ calculated as an average over 10^4 runs using uniform and density-uniform initial distributions. The maximal time t_{MAX} allowed for leader election in (a) and (b) is 300. Figures (a) and (b) show the ratio of runs that end in a fixed point with a single active cell (1 L FP), two active cells (2 Ls FP), three or more active cells (3+ Ls FP), or no fixed point (No FP). Figure (c) plots the number of ones in a final configuration.

5.4.2.3 Leader Election Targeting $N = 19^2$

The best CA for the LE 19^2 task has significantly higher performance than any of the density-minimizing CAs. For $N = 19^2$, performance reaches 0.987 for uniform and 0.807 for density-uniform distributions (Table 5.12). Note that density-uniform distribution has (compared to uniform configurations) a bias for configurations with low densities, which are generally more difficult, especially because of the low radius $r = 1$.

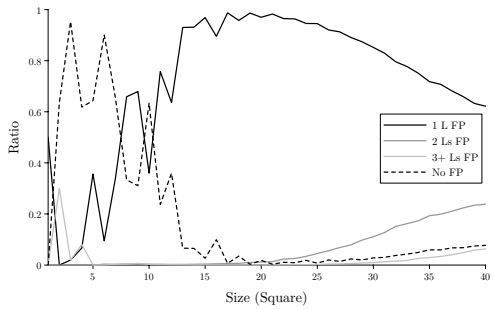


(a) t_0 (b) t_{40} (c) t_{80} (d) t_{120}

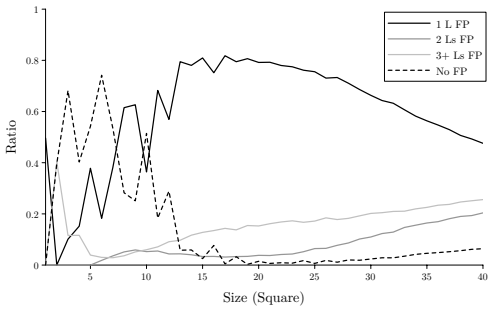
(e) t_{160} (f) t_{180} (g) t_{200} (h) t_{215}

Figure 5.15: Example space-time diagrams of the best-performing leader-electing CA targeting $N = 19^2$ on lattice size $N = 40^2$. Figures show a CA computation starting from an initial configuration generated by a uniform distribution (time t_0), followed by 7 state snapshots. The CA reaches a final configuration with a single active cells at time t_{215}.

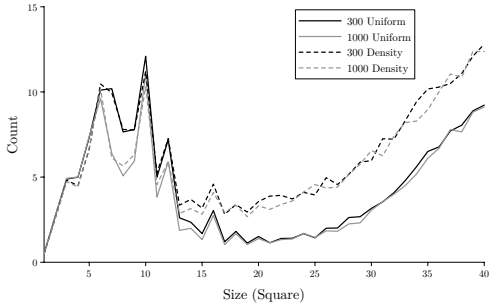
Also, the system's scalability improved and for $N = 39^2$ performance reaches 0.633 for uniform and 0.493 for density-uniform initial distributions (Figure 5.16). The execution time increased to around 220 steps for $N = 19^2$. That is due to the emergence of regular propagating regions, domains, and particles similar to those found in space-time diagrams of one-dimensional leader election. As presented in Figure 5.15 the initial random configuration with ones and zeros uniformly distributed splits into different slowly-contracting regions, which keep connected by lines of active cells. These regions propagate left, sweep any remaining active cells, transform to a single moving line that shrinks from both sides, and finally contracts to a single active cell.



(a) uniform



(b) density-uniform



(c) final one-count

Figure 5.16: Performance of the best leader-electing CA targeting 19^2 for the square sizes $N = 1^2, \ldots, 40^2$ calculated as an average over 10^4 runs using uniform and density-uniform initial distributions. The maximal time t_{MAX} allowed for leader election in (a) and (b) is 300. Figures (a) and (b) show the ratio of runs that end in a fixed point with a single active cells (1 L FP), two active cells (2 Ls FP), three or more active cells (3+ Ls FP), or no fixed point (No FP). Figure (c) plots the number of ones in a final configuration.

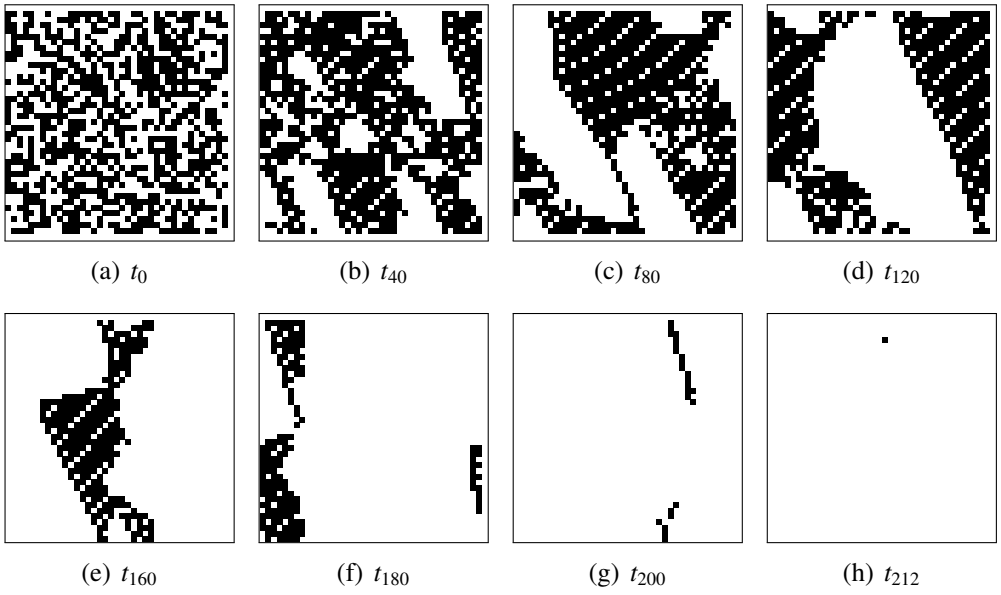
N	Uniform	Density-Uniform
19^2	0.987	0.807
29^2	0.874	0.686
39^2	0.633	0.493

Table 5.12: Performance of the best leader-electing CA targeting $N = 19^2$ using uniform and density-uniform initial distributions.

5.4.2.4 Leader Election Targeting $N = 29^2$

Since the LE 29^2 task targets larger system's sizes, performance of the best leader-electing CA stays above 0.9 for $N = 19^2$ up to 40^2. Performance for $N = 19^2$ is 0.9672 for uniform and 0.768 for density-uniform initial distributions (Table 5.13). The system scales well and for $N = 39^2$ performance reaches 0.9078 for uniform and 0.6918 for density-uniform distribution (Figure 5.18).

This is the best performing and most scalable CA, yet it is also the slowest. Time complexity is, due to two-dimensional information spreading, still sublinear. Similarly to LE 19^2, a large execution time is caused by propagating regions that move around the lattice, contract, and eliminate any remaining active cells. Performance is stable and reaches > 0.9 for all $N \geq 19^2$.



(a) t_0 (b) t_{40} (c) t_{80} (d) t_{120}

(e) t_{160} (f) t_{180} (g) t_{200} (h) t_{212}

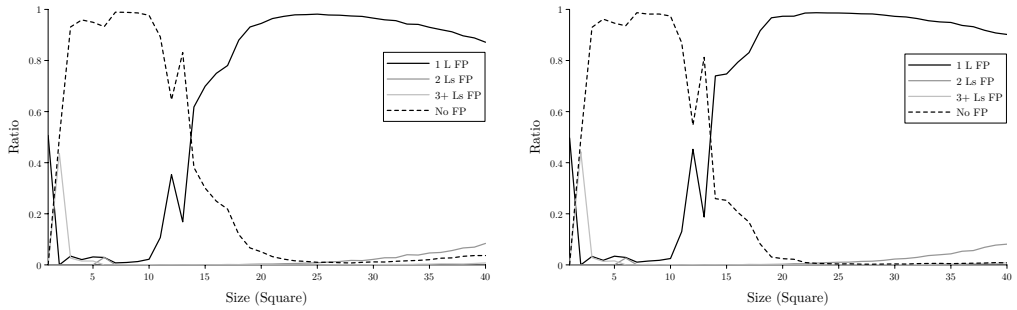
Figure 5.17: Example space-time diagrams of the best-performing leader-electing CA targeting $N = 29^2$ on lattice size $N = 40^2$. Figures show a CA computation starting with a uniform initial distribution (time t_0), followed by 7 state snapshots. The CA reaches a final configuration with a single active cells at time t_{212}.

The main improvement is the occurrence of high-density regular domains $(111110)^*$ and $(11111110)^*$. These domains contract at a slower pace, and so reduce the risk of disconnection. Having more active cells in a configuration helps it to keep propagating and contracting fronts together. Interestingly, because of an initial increase in density, the system is too unstable for small lattices for which the CA cycles and never reaches a fixed point. Compared to previous CA, the best LE 29^2 CA performs poorer on density-uniform initial distributions. Low-density configurations, frequent among density-uniform distributions, harm domains, which require a lot of active cells.

Last but not least, need for global coordination resulted in a higher occurrence of modulo-dependent CA (Figure 5.12). Yet unlike the one-dimensional case, the best-performing CAs are not necessarily more N-modulo restricted. That might be due to a smaller radius $r = 1$, which restricts periodicity of moving particles or propagating patterns, and so interaction results are less phase-dependent.

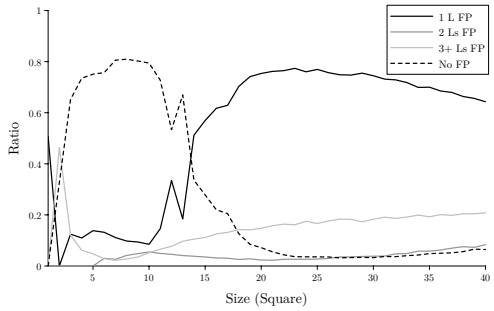
N	Uniform		Density-Uniform	
	$t_{MAX} = 300$	$t_{MAX} = 1000$	$t_{MAX} = 300$	$t_{MAX} = 1000$
19^2	0.9312	0.9672	0.7413	0.768
29^2	0.9726	0.9779	0.7549	0.7649
39^2	0.8884	0.9078	0.656	0.6918

Table 5.13: Performance of the best leader-electing CA targeting $N = 29^2$ using uniform and density-uniform initial distributions and the maximal time $t_{MAX} = 300$ and $t_{MAX} = 1000$.

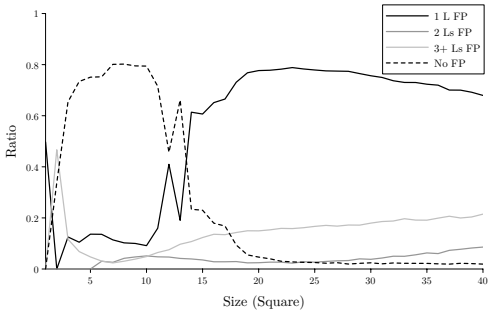


(a) uniform, $t_{MAX} = 300$

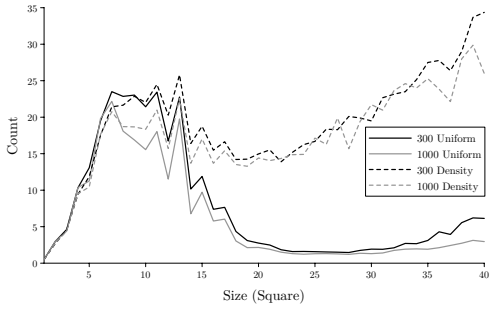
(b) uniform, $t_{MAX} = 1000$



(c) density-uniform, $t_{MAX} = 300$



(d) density-uniform, $t_{MAX} = 1000$



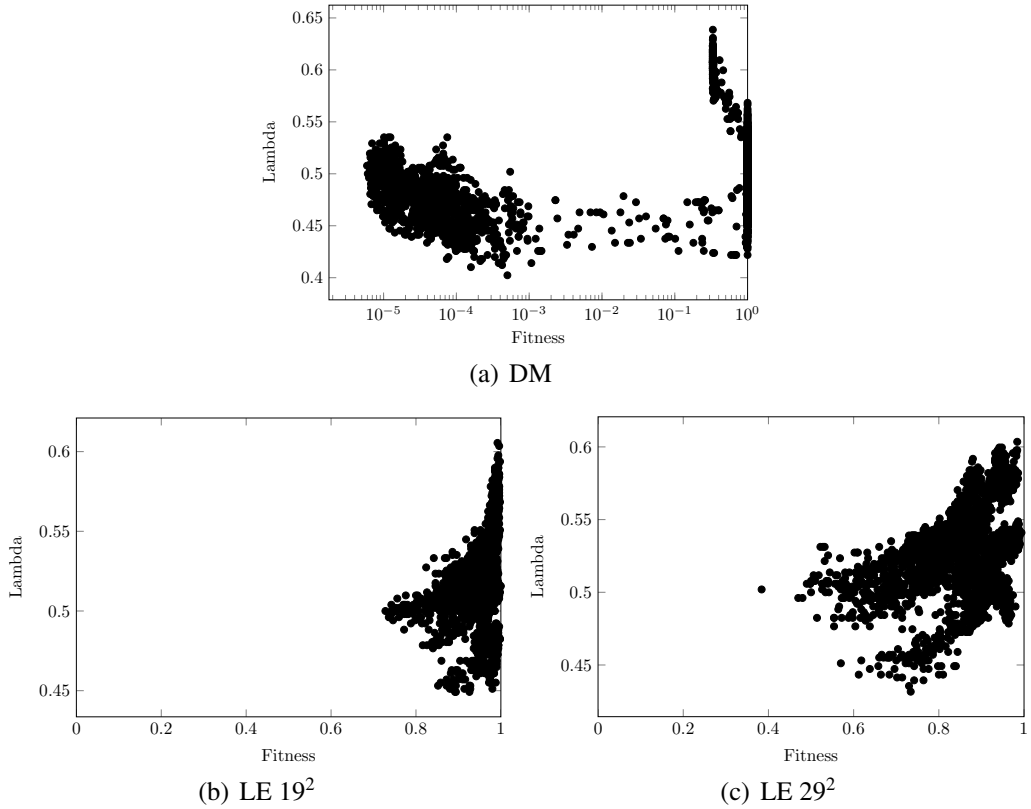
(e) final one-count

Figure 5.18: Performance of the best leader electing CA targeting 29^2 for the square sizes $N = 1^2, \ldots, 40^2$ calculated as an average over 10^4 runs using uniform and density-uniform initial distributions. The maximal time t_{MAX} allowed for leader election is 300 in (a) and (b), and 1000 in (c) and (d). Figures (a-d) show the ratio of runs that end in a fixed point with a single active cell (1 L FP), two active cells (2 Ls FP), three or more active cells (3+ Ls FP), or no fixed point (No FP). Figure (e) plots the number of ones in a final configuration.

5.4.2.5 Transition Table Density

Recall that λ is transition table density, i.e., the fraction of ones in a transition table. Figure 5.19 shows the fitness-λ correlation of the best chromosomes from each population for the DM, LE 19^2, and LE 29^2 tasks. For all tasks, the λ-critical regions enabling leader election are present. The critical region of the density minimization task is $(0.46 - 0.52)$ with fitness (error) $< 10^{-5}$, and $(0.45 - 0.6)$ and $(0.48 - 0.6)$ with fitness > 0.95 for the LE 19^2 and LE 29^2 respectively.

Recall that one-dimensional leader-electing CAs have optimal λ around 0.46.



(a) DM

(b) LE 19^2

(c) LE 29^2

Figure 5.19: Relation between the fitness and λ (transition table density) of CAs from the three evolutionary sets: DM, LE 19^2, and LE 29^2. Note that a critical high-performing region for leader election (b-c) correlates with $\lambda \in (0.48, 0.6)$ excluding 0.5.

Because Moore neighborhoods in two-dimensional CAs has a small radius, the risk of disconnecting active regions is higher. Further, a transition function must output an active state more often and the λ-critical regions for leader election are clustered on average for densities > 0.5. Also, the best-performing region has a gap in densities around $\lambda = 0.5$, because $\lambda = 0.5$ transition tables are more likely to be symmetric, which would impose an additional constraint on leader election and solvable configurations (Section 6.1).

5.4.2.6 Derrida Measure

To investigate a dynamical regime of the evolved CAs we calculate their sensitivity to perturbations as defined by the Derrida measure [34]. More precisely, we trace the convergence or divergence of two perturbed configurations after one update. Given perturbation strength $p \leq N$, we first generate an initial configuration $\mathbf{s^1}$ using uniform distribution and create another initial configuration $\mathbf{s^2}$ by randomly flipping p bits in $\mathbf{s^1}$. The Hamming distance d_t is therefore $d_t = d(\mathbf{s^1}, \mathbf{s^2}) = p$. Next, we run the CA on both configurations and again calculate the Hamming distance at time $t + 1$ as $d_{t+1} = d(\Phi(\mathbf{s^1}), \Phi(\mathbf{s^2}))$. Now, the Derrida measure [86], an annealed approximation of the system's stability, is obtained by plotting d_t against d_{t+1} for different starting Hamming distances averaged over sufficiently large numbers of samples and normalized by the system size N.

The Derrida measure, sometimes called the Derrida plot, describes how fast a perturbation spreads to other cells in one time step. The identity line $d_t = d_{t+1}$ is important here, since it separates the dynamical regimes. The curves below the identity line correspond to ordered systems, which are not sensitive to perturbations, so information does not spread and eventually die out. If the $\frac{d_{t+1}}{d_t}$ ratio is

larger than one the CA is chaotic, and so even a small state change diverges and disturbs the system's dynamics. Finally, the curves tangent to the identity line are critical. In this region, also called the edge of chaos, a perturbation will not die out nor spread out. The critical regime has been shown optimal for information processing and computing. Furthermore, the most important part of the Derrida plot is for the small Hamming distances. The more the Derrida curve lies above the main diagonal for small values d_t, the more chaotic the system.

Figure 5.20 presents the Derrida measure for randomly generated twodimen-

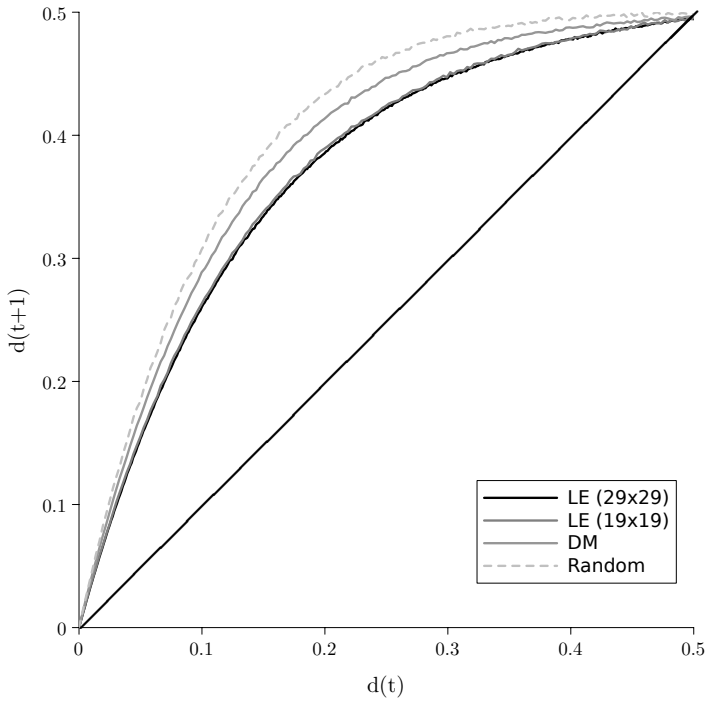


Figure 5.20: Derrida curves for 100 randomly generated two-dimensional CAs with Moore neighborhood, and the last best CAs from all evolutions for the density minimization task (DM), and the leader election task with $N = 19^2$ (LE 19^2) and $N = 29^2$ (LE 29^2). Averages over all CAs and 100 configurations per each Hamming distance d_t are plotted. Note that the identity line represents the critical dynamical regime, i.e., the closer to the line, the more complex the dynamics. The portion shown in the plot is limited to $d_t \leq 0.5$.

sional CAs with Moore neighborhoods, and the best CAs for the three evolutionary tasks. Because Moore neighborhoods contains 9 cells, hence the in-degree of each cell is 9, chaotic dynamics, where even a small perturbation results in trajectories that diverge rapidly, is more likely. That has been shown by Kauffman [58] for random Boolean networks, but it qualitatively applies also to CAs. That coheres with our findings, since the Derrida curve for random CAs is far above the line of criticality.

As expected, the dynamics of the density-minimizing CAs are more ordered, since their Derrida curve lies bellow the curve for random CAs. The density minimization task is, however, not as challenging as leader election, whose Derrida curve is closest to the identity line and its dynamics are, therefore, most complex. In a nutshell, the more complicated the task, the more complex the CA dynamics required to solve the problem.

5.4.2.7 Damage Spreading

Similarly to the Derrida measure, the damage spreading measure [83, 68] is a form of sensitivity analysis. The idea is to track how fast the smallest possible state perturbation of a single bit spreads throughout the CA.

By definition, the damage spreading $\overline{d}(T)$ is simply the Hamming distance of two trajectories of the same system after time T starting from two configurations that differ only in a single position (bit) averaged over multiple runs. Here we plot $\overline{d}(T)$ for the last best chromosomes from our three evolutionary sets and 100 randomly generated two-dimensional CAs with Moore neighborhood. We performed 1000 runs for each CA and tracked the distance for time $T = 1$ up to 300. The curves show how fast a perturbation disturbs the system. Since leader elec-

tion as well as density reduction tasks are supposed to minimize the number of active cells after the initial expansion the Hamming distance decreases and finally reaches low values around 10.

The most interesting part of these curves is the initial (steep) portion, which shows the rate of convergence or divergence. That tells us how information transmission and noise propagation are embedded in the system. In fact, $\overline{d}(1)$, the Hamming distance after a single update, could be used for calculation of a variant of Lyapunov stability (exponent) for discrete systems [17]. More precisely, the Lyapunov exponent $\lambda = \lim_{t\to\infty} \frac{1}{T}\left(\frac{d(T)}{d(0)}\right)$ could be approximated as

$$\lambda = ln(\frac{\overline{d}(1)}{\overline{d}(0)}).$$

Since $\overline{d}(0)$, the size of initial perturbation, is 1, $\lambda = ln\,\overline{d}(1)$. Note that the notation for the Lyapunov exponent, λ, conflicts with a symbol used for the transition table density introduced earlier. Since we present these two areas separately we follow

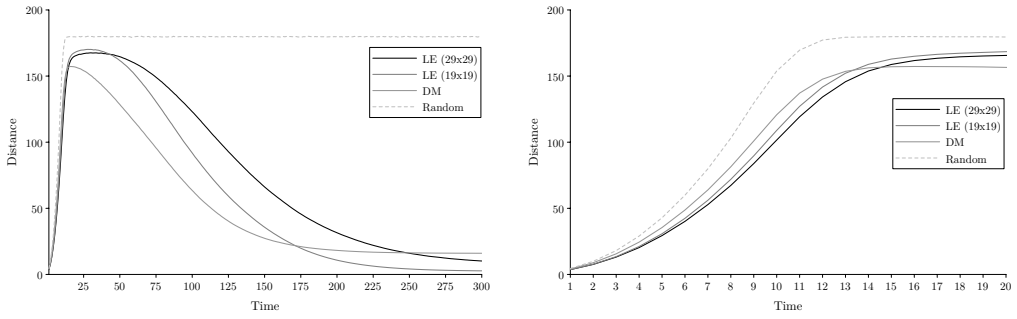


Figure 5.21: Damage spreading for 100 randomly generated two-dimensional CAs with Moore neighborhood, and the last best CAs from all evolutions for the density minimization task (DM) and the leader election task with $N = 19^2$ (LE 19^2) and $N = 29^2$ (LE 29^2). Averages for all CAs over 1000 runs are plotted.

the original notations in both cases.

The critical line for the Lyapunov exponent is $\lambda = 1$. For $\lambda > 1$ the system's trajectories tend to diverge; the system is unstable and chaotic. For $\lambda < 1$ the system converges or contracts, hence it is stable and ordered. At the threshold of these regions, around $\lambda = 1$, complex dynamics combining ordered and chaotic properties occurs. This region promotes efficient information transfer and processing.

Our results are consistent with the Derrida measure presented earlier. The CA from LE 29^2, solving the most complicated instance of the problem, requiring global coordination of cells, have low $\lambda = 1.288$ showing their dynamics are the most complex. The CAs of the LE 19^2 task has slightly higher $\lambda = 1.305$, the DM is already quite chaotic and reaches $\lambda = 1.414$, and finally randomly-generated CAs are the most chaotic with $\lambda = 1.499$.

5.4.3 Asynchronous Leader Election

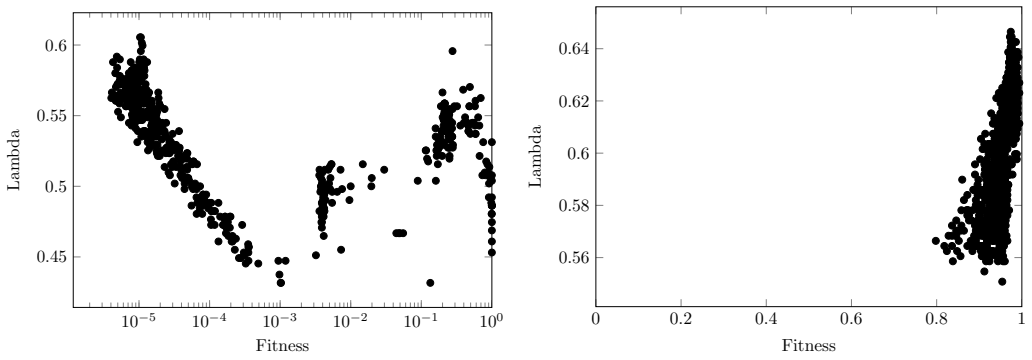
We demonstrated successful leader election in two-dimensional CAs. Here we ask whether the synchronous update employed in standard CA architecture is required to solve this task. Namely, we evolve and analyze leader election in an asynchronous CA, where the global update function Φ is implemented by invocation of each cell's update $\phi(\eta_i)$ independently in a random order (permutation of cells) during each global step. Note that because the order of asynchronous update is random, the CA architecture is not deterministic anymore.

As in the synchronous case the leader election task expects a final configuration to be a fixed point with a single active cell. Because of nondeterminism

we simply assume that if the last two configurations are the same the system is in a (weak) fixed point, which, however, does not have to hold if the CA's run continues. We again performed two sets of evolutions: an intermediate density minimization task followed by the actual leader election task on $N = 19^2$ cells. Note that solving asynchronous leader election was not our priority, and so we performed only a limited number of evolutions; 4 for the density minimization task, as well as for the leader election task.

Figure 5.22 presents a relation between fitness and transition table density λ. Similarly to synchronous leader election, we found a critical region $\lambda \in (0.61, 0.64)$. This is higher compared to the synchronous instance, because in the asynchronous case a transition table implementing leader election needs to output ones more often than zeros to make configurations more dense (less breakable).

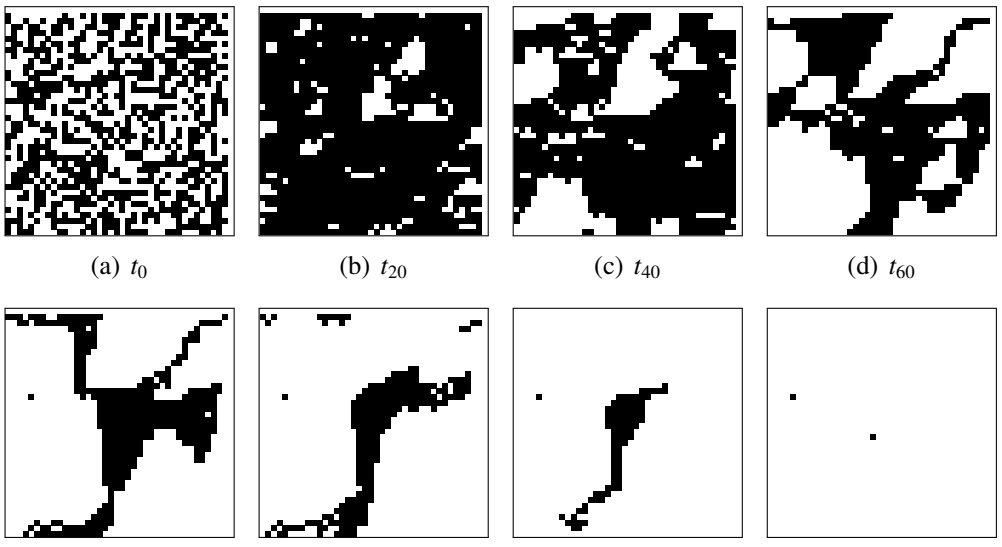
Our results show that asynchronous leader election is indeed possible, and for $N = 19^2$ performance of the best CA reaches 0.965 for uniform and 0.735 for



(a) DM Async (b) LE Async

Figure 5.22: Relation between the fitness and λ (transition table density) of asynchronous CA from the density minimization (DM Async)and leader election (LE Async) evolutionary sets. Note that a critical high-performing region for leader election (b) correlates with $\lambda \in (0.61, 0.64)$.

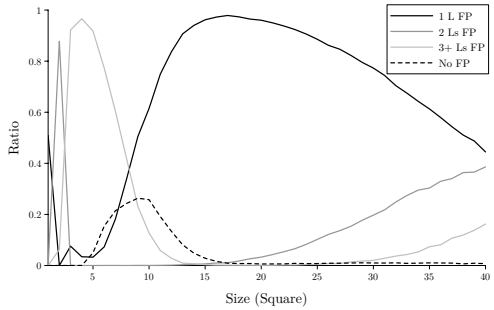
density-uniform initial distributions (Figure 5.24). Yet because of nondeterministic update, the asynchronous CA could not benefit from regular propagated patterns enabling coordinated exchange of information over distances. Therefore, the strategy of asynchronous leader election is to increase density of the contracting regions in order to reduce the risk of disconnection (Figure 5.23). This strategy is statistical and localistic, and so it works only partially and the CA scales poorly. Performance for $N = 39^2$ and uniform distribution is 0.487. Due to an absence of particles, there is no modularity and execution is fast (100 time steps for $N = 19^2$). Yet nondeterminism makes runtime more variable. We admit that more thorough investigation of asynchronous leader election is required, however, we leave that research avenue for future consideration.



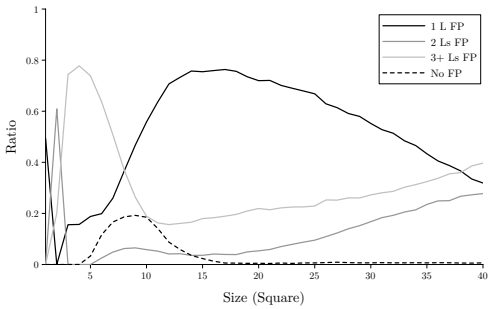
(a) t_0 (b) t_{20} (c) t_{40} (d) t_{60}

(e) t_{80} (f) t_{90} (g) t_{100} (h) t_{116}

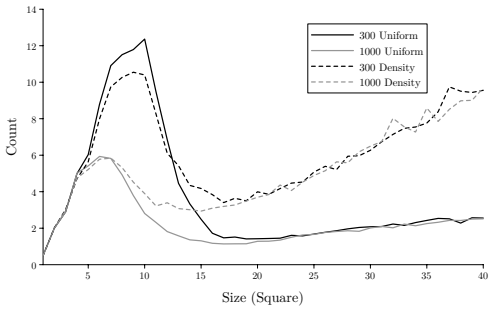
Figure 5.23: Example space-time diagrams of the best-performing asynchronous leader-electing CA on lattice size $N = 40^2$. Figures show a CA computation starting from an initial configuration generated by using uniform distribution (time t_0), followed by 7 state snapshots. The CA fails to elect a leader and reaches a final configuration with 2 active cells at time t_{116}.



(a) uniform



(b) density-uniform



(c) final one-count

Figure 5.24: Performance of the best asynchronous leader-electing CA for the square sizes $N = 1^2, \ldots, 40^2$ calculated as an average over 10^4 runs using uniform and density-uniform initial distributions. The maximal time t_{MAX} allowed for leader election is 300 in (a) and (b). Figures (a) and (b) show the ratio of runs that end in a fixed point with a single active cells (1 L FP), two active cells (2 Ls FP), three or more active cells (3+ Ls FP), or no fixed point (No FP). Figure (c) plots the number of ones in a final configuration.

Symmetry,...,is one idea by which man through the ages has tried to comprehend and create order, beauty and perfection.

Hermann Weyl (1885 - 1955)

6

Limitations and Performance Upper Bound

Previously we identified and analyzed various one- and two-dimensional binary CAs solving the problem of anonymous leader election. The most successful strategies employed collective, global, particle-based computation [1, 28] and reached very satisfactory performance of about 99% for a uniform distribution of initial configurations.

It is now natural to ask whether a better-performing CA could exist. In this chapter, we ask whether the performance could be further improved, and if so, to what extent. Which initial configurations are insolvable for leader election? What is a theoretical upper bound on the performance rate?

Let us recall that the leader election task is defined as a transformation of an initial configuration to a final fixed point configuration with exactly one active cell, where an active cell is a cell in the leader state $1 \in \Sigma$. Note that for a binary

CA, all non-active cells must be in the state 0. Formally

Definition 6.0.1. *The* leader election *task T for the state set Σ (alphabet) and the leader state $1 \in \Sigma$ is*

$$T : \Sigma^N \to \{\mathbf{s} \in \Sigma^N |_{\#_1} \mathbf{s} = 1, \Phi(\mathbf{s}) = \mathbf{s}\},$$

where $\#_1\mathbf{s}$ is the number of 1's in a configuration $\mathbf{s}$.

We say that CA for leader election is *self-stabilizing* whenever the system eventually reaches a correct state with a single active (leader) cell, regardless of the initial configuration. Note that the original definition [89] expects the processors to be initially all in the same state. As we will show, the self-stabilizing property cannot hold for all configurations.

We identify general limitations that no one- or two-dimensional CA can overcome. We show that a minimal, fully uniform and anonymous architecture of CA cannot produce a correct output from all input configurations. We enumerate such *insolvable configurations*, including both symmetric and loosely-coupled configurations, and we formulate a universal upper bound on performance for the anonymous one-dimensional leader election problem. The first limitation is due to *symmetry* of configurations, which is preserved during computation (Section 6.1). The second one is a result of the stable point requirement on a final leader election configuration, which implies insolvability of *loosely-coupled* configurations, where the distance between active cells is too large (Section 6.2).

The calculated theoretical upper bound performance of one- and two-dimensional CAs for the leader election problem is greater than performance of the best one- and two-dimensional strategies, such as, the improved strategy of mirror particles

(Section 5.3.2.6) and the best two-dimensional CA targeting $N = 19^2$ (Section 5.4.2.3) and $N = 29^2$ (Section 5.4.2.4).

6.1 Symmetric Configurations

In this section we show that a specific class of initial configurations, namely symmetric configurations, are insolvable. A symmetric configuration consists of a single sequence (pattern) replicated along the configuration in a certain direction without any gaps. Note that the purpose of leader election and the presence of symmetry are closely related. In fact, leader election is often referred to as symmetry breaking. Also, note that symmetric configurations as opposed to loosely-coupled configurations discussed in Section 6.2 do not rely on specific radius nor neighborhood function. The only assumption is that the neighborhood as well as transition function are uniform.

6.1.1 One-Dimensional Symmetric Configurations

We start with a simple case of uniform configurations. As stated by Angluin [3], uniform configurations are insolvable by any anonymous deterministic algorithm (including one-dimensional CAs). Uniformity of CA can be manifested in its transition function, deterministic update, synchronicity, topology, configuration, and cells' anonymity. Basically a fully symmetric or uniform system in terms of its structure, configuration and computational mechanisms cannot produce any reasonable or complex dynamics. We demonstrate this intuitive assumption in the following lemma.

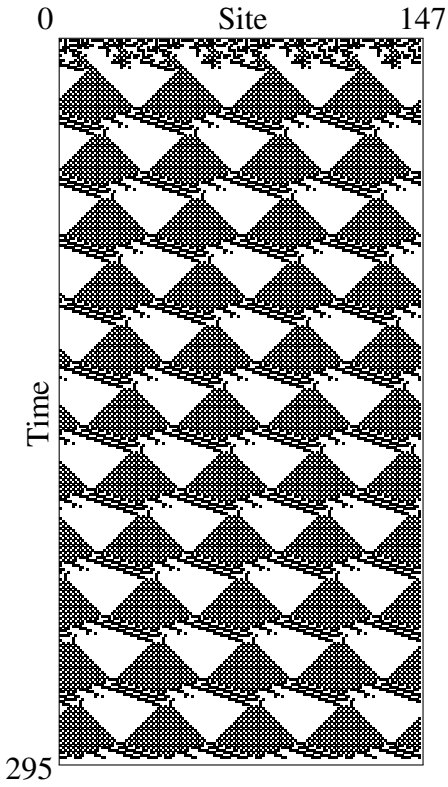


Figure 6.1: A space-time diagram of CA computation on a one-dimensional symmetric configuration. Note that leader election from a symmetric configuration is impossible.

Lemma 6.1.1. *A uniform configuration, which is any configuration consisting of all cells in the same state (e.g.,* 0^N*, and* 1^N*), is insolvable for leader election.*

Proof By definition, $\Phi(\mathbf{s}) = (\phi(\eta_0), \ldots, \phi(\eta_{N-1}))$. Since $\mathbf{s} = a^N, a \in \Sigma$ is a uniform configuration, $\Phi(\mathbf{s}) = (\phi(a^{2r+1}))^N = b^N$ for some $b \in \Sigma$, which is again a uniform configuration. Therefore, only a stable point or a periodic cycle of uniform configurations can be obtained. □

We now formally define a much larger class of symmetric configurations, which include uniform configurations as a special case. Figure 6.1 shows a CA computation on a one-dimensional symmetric configuration.

Definition 6.1.1. *A configuration* $\mathbf{s} \in \Sigma^N$ *is* symmetric *iff* $\mathbf{s} = \mathbf{q}^m$*, where* $\mathbf{q} \in \Sigma^l$ *for some integers* $m > 1$ *and* $l > 0$*.*

Corollary 6.1.2. *For any* $a \in \Sigma$*, a uniform configuration of the form* $\mathbf{s} = a^N$ *is symmetric.*

Corollary 6.1.3. *For any symmetric configuration* $\mathbf{s} = \mathbf{q}^m \in \Sigma^N$*, it must be the case that* $m \mid N$*.*

Remark: Since our CA's topology is cyclic, a symmetric configuration might have the form $\mathbf{s} = \mathbf{bq}^{m-1}\mathbf{a}$, where $\mathbf{ab} = \mathbf{q}$. Such a configuration $\mathbf{s}$ is, however, equal to $\mathbf{ba} \ldots \mathbf{ba}$, hence $\mathbf{s} = \mathbf{r}^m$, where $\mathbf{r} = \mathbf{ba}$, which is a symmetric configuration in the original non-cyclic form.

Lemma 6.1.4. *The neighborhoods in a symmetric configuration* $\mathbf{s} = \mathbf{q}^m \in \Sigma^N$ *are symmetric. Specifically, for any* $i \in \{0, \ldots, N-1\}$*, the neighborhoods in* $\mathbf{s}$ *satisfy*

$$\eta_i = \eta_{i+l(\mathrm{mod\,N})},$$

where $l = N/m$*.*

Proof Suppose there exists an i for which $\eta_i \neq \eta_{i+l}$, (mod N), and suppose η_i has radius r in a symmetric configuration $\mathbf{s} = \mathbf{q}^m$. Then (with indices taken mod N),

$$\eta_i = (s_{i-r}, \ldots, s_{i+r}) \neq (s_{i+l-r}, \ldots, s_{i+l+r}) = \eta_{i+l}$$

and there exists some j ($-r \leq j \leq r$) such that $s_{i+j} \neq s_{i+l+j}$, which contradicts the assumption that $\mathbf{s}$ is symmetric. □

Lemma 6.1.5. *If a configuration* $\mathbf{s}$ *is symmetric then* $\Phi(\mathbf{s})$ *is symmetric, for any uniform global transition rule* Φ.

Proof By Lemma 6.1.4,

$$\Phi(\mathbf{s}) = \Phi(\mathbf{q}^m) = (\phi(\eta_0)\dots\phi(\eta_{l-1}))^m.$$

□

Theorem 6.1.6. *Leader election from a symmetric configuration is not possible.*

Proof Let $1 \in \Sigma$ be the leader state and $\mathbf{s} = \mathbf{q}^m$ for some $m > 1$ be a symmetric configuration. Using Lemma 6.1.5, $\Phi^n(\mathbf{s})$ is symmetric for any positive integer n, so that $\Phi^n(\mathbf{s}) = \mathbf{w}^m$. Let the number of 1-occurrences in $\Phi^n(\mathbf{s})$ be k. Trivially, $k =_{\#_1} \Phi^n(\mathbf{s}) = m_{\#_1}\mathbf{w}$, and therefore $m \mid k$. Since $m > 1$, either $k = 0$ or $k > 1$ and ${}_{\#_1}\Phi^n(\mathbf{s}) \neq 1$, which violates the requirement of leader election that only one cell is active in the final configuration. □

Now, we will enumerate all symmetric configurations of size N. Note that all theorems and proofs assume we have a one-dimensional CA.

Definition 6.1.2. *We denote by* $S_N(l) = \{\mathbf{q}^m | \mathbf{q} \in \Sigma^l, N = lm\}$ *the set of all symmetric configurations of length N over the alphabet* Σ *with pattern size l.*

Corollary 6.1.7. *For any* $l|N$, *we have* $|S_N(l)| = |\Sigma^l| = |\Sigma|^l$.

Lemma 6.1.8. $S_N(l_1) \cap S_N(l_2) \subseteq S_N(\gcd(l_1, l_2))$

Proof Let $\mathbf{s} \in S_N(l_1) \cap S_N(l_2)$, so that $s_i = s_j$ whenever $i \equiv j \pmod{l_1}$ or $i \equiv j \pmod{l_2}$. Let $d = \gcd(l_1, l_2)$ and recall $\exists m, n \in \mathbb{Z}$ with $d = ml_1 + nl_2$. If $i \equiv j \pmod{}$

d) then $j = i + qd$ for some $q \in \mathbb{Z}$. Thus

$$s_j = s_{i+qml_1+qnl_2} = s_{i+qml_1} = s_i,$$

and $\mathbf{s} \in S_N(d)$ as desired. □

Lemma 6.1.9. $S_N(l_1) \subseteq S_N(l_2) \iff l_1 \mid l_2$.

Proof ($\Rightarrow$). Suppose $S_N(l_1) \subseteq S_N(l_2)$, and let $d = \gcd(l_1, l_2)$. Then

$$S_N(l_1) = S_N(l_1) \cap S_N(l_2) \subseteq S_N(d),$$

by Lemma 6.1.8. So, by Corollary 6.1.7, we must have $l_1 \leq d$. But certainly $d|l_1$, so $d = l_1$, which implies $l_1|l_2$.

($\Leftarrow$). Let $l_2 = cl_1$ and $\mathbf{s} \in S_N(l_1)$. There exists $\mathbf{q_1} \in \Sigma^{l_1}$ such that $\mathbf{s} = \mathbf{q_1}^{N/l_1}$. Now, let $\mathbf{q_2} = \mathbf{q_1}^c$. Since $\mathbf{q_2} \in \Sigma^{l_2}$ and $l_2 \mid N$, $\mathbf{q_2} \in S_N(l_2)$. □

Lemma 6.1.10. $S_N(\gcd(l_1, l_2)) = S_N(l_1) \cap S_N(l_2)$

Proof Let $d = \gcd(l_1, l_2)$. Since $d|l_1$ and $d|l_2$, Lemma 6.1.9 implies $S_n(d) \subseteq S_N(l_1)$ and $S_N(d) \subseteq S_N(l_2)$, forcing $S_N(d) \subseteq S_N(l_1) \cap S_N(l_2)$. The reverse inclusion holds by Lemma 6.1.8. □

Corollary 6.1.11.

$$|S_N(l_1) \cup S_N(l_2)| = |S_N(l_1)| + |S_N(l_2)| - |S_N(\gcd(l_1, l_2))|.$$

Proof Immediate by Lemma 6.1.10 and inclusion-exclusion. □

Definition 6.1.3. *We denote by S_N the set of all symmetric configurations of length N over the alphabet Σ, so that*

$$S_N = \bigcup_{l|N, l<N} S_N(l).$$

Lemma 6.1.12. *Let $N = \prod_{i=1}^{\omega(N)} p_i^{\alpha_i}$ be the prime factorization of N, where $\omega(N)$ denotes the number of distinct prime factors. Then*

$$S_N = \bigcup_{i=1}^{\omega(N)} S_N\left(N/p_i\right).$$

Proof ($\subseteq$). Let $\mathbf{s} \in S_N$, so that $s \in S_N(l)$ for some $l < N$ with $l|N$. We may write $l = \prod_{i=1}^{\omega(N)} p_i^{\beta_i}$, where $\beta_i \leq \alpha_i$ for each i. Since $l < N$ there must be some j such that $\beta_j < \alpha_j$, and therefore $l|(N/p_j)$. By Lemma 6.1.9 $S_N(l) \subseteq S_N(N/p_j)$, so $\mathbf{s} \in S_N(N/p_j)$.

($\supseteq$). Immediate by Definition 6.1.3. □

Theorem 6.1.13. *Let $N = \prod_{i=1}^{\omega(N)} p_i^{\alpha_i}$ be the prime factorization of N, where $\omega(N)$ denotes the number of distinct prime factors. Then*

$$|S_N| = \sum_{i=1}^{\omega(N)} (-1)^{i+1} \sum_{\substack{J \subseteq \{1,\ldots,\omega(N)\} \\ |J|=i}} |\Sigma|^{N/\prod_{j\in J} p_j}$$

Proof By Lemma 6.1.12 and the inclusion-exclusion principle

$$\begin{aligned} |S_N| &= \left| \bigcup_{i=1}^{\omega(N)} S_N\left(N/p_i\right) \right| \\ &= \sum_{i=1}^{\omega(N)} (-1)^{i+1} \sum_{\substack{J \subseteq \{1,\ldots,\omega(N)\} \\ |J|=i}} \left| \bigcap_{j\in J} S_N\left(N/p_j\right) \right| \end{aligned}$$

By Lemma 6.1.10, for each $J \subseteq \{1, ..., \omega(N)\}$ we have

$$\bigcap_{j \in J} S_N\left(N/p_j\right) = S_N(d)$$

where $d = \gcd\left\{N/p_j\right\}_{j\in J} = N/\prod_{j\in J} p_j$. Finally, using Corollary 6.1.7

$$|S_N(d)| = |\Sigma|^{N/\prod_{j\in J} p_j}.$$

□

Example: Let $N = p_1^{\alpha_1} p_2^{\alpha_2}$, then $|S_N| = |\Sigma|^{\frac{N}{p_1}} + |\Sigma|^{\frac{N}{p_2}} - |\Sigma|^{\frac{N}{p_1 p_2}}$. For example, when $N = 24$ and $|\Sigma| = 2$ we have $|S_{24}| = 2^{12} + 2^8 - 2^4 = 4336$.

Example: Let $N = p_1^{\alpha_1} p_2^{\alpha_2} p_3^{\alpha_3}$, then $|S_N| = |\Sigma|^{\frac{N}{p_1}} + |\Sigma|^{\frac{N}{p_2}} + |\Sigma|^{\frac{N}{p_3}} - |\Sigma|^{\frac{N}{p_1 p_2}} - |\Sigma|^{\frac{N}{p_1 p_3}} - |\Sigma|^{\frac{N}{p_2 p_3}} + |\Sigma|^{\frac{N}{p_1 p_2 p_3}}$. For example, when $N = 60$ and $|\Sigma| = 2$ we have $|S_{60}| = 2^{30} + 2^{20} + 2^{12} - 2^{10} - 2^6 - 2^4 + 2^2 = 1074793396$.

Definition 6.1.4. *For any state $a \in \Sigma$ and $N, k \in \mathbb{N}$, we define the set $D^a_{N,k}$ to be the set of all configurations with exactly k sites in state a:*

$$D^a_{N,k} = \{\mathbf{s} \in \Sigma^N \,|\, {}_{\#_a}\mathbf{s} = k\}.$$

Accordingly, we denote by $S^a_{N,k}$ the set of such configurations that are symmetric, so that

$$S^a_{N,k} = S_N \cap D^a_{N,k}.$$

And for any $l \in \mathbb{N}$, $S^a_{N,k}(l)$ denotes the set of those configurations in $S^a_{N,k}$ that have

pattern size l, so that

$$S^a_{N,k}(l) = S_N(l) \cap D^a_{N,k}.$$

Corollary 6.1.14. *For any state $a \in \Sigma$ and any $N, k, l \in \mathbb{N}$,*

$$S^a_{N,k}(l) \neq \emptyset$$

iff $\frac{N}{l}$ is an integer that divides k.

Lemma 6.1.15. *For a given state $a \in \Sigma$, and m such that $m \mid N$ and $m \mid k$,*

$$\left|S^a_{N,k}\left(\frac{N}{m}\right)\right| = \binom{\frac{N}{m}}{\frac{k}{m}}(|\Sigma| - 1)^{\frac{N-k}{m}}$$

Proof Let $\mathbf{s} \in S^a_{N,k}(\frac{N}{m})$. Then $\mathbf{s} = \mathbf{q}^m$ for some $\mathbf{q} \in \Sigma^{\frac{N}{m}}$, and so $\#_a\mathbf{q} = k/m$. To enumerate the number of such configurations, we first have to choose k/m of the N/m sites in $\mathbf{q}$ to be in state a, and then fill the remaining $N/m - k/m$ sites of $\mathbf{q}$ with states from $\Sigma \setminus \{a\}$. □

Theorem 6.1.16. *Pick $N, k \in \mathbb{N}$ with $k \leq N$ and let $d = \gcd(k, N)$. Let $N = \prod_{i=1}^{\omega(N)} p_i^{\alpha_i}$, $k = \prod_{i=1}^{\omega(k)} q_i^{\beta_i}$, and $d = \prod_{i=1}^{\omega(d)} r_i^{\gamma_i}$ be the prime factorizations of N, k, d, respectively. Then for any $a \in \Sigma$,*

$$|S^a_{N,k}| = \sum_{i=1}^{\omega(d)} (-1)^{i+1} \sum_{\substack{J \subseteq \{1,\ldots,\omega(d)\} \\ |J|=i}} \binom{\frac{N}{\prod_{j\in J} r_j}}{\frac{k}{\prod_{j\in J} r_j}} (|\Sigma| - 1)^{(N-k)/\prod_{j\in J} r_j}.$$

Proof Using Definition 6.1.4, Lemma 6.1.12, and Corollary 6.1.14,

$$\begin{aligned} S^a_{N,k} &= \left(\bigcup_{i=1}^{\omega(N)} S_N(N/p_i)\right) \bigcap D^a_{N,k} \\ &= \bigcup_{i=1}^{\omega(N)} S^a_{N,k}(N/p_i) \\ &= \bigcup_{i=1}^{\omega(d)} S^a_{N,k}(N/r_i). \end{aligned}$$

By the inclusion-exclusion principle

$$|S^a_{N,k}| = \sum_{i=1}^{\omega(d)} (-1)^{i+1} \sum_{\substack{J \subseteq \{1,\ldots,\omega(d)\} \\ |J|=i}} \left| \bigcap_{j \in J} S^a_{N,k}(N/r_j) \right|.$$

Now, by Lemma 6.1.10

$$\begin{aligned} \bigcap_{j \in J} S^a_{N,k}(N/r_j) &= \left(\bigcap_{j \in J} S_N(N/r_j) \right) \cap D^a_{N,k} \\ &= S_N(\gcd\{N/r_j \mid j \in J\}) \cap D^a_{N,k} \\ &= S_N(N/\textstyle\prod_{j \in J} r_j) \cap D^a_{N,k} \end{aligned}$$

Finally using Lemma 6.1.15,

$$|S_N(N/\textstyle\prod_{j \in J} r_j) \cap D^a_{N,k}| = \left(\dfrac{N}{\prod_{j \in J} r_j} \right) (|\Sigma| - 1)^{(N-k)/\prod_{j \in J} r_j}.$$

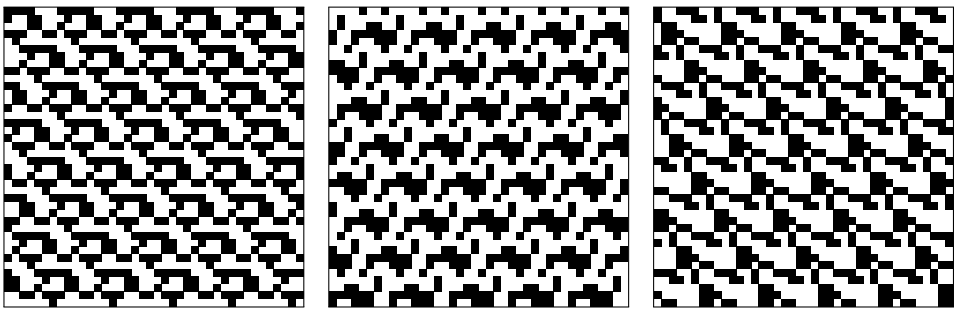
□

Corollary 6.1.17. *For any state set Σ and any state $a \in \Sigma$, the set $S^a_{N,0}$ equals the set S_N for the state set $\Sigma \setminus \{a\}$.*

Corollary 6.1.18. *The number of binary symmetric configurations ($|\Sigma| = 2$) with k sites in state a is given by*

$$|S^a_{N,k}| = \sum_{i=1}^{\omega(d)} (-1)^{i+1} \sum_{\substack{J \subseteq \{1,\ldots,\omega(d)\} \\ |J|=i}} \binom{\frac{N}{\prod_{j\in J} r_j}}{\frac{k}{\prod_{j\in J} r_j}}.$$

Example: Let $N = p_1^{\alpha_1} p_2^{\alpha_2}$, and $k = p_1 p_2$. Then for any $a \in \Sigma$, we have $|S^a_{N,k}| = \binom{\frac{N}{p_1}}{\frac{k}{p_1}}(|\Sigma|-1)^{\frac{N-k}{p_1}} + \binom{\frac{N}{p_2}}{\frac{k}{p_2}}(|\Sigma|-1)^{\frac{N-k}{p_2}} - \binom{\frac{N}{p_1 p_2}}{\frac{k}{p_1 p_2}}(|\Sigma|-1)^{\frac{N-k}{p_1 p_2}}$. For example, when $N = 24$, $k = 6$, and $|\Sigma| = 2$, we have $|S^a_{24,6}| = \binom{12}{3} + \binom{8}{2} - \binom{4}{1} = 244$.



(a) t_0 (b) t_1 (c) t_2

Figure 6.2: Space-time diagrams of CA computation on a two-dimensional symmetric configuration showing a lattice at three consecutive time steps. Note that leader election from a symmetric configuration is impossible.

6.1.2 Two-Dimensional Symmetric Configurations

Compared to the one-dimensional case, two dimensions allow more possibilities for symmetries. In this section we generalize the concept of configuration symmetry by employing vector projections and group theory. In fact, the formulas and methodology we use to enumerate two-dimensional symmetric configurations

could be easily extended to arbitrary many dimensions. Also, we deal with two-dimensional square configurations only, but we suggest most of the Lemmas and Theorems could be extended to incorporate arbitrary rectangle shapes. For consistency, however, we leave the rectangle extension for future consideration. Figure 6.2 shows a CA computation on a two-dimensional symmetric configuration.

Definition 6.1.5. *For a non-zero vector (pattern shift) $v \in \mathbb{Z}_n \times \mathbb{Z}_n$ we denote by*

$$S_{n\times n}(v) = \{\mathbf{s} \in \Sigma^{n\times n} \mid \forall u \in \mathbb{Z}_n \times \mathbb{Z}_n : s_u = s_{u\oplus v}\}$$

the set of all symmetric square configurations of size $N = n^2$ over the alphabet Σ, where $\oplus$ is a coordination-wise addition operation defined on $\mathbb{Z}_n$.

It means that any projection by a non-zero vector v defines a configuration symmetry. Having that, we can bridge symmetric configurations with the group theory. From now on we will call a vector we use for state projection a group generator.

Corollary 6.1.19. *For a non-zero vector (generator) $v = (l_1, l_2) \in \mathbb{Z}_n \times \mathbb{Z}_n$*

$$S_{n\times n}(v) = \{\mathbf{s} \in \Sigma^{n\times n} \mid \forall u \in \mathbb{Z}_n \times \mathbb{Z}_n : \forall w \in \langle v\rangle : s_u = s_{u\oplus w}\},$$

where $\langle v\rangle$ is a cyclic subgroup $\{(c\,l_1, c\,l_2) \mid c \in \mathbb{N}\}$ over additive group $\mathbb{Z}_n \times \mathbb{Z}_n$, i.e. $\langle v\rangle \leq \mathbb{Z}_n \times \mathbb{Z}_n$. Trivially, the order of $\langle v\rangle$, written as $|\langle v\rangle|$ divides $|\mathbb{Z}_n \times \mathbb{Z}_n| = n^2$.

Lemma 6.1.20. $|S_{n\times n}(l_1, l_2)| = |\Sigma|^{n\,\gcd(l_1,l_2,n)}$.

Proof Since a vector (l_1, l_2) applied to a state fixes $|\langle(l_1, l_2)\rangle|$ states in total, the number of state choices and therefore also the size of each symmetric propagated

pattern is $|S_{n\times n}(l_1, l_2)| = \frac{n^2}{|\langle(l_1,l_2)\rangle|}$. Furthermore, for $l \in \mathbb{Z}_n$, $|\langle l\rangle| = \frac{n}{\gcd(l,n)}$, so

$$|\langle l_1, l_2\rangle| = \text{lcm}(\frac{n}{\gcd(l_1, n)}, \frac{n}{\gcd(l_2, n)}) = \frac{n}{\gcd(l_1, l_2, n)},$$

where lcm is a least common multiplier. Finally,

$$|S_{n\times n}(l_1, l_2)| = \frac{n^2}{\frac{n}{\gcd(l_1,l_2,n)}} = n\gcd(l_1, l_2, n)$$

□

Remark: Note that if $l_1 = 0$ (or $l_2 = 0$), $\gcd(0, n) = n$, and $|\langle(0, l_2)\rangle| = |\langle l_2\rangle|$, which is a special one-dimensional case we handled in the previous section.

Lemma 6.1.21. *The neighborhoods in a symmetric square configuration defined by any non-zero vector $v \in \mathbb{Z}_n\times\mathbb{Z}_n$ are symmetric. Specifically, for any $w \in \mathbb{Z}_n\times\mathbb{Z}_n$, the neighborhoods satisfy*

$$\eta_w = \eta_{w\oplus v},$$

where $\oplus$ is addition defined on $\mathbb{Z}_n$.

Proof Suppose the neighborhood function is defined by (relative) vectors $u_1, \ldots, u_m$, i.e., $\eta_w = (s_{w\oplus u_1}, \ldots, s_{w\oplus u_m})$ and assume the Lemma does not hold, i.e., there exists w for which $\eta_w \neq \eta_{w\oplus v}$. Then,

$$\eta_w = (s_{w\oplus u_1}, \ldots, s_{w\oplus u_m}) \neq (s_{(w\oplus v)\oplus u_1}, \ldots, s_{(w\oplus v)\oplus u_m}) = \eta_{w\oplus v}$$

and so there exists some u_j such that $s_{w\oplus u_j} \neq s_{(w\oplus v)\oplus u_j}$, i.e., $s_{w\oplus u_j} \neq s_{(w\oplus u_j)\oplus v}$, which contradicts the assumption that **s** is symmetric by vector v. □

Lemma 6.1.22. *If symmetry of a square configuration* $\mathbf{s}$ *is defined by vector* v *then* $\Phi(\mathbf{s})$ *is also* v*-symmetric, for any uniform global transition rule* Φ*.*

Proof Suppose $\mathbf{q} = \Phi(\mathbf{s})$ is not symmetric by v. Then, there exists $u \in \mathbb{Z}^n \times \mathbb{Z}^n$, such that $q_u \neq q_{u\oplus v}$. By Lemma 6.1.21, $\eta_{s_u} = \eta_{s_{u\oplus v}}$, and so

$$q_u = \phi(\eta_{s_u}) = \phi(\eta_{s_{u\oplus v}}) = q_{u\oplus v},$$

which is a contradiction. □

Theorem 6.1.23. *Leader election from a symmetric square configuration* $\mathbf{s}$ *is not possible.*

Proof Let $1 \in \Sigma$ be the leader state and $\mathbf{s}$ be a symmetric configuration defined by vector $v = (l_1, l_2)$. Using Lemma 6.1.22, $\mathbf{q} = \Phi^m(\mathbf{s})$ is also symmetric by vector v for any positive integer m. Let the number of 1-occurrences in $\mathbf{q}$ be k. Trivially, $|\langle v\rangle|$ divides k. Since $v \neq (0,0)$, $|\langle v\rangle| = \frac{n}{\gcd(l_1,l_2,n)} > 1$, and either $k = 0$ or $k > 1$. Since $k \neq 1$, the requirement of leader election that only one cell is active in the final configuration is violated. □

Now, we will enumerate all symmetric square two-dimensional configurations of size $N = n^2$. To further investigate the relations among symmetric configurations we need to define the symmetric configurations over several generators.

Definition 6.1.6. *We denote by*

$$S_{n\times n}\mathbb{L} = \{\mathbf{s} \in \Sigma^{n\times n} \mid \forall u \in \mathbb{Z}_n \times \mathbb{Z}_n,\ \forall v \in \langle \mathbb{L}\rangle : s_u = s_{u\oplus v}\}$$

the set of all symmetric square configurations of size $N = n^2$ over the alphabet Σ with generator set $\mathbb{L} \subseteq \mathbb{Z}_n \times \mathbb{Z}_n$, where $\oplus$ is a coordinate-wise addition operation defined on $\mathbb{Z}_n$, and $\langle \mathbb{L} \rangle = \{c_1 v_1 \oplus \ldots \oplus c_{|\mathbb{L}|} v_{|\mathbb{L}|} \,|\, c_i \in \mathbb{Z}_n\}$

Corollary 6.1.24. $|S_{n\times n}\mathbb{L}| = \frac{n^2}{|\langle \mathbb{L} \rangle|}$.

Corollary 6.1.25. *For vectors $v_1, v_2 \in \mathbb{Z}_n \times \mathbb{Z}_n$*

$$S_{n\times n}(v_1) \cap S_{n\times n}(v_2) = S_{n\times n}\{v_1, v_2\}.$$

Lemma 6.1.26. *For two vectors $v_1, v_2 \in \mathbb{Z}_n \times \mathbb{Z}_n$*

$$|\langle v_1, v_2 \rangle| = \frac{|\langle v_1 \rangle||\langle v_2 \rangle|}{|\langle v_1 \rangle \cap \langle v_2 \rangle|}.$$

Proof By basic linear algebra $\langle v_1, v_2 \rangle = \{c_1 v_1 + c_1 v_2 \,|\, c_1, c_2 \in \mathbb{Z}_n\}$, hence there are $|\langle v_1 \rangle||\langle v_2 \rangle|$ selections of vectors from $\langle v_1 \rangle$ and $\langle v_2 \rangle$. However, each vector from $\langle v_1, v_2 \rangle$ is included $|\langle v_1 \rangle \cap \langle v_2 \rangle|$ times. □

Corollary 6.1.27. $|S_{n\times n}(v_1) \cap S_{n\times n}(v_2)| = \frac{n^2 |\langle v_1 \rangle \cap \langle v_2 \rangle|}{|\langle v_1 \rangle||\langle v_2 \rangle|}$.

Corollary 6.1.28. $|S_{n\times n}(v_1) \cup S_{n\times n}(v_2)| =$

$$|S_{n\times n}(v_1)| + |S_{n\times n}(v_2)| - |S_{n\times n}(v_1) \cap S_{n\times n}(v_2)| = n^2\left(\frac{1}{|\langle v_1 \rangle|} + \frac{1}{|\langle v_1 \rangle|} - \frac{|\langle v_1 \rangle \cap \langle v_2 \rangle|}{|\langle v_1 \rangle||\langle v_2 \rangle|}\right)$$

.

Lemma 6.1.29. $S_{n\times n}(v_1) \subseteq S_{n\times n}(v_2) \iff \langle v_1 \rangle \geq \langle v_2 \rangle$.

Proof ($\Rightarrow$). Suppose $S_{n\times n}(v_1) \subseteq S_{n\times n}(v_2)$. Then $S_{n\times n}(v_1) = S_{n\times n}(v_1) \cap S_{n\times n}(v_2) = S_{n\times n}\{v_1, v_2\}$ by Corollary 6.1.25. Then $|\langle v_1 \rangle| = |\langle v_1, v_2 \rangle|$, and therefore $\langle v_2 \rangle$ does not add any extra element and so it must be a subgroup $\langle v_1 \rangle \geq \langle v_2 \rangle$ as desired.

($\Leftarrow$). Suppose $S_{n\times n}(v_1) \not\subseteq S_{n\times n}(v_2)$ and $\langle v_1\rangle \geq \langle v_2\rangle$. Let $\mathbf{s} \in S_{n\times n}(v_1)$ such that $\mathbf{s} \notin S_{n\times n}(v_2)$, hence $\mathbf{s}$ is symmetric under v_1, but not under v_2. Hence, there exists s_u such that $s_u \neq s_{u\oplus v_2}$. However, since $\langle v_2\rangle$ is a subgroup of $\langle v_1\rangle$, $v_2 \in \langle v_1\rangle$, and because $\mathbf{s} \in S_{n\times n}(v_1)$, $s_{u\oplus} = s_{u\oplus v_2}$, which is a contradiction. □

Definition 6.1.7. *We denote by $S_{n\times n}$ the set of all square symmetric configurations of length $N = n^2$ over the alphabet Σ, so that*

$$S_{n\times n} = \bigcup_{v\in\mathbb{Z}_n\times\mathbb{Z}_n} S_{n\times n}(v).$$

Lemma 6.1.30. *For a prime p that divides n and $0 \leq i < n$, a cyclic group $\langle(\frac{n}{p}, i\frac{n}{p})\rangle$ is minimal, i.e., its only strict subgroup is trivial* $(0, 0)$.

Proof Since each subgroup of a cyclic group is cyclic the order of group must be divisible by the order of subgroup. Hence, for $\langle v\rangle \leq \langle(\frac{n}{p}, i\frac{n}{p})\rangle$

$$\begin{aligned} |\langle v\rangle| \quad &\text{divides} \quad \left|\left\langle\left(\frac{n}{p}, i\frac{n}{p}\right)\right\rangle\right| \\ |\langle v\rangle| \quad &\text{divides} \quad \frac{n}{\gcd(\frac{n}{p}, i\frac{n}{p}, n)} = p \end{aligned}$$

Therefore, the order of a subgroup $\langle v\rangle$ is either 1 or p, i.e. $\langle v\rangle$ is either $\langle(0, 0)\rangle$ or $\langle(\frac{n}{p}, i\frac{n}{p})\rangle$. □

Remark: By swapping the coordinates, the proof applies also to each subgroup of the form $\langle(i\frac{n}{p}, \frac{n}{p})\rangle$.

Lemma 6.1.31. *Let $n = \prod_{i=1}^{\omega(n)} p_i^{\alpha_i}$ be the prime factorization of n, where $\omega(n)$ denotes the number of distinct prime factors. Then*

$$S_{n\times n} = \bigcup_{w\in G_n} S_{n\times n}(w),$$

where G_n is the set of generators

$$G_n = \bigcup_{j=1}^{\omega(n)} G_n(p_j)$$

and

$$G_n(p) = \bigcup_{i=0}^{p-1}\left(\frac{n}{p}, i\frac{n}{p}\right)\bigcup\left\{\left(0, \frac{n}{p}\right)\right\}.$$

Proof ($\subseteq$). Let $\mathbf{s} \in S_{n\times n}$, so that $\mathbf{s} \in S_{n\times n}(v)$ for some $v = (a, b) \in \mathbb{Z}_n \times \mathbb{Z}_n$. We will show that for some $w \in G_n$, $\langle w\rangle \leq \langle v\rangle$, which by using Lemma 6.1.29 $S_{n\times n}(v) \subseteq S_{n\times n}(w)$ and $\mathbf{s} \in S_{n\times n}(w)$.

Now, we have two options to consider:

- Let $\gcd(a, n) \neq 1$. We assume $\gcd(a, b, n) = 1$ (if not we can divide all the numbers by the gcd first). We pick a vector $v' = (0, b\frac{n}{a})$, which generates a subgroup of $\langle v\rangle$. Because $\gcd(a, b, n) = 1$, $b\frac{n}{a}$ is non-zero in $\mathbb{Z}_n$ and there exists a prime p such that $w = (0, \frac{n}{p})$, a generator from $G_n(p)$, generates a non-trivial subgroup of $\langle v'\rangle$ and transitively $\langle w\rangle \leq \langle v\rangle$ as required.

- Let $\gcd(a, n) = 1$. Let $i = \frac{b}{a}$ in modular $\mathbb{Z}_n$ arithmetic, and p be a prime of n. Then the vectors v and $w = (\frac{n}{p}, i\frac{n}{p}) \in G_n(p)$ are linearly dependent since the determinant

$$|v\, w| = \begin{vmatrix} a & \frac{n}{p} \\ b & i\frac{n}{p} \end{vmatrix} = 0.$$

Therefore,

$$\langle v\rangle \cap \langle w\rangle = \langle \gcd(n, \operatorname{lcm}(a, \frac{n}{p})), \gcd(n, \operatorname{lcm}(b, i\frac{n}{p}))\rangle,$$

and also

$$|\langle v\rangle \cap \langle w\rangle| = \gcd(|\langle v\rangle|, |\langle w\rangle|).$$

Now by Lemma 6.1.30

$$|\langle v\rangle \cap \langle w\rangle| = \gcd(|\langle v\rangle|, p).$$

Since $|\langle v\rangle|$ divides n there exists p_j such that p_j divides $|\langle v\rangle|$, and so $\gcd(|\langle v\rangle|, p_j) = p_j$. That means $\langle v\rangle \cap \langle w\rangle = \langle w\rangle$ and $\langle w\rangle \leq \langle v\rangle$.

($\supseteq$). Immediate by Definition 6.1.7. □

Corollary 6.1.32.

$$|G_n| = \sum_{i=1}^{\omega(n)} p_i + \omega(n).$$

Lemma 6.1.33. *The generators from G_n are mutually linearly independent.*

Proof Let $u \in G_n(p)$, $v \in G_n(q), u \neq v$, then $|\langle u\rangle| = p$ and $|\langle v\rangle| = q$. If $p \neq q$ then $\langle u\rangle \cap \langle v\rangle = \langle (0,0)\rangle$, since p and q are primes. For $p = q$, we have two options:

- Let $u = (\frac{n}{p}, i\frac{n}{p})$ and $v = (\frac{n}{p}, j\frac{n}{p})$ for $i \neq j$. If u and v were linearly dependent in $\mathbb{Z}_n \times \mathbb{Z}_n$, $u' = (1, i)$ and $v' = (1, j)$ would be linearly dependent in $\mathbb{Z}_p \times \mathbb{Z}_p$, and so the determinant $|u'\, v'| = (j - i) = 0$. That is however not possible since $i, j < p$ and $i \neq j$.

- Let $u = (0, \frac{n}{p})$ and $v = (\frac{n}{p}, j\frac{n}{p})$. Since $\frac{n}{p} \neq 0$ the vectors u and v are clearly linearly independent.

Therefore any pair of generators from G_n is linearly independent as required. □

Lemma 6.1.34. *For $u, v \in G_n(p), u \neq v$,*

$$\langle u, v\rangle = \left\langle \left(\frac{n}{p}, 0\right), \left(0, \frac{n}{p}\right)\right\rangle$$

and

$$|\langle u, v\rangle| = p^2$$

Proof ($\subseteq$). Let pick $i \neq j$, and assign $u = (\frac{n}{p}, i\frac{n}{p})$ and $v = (\frac{n}{p}, j\frac{n}{p})$. Then $u = (\frac{n}{p}, 0) + i(0, \frac{n}{p}) \in \langle(\frac{n}{p}, 0), (0, \frac{n}{p})\rangle$. Similarly, $v = (\frac{n}{p}, 0) + j(0, \frac{n}{p}) \in \langle(\frac{n}{p}, 0), (0, \frac{n}{p})\rangle$. Trivially also the case where $u = (\frac{n}{p}, i\frac{n}{p})$ and $v = (0, \frac{n}{p})$ holds for the same reason.

($\supseteq$). Let pick $i \neq j$, and assign $u = (\frac{n}{p}, i\frac{n}{p})$ and $v = (\frac{n}{p}, j\frac{n}{p})$. Then $u - v = (0, (i-j)\frac{n}{p}) \in \langle u, v\rangle$ as well as $ju - iv = ((j-i)\frac{n}{p}, 0) \in \langle u, v\rangle$. Therefore also $(\frac{n}{p}, 0) \in \langle u, v\rangle$, and $(0, \frac{n}{p}) \in \langle u, v\rangle$ since the order of $u - v$ and $ju - iv$ is p. If $u = (\frac{n}{p}, i\frac{n}{p})$ and $v = (0, \frac{n}{p})$, $u - iv = (\frac{n}{p}, 0) \in \langle u, v\rangle$. □

Theorem 6.1.35. *Let $n = \prod_{i=1}^{\omega(n)} p_i^{\alpha_i}$ be the prime factorization of n, where $\omega(n)$ denotes the number of distinct prime factors. Then*

$$|S_{n\times n}| = \sum_{\substack{0\leq l_1\leq(p_1+1)\\ \cdots \\ 0\leq l_{\omega(n)}\leq(p_{\omega(n)}+1)}} (-1)^{1+\sum_{i=1}^{\omega(n)} l_i} \left(\prod_{i=1}^{\omega(n)} \binom{p_i+1}{l_i}\right) |\Sigma|^{\frac{n^2}{\prod_{i=1}^{\omega(n)} p_i^{\min(l_i,2)}}}.$$

Proof By Lemma 6.1.31, the inclusion-exclusion principle, and Corollary 6.1.25

$$|S_{n\times n}| = \left|\bigcup_{w\in G_n} S_{n\times n}(w)\right| = \sum_{i=1}^{|G_n|}(-1)^{i+1}\sum_{\substack{J\subseteq G_n\\|J|=i}}\left|\bigcap_{j\in J} S_{n\times n}(j)\right| = \sum_{J\subseteq G_n}(-1)^{|J|+1}\,|S_{n\times n}J|$$

Since $G_n = \bigcup_{j=1}^{\omega(n)} G_n(p_j)$, we have $\omega(n)$ sets to choose the elements of J from, so

$$|S_{n\times n}| = \sum_{\substack{J_1\subseteq G_n(p_1)\\ \cdots\\ J_{\omega(n)}\subseteq G_n(p_{\omega_n})}} (-1)^{1+\sum_{i=1}^{\omega(n)}|J_i|}\left|S_{n\times n}(\bigcup_{i=1}^{\omega(n)} J_i)\right|$$

Since $J_i \subseteq G_n(p_i)$, by Lemma 6.1.34 $\langle J_i\rangle = \langle(\frac{n}{p_i},0),(0,\frac{n}{p_i})\rangle$ for $|J_i|\geq 2$. So for $i\neq j$, $\langle J_i\rangle$ and $\langle J_j\rangle$ are linearly independent for any size $|J_i|$ and $|J_j|$. Since $|\langle J_i\rangle| = p_i$ for $|J_i| = 1$ and $|\langle J_i\rangle| = p_i^2$ for $|J_i|\geq 2$

$$\left|\left\langle\bigcup_{i=1}^{\omega(n)} J_i\right\rangle\right| = \left|\left\langle\bigcup_{i=1}^{\omega(n)}\langle J_i\rangle\right\rangle\right| = \prod_{i=1}^{\omega(n)}|\langle J_i\rangle| = \prod_{i=1}^{\omega(n)} p_i^{\min(|J_i|,2)}$$

When we plug in the expression for $|\langle\cup_{i=1}^{\omega(n)} J_i\rangle|$, we obtain

$$|S_{n\times n}| = \sum_{\substack{J_1\subseteq G_n(p_1)\\ \cdots\\ J_{\omega(n)}\subseteq G_n(p_{\omega_n})}} (-1)^{1+\sum_{i=1}^{\omega(n)}|J_i|}\left(|\Sigma|^{\frac{n^2}{\prod_{i=1}^{\omega(n)} p_i^{\min(|J_i|,2)}}}\right)$$

Now, because the content of J_i is irrelevant and we care only about the cardinality $|J_i|$, for each size $l_i = |J_i|$ we have $\binom{|G_n(p_i)|}{l_i} = \binom{p_i+1}{l_i}$ ways of choosing l_i elements from $G_n(p_i)$, which produces the final formula as required. □

Lemma 6.1.36. *Let $n = \prod_{i=1}^{\omega(n)} p_i^{\alpha_i}$ be the prime factorization of n, where $\omega(n)$ denotes the number of distinct prime factors. Then an alternative counting of*

$|S_{n\times n}|$ *is*

$$|S_{n\times n}| = \sum_{\substack{0\le k_1\le 2\\ \cdots\\ 0\le k_{\omega(n)}\le 2}} |\Sigma|^{\frac{n^2}{\prod_{i=1}^{\omega(n)} p_i^{k_i}}} \left(\sum_{\substack{k_1\le l_1\le \mathrm{top}(k_1)\\ \cdots\\ k_{\omega(n)}\le l_{\omega(n)}\le \mathrm{top}(k_{\omega(n)})}} (-1)^{1+\sum_{i=1}^{\omega(n)} l_i} \prod_{i=1}^{\omega(n)} \binom{p_i+1}{l_i} \right),$$

where

$$\mathrm{top}(k_i) = \begin{cases} k_i & \text{if } k_i < 2 \\ p_i + 1 & \text{if } k_i = 2. \end{cases}$$

Proof We know that the exponent of each p_i in $S_{n\times n}$ from Theorem 6.1.35 is at most 2. Therefore for given $k_1, \ldots, k_{\omega(n)} \in \{0, 1, 2\}$ we could combine all binomial expressions associated with $|\Sigma|^{\frac{n^2}{\prod_{i=1}^{\omega(n)} p_i^{k_i}}}$. If $k_i \le 1$ then we have $\binom{p_i+1}{k_i}$ selections from $G_n(p_i)$, and $\bigcup_{l_i=2}^{p_i+1} \binom{p_i+1}{l_i}$ for $k_i = 2$. These two expressions could be generalized as $\bigcup_{l_i=k_i}^{\mathrm{top}(k_i)} \binom{p_i+1}{l_i}$ using *top* function defined above. Therefore the total coefficient of $|\Sigma|^{\frac{n^2}{\prod_{i=1}^{\omega(n)} p_i^{k_i}}}$ is

$$\sum_{\substack{k_1\le l_1\le \mathrm{top}(k_1)\\ \cdots\\ k_{\omega(n)}\le l_{\omega(n)}\le \mathrm{top}(k_{\omega(n)})}} (-1)^{1+\sum_{i=1}^{\omega(n)} l_i} \prod_{i=1}^{\omega(n)} \binom{p_i+1}{l_i}$$

as required. □

Example: Let $n = 2^{\alpha_1}3^{\alpha_2}$, then using counting from Theorem 6.1.35 $|S_{n\times n}| =$

$$
\begin{aligned}
&\binom{3}{1}|\Sigma|^{\frac{n^2}{2}} + \binom{4}{1}|\Sigma|^{\frac{n^2}{3}} \\
&-\binom{3}{2}|\Sigma|^{\frac{n^2}{2^2}} - \binom{3}{1}\binom{4}{1}|\Sigma|^{\frac{n^2}{2\,3}} - \binom{4}{2}|\Sigma|^{\frac{n^2}{3^2}} \\
&+\binom{3}{3}|\Sigma|^{\frac{n^2}{2^2}} + \binom{3}{2}\binom{4}{1}|\Sigma|^{\frac{n^2}{2^2\,3}} + \binom{3}{1}\binom{4}{2}|\Sigma|^{\frac{n^2}{2\,3^2}} + \binom{4}{3}|\Sigma|^{\frac{n^2}{3^2}} \\
&-\binom{3}{3}\binom{4}{1}|\Sigma|^{\frac{n^2}{2^2\,3}} - \binom{3}{2}\binom{4}{2}|\Sigma|^{\frac{n^2}{2^2\,3^2}} - \binom{3}{1}\binom{4}{3}|\Sigma|^{\frac{n^2}{2\,3^2}} - \binom{4}{4}|\Sigma|^{\frac{n^2}{3^2}} \\
&+\binom{3}{3}\binom{4}{2}|\Sigma|^{\frac{n^2}{2^2\,3^2}} + \binom{3}{2}\binom{4}{3}|\Sigma|^{\frac{n^2}{2^2\,3^2}} + \binom{3}{1}\binom{4}{4}|\Sigma|^{\frac{n^2}{2\,3^2}} \\
&-\binom{3}{3}\binom{4}{3}|\Sigma|^{\frac{n^2}{2^2\,3^2}} - \binom{3}{2}\binom{4}{4}|\Sigma|^{\frac{n^2}{2^2\,3^2}} \\
&+\binom{3}{3}\binom{4}{4}|\Sigma|^{\frac{n^2}{2^2\,3^2}}
\end{aligned}
$$

and by Lemma 6.1.36, $|S_{n\times n}| =$

$$
\begin{aligned}
&|\Sigma|^{\frac{n^2}{2}}\left[+\binom{3}{1}\right] + \\
&|\Sigma|^{\frac{n^2}{3}}\left[+\binom{4}{1}\right] + \\
&|\Sigma|^{\frac{n^2}{2\,3}}\left[-\binom{3}{1}\binom{4}{1}\right] + \\
&|\Sigma|^{\frac{n^2}{2^2}}\left[-\binom{3}{2} + \binom{3}{3}\right] + \\
&|\Sigma|^{\frac{n^2}{3^2}}\left[-\binom{4}{2} + \binom{4}{3} - \binom{4}{4}\right] + \\
&|\Sigma|^{\frac{n^2}{2^2\,3}}\left[+\binom{3}{2}\binom{4}{1} - \binom{3}{3}\binom{4}{1}\right] + \\
&|\Sigma|^{\frac{n^2}{2\,3^2}}\left[+\binom{3}{1}\binom{4}{2} - \binom{3}{1}\binom{4}{3} + \binom{3}{1}\binom{4}{4}\right] + \\
&|\Sigma|^{\frac{n^2}{2^2\,3^2}}\left[-\binom{3}{2}\binom{4}{2} + \binom{3}{3}\binom{4}{2} + \binom{3}{2}\binom{4}{3} - \binom{3}{3}\binom{4}{3} - \binom{3}{2}\binom{4}{4} + \binom{3}{3}\binom{4}{4}\right]
\end{aligned}
$$

Definition 6.1.8. *For any state $a \in \Sigma$ and $n, k \in \mathbb{N}$, we define the set $D^a_{n\times n,k}$ to be the set of all square configurations with exactly k sites in state a:*

$$D^a_{n\times n,k} = \{\mathbf{s} \in \Sigma^{n\times n} \mid {}_{\#_a}\mathbf{s} = k\}.$$

Accordingly, we denote by $S^a_{n\times n,k}$ the set of such configurations that are symmetric, so that

$$S^a_{n\times n,k} = S_{n\times n} \cap D^a_{n\times n,k}.$$

And for any vector $v \in \mathbb{Z}_n \times \mathbb{Z}_n$, $S^a_{n\times n,k}(v)$ denotes the set of those configurations in $S^a_{n\times n,k}$ that are generated by vector v, so that

$$S^a_{n\times n,k}(v) = S_{n\times n}(v) \cap D^a_{n\times n,k}.$$

Corollary 6.1.37. *For any state $a \in \Sigma$ and $n, k \in \mathbb{N}$, and $v = (l_1, l_2) \in \mathbb{Z}_n \times \mathbb{Z}_n$*

$$S^a_{n\times n,k}(v) \neq \emptyset$$

iff $|\langle v\rangle| = \frac{n}{\gcd(l_1,l_2,n)}$ is an integer that divides k.

Lemma 6.1.38. *For a given state $a \in \Sigma$, $k \in \mathbb{N}$, and vector $v \in \mathbb{Z}_n \times \mathbb{Z}_n$ such that $|\langle v\rangle|$ divides k*

$$\left|S^a_{n\times n,k}(v)\right| = \binom{\frac{n^2}{|\langle v\rangle|}}{\frac{k}{|\langle v\rangle|}}(|\Sigma| - 1)^{\frac{n^2-k}{|\langle v\rangle|}}$$

Proof Let $\mathbf{s} \in S^a_{n\times n,k}(v)$. Then the number of selections of state in $\mathbf{s}$, i.e. the pattern size, is $n^2/|\langle v\rangle|$. To enumerate the number of such configurations, we first

have to choose $k/|\langle v\rangle|$ over $n^2/|\langle v\rangle|$ sites to be in state a, and then fill the remaining $n^2/|\langle v\rangle| - k/|\langle v\rangle|$ sites with states from $\Sigma \setminus \{a\}$. □

Theorem 6.1.39. *Pick $n, k \in \mathbb{N}$ with $k \leq n$ and let $d = \gcd(k, n)$. Let $n = \prod_{i=1}^{\omega(n)} p_i^{\alpha_i}$, $k = \prod_{i=1}^{\omega(k)} q_i^{\beta_i}$, and $d = \prod_{i=1}^{\omega(d)} r_i^{\gamma_i}$ be the prime factorizations of n, k, d, respectively. Then for any $a \in \Sigma$,*

$$|S^{a}_{n\times n,k}| = \sum_{\substack{0\leq l_1\leq (r_1+1)\\ \cdots \\ 0\leq l_{\omega(d)}\leq (r_{\omega(d)}+1)}} (-1)^{1+\sum_{i=1}^{\omega(n)} l_i} \left(\prod_{i=1}^{\omega(d)} \binom{r_i+1}{l_i}\right) \binom{\frac{n^2}{\prod r_i}}{\frac{k}{\prod r_i}} (|\Sigma|-1)^{\frac{n^2-k}{\prod r_i}},$$

where $\prod r_i = \prod_{i=1}^{\omega(d)} r_i^{\min(l_i,2)}$.

Proof Using Definition 6.1.8, Lemma 6.1.31, and Corollary 6.1.37,

$$\begin{aligned} S^{a}_{n\times n,k} &= \left(\bigcup_{w\in G_n} S_{n\times n}(w)\right) \bigcap D^{a}_{n\times n,k} \\ &= \bigcup_{i=1}^{\omega(N)} \bigcup_{w\in G_n(p_i)} S^{a}_{n\times n,k}(w) \\ &= \bigcup_{i=1}^{\omega(d)} \bigcup_{w\in G_n(r_i)} S^{a}_{n\times n,k}(w). \end{aligned}$$

By the inclusion-exclusion principle

$$|S^{a}_{n\times n,k}| = \sum_{\substack{J_1\subseteq G_n(r_1)\\ \cdots \\ J_{\omega(d)}\subseteq G_n(r_{\omega(d)})}} (-1)^{1+\sum_{i=1}^{\omega(d)} |J_i|} \left|\bigcap_{w\in\cup_i J_i} S^{a}_{n\times n}(w)\right|.$$

Now, by Corollary 6.1.25

$$\begin{aligned}\bigcap_{w\in\cup_{i=1}^{\omega(d)}J_i} S^a_{n\times n,k}(w) &= \left(\bigcap_{w\in\cup_{i=1}^{\omega(d)}J_i} S_{n\times n}(w)\right)\cap D^a_{n\times n,k}\\ &= S_{n\times n}(\cup_{i=1}^{\omega(d)}J_i)\cap D^a_{n\times n,k}\\ &= S^a_{n\times n,k}(\cup_{i=1}^{\omega(d)}J_i)\end{aligned}$$

Finally let $m = |\langle\cup_{i=1}^{\omega(d)}J_i\rangle|$, then using Lemma 6.1.38,

$$|S^a_{n\times n,k}(\cup_{i=1}^{\omega(d)}J_i)| = \binom{\frac{n^2}{m}}{\frac{k}{m}}(|\Sigma|-1)^{\frac{n^2-k}{m}},$$

where $m = |\langle\cup_{i=1}^{\omega(d)}J_i\rangle| = \prod_{i=1}^{\omega(d)} r_i^{\min(|J_i|,2)}$. □

Corollary 6.1.40. *For any state set Σ and state $a\in\Sigma$, the set $S^a_{n\times n,0}$ equals the set $S_{n\times n}$ for the state set $\Sigma\setminus\{a\}$.*

Corollary 6.1.41. *The number of binary symmetric configurations ($|\Sigma| = 2$) with k sites in state a is given by*

$$|S^a_{n\times n,k}| = \sum_{\substack{0\le l_1\le(r_1+1)\\ \vdots\\ 0\le l_{\omega(d)}\le(r_{\omega(d)}+1)}} (-1)^{1+\sum_{i=1}^{\omega(n)}l_i}\left(\prod_{i=1}^{\omega(d)}\binom{r_i+1}{l_i}\right)\binom{\frac{n^2}{\prod r_i}}{\frac{k}{\prod r_i}},$$

where $\prod r_i = \prod_{i=1}^{\omega(d)} r_i^{\min(l_i,2)}$.

Lemma 6.1.42. *Pick $n, k \in \mathbb{N}$ with $k \leq n$ and let $d = \gcd(k, n)$. Let $n = \prod_{i=1}^{\omega(n)} p_i^{\alpha_i}$, $k = \prod_{i=1}^{\omega(k)} q_i^{\beta_i}$, and $d = \prod_{i=1}^{\omega(d)} r_i^{\gamma_i}$ be the prime factorizations of n, k, d, respectively. Then for any $a \in \Sigma$,*

$$|S^a_{n\times n,k}| = \sum_{\substack{0\leq k_1\leq 2\\ \dots \\ 0\leq k_{\omega(d)}\leq 2}} \binom{\frac{n^2}{\prod r_i}}{\frac{k}{\prod r_i}} (|\Sigma|-1)^{\frac{n^2-k}{\prod r_i}} \left(\sum_{\substack{k_1\leq l_1\leq \mathrm{top}(k_1)\\ \dots \\ k_{\omega(d)}\leq l_{\omega(d)}\leq \mathrm{top}(k_{\omega(d)})}} (-1)^{1+\sum_{i=1}^{\omega(d)} l_i} \prod_{i=1}^{\omega(d)} \binom{r_i+1}{l_i} \right),$$

where

$$\mathrm{top}(k_i) = \begin{cases} k_i & \text{if } k_i < 2 \\ r_i + 1 & \text{if } k_i = 2. \end{cases}$$

and $\prod r_i = \prod_{i=1}^{\omega(d)} r_i^{\min(k_i,2)}$.

Proof Similarly to the proof of Lemma 6.1.36 the coefficient of each expression $\binom{\frac{n^2}{\prod r_i}}{\frac{k}{\prod r_i}} |\Sigma - 1|^{\frac{n^2}{\prod r_i}}$ for given $k_1, \dots, k_{\omega(n)} \in \{0, 1, 2\}$ is

$$\sum_{\substack{k_1\leq l_1\leq \mathrm{top}(k_1)\\ \dots \\ k_{\omega(d)}\leq l_{\omega(d)}\leq \mathrm{top}(k_{\omega(d)})}} (-1)^{1+\sum_{i=1}^{\omega(d)} l_i} \prod_{i=1}^{\omega(d)} \binom{r_i+1}{l_i}$$

. □

Example: Let $n = 2^{\alpha_1}3^{\alpha_2}$, $a \in \Sigma$, and $k = 2^{\beta_1}3^{\beta_2}$, where $\beta_1 \leq \alpha_1, \beta_2 \leq \alpha_2$ then using counting from Theorem 6.1.39 $|S^a_{n\times n,k}| =$

$$
\begin{aligned}
&\binom{3}{1}\binom{\frac{n^2}{2}}{\frac{k}{2}}(|\Sigma|-1)^{\frac{n^2-k}{2}} + \binom{4}{1}\binom{\frac{n^2}{3}}{\frac{k}{3}}(|\Sigma|-1)^{\frac{n^2-k}{3}} \\
&-\binom{3}{2}\binom{\frac{n^2}{2^2}}{\frac{k}{2^2}}(|\Sigma|-1)^{\frac{n^2-k}{2^2}} - \binom{3}{1}\binom{4}{1}\binom{\frac{n^2}{2\,3}}{\frac{k}{2\,3}}(|\Sigma|-1)^{\frac{n^2-k}{2\,3}} - \binom{4}{2}\binom{\frac{n^2}{3^2}}{\frac{k}{3^2}}(|\Sigma|-1)^{\frac{n^2-k}{3^2}} \\
&+\binom{3}{3}\binom{\frac{n^2}{2^2}}{\frac{k}{2^2}}(|\Sigma|-1)^{\frac{n^2-k}{2^2}} + \binom{3}{2}\binom{4}{1}\binom{\frac{n^2}{2^2\,3}}{\frac{k}{2^2\,3}}(|\Sigma|-1)^{\frac{n^2-k}{2^2\,3}} + \binom{3}{1}\binom{4}{2}\binom{\frac{n^2}{2\,3^2}}{\frac{k}{2\,3^2}}(|\Sigma|-1)^{\frac{n^2-k}{2\,3^2}} + \binom{4}{3}\binom{\frac{n^2}{3^2}}{\frac{k}{3^2}}(|\Sigma|-1)^{\frac{n^2-k}{3^2}} \\
&-\binom{3}{3}\binom{4}{1}\binom{\frac{n^2}{2^2\,3}}{\frac{k}{2^2\,3}}(|\Sigma|-1)^{\frac{n^2-k}{2^2\,3}} - \binom{3}{2}\binom{4}{2}\binom{\frac{n^2}{2^2\,3^2}}{\frac{k}{2^2\,3^2}}(|\Sigma|-1)^{\frac{n^2-k}{2^2\,3^2}} - \binom{3}{1}\binom{4}{3}\binom{\frac{n^2}{2\,3^2}}{\frac{k}{2\,3^2}}(|\Sigma|-1)^{\frac{n^2-k}{2\,3^2}} - \binom{4}{4}\binom{\frac{n^2}{3^2}}{\frac{k}{3^2}}(|\Sigma|-1)^{\frac{n^2-k}{3^2}} \\
&+\binom{3}{3}\binom{4}{2}\binom{\frac{n^2}{2^2\,3^2}}{\frac{k}{2^2\,3^2}}(|\Sigma|-1)^{\frac{n^2-k}{2^2\,3^2}} + \binom{3}{2}\binom{4}{3}\binom{\frac{n^2}{2^2\,3^2}}{\frac{k}{2^2\,3^2}}(|\Sigma|-1)^{\frac{n^2-k}{2^2\,3^2}} + \binom{3}{1}\binom{4}{4}\binom{\frac{n^2}{2\,3^2}}{\frac{k}{2\,3^2}}(|\Sigma|-1)^{\frac{n^2-k}{2\,3^2}} \\
&-\binom{3}{3}\binom{4}{3}\binom{\frac{n^2}{2^2\,3^2}}{\frac{k}{2^2\,3^2}}(|\Sigma|-1)^{\frac{n^2-k}{2^2\,3^2}} - \binom{3}{2}\binom{4}{4}\binom{\frac{n^2}{2^2\,3^2}}{\frac{k}{2^2\,3^2}}(|\Sigma|-1)^{\frac{n^2-k}{2^2\,3^2}} \\
&+\binom{3}{3}\binom{4}{4}\binom{\frac{n^2}{2^2\,3^2}}{\frac{k}{2^2\,3^2}}(|\Sigma|-1)^{\frac{n^2-k}{2^2\,3^2}}
\end{aligned}
$$

and by Lemma 6.1.42 $|S^{a}_{n\times n,k}| =$

$$\binom{\frac{n^2}{2}}{\frac{k}{2}}(|\Sigma|-1)^{\frac{n^2-k}{2}}\left[+\binom{3}{1}\right]+$$
$$\binom{\frac{n^2}{3}}{\frac{k}{3}}(|\Sigma|-1)^{\frac{n^2-k}{3}}\left[+\binom{4}{1}\right]+$$
$$\binom{\frac{n^2}{2\,3}}{\frac{k}{2\,3}}(|\Sigma|-1)^{\frac{n^2-k}{2\,3}}\left[-\binom{3}{1}\binom{4}{1}\right]+$$
$$\binom{\frac{n^2}{2^2}}{\frac{k}{2^2}}(|\Sigma|-1)^{\frac{n^2-k}{2^2}}\left[-\binom{3}{2}+\binom{3}{3}\right]+$$
$$\binom{\frac{n^2}{3^2}}{\frac{k}{3^2}}(|\Sigma|-1)^{\frac{n^2-k}{3^2}}\left[-\binom{4}{2}+\binom{4}{3}-\binom{4}{4}\right]+$$
$$\binom{\frac{n^2}{2^2\,3}}{\frac{k}{2^2\,3}}(|\Sigma|-1)^{\frac{n^2-k}{2^2\,3}}\left[+\binom{3}{2}\binom{4}{1}-\binom{3}{3}\binom{4}{1}\right]+$$
$$\binom{\frac{n^2}{2\,3^2}}{\frac{k}{2\,3^2}}(|\Sigma|-1)^{\frac{n^2-k}{2\,3^2}}\left[+\binom{3}{1}\binom{4}{2}-\binom{3}{1}\binom{4}{3}+\binom{3}{1}\binom{4}{4}\right]+$$
$$\binom{\frac{n^2}{2^2\,3^2}}{\frac{k}{2^2\,3^2}}(|\Sigma|-1)^{\frac{n^2-k}{2^2\,3^2}}\left[-\binom{3}{2}\binom{4}{2}+\binom{3}{3}\binom{4}{2}+\binom{3}{2}\binom{4}{3}-\binom{3}{3}\binom{4}{3}-\binom{3}{2}\binom{4}{4}+\binom{3}{3}\binom{4}{4}\right]$$

6.2 Loosely-Coupled Configurations

As stated in Definition 6.0.1, a final configuration $\mathbf{s}$ for the leader election task must be a fixed point ($\Phi(\mathbf{s}) = \mathbf{s}$) containing a single active cell ($_{\#_1}\mathbf{s} = 1$). A crucial aspect of this definition is that once a leader is elected, the configuration must *freeze* all following steps. This section discusses how the fixed point requirement manifests in the underlying transition rule and how it affects the CA's performance.

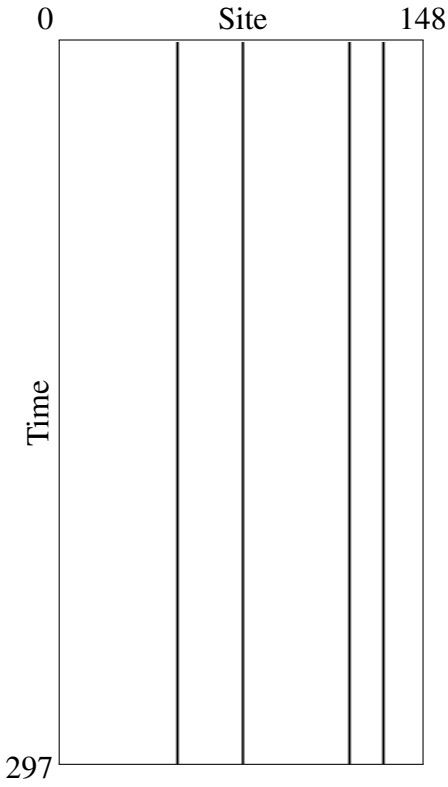


Figure 6.3: A space-time diagram of leader-electing CA on a one-dimensional loosely-coupled configuration, which is a configuration where distance of active cells $\geq 2r + 1$. Note that leader election from a loosely-coupled configuration (a fixed point) is impossible.

6.2.1 One-Dimensional Loosely-Coupled Configurations

Now we analyze the mandatory conditions that have to be satisfied by any transition rule of a one-dimensional CA solving the leader election problem. We will use these mandatory (leader preserving) transitions to identify another set of principally insolvable configurations. The leader preserving transitions are transitions that guarantee a final leader configuration to be a fixed point. For instance, Table 6.1 shows the leader preserving transitions of a one-dimensional binary CA with $r = 3$. In a final fixed point configuration, the leader cell uses the first rule, its neighbors use the rules $2 - 7$, and the rest of the cells use the rule 8. Figure 6.3

shows a CA computation on a one-dimensional loosely-coupled configuration.

Num	η	$\phi(\eta)$	Num	η_i	$\phi(\eta)$
1	0001000	1	5	0000100	0
2	1000000	0	6	0000010	0
3	0100000	0	7	0000001	0
4	0010000	0	8	0000000	0

Table 6.1: The mandatory rows for any leader preserving ϕ of a binary CA with $r = 3$.

Definition 6.2.1. *Suppose ϕ is a transition rule of a uniform one-dimensional CA with radius r. We say ϕ is* leader preserving *(LP) if and only if*

$$\phi(\eta_i) = s_i \qquad \textit{whenever} \qquad {}_{\#_1}\eta_i \leq 1,$$

for any neighborhood $\eta_i = (s_{i-r}, \ldots, s_i, \ldots, s_{i+r})$.

Remark: In applications where $|\Sigma| > 2$, it could be desirable to weaken Definition 6.0.1 of the leader election task together with the implied constraint above by only requiring $\phi(\eta_i) = s_i$ for a smaller collection of neighborhoods. For example, a ternary CA could be designed to aim for final configurations where all non-leader cells are in a specified state $0 \neq 1$. In such cases, it would make sense to refer to the condition (6.2.1) as *strongly* LP, and to the desired modification as *weakly* LP, since the modified constraint would fix fewer configurations. For consistency, however, we follow Definition 6.2.1 and leave weakly LP transitions for future consideration.

Lemma 6.2.1. *Let $\mathcal{F}$ denote the set of neighborhoods constrained in Definition*

6.2.1, so that $\mathcal{F} = \{\eta \in \Sigma^{2r+1} |_{\#_1} \eta \leq 1\}$. *Then*

$$|\mathcal{F}| = (|\Sigma| - 1)^{2r}(2r + 1) + (|\Sigma| - 1)^{2r+1}.$$

Accordingly, every transition table for a leader election contains $|\mathcal{F}|$ *mandatory transitions (rows).*

Proof There are exactly $(|\Sigma| - 1)^{2r}(2r + 1)$ neighborhoods with ${}_{\#_1}\eta = 1$ and $(|\Sigma| - 1)^{2r+1}$ with ${}_{\#_1}\eta = 0$. □

Definition 6.2.2. *A configuration* $\mathbf{s} \in \Sigma^N$ *is called* loosely coupled *if* $\eta_i \in \mathcal{F}$ *for all* i $(0 \leq i \leq N - 1)$.

Corollary 6.2.2. *A configuration* **s** *is* loosely coupled *whenever*

$$s_i = s_j = 1 \qquad \textit{implies} \qquad 2r + 1 \leq |i - j| \leq N - 2r - 1$$

for any distinct i, j $(0 \leq i, j \leq N - 1)$ *and radius* r.

We note that the restriction $|i - j| \leq N - 2r - 1$ is needed above since our configurations are read cyclically.

Corollary 6.2.3. *If* ϕ *is uniform and leader preserving, and* **s** *is a loosely-coupled configuration then* $\Phi(\mathbf{s}) = \mathbf{s}$.

Along with the desirable final configurations for leader election, the above corollary indicates that there may be undesirable ones which must be fixed as well. Our next definition will help us discuss the issues that arise when the distance between cells in state 1 exceeds the length of the neighborhoods of the transition rule ϕ.

Definition 6.2.3. *We denote by L_N the set of all loosely-coupled configurations in Σ^N.*

Theorem 6.2.4. *Leader election from a loosely-coupled configuration $\mathbf{s} \in L_N$, where ${}_{\#_1}\mathbf{s} \neq 1$ is not possible.*

Proof By Corollary 6.2.3, any $\mathbf{s} \in L_N$ is a fixed point. Therefore, after n applications of Φ, the number of active cells in $\Phi^n(\mathbf{s}) = \mathbf{s}$ stays the same. □

We defined the loosely-coupled configurations, which are those configurations where the distance between any two active cells is at least $2r + 1$. Also, we showed that loosely-coupled configurations are fixed points for any leader electing or leader preserving CA. Now we will enumerate all such configurations.

Definition 6.2.4. *The set of loosely-coupled configurations with exactly k active cells (i.e., cells in state $1 \in \Sigma$) is*

$$L_{N,k} = L_N \cap D^1_{N,k},$$

where $D^1_{N,k} = \{\mathbf{s} \in \Sigma^N |_{\#_1}\mathbf{s} = k\}$.

Theorem 6.2.5. *Let k be the number of active cells in a loosely-coupled configuration with radius r, and $f = N - k(2r + 1)$ be the number of free cells, which do not belong to the neighborhood of any active cell. If $f \geq 0$*

$$|L_{N,k}| = \left(\binom{k+f-1}{k-1}(2r+1) + \binom{k+f-1}{k}\right)(|\Sigma| - 1)^{N-k}.$$

Proof Since $\mathbf{s} \in L_{N,k}$ iff the neighborhoods of active cells do not overlap, the problem of enumerating $L_{N,k}$ can be transformed to a problem of non-intersecting

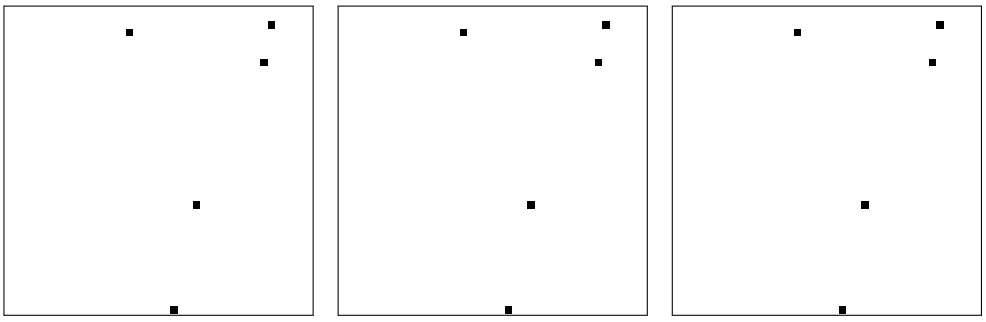
intervals. Specifically, we want to place k disjoint neighborhoods (intervals) of size $2r + 1$ over N sites. There are $\binom{k+f}{k}$ ways to form a sequence of length $k + f$ by distributing f free cells between k disjoint intervals.

However, since our configurations are cyclic, we must adjust this number. In particular, each sequence that begins with an interval corresponds with $2r + 1$ shifted configurations, depending on which cell in that active neighborhood is cell 0 of the configuration. As a result, there are $\binom{k+f-1}{k-1}(2r + 1)$ placements starting with an active neighborhood and $\binom{k+f-1}{k}$ starting with a free cell.

Once all active cells are placed, there are $N - k$ remaining cells to be filled with inactive states, which can be done in $(|\Sigma| - 1)^{N-k}$ ways. □

Corollary 6.2.6. *The number of active cells in a loosely-coupled configuration is at most*

$$k_{\max} = \left\lfloor \frac{N}{2r + 1} \right\rfloor.$$



(a) t_0 (b) t_1 (c) t_2

Figure 6.4: Space-time diagrams of CA computation on a two-dimensional loosely-coupled configuration, which is a configuration where distance of active cells $\geq 2r + 1$. Note that leader election from loosely-coupled configurations (fixed points) is impossible.

6.2.2 Two-Dimensional Loosely-Coupled Configurations

As defined in Section 3.1.2, a square neighborhood of two-dimensional CAs with radius r contains $(2r+1)^2$ cells. Similarly to the one-dimensional case, each transition table for two-dimensional leader election must contain the leader preserving transitions, which guarantee a final leader configuration to be a fixed point. For instance, Table 6.2 shows the leader preserving transitions of a two-dimensional binary CA with Moore neighborhood. In a final fixed point configuration, the leader cell uses the first rule, its neighbors use the rules $2-9$, and the rest of the cells use the rule 10. Figure 6.4 shows a CA computation on a two-dimensional loosely-coupled configuration.

Num	η	$\phi(\eta)$	*Num*	η_i	$\phi(\eta)$
1	000 010 000	1	6	000 001 000	0
2	100 000 000	0	7	000 000 100	0
3	010 000 000	0	8	000 000 010	0
4	001 000 000	0	9	000 000 001	0
5	000 100 000	0	10	000 000 000	0

Table 6.2: The mandatory rows for any leader preserving ϕ of a binary two-dimensional CA with Moore neighborhood.

Definition 6.2.5. *Suppose ϕ is a transition rule of a uniform two-dimensional CA*

with radius r. We say ϕ is leader preserving *(LP) if and only if*

$$\phi(\eta_u) = s_u \qquad \textit{whenever} \qquad {}_{\#_1}\eta_u \leq 1,$$

for any location $u \in \mathbb{Z}_n \times \mathbb{Z}_n$.

Lemma 6.2.7. *Let $\mathcal{F}$ denote the set of neighborhoods constrained in Definition 6.2.5, so that $\mathcal{F} = \{\eta \in \Sigma^{(2r+1)^2} |_{\#_1}\eta \leq 1\}$. Then*

$$|\mathcal{F}| = (|\Sigma| - 1)^{(2r+1)^2-1}(2r+1)^2 + (|\Sigma| - 1)^{(2r+1)^2}.$$

Accordingly, every transition table for a two-dimensional leader election contains $|\mathcal{F}|$ mandatory transitions (rows).

Proof There are exactly $(|\Sigma| - 1)^{(2r+1)^2-1}(2r+1)^2$ neighborhoods with ${}_{\#_1}\eta = 1$ and $(|\Sigma| - 1)^{(2r+1)^2}$ with ${}_{\#_1}\eta = 0$. □

Definition 6.2.6. *A two-dimensional configuration $\mathbf{s} \in \Sigma^N$ is called* loosely coupled *if $\eta_u \in \mathcal{F}$ for all $u \in \mathbb{Z}_n \times \mathbb{Z}_n$ ($N = n^2$).*

Corollary 6.2.8. *A two-dimensional configuration* **s** *is* loosely coupled *whenever*

$$s_u = s_v = 1 \qquad \textit{implies} \qquad 2r+1 \leq |u_1 - v_1| \leq n-2r-1 \wedge 2r+1 \leq |u_2 - v_2| \leq n-2r-1$$

for any distinct $u = (u_1, u_2), v = (v_1, v_2) \in \mathbb{Z}_n \times \mathbb{Z}_n$ and radius r.

Corollary 6.2.9. *If ϕ is uniform and leader preserving, and* **s** *is a loosely-coupled configuration then $\Phi(\mathbf{s}) = \mathbf{s}$.*

Definition 6.2.7. *We denote by $L_{n\times n}$ the set of all loosely-coupled two-dimensional (square) configurations in $\Sigma^n \times \Sigma^n$.*

Theorem 6.2.10. *Leader election from a loosely-coupled configuration $\mathbf{s} \in L_{n\times n}$, where ${}_{\#_1}\mathbf{s} \neq 1$ is not possible.*

Proof By Corollary 6.2.9, any $\mathbf{s} \in L_{n\times n}$ is a fixed point. Therefore, after n applications of Φ, the number of active cells in $\Phi^n(\mathbf{s}) = \mathbf{s}$ stays the same. □

We defined the loosely-coupled two-dimensional configurations, which are those configurations where the distance between any two active cells is at least $2r + 1$ for each axis.

Definition 6.2.8. *The set of loosely-coupled two-dimensional configurations with exactly k active cells (i.e., cells in state $1 \in \Sigma$) is*

$$L_{n\times n,k} = L_N \cap D^1_{n\times n,k},$$

where $D^1_{n\times n,k} = \{\mathbf{s} \in \Sigma^{n\times n} \mid {}_{\#_1}\mathbf{s} = k\}$.

As opposed to the one-dimensional case, enumeration of two-dimensional loosely-coupled configurations is substantially more difficult. In fact, the problem of enumerating loosely-coupled configurations is a generalized problem of non-attacking kings. More precisely, the kings problem [65] is to calculate how many different ways k kings can be placed on a chessboard $n \times n$ so that no two kings can attack each other. This problem has not been solved universally for arbitrary k and n, however, there exists several upper and lower bounds and approximations [19]. Furthermore, a generating function [18] for a recursive formula has been found only for a specific k (low values up to 7).

Note that a kings placement is non-attacking iff each pair of kings is at least 2 sites apart (on each axis). Clearly, the enumeration of loosely-coupled configurations is even more difficult, since a part of the problem is to enumerate different placements of k active cells on lattice with cycled boundaries (toroid) where active cells keep distance of at least $2r + 1$. As a matter of fact, rather than deriving an explicit formula for $|L_{n\times n,k}|$, we numerically obtain approximate values by statistical sampling. More specifically, we randomly draw k positions for active cells, and then check whether the configuration is loosely coupled. After we repeat this process $30,000$ times, we multiple the fraction of loosely-coupled configurations with the total number of placements of k active cells $\binom{n^2}{k}$. Finally, we can place the remaining $n^2 - k$ inactive cells on lattice in $(|\Sigma| - 1)^{n^2-k}$ ways. For faster enumeration we can bound the maximal number of active cells as follows.

Lemma 6.2.11. *The number of active cells in a square loosely-coupled configuration is at most*

$$k_{\max} = \left\lfloor \frac{n^2}{(2r+1)^2} \right\rfloor.$$

6.3 Upper Bound on Performance

We have shown that symmetric and loosely-coupled configurations are insolvable for leader election. To combine the enumerations of these configuration types, we first characterize the configurations that are both symmetric and loosely-coupled and then derive an overall upper bound on performance.

6.3.1 One-Dimensional Upper Bound on Performance

Lemma 6.3.1. *Fix any positive integers $m \leq k \leq N$, where $m|N$ and $m|k$. Then the set of all symmetric loosely-coupled configurations of length N over the alphabet Σ with pattern size $\frac{N}{m}$ and k active cells satisfies*

$$|S_N(N/m) \cap L_{N,k}| = |L_{\frac{N}{m},\frac{k}{m}}|.$$

Proof Pick any element $\mathbf{s}$ of the left-hand side. Since $\mathbf{s} \in S_N(\frac{N}{m})$, we know that $\mathbf{s} = \mathbf{q}^m$ for some $\mathbf{q} \in \Sigma^{\frac{N}{m}}$. Since $\mathbf{s} \in L_{N,k}$, it must also be the case that ${}_{\#_1}\mathbf{q} = k/m$. Indeed, since $\mathbf{s} = \mathbf{q}^m \in L_{N,k}$, the 1's in $\mathbf{q}$ must keep a (cyclic) distance of at least $2r+1$ apart. Therefore $\mathbf{q}$ is a loosely-coupled (sub)configuration of size N/m with k/m active cells, *i.e.*, $\mathbf{q} \in L_{\frac{N}{m},\frac{k}{m}}$.

Conversely, any element $\mathbf{q}$ of the right-hand side gives rise to a unique element $\mathbf{s} = \mathbf{q}^m$ of the left-hand side. □

Theorem 6.3.2. *Pick $N, k \in \mathbb{N}$ with $k \leq N$ and let $d = \gcd(k, N)$. Let $N = \prod_{i=1}^{\omega(N)} p_i^{\alpha_i}$, $k = \prod_{i=1}^{\omega(k)} q_i^{\beta_i}$, and $d = \prod_{i=1}^{\omega(d)} r_i^{\gamma_i}$ be the prime factorizations of N, k, d, respectively. Then the number of symmetric loosely-coupled configurations of size N with k active cells is*

$$|S\,L_{N,k}| = |S^1_{N,k} \cap L_{N,k}| = \sum_{i=1}^{\omega(d)} (-1)^{i+1} \sum_{\substack{J \subseteq \{1,\ldots,\omega(d)\} \\ |J|=i}} \left| L_{\frac{N}{\prod_{j\in J} r_j}, \frac{k}{\prod_{j\in J} r_j}} \right|.$$

Proof As in the proof of Theorem 6.1.16, $S^1_{N,k} = \bigcup_{i=1}^{\omega(d)} S^1_{N,k}(N/r_i)$. So by the

inclusion-exclusion principle

$$|S^1_{N,k} \cap L_{N,k}| = \sum_{i=1}^{\omega(d)} (-1)^{i+1} \sum_{\substack{J \subseteq \{1,\ldots,\omega(d)\} \\ |J|=i}} \left| \bigcap_{j \in J} S^1_{N,k}(N/r_j) \cap L_{N,k} \right|.$$

Now, by Lemma 6.1.10, and since $L_{N,k} \subseteq D^1_{N,k}$

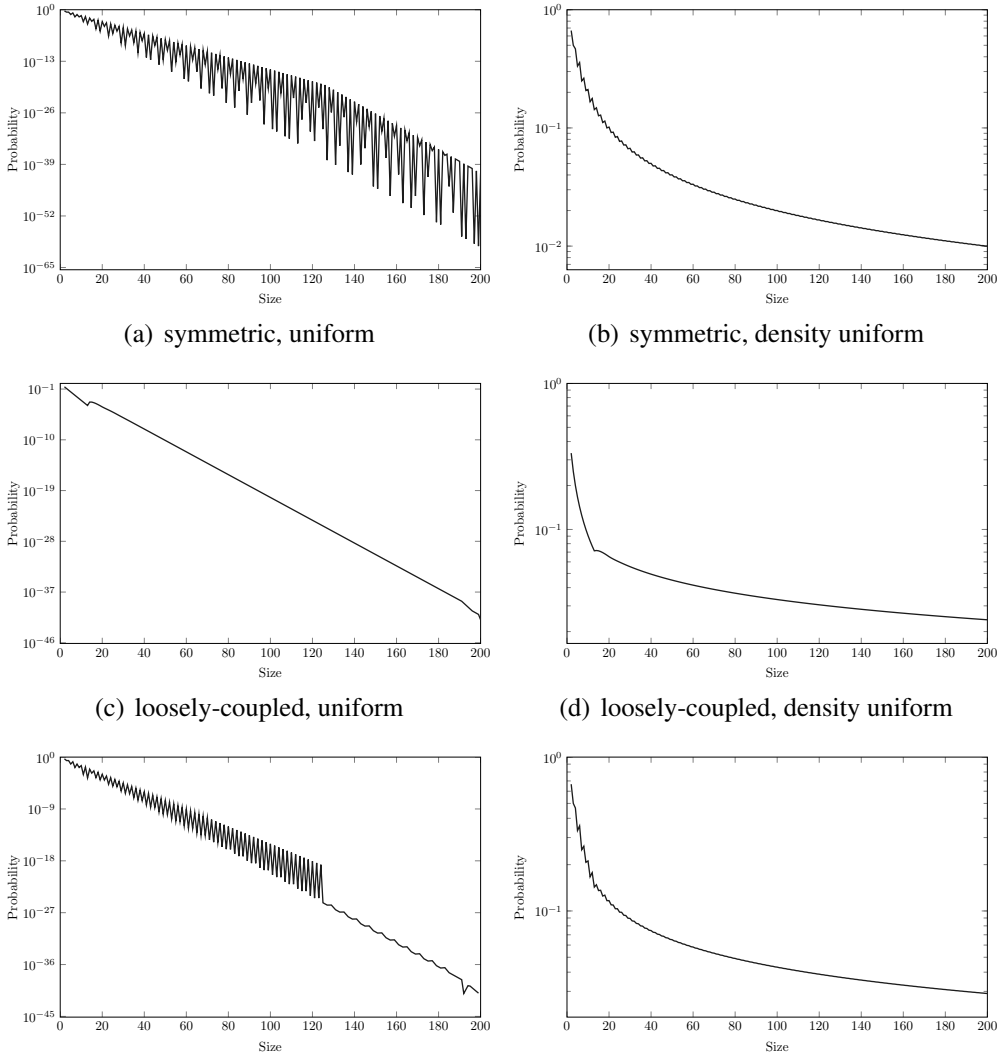
$$\begin{aligned} \bigcap_{j \in J} S^1_{N,k}(N/r_j) \cap L_{N,k} &= S_N\left(\frac{N}{\prod_j r_j}\right) \cap D^1_{N,k} \cap L_{N,k} \\ &= S_N\left(\frac{N}{\prod_j r_j}\right) \cap L_{N,k} \end{aligned}$$

Finally using Lemma 6.3.1,

$$\left| S_N\left(\frac{N}{\prod_j r_j}\right) \cap L_{N,k} \right| = \left| L_{\frac{N}{\prod_j r_j}, \frac{k}{\prod_j r_j}} \right|.$$

□

To calculate the probability that a randomly drawn configuration is insolvable, we can either use a uniform distribution, where the probability of selecting each symbol from Σ for s_i in configuration $\mathbf{s}$ is the same, or we can use a density uniform distribution, where the probability of selecting k active cells $_{\#_1}\mathbf{s} = k$ is uniform. The probability that a configuration is insolvable is, therefore, given below.



(a) symmetric, uniform

(b) symmetric, density uniform

(c) loosely-coupled, uniform

(d) loosely-coupled, density uniform

(e) symmetric or loosely-coupled, uniform

(f) symmetric or loosely-coupled, density uniform

Figure 6.5: Probability of selecting insolvable binary configurations for one-dimensional symmetric and/or loosely-coupled configurations using uniform and density-uniform distributions and radius $r = 3$.

$$P_{\text{unif}}(\mathbf{s} \in S_N \cup L_N) = \frac{1}{|\Sigma|^N} \sum_{k \in \{0,2,\ldots k_{\max}\}} |S_{N,k}| + |L_{N,k}| - |S L_{N,k}|$$

$$P_{\text{dens}}(\mathbf{s} \in S_N \cup L_N) = \frac{1}{N+1} \sum_{k \in \{0,2,\ldots k_{\max}\}} \frac{|S_{N,k}| + |L_{N,k}| - |S L_{N,k}|}{\binom{N}{k}(|\Sigma|-1)^{N-k}}$$

An upper bound on performance for these distribution types are $1 - P_{\text{unif}}(\mathbf{s} \in S_N \cup L_N)$ and $1 - P_{\text{dens}}(\mathbf{s} \in S_N \cup L_N)$. As presented in Figure 6.5, the probability of selecting an insolvable configuration using a uniform distribution decreases by a logarithmic scale, and, for $N > 11$, it drops below 0.01 with $|\Sigma| = 2$ and radius $r = 3$. On the other hand, the insolvability for a density uniform distribution decreases a magnitude slower and reaches 0.03 even for $N = 200$. That is due to the fact that configurations with very few active cells are very often loosely-coupled. Also fewer active cells means more symmetric configurations, since the repeated patterns could be longer. Figure 6.3 compares an upper bound with the performance of the best one-dimensional leader election CA—the improved strategy of mirror particles.

N	Uniform Dist.		Density Uniform Dist.	
	Performance	Upper Bound	Performance	Upper Bound
149	0.997	~ 1.0	0.945	0.966
593	0.997	~ 1.0	0.973	0.988
1001	0.996	~ 1.0	0.979	0.993
1301	0.995	~ 1.0	0.979	0.994

Table 6.3: Performance of the improved strategy of mirror particles, the best one-dimensional binary cellular automaton with radius $r = 3$, compared to theoretical upper bound performance. Both uniform and density-uniform distributions are considered.

6.3.2 Two-Dimensional Upper Bound on Performance

As we showed in Section 6.2.2 enumeration of two-dimensional loosely-coupled configurations is a generalized kings-placement problem, for which exists no universal closed formula. Here we are going to merge symmetric and loosely-coupled configurations in order to derive upper bound on performance for two-dimensional CAs. Similarly to the one-dimensional case, we need to enumerate configurations that are both symmetric and loosely-coupled. However, since the elements of $L_{n\times n,k}$ could not be properly characterized, we do not attempt to derive a closed formula for symmetric loosely-coupled configurations $|S_{n\times n}(v) \cap L_{n\times n,k}|$ given a vector $v \in \mathbb{Z}^n \times \mathbb{Z}^n$ either.

Recall that as a result of Corollary 6.1.37, $S_{n\times n}(v) \cap L_{n\times n,k} \neq \emptyset$ implies $|\langle v\rangle| \mid k$. Note that the opposite implication does not have to hold. Also, the precise number of symmetric loosely-coupled configurations depends on vector v and its periodicity in both dimensions, however, we can bound it as

$$|S_{n\times n}(v) \cap L_{n\times n,k}| \leq |L_{\frac{n\times n}{|\langle v\rangle|},\frac{k}{|\langle v\rangle|}}|.$$

The equality holds for instance if v is diagonal, i.e., $v = (a, a)$, however, if for instance $v = (1, 0)$ then $|S_{n\times n}(v) \cap L_{n\times n,k}| = 0$ for $k > 0$.

Definition 6.3.1. *The square symmetric loosely-coupled configurations of size n^2 with k active cells is*

$$S\,L_{n\times n,k} = S^1_{n\times n,k} \cap L_{n\times n,k}.$$

Similarly to loosely-coupled configurations we approximate $|S\,L_{n\times n,k}|$ by sta-

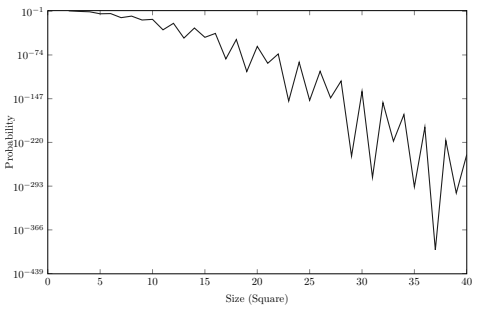
tistical sampling. More specifically, we randomly generate a symmetric configuration with k active cells, and then check whether the configuration is loosely-coupled. We repeat this process $30,000$ times, and we multiple the fraction of those configurations that are loosely-coupled with the total number of symmetric configurations with k active cells $|S^{1}_{n\times n,k}|$. For faster enumeration we can handle the specific number of active cells without sampling. Trivially if the number of active cells is zero, all symmetric loosely-coupled configurations are loosely coupled. There exists no loosely-coupled configuration with a single active cell (besides $N = 1$ case), and so neither a symmetric loosely-coupled configuration. For a non-zero count the prime factors of N must divide the number of active cells k.

Similarly to the one-dimensional case, we calculate the probability that a randomly drawn configuration is insolvable using uniform and density-uniform distributions. The probability that a square two-dimensional configuration is insolvable is, therefore, given below.

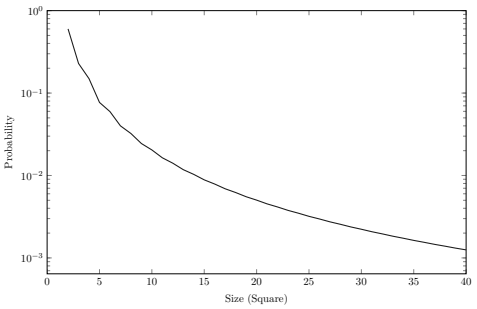
$$P_{\text{unif}}(\mathbf{s} \in S_{n\times n} \cup L_{n\times n}) = \frac{1}{|\Sigma|^{n^2}} \sum_{k\in\{0,2,\ldots k_{\max}\}} |S_{n\times n,k}| + |L_{n\times n,k}| - |S\,L_{n\times n,k}|$$

$$P_{\text{dens}}(\mathbf{s} \in S_{n\times n} \cup L_{n\times n}) = \frac{1}{n^2+1} \sum_{k\in\{0,2,\ldots k_{\max}\}} \frac{|S_{n\times n,k}| + |L_{n\times n,k}| - |S\,L_{n\times n,k}|}{\binom{n^2}{k}(|\Sigma|-1)^{n^2-k}}$$

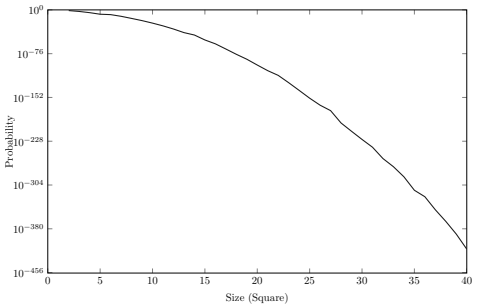
An upper bound on performance for these distribution types are $1 - P_{\text{unif}}(\mathbf{s} \in S_{n\times n} \cup L_{n\times n})$ and $1 - P_{\text{dens}}(\mathbf{s} \in S_{n\times n} \cup L_{n\times n})$. As presented in Figure 6.6, the probability of selecting an insolvable configuration using a uniform distribution decreases by a logarithmic scale. On the other hand, the insolvability for density-uniform dis-



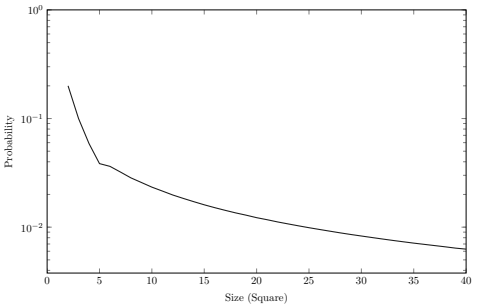
(a) symmetric, uniform



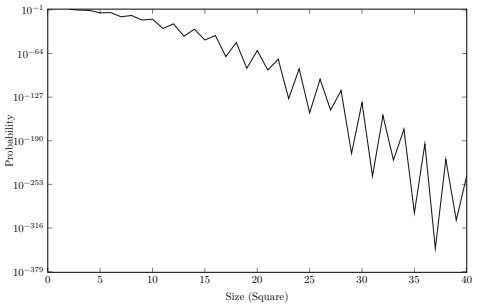
(b) symmetric, density uniform



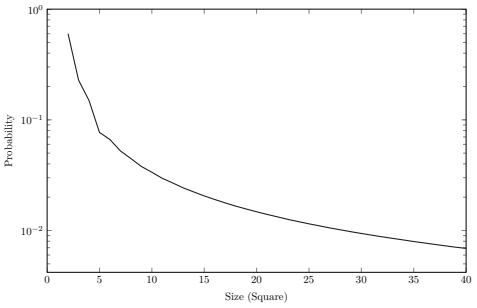
(c) loosely-coupled, uniform



(d) loosely-coupled, density uniform



(e) symmetric or loosely-coupled, uniform



(f) symmetric or loosely-coupled, density uniform

Figure 6.6: Probability of selecting insolvable binary configurations for two-dimensional symmetric and/or loosely-coupled configurations using uniform and density-uniform distributions and Moore neighborhood, i.e., a square neighborhood with radius $r = 1$.

tribution decreases a magnitude slower and reaches 0.0069 even for $N = 40^2$. Similarly to the one-dimensional case that is due to the fact that density uniform distribution prefers configurations with very few active cells, which are very often

$N = n^2$	Uniform Dist.		Density Uniform Dist.	
	Performance	Upper Bound	Performance	Upper Bound
19^2	0.987	~ 1.0	0.807	0.984
29^2	0.978	~ 1.0	0.765	0.990
39^2	0.908	~ 1.0	0.692	0.992

Table 6.4: Maximal performance of the best two-dimensional binary cellular automata with Moore neighborhood from Sections 5.4.2.3 and 5.4.2.4 compared to theoretical upper bound performance. Both uniform and density-uniform distributions are considered.

loosely-coupled and symmetric. Figure 6.4 compares an upper bound with the performance of the best two-dimensional leader-electing CAs.

The traveller has reached the end of the journey!
Edmund Burke (1729-1797)

7

Conclusion

Leader election is a fundamental procedure of many distributed protocols and biological societies. In this thesis we successfully analyzed and solved the leader election problem for one- and two-dimensional CAs. We showed that even mere distributed system consisting of indistinguishable and uniform cells operating just with a binary state is capable of emergent and complex dynamics enabling global coordination of cells and ultimately leader election. Evolutionary process [28] pushed CAs towards a creation of remarkable collective patterns known as domains, particles and interactions [26]. The anonymous leader election implemented by CAs is possible due to particle-mediated information exchange across distances. In two dimensions, slowly-contracting regions connected by lines of active cells propagate throughout the lattice and sweep any remaining active cells, before shrinking to a single cell (leader). We also analyzed the perturbation stability of two-dimensional CAs and showed that the complex dynamics correlate with CA's performance on leader election.

Our final evolutionary one- and two-dimensional CAs reached remarkably high performance of 99% that scale well with respect to the system size. However, the best one-dimensional CAs, such as, the (improved) strategy of mirror particles, are often N-modulo restricted. Our model substantially reduced architectural requirements of the problem and compared to known state-of-art distributed algorithms we can claim that it is the simplest system satisfactory dealing with most difficult anonymous, deterministic instance of leader election.

Furthermore, we showed that anonymous leader election, which is the most difficult instance of the problem, implemented in one- or two-dimensional CAs is principally insolvable. We identified two classes of configurations, symmetric and loosely-coupled, which no CA could transform to a state with a single active cell (a leader). We enumerated such configurations and formulated a corresponding upper bound on the performance rate. Loosely-coupled configurations, where active cells are too far from each other, are insolvable due to the fixed point requirement for a final configuration. That is analogous to the so-called silent self-stabilization [84] where a final, desired state of the system cannot change unless it is perturbed from outside. To weaken the leader election definition and to make loosely-coupled configurations solvable, we may say that the final configurations do not have to be stable points and it is sufficient if only a leader cell keeps its state fixed but all remaining cells are free to change to any non-leader state. That would mean that the CA can compute even after a leader is elected. A new mandatory constraint on the transition rule ϕ would be

$$\phi(\eta_i) = \left\{ \begin{array}{ll} 1 & \Longleftrightarrow \ {}_{\#_1}\eta_i = 1, s_i = 1 \\ \in \Sigma \setminus \{1\} & \Longleftrightarrow \ {}_{\#_1}\eta_i = 1, s_i \neq 1 \\ \in \Sigma \setminus \{1\} & \Longleftrightarrow \ {}_{\#_1}\eta_i = 0 \end{array} \right\}.$$

Note that for binary CAs, the original and weakened definitions are equal. Therefore, to exploit the weakened definition for loosely-coupled configurations we need more than 2 states. Also, because the second and third case allow transitions to any of the $\Sigma \setminus \{1\}$ states, there exist many possible transition rules that meet the weakened fixed-point requirement when $|\Sigma| > 2$.

The best one- and two-dimensional CAs, found by genetic algorithms, reach the performance of 99% for uniform distribution, which is close to the theoretical upper bound. Using a uniform distribution, the probability of selecting an insolvable configuration decreases rapidly with the number of cells. For instance, the number of insolvable configurations for one-dimensional binary CA with 25 cells is just 10^{-5}. Therefore, the reliability of anonymous leader election could be, in principle, minimized to a desired tolerance based on specific application requirements.

Selected CA transition tables presented in this dissertation are provided in our COEL framework at `http://coel-sim.org/download`. COEL framework is an enterprise Java/Scala project built on Grails, Spring, Hibernate, GridGain technology stack providing a unified web environment for the definition and manipulation of several unconventional computational models. The part relevant for this thesis is the network module responsible for modeling, execution, performance evaluation, and dynamics analysis of various networks, such as, CAs. If you wish

to obtain an account please contact the author.

The presented results are generally applicable in the theory of distributed algorithms and also at the elementary level of biology and social science. Last, but not least we hope that this thesis illustrated how powerful self-organization is and what processes are responsible for complex computation implemented by nature. Leader election is also a key routine of cell differentiation, where initial symmetry of newly developed organism has to be broken to allow structural heterogeneity and a cell specialization.

Bibliography

[1] Andrew Adamatzky. *Collision-based computing*. Springer, 2002.

[2] David Andre, Forrest H Bennett III, and John R Koza. Evolution of intricate long-distance communication signals in cellular automata using genetic programming. In *Proceedings of the Fifth International Workshop on the Synthesis and Simulation of Living Systems. Cambridge, MA: MIT Press*, 1996.

[3] Dana Angluin. Local and global properties in networks of processors. In *STOC '80: Proceedings of the twelfth annual ACM symposium on Theory of computing*, pages 82–93, New York, NY, USA, 1980. ACM.

[4] Daniel Ashlock. *Evolutionary computation for modeling and optimization*, volume 103. Springer, 2006.

[5] Baruch Awerbuch and Rafail Ostrovsky. Memory-efficient and self-stabilizing network reset (extended abstract). In *PODC '94: Proceedings of the thirteenth annual ACM symposium on Principles of distributed computing*, pages 254–263, New York, NY, USA, 1994. ACM.

[6] Thomas Bäck. *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford university press, 1996.

[7] Rena Bakhshi, Wan Fokkink, Jun Pang, and Jaco van de Pol. *Leader Election in Anonymous Rings: Franklin Goes Probabilistic*, volume 273 of *IFIP International Federation for Information Processing*, pages 52–72. Springer Boston, 2009.

[8] P. Banda. Cellular automata evolution and leader election in a symmetric ring. In Maria Bielikova, editor, *Proceedings in Informatics and Information Technologies, IIT.SRC 2008 - Student Research Conference*, pages 303–310. STU, 2008.

[9] P. Banda. Computational mechanics, evolution of cellular automata and leader election problem. In J. Rybar J. Kelemen, V. Kvasnicka, editor, *Cognition and Artificial Life '09*, pages 19–28. SLU Opava, 2009.

[10] P. Banda. Upper bound performance of cellular automata for the leader election problem in arbitrary ring. In Maria Bielikova, editor, *Proceedings in Informatics and Information Technologies, IIT.SRC 2009 - Student Research Conference*, pages 158–165. STU, 2009.

[11] Peter Banda. Cellular automata evolution of leader election. In George Kampis, István Karsai, and Eörs Szathmáry, editors, *Advances in Artificial Life. Darwin Meets von Neumann*, volume 5778 of *Lecture Notes in Computer Science*, pages 310–317. Springer Berlin / Heidelberg, 2011.

[12] Peter Banda, John IV Caugmann, and Jiri Pospichal. Configuration symmetry and performance upper bound of one-dimensional cellular automata for the leader election problem. *Journal of Cellular Automata*, 10(1-2):1–21, 2015.

[13] J. Beauquier, M. Gradinariu, and C. Johnen. Memory space requirements for self-stabilizing leader election protocols. In *In PODC '99: Proceedings of the eighteenth annual ACM symposium on Principles of distributed computing*, 1999.

[14] Charles H Bennett, Geoffrey Grinstein, et al. Role of irreversibility in stabilizing complex and nonergodic behavior in locally interacting discrete systems. *Physical review letters*, 55(7):657–660, 1985.

[15] J. E. Burns. Self-stabilizing rings without demons. Technical report, Georgia Inst of Technology, 1987.

[16] J. E. Burns and Jan K. Pachl. Uniform self-stabilizing rings. *ACM Trans. Program. Lang. Syst.*, 11(2):330–344, 1989.

[17] Lars Büsing, Benjamin Schrauwen, and Robert Legenstein. Connectivity, dynamics, and memory in reservoir computing with binary and analog neurons. *Neural computation*, 22(5):1272–1311, 2010.

[18] Neil Calkin, Shihwei Chao, Catherine Davison, Krista Kelly, and Ana-Andreea Stoica. Reu kings write-up. 2013.

[19] Neil J Calkin, Kevin James, Shannon Purvis, S Race, K Schneider, and M Yancey. Counting kings: As easy as lambda˜ 1, lambda˜ 2, lambda˜ 3. *Congressus Numerantium*, 183:83, 2006.

[20] Martin Cenek. *Information Processing in Two-Dimensional Cellular Automata*. PhD thesis, Portland State University, 2011.

[21] Martin Cenek and Melanie Mitchell. Evolving cellular automata. In *Encyclopedia of Complexity and Systems Science*, pages 3233–3242, 2009.

[22] Ernest Chang and Rosemary Roberts. An improved algorithm for decentralized extrema-finding in circular configurations of processes. *Commun. ACM*, 22(5):281–283, 1979.

[23] E. F. Codd. *Cellular automata*. Academic Press, 1968.

[24] Larissa Conradt and Christian List. Group decisions in humans and animals: a survey. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 364(1518):719–742, 2008.

[25] Larissa Conradt and Timothy J. Roper. Consensus decision making in animals. *Trends in Ecology & Evolution*, 20(8):449–456, August 2005.

[26] J. P. Crutchfield. The calculi of emergence: Computation, dynamics and induction. *Physica D*, 75:11–53, 1994.

[27] J. P. Crutchfield and J. E. Hanson. Turbulent pattern bases for cellular automata. *Physica D*, 69(3/4):279–301, 1993.

[28] J. P. Crutchfield, M. Mitchell, and R. Das. Evolutionary design of collective computation in cellular automata. In *Evolutionary dynamics: Exploring the interplay of selection, accident, neutrality, and function*, pages 361–411. Oxford, 2003.

[29] J. P. Crutchfield and P. Schuster. Evolutionary dynamics: Exploring the interplay of selection, accident, neutrality and function, 2003.

[30] J. P. Crutchfield and K. Young. Inferring statistical complexity. *Physical review letters*, 63:105–108, 1989.

[31] James P. Crutchfield and Melanie Mitchell. The evolution of emergent computation. Technical Report 94-03-012, 1994.

[32] Rajarshi Das, James P. Crutchfield, Melanie Mitchell, and James E. Hanson. Evolving globally synchronized cellular automata. In Larry Eshelman, editor, *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 336–343, San Francisco, CA, 1995. Morgan Kaufmann.

[33] Rajarshi Das, Melanie Mitchell, and James P Crutchfield. A genetic algorithm discovers particle-based computation in cellular automata. In *Parallel problem solving from nature—PPSN III*, pages 344–353. Springer, 1994.

[34] Bernard Derrida and Yves Pomeau. Random networks of automata: a simple annealed approximation. *EPL (Europhysics Letters)*, 1(2):45, 1986.

[35] Edsger W. Dijkstra. Self-stabilizing systems in spite of distributed control. *Commun. ACM*, 17(11):643–644, 1974.

[36] Danny Dolev, Maria Klawe, and Michael Rodeh. An o(n log n) unidirectional distributed algorithm for extrema finding in a circle, 1982.

[37] P. Érdi. *Complexity explained*. Springer, 2007.

[38] G Bard Ermentrout and Leah Edelstein-Keshet. Cellular automata approaches to biological modeling. *Journal of theoretical Biology*, 160(1):97–133, 1993.

[39] Faith E. Fich and Colette Johnen. A space optimal, deterministic, self-stabilizing, leader election algorithm for unidirectional rings. In *DISC '01: Proceedings of the 15th International Conference on Distributed Computing*, pages 224–239, London, UK, 2001. Springer-Verlag.

[40] Michael Fischer and Hong Jiang. Self-stabilizing leader election in networks of finite-state anonymous agents. In *Principles of Distributed Systems*, pages 395–409. Springer-Verlag, 2006.

[41] G. N. Frederickson and N. A. Lynch. Electing a leader in a synchronous ring. *JACM*, 34(1):98–115, 1987.

[42] Edward Fredkin and Tommaso Toffoli. Conservative logic. *International Journal of theoretical physics*, (21):219–253, 1982.

[43] Niloy Ganguly, Pradipta Maji, Sandip Dhar, K. Biplab Sikdar, and P. Pal Chaudhuri. Evolving cellular automata as pattern classifier. In *ACRI '01: Proceedings of*

the 5th International Conference on Cellular Automata for Research and Industry, pages 56–68, London, UK, 2002. Springer-Verlag.

[44] M Gardner. The fantastic combinations of john conway's new solitaire game "life". *Scientific American*, 223:120–123, 1970.

[45] Max Garzon. Cellular automata and discrete neural networks. *Physica D: Nonlinear Phenomena*, 45(1):431–440, 1990.

[46] David Edward Goldberg et al. *Genetic algorithms in search, optimization, and machine learning*, volume 412. Addison-wesley Reading Menlo Park, 1989.

[47] David Griffeath and Cristopher Moore. *New constructions in cellular automata*. Oxford University Press, 2003.

[48] J. E. Hanson. *The computation mechanics of cellular automata*. PhD thesis, University Microfilms Intl, Ann Arbor, Michigan, 1993.

[49] J. E. Hanson and J. P. Crutchfield. The attractor-basin portrait of a cellular automaton. *Journal of statistical physics*, 66(5/6):1415–1462, 1992.

[50] L. Higham and S. Myers. Self-stabilizing token circulation on anonymous message passing rings. Technical report, University of Calgary, 1999.

[51] D. S. Hirschberg and J. B. Sinclair. Decentralized extrema-finding in circular configurations of processors. *Commun. ACM*, 23(11):627–628, 1980.

[52] J. H. Holland. *Adaptation in natural and artificial systems*. University of Michigan Press, 1975.

[53] W. Hordijk. *Dynamics, emergent computation, and evolution in cellular automata*. PhD thesis, University of New Mexico, Albuquerque, NM, 2000.

[54] Wim Hordijk. The evca project: A brief history. *Complexity*, 18(5):15–19, 2013.

[55] Wim Hordijk, James P. Crutchfield, and Melanie Mitchell. Mechanisms of emergent computation in cellular automata. In *PPSN V: Proceedings of the 5th International Conference on Parallel Problem Solving from Nature*, pages 613–622, London, UK, 1998. Springer-Verlag.

[56] A. Itai and M. Rodeh. Symmetry breaking in distributed network. In *Proceeding of the 22nd Annual IEEE Symp. of Foundations of Computer Science (FOCS)*, pages 245–260. IEEE Press, 1994.

[57] Gene Itkis, Chengdian Lin, and Janos Simon. Deterministic, constant space, self-stabilizing leader election on uniform rings. In *WDAG '95: Proceedings of the 9th International Workshop on Distributed Algorithms*, pages 288–302, London, UK, 1995. Springer-Verlag.

[58] Stuart A Kauffman. Emergent properties in random complex automata. *Physica D: Nonlinear Phenomena*, 10(1):145–156, 1984.

[59] David B. Knoester, Philip K. McKinley, and Charles A. Ofria. Using group selection to evolve leadership in populations of self-replicating digital organisms. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 293–300. ACM, 2007.

[60] J. R. Koza. *Genetic programming: on the programming of computers by means of natural selection*. The MIT Press, Cambridge, MA, 1992.

[61] J. Lamprecht. Variable leadership in bar-headed geese (anser indicus) - an analysis of pair and family departures. *Trends in Ecology & Evolution*, 122:105–120, 1992.

[62] Chris G Langton. Computation at the edge of chaos: Phase transitions and emergent computation. *Physica D: Nonlinear Phenomena*, 42(1):12–37, 1990.

[63] Christopher G Langton. Self-reproduction in cellular automata. *Physica D: Nonlinear Phenomena*, 10(1):135–144, 1984.

[64] G. Le Lann. Distributed systems, towards a formal approach. *IFIP Congress*, pages 155–160, 1977.

[65] Michael Larsen. The problem of kings. *the electronic journal of combinatorics*, 2(1):R18, 1995.

[66] Peter A. Lawrence. *The Making of a Fly: The Genetics of Animal Design*. Wiley-Blackwell, April 1992.

[67] Joseph T Lizier, Mikhail Prokopenko, and Albert Y Zomaya. Detecting non-trivial computation in complex dynamics. In *Advances in Artificial Life*, pages 895–904. Springer, 2007.

[68] Qiming Lu and Christof Teuscher. Damage spreading in spatial and small-world random boolean networks. *arXiv preprint arXiv:0904.4052*, 2009.

[69] David Lusseau and Larissa Conradt. The emergence of unshared consensus decisions in bottlenose dolphins. *Behavioral Ecology and Sociobiology*, 2009.

[70] Nancy A. Lynch. *Distributed Algorithms*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1996.

[71] Manuel Marques-Pita and Luis Mateus Rocha. Conceptual structure in cellular automata-the density classification task. In *ALIFE*, pages 390–397, 2008.

[72] Alain Mayer, Yoram Ofek, Rafail Ostrovsky, and Moti Yung. Self-stabilizing symmetry breaking in constant-space (extended abstract). In *STOC '92: Proceedings of the twenty-fourth annual ACM symposium on Theory of computing*, pages 667–678, New York, NY, USA, 1992. ACM.

[73] M. Mitchell. *An introduction to genetic algorithms*. MIT Press, 1996.

[74] M. Mitchell, J. P. Crutchfield, and R. Das. Computer science application: Evolving cellular automata to perform computations. In T. Bäck, D. Fogel, and Z. Michalewicz, editors, *HandBook of Evolutionary Computation*. Oxford University Press, 1997.

[75] Melanie Mitchell. *Complexity: A Guided Tour*. Oxford University Press, USA, April 2009.

[76] Radhika Nagpal. A catalog of biologically-inspired primitives for engineering self-organization. In *Engineering Self-Organising Systems*, volume 2977 of *Lecture Notes in Computer Science*, pages 53–62. Springer, 2003.

[77] J. Von Neumann. *Theory of self-reproducing automata - edited and completed by Burks*. Illinois Press, 1966.

[78] Codrin Nichitiu, Jacques Mazoyer, and Eric Rémila. Algorithms for leader election by cellular automata. *J. Algorithms*, 41(2):302–329, 2001.

[79] Norman H Packard. *Adaptation toward the edge of chaos*. University of Illinois at Urbana-Champaign, Center for Complex Systems Research, 1988.

[80] Gary L. Peterson. An o(nlog n) unidirectional algorithm for the circular extrema problem. *ACM Trans. Program. Lang. Syst.*, 4(4):758–762, 1982.

[81] Rene Reynaga and Eligio Amthauer. Two-dimensional cellular automata of radius one for density classification task ρ= 12. *Pattern recognition letters*, 24(15):2849–2856, 2003.

[82] Fred C. Richards, Thomas P. Meyer, and Norman H. Packard. Extracting cellular automaton rules directly from experimental data. pages 189–202, 1990.

[83] Thimo Rohlf, Natali Gulbahce, and Christof Teuscher. Damage spreading and criticality in finite random dynamical networks. *Physical review letters*, 99(24):248701, 2007.

[84] Marco Schneider. Self-stabilization. *ACM Comput. Surv.*, 25(1):45–67, 1993.

[85] Cosma Rohilla Shalizi, Robert Haslinger, Jean-Baptiste Rouquier, Kristina Lisa Klinkner, and Cristopher Moore. Automatic filters for the detection of coherent structure in spatiotemporal systems. *Physical Review E*, 73(3):036104, 2006.

[86] Ilya Shmulevich and Stuart A Kauffman. Activities and sensitivities in boolean network models. *Physical Review Letters*, 93(4):048701, 2004.

[87] M Sipper. Studying artificial life using a simple, general cellular model. *Artificial Life Journal*, (2):1–35, 1995.

[88] A Smith. Simple computation-universal spaces. *Journal of ACM*, 18:339–353, 1971.

[89] A. R. III Smith. Two-dimensional formal languages and pattern recognition by cellular automata. In *Proceeding of the 22nd Annual IEEE Symp. of Foundations of Computer Science (FOCS)*, pages 144–152. IEEE Press, 1971.

[90] C. Sueura and O. Petit. Shared or unshared consensus decision in macaques? *Behavioural Processes*, 78(1):84–92, 2008.

[91] Gerard Tel. *Introduction to Distributed Algorithms*. Cambridge University Press, New York, NY, USA, 2001.

[92] Tommaso Toffoli. Cellular automata as an alternative to (rather than an approximation of) differential equations in modeling physics. *Physica D: Nonlinear Phenomena*, 10(1):117–127, 1984.

[93] Gérard Y Vichniac. Simulating physics with cellular automata. *Physica D: Nonlinear Phenomena*, 10(1):96–116, 1984.

[94] GY Vichniac, P Tamayo, and H Hartman. Annealed and quenched inhomogeneous cellular automata (inca). *Journal of Statistical Physics*, 45(5-6):875–883, 1986.

[95] S. Wolfram. *Theory and Application of Cellular Automata*. Singapore: World Scientific, 1986.

[96] Stephen Wolfram. Cellular automata as models of complexity. *Nature*, 311(5985):419–424, October 1984.

[97] Stephen Wolfram. Cellular automata and complexity, 1994.

[98] Dietmar Wolz and Pedro De Oliveira. Very effective evolutionary techniques for searching cellular automata rule spaces. *Journal of Cellular Automata*, 3(4), 2008.

[99] Andrew Wuensche. Classifying cellular automata automatically. Santa Fe Institute Santa Fe, NM, 1998.

[100] Victor Zhirnov, Ralph Cavin, Greg Leeming, and Kosmas Galatsis. An assessment of integrated digital cellular automata architectures. *IEEE computer*, 41(1):38–44, 2008.

List of Symbols and Acronyms

Acronym	Description
CA	Cellular automaton
CM	Computational mechanics
IC	Initial configuration
EvCA	Evolutionary Cellular Automata research group
DFA	Deterministic finite state automaton
GA	Genetic algorithm(s)