ISSN: 1803-3814 (Printed), 2571-3701 (Online)
https://doi.org/10.13164/mendel.2019.1.147

ANT COLONY OPTIMISATION FOR PERFORMING COMPUTATIONAL TASK IN CELLULAR AUTOMATA

Michal Bidlo[✉], **Jakub Korgo**

IT4Innovations Centre of Excellence, Faculty of Information Technology, Brno University of Technology, Czech Republic
bidlom@fit.vutbr.cz[✉], jakub235@seznam.cz

Abstract
A method is presented for the design of cellular automata rules by means of ant algorithms. In particular, Elitist Ant System and a modified MAX-MIN Ant System are applied to search for transition functions of 1D cellular automata that are able to calculate squares of given input values. It will be shown that the proposed MAX-MIN Ant System can perform significantly better than the standard variant of Elitist Ant System. In particular, in the most advanced case study, the ant algorithm showed an ability to design a complete set of elementary cellular automata rules that fulfil the required square calculations. Some selected results will be presented and their features discussed.

Keywords: *ant colony optimisation, MAX-MIN ant system, cellular automaton, transition function, square calculation.*

Received: 11 April 2019
Accepted: 30 May 2019
Published: 24 June 2019

1 Introduction

Biologically inspired algorithms represent popular techniques for modeling, optimisation and design with applications in various areas. Evolutionary algorithms (with their most popular variants like genetic algorithms [9], genetic programming [11] or evolutionary strategies [18]), inspired by natural evolution, are applied primarily on optimisation problems for which no analytical solution is known or their solution is intractable by means of conventional techniques. Such algorithms utilise a stochastic search in a search space induced by a proper (problem-specific) representation the goal of which is to fulfil some criteria specified by the designer. The solution of a problem is then treated as finding its suitable form with respect to the representation while optimising an objective function based on the given criteria.

However, the evolution is not the only natural phenomenon that may be taken as an inspiration for creating optimisation metaheuristics. Some other (nowadays popular) techniques are inspired by processes that can be observed in socially living organisms. Typically, swarm intelligence [13] or various insect-based behaviour like ant colonies [7] or bee colonies [12] have adopted such strategies in order to construct robust and powerful optimisation algorithms. In this paper our attention will be focused on ant algorithms.

The idea of ant-inspired algorithms was originally mentioned by Colorni, Dorigo and Maniezzo in early 1990s and proposed as a method for optimising the Traveling Salesman Problem (TSP) [3]. Later, new variants of the original algorithm arose and some other applications were involved, e.g. routing, scheduling, machine learning and others [7]. These techniques are currently referred to as Ant Colony Optimisation (ACO).

The goal of this paper is to apply the ACO principles on the problem of designing transition rules of cellular automata (CA). In particular, we will show that by introducing a proper form of the construction graph for the ant algorithm, a transition function (that is composed of the transition rules) for the CA may be derived by a stochastic traverse of ants through the graph as performed in the original ACO experiment. It will be demonstrated that non-trivial multi-state cellular automata can be constructed that exhibit a given computational task – the square calculation. Although the ACO in combination with CA has already been studied before (see Section 2), an approach for generating transition rules on an elementary level and its analysis has not yet been performed.

2 Related Work

In this section the related work regarding cellular automata and ACO algorithms is summarised. In particular, some relevant references involving combination of ACO and CA will be mentioned.

2.1 Ant colony optimisation

Ant-inspired techniques have extensively been studied since the introduction of their simplest form — the Ant System — in 1991, that was later revisited and extended in [6]. Probably the most known ACO variants are the

Ant Colony System [5], Elitist Ant System [6] or MAX-MIN Ant System [20]. Although originally designed for discrete (combinatorial) optimisation, the ACO approach was subsequently applied to continuous optimisation (e.g. see [17]). ACO has recently been studied in various application domains. For example, Martens et al. proposed an AntMiner+ — a technique based on MMAS — in the field of data mining and rule-based classification [15]. In [8] the authors proposed a novel block cipher using reversible and irreversible 1D CA with an ACO-based extension in order to establish a more secure communication channel. Another CA-based approach supported by an ant algorithm was proposed in [23] in order to optimise rules for obstacle avoidance in robot soccer. Thilak and Amuthan studied a CA-based improved ACO algorithm for mitigating DDoS attacks in vehicular ad hoc networks [21]. As the last example we mention a study regarding the utilisation of geographical cellular automata optimised by ACO for simulating the evolution of complex geographical phenomena [14]. As evident from this survey, there are some resources regarding the study of CA using ACO mainly in specific application areas. A more basic approach, however, which could reveal some elementary knowledge of designing and optimising CA by means of ant algorithms rather represents a rare case. Therefore, this paper tries to achieve this objective.

2.2 Cellular automata

The concept of cellular automata was introduced by von Neumann in [16] as a mathematical model for studying the behaviour of complex dynamical systems. The regular structure with only local interactions of the cells and massively parallel nature makes the CA potentially suitable computational platform for future technologies. Cellular automata have been studied both theoretically and in real-world applications. For example, some analysis of computational power of CA were published, in addition to von Neumann's original work, by Codd [2], Sipper [19], Ilachinski [10] or Dewdney [4]. These studies showed that CA can be considered as a computational platform, similar to an ordinary computer, that may be "programmed" to perform a given task. The problem adopted for this paper considers a specific operation – squaring of integer numbers – which is inspired by one of its solutions proposed by Wolfram in [22]. This operation was recently investigated by Bidlo in [1] and it was shown that a *generic* solution of squaring in 1D CA may successfully be solved by means of genetic algorithms. The utilisation of evolutionary approach reflects the fact that designing transition functions for CA represents a difficult task because it is not possible to use conventional programming techniques. In this paper, we adopt ant algorithms for designing cellular automata rules, the details of the case study chosen for our experiments will be described in Section 4.2.

3 The ACO Techniques

In the following paragraphs a technical description of relevant ant algorithms is provided which represents a basis for creating our variant of ACO for designing the CA rules. The proposed method utilises the Elitist Ant System (EAS) and MAX-MIN Ant System (MMAS) which both are based on the basic Ant System (AS). Therefore the AS will be described first.

3.1 Ant System

The Ant System in its original form was designed for optimising the solutions of TSP [6]. A set of cities N is represented as nodes in a graph $G = (N, E)$, where E is a set of edges (i, j), representing paths between cities $i, j \in N$ of a given length $\eta_{ij} = distance(i, j)$. The Ant System simulates traversing a population of m artificial ants through the graph in order to search for the shortest path that visits each city (node) exactly once. As an analogy to natural ants, an ant determines its "step" from city i to j on the basis of a pheromone intensity associated with every $(i, j) \in E$ which indicate a suitability of a particular path. At the beginning of the algorithm, the pheromones acquire an initial value

$$\tau_0 = \tau_{ij} = m/F^{nn}, \quad \forall (i, j) \in E, \tag{1}$$

where F^{nn} is a suitable constant. A current "generation" of ants traverses through the graph, each ant k $(k = 1, \ldots, m)$ builds a path, measures its length and subsequently puts an amount of pheromone $\Delta\tau_{ij}^k$ on each edge of the traversed path $T^k \subset E$ with respect to its total length F^k. Therefore, the pheromone update of (i, j) by all ants may be expressed as

$$\tau_{ij} \longleftarrow \tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}^k, \quad \forall (i, j) \in E, \tag{2}$$

where

$$\Delta\tau_{ij}^k = \begin{cases} 1/F^k, & (i, j) \in T^k \\ 0, & (i, j) \notin T^k. \end{cases} \tag{3}$$

Each ant k of the next generation, that builds its path, chooses a "step" from a node i to a next node j stochastically on the basis of probabilities calculated from the pheromone values and a problem-specific information η_{ij} (e.g. in TSP for $\forall(i,j) \in E : \eta_{ij} = distance(i,j)$):

$$p_{ij}^k = \begin{cases} \frac{[\tau_{ij}]^\alpha [\eta_{ij}]^\beta}{\sum_{l \in \mathcal{N}_i^k} [\tau_{il}]^\alpha [\eta_{il}]^\beta}, & j \in \mathcal{N}_i^k \\ 0, & j \notin \mathcal{N}_i^k, \end{cases} \tag{4}$$

where α, respective β, allows influencing the preference of pheromones, respective the problem-specific information, during building the path and $\mathcal{N}_i^k$ is a set of cities not yet visited by ant k while being in city i (note that visiting a city more than once violates the rule of TSP, hence the probability of going to a city $j \notin \mathcal{N}_i^k$ has assigned probability 0). The probabilistic selection prefers shorter paths and by performing a sufficient number of generations the total path length is optimised. At the beginning of each generation, the pheromones undergo an "evaporation", that also occurs in nature, using an evaporation factor $p \in (0;1)$:

$$\tau_{ij} \longleftarrow (1-p)\tau_{ij}, \quad \forall(i,j) \in E. \tag{5}$$

This process repeats until a generation limit is reached or a satisfactory solution found.

3.2 Elitist Ant System and MAX-MIN Ant System

In order to improve the AS performance, Elitist Ant System (EAS) was introduced [6]. In addition to AS, the pheromone update is influenced by the best path T^{best} found so far the value of which is denoted as Δt_{ij}^{best} and calculated according to (3). Moreover, a new parameter e is introduced which allows determining the amount of pheromone contribution from the best path. The pheromones are updated as follows:

$$\tau_{ij} \longleftarrow \tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}^k + e\Delta\tau_{ij}^{best}, \quad \forall(i,j) \in E. \tag{6}$$

The MAX-MIN Ant System (MMAS) [20] represents an advanced ACO algorithm the goal of which is mainly to improve its ability to recover from getting stuck in local optima. The following modifications were introduced:

1. Pheromones are updated on the basis of the best solution only (found in the current iteration or found so far).
2. A pheromone range $\langle\tau_{min}, \tau_{max}\rangle$ was introduced.
3. In case that a stagnation is detected, MMAS may reinitialise the pheromone values to τ_{max}.

After the evaporation phase, the MMAS updates the pheromones as follows:

$$\tau_{ij} \longleftarrow \tau_{ij} + \Delta\tau_{ij}^{best}, \quad \forall(i,j) \in E, \tag{7}$$

where $\Delta\tau_{ij}^{best}$ may be calculated either from the best path found in the current generation as $\Delta\tau_{ij}^{best} = 1/F^{\mathrm{bgener}}$ or from the best path found so far using $\Delta\tau_{ij}^{best} = 1/F^{\mathrm{bsofar}}$. This choice is problem specific. For more details the reader is referred to [7].

4 ACO for Cellular Automata Design

In this paper, experiments were conducted regarding the design of CA rules by means of ACO algorithms. In particular, the EAS and a modified MMAS were applied to design transition rules of 1D CA with the calculation of the square of integer numbers as a case study. This section describes the proposed construction graph for the ACO, the details of the case study together with the way of evaluating candidate solutions and the experimental setup used for obtaining the results.

4.1 The construction graph

In order to design transition functions for CA using ant algorithms, a suitable form of construction graph needs to be developed. As we ought to design a complete transition function (i.e. for every possible combination of states in cellular neighbourhood a new state needs to be determined), a fully interconnected graph is not an option (e.g. we could not ensure generating all rules or a test of duplicite rules would be needed).

For simplicity, consider a binary CA with 3-cell neighbourhood. The following concept, based on an oriented construction graph, shown in Fig. 1, will be considered. There are two "rows" of nodes, the first row includes all possible rules with the new state 0, the second row includes all possible rules with the new state 1 (see the description inside the nodes). In particular, for the sample binary CA with 3-cell neighbourhood, the construction graph would consist of 2 rows, each containing $2^3 = 8$ nodes. In general, for a q-state CA, the graph will be composed of q rows, each containing q^3 nodes. An additional Start-node is created that represents a starting point for ants.

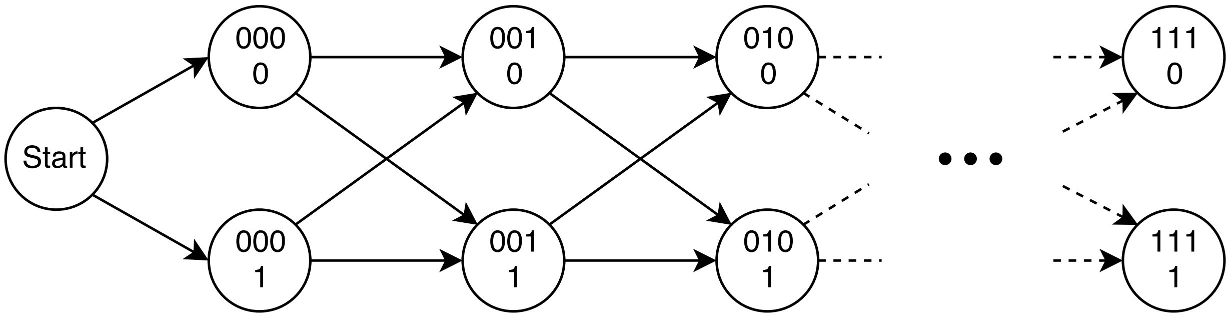
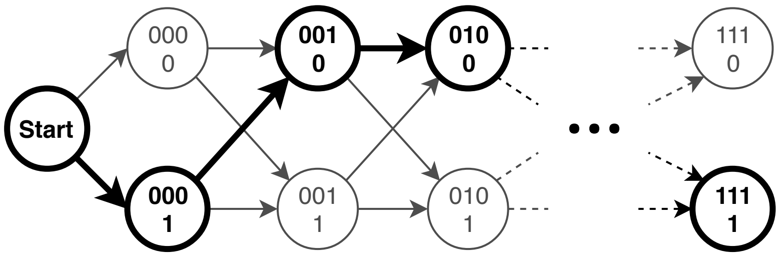


Figure 1: A construction graph for designing transition functions of binary CA with 3-cell neighbourhood by means of ant algorithms. The description of the nodes represents a combination state in the neighbourhood and a new state.

An ant constructs a transition function by traversing through the graph from the Start-node to a right-most node. In each step the ant decides which row to choose from a current node. The node in row r, that the ant chose, represents a rule $c_{i-1}c_ic_{i+1} \rightarrow c_i^{new}$, where c_{i-1}, c_i, c_{i+1} denote a combination of states in the neighbourhood of the i^{th} cell and c_i^{new} is a new state of the i^{th} cell determined for this combination as specified inside the nodes. For example, consider a path of an ant through the graph denoted by thick edges in Fig. 2a. From the Start-node the ant chose a node in the second row which represents rule $000 \rightarrow 1$, the next node chosen by the ant is in the first row, producing rule $001 \rightarrow 0$. Subsequently, a rule $010 \rightarrow 0$ is specified by the next node in the first row and finally (after the last step of the ant), rule $111 \rightarrow 1$ is used. The corresponding transition function composed of these rules is given by a table in Fig. 2b.



(a) A sample tranverse of ant through the graph from Fig. 1.

c_{i-1}	c_i	c_{i+1}	c_i^{new}
0	0	0	1
0	0	1	0
0	1	0	0
	...		...
1	1	1	1

(b) The corresponding rules of the resulting transition function.

Figure 2: Example of creating a transition function for CA by traversing an ant through the construction graph.

4.2 The case study: square calculations in 1D CA

In this paper, 1D uniform cellular automata are considered, the transition rules of which will be designed by the aforementioned EAS and MMAS. The CA is treated as a linear arrangement of cells, each of which may at a given moment acquire a state from a finite set of states. A new state of the i^{th} cell depends on the states of its neighbours ($c_{i-1}c_ic_{i+1}$), i.e. it is a case of the 3-cell neighbourhood. *A step* of the CA is considered as a synchronous update of state values of all its cells according to a given transition function. For the practical implementation purposes, zero boundary conditions are considered (i.e. the non-existent neighbours of the boundary cells are considered as cells in state 0). However, it is important to note that CAs with sufficient sizes are used in this paper in order to avoid affecting its behaviour by the boundary cells.

The objective is to find (using ACO) suitable transition rules in order to calculate the square of integer numbers encoded in the CA state as follows. A value x is represented in the initial state as a continuous sequence of cells in state 1 the length of which corresponds to x; the other cells possess state 0. For example, the state of a 12-cell CA, which encodes $x = 3$, can appear as 0000011100000. The result $y = x^2$, that will emerge from the initial state in a finite number of CA steps, is required to be a stable continuous sequence of cells in arbitrary non-zero states the length of which equals x^2, the other cells are required in state 0. For the aforementioned example, the result can appear as 002222222220 or even 023231323200 (there is a sequence of non-zero cells of length $3^2 = 9$ cells). This is a generalised interpretation based on the original idea presented in [22], page 639.

4.3 Evaluation of solutions created by ants

The goal of the ant algorithms is to search for transition rules for the CA that will be able to calculate the square for a given value x from an integer range $\langle x_{min}; x_{max}\rangle$. Since the computational effort needed to evaluate a CA depends on the value of x, the number of cells w and the maximal number of steps t_{max} is set to $w = 4x^2$ and $t_{max} = 5x^2$ in order to reduce the evaluation time. This settings was determined experimentally in order to provide a sufficient resources (cells) and time to calculate the square. For example, if $x = 3$, the CA consists of $w = 4 \cdot 3^2 = 36$ cells (with the initial state 000000000000111000000000000) and the maximal number of steps $t_{max} = 5 \cdot 3^2 = 45$. In order to evaluate candidate transition rules created by an ant, the CA is developed from its initial state until one of the following events occur. This event will influence the fitness F_x of the CA, calculated for a given input value x, which represents the evaluation of the rules with respect to calculating the power x^2.

1. If the state of a boundary cell is changed, $F_x = 0$ as we do not want the behaviour to be influenced by the boundary cells.
2. If the number of steps exceeds t_{max} and the CA has not reached a stable state, then $F_x = 0$.
3. If no cell changed its state since the last step (i.e. a stable state was reached), the state is evaluated as described in the following paragraph.

Let p denote a position of a cell, $0 < p < w - x^2$, determining a continuous sequence of cells at positions $p, p+1, \ldots, p+x^2-1$ in the CA. Let g_p be the number of cells in non-zero states inside this sequence and b_p be the number of cells in non-zero states outside this sequence. The evaluation of the stable CA state is performed as finding such a p for which the difference $g_p - b_p$ is maximal. In other words, for the correct result of x^2, represented by the CA state, it is required this sequence to contain all cells in non-zero states, the remaining cells are required to be in state 0. Mathematically expressed, for $0 \leq i < w$ and $0 < p < w - x^2$:

$$r_i = \begin{cases} 0 & \text{if } c_i = 0 \\ 1 & \text{otherwise} \end{cases} \tag{8}$$

$$g_p = \sum_{j=p}^{p+x^2-1} r_j \tag{9}$$

$$b_p = \sum_{j=0}^{p-1} r_j + \sum_{j=p+x^2}^{w-1} r_j \tag{10}$$

$$f = \max_{0<p<w-x^2} (g_p - b_p) \tag{11}$$

Then the fitness value F_x of a solution produced by an ant is evaluated as follows (note the square root from a normalized value, that we determined experimentally, which showed very helpful especially for improving the convergence of EAS):

$$F_x = \begin{cases} \sqrt{\frac{f}{x^2}} & \text{for } f \geq 0 \\ 0, & \text{for } f < 0 \end{cases} \tag{12}$$

In order to calculate fitness for a range of values $\langle x_{min}; x_{max}\rangle$, we used a weighted average

$$F_{\langle x_{min}; x_{max}\rangle} = \frac{\sum_{x=x_{min}}^{x_{max}} xF_x}{\sum_{x=x_{min}}^{x_{max}} x} \tag{13}$$

This value is used to calculate the pheromone updates during computation of the ant algorithm using (3). Analogically, the best fitness of the current generation F^{bgener} and the best fitness found so far F^{bsofar} are used as described in Section 3.2.

4.4 Experimental settings of the ant systems

Since the ACO techniques allow setting of a higher range of control parameters, it is intractable due to a significant computational effort to perform an in-depth analysis in order to determine the optimal settings. Therefore, the parameter values used in our experiments were chosen according to the literature and our initial experimentation with the system. The settings for the EAS and MMAS are summarised in Table 1. Note that the product of the number of ants and the number of generations is the same, therefore, we can compare adequately the properties of both algorithms.

Table 1: Parameter settings for the EAS and MMAS utilised for experiments.

Parameter name	**Elitist AS**	**MAX-MIN AS**
The number of ants (`ANT_COUNT`)	60000	750
Maximal number of generations	250	20000
The amount of initial pheromones	`ANT_COUNT`/100	τ_{max}
Pheromone evaporation factor v	0.3	0.005
Elitist factor e	`ANT_COUNT`/200	-
Global best solution preference	1 (always)	0.85
`PHEROMONE_RANGE` (to determine τ_{min} from τ_{max})	-	10000
`STAGNATION_CHECK` (the num. of iterations)	-	20
Pheromone `RESET_LIMIT` (the num. of stagnations)	-	6
The influence s for reseting pheromones towards τ_{max}	-	0.025

A significant feature of the proposed ant system is that the pheromones are associated with *nodes* of the construction graph. This approach was chosen because no (measurable) dependency is known between various transition rules that are generated by traversing the graph. Hence there is no problem-specific information (in addition to the transition rules represented by the nodes) that could be directly associated with the graph. This means that only the pheromone values, calculated on the basis of the fitness given by (13), are used to control the ants traversing the graph. As a consequence, equation (4) for calculating the probability of selecting a next node j from a current node i by ant k can be simplified to

$$p_j^k = \frac{\tau_j}{\sum_{l \in \mathcal{N}_i^k} \tau_l}, \tag{14}$$

where $i, l \in \mathcal{N}_i^k$ and $\mathcal{N}_i^k$ is a set of nodes to which ant k can go from node i and τ_l is the pheromone value of each node from this set.

In order to improve the performance, several modification were introduced into the MMAS as described in the following paragraphs. These modifications are based on our initial experiments in which they showed significant improvements in the convergence of the MMAS and reduction of a risk to get stuck in local optima. The results from the modified MMAS will be compared to the approach using EAS (that was described in Section 3.2).

At the beginning, a CA that preserves the initial state (i.e. with transition rules of the form $c_{i-1}\, c_i\, c_{i+1} \to c_i$) is considered and its fitness F is calculated as described in Section 4.3. Then the initial values of τ_{max} and τ_{min} are determined as

$$\tau_{max} = F/v \text{ (for } v \text{ see Table 1)}, \tag{15}$$

$$\tau_{min} = \tau_{max}/\texttt{PHEROMONE\_RANGE}. \tag{16}$$

The value of τ_{max} is set as the initial pheromone value.

After each iteration of the MMAS, a τ_{tmp} is calculated according to (15) using the fitness of the best ant. If $\tau_{tmp} > \tau_{max}$, then $\tau_{max} = \tau_{tmp}$ and τ_{min} is updated according to (16). After updating the pheromones, their values are adapted according to τ_{min} and τ_{max}.

In case that a fitness stagnation is detected, a reinitialisation of pheromone values is performed in order to support exploration activity of the ant system. The average fitness of the ant population is utilised for this purpose. In particular, after each `STAGNATION_CHECK` iterations, the average fitness is calculated and compared to the previous value in order to detect stagnation. The stagnation is indicated if the average fitness or global maximal fitness has not increased since the last stagnation check. If the stagnation is detected in a series of `RESET_LIMIT` checks, then the pheromone reinitialisation is performed as follows.

Instead of reseting the values directly to τ_{max} (as usual in the original version of MMAS), a new parameter s was introduced which allows us to influence a rate of reseting the pheromones *towards* τ_{max} as follows:

$$\tau_i \longleftarrow \tau_i + s(\tau_{max} - \tau_i). \tag{17}$$

5 Experimental Results

In order to evaluate the proposed ant algorithms, experiments were conducted using various specifications of the case study. The applied algorithms (in particular, the Elitist Ant System and modified MAX-MIN Ant System) showed significant differences in their performance, therefore we present the results separately for each specification and provide a relevant discussion. In all experiments, the CA working with 6 cell states were considered.

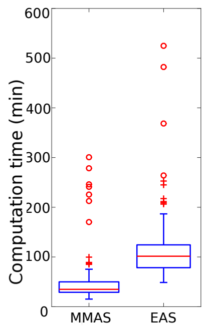
5.1 Calculating the square for a single value

The simplest variant of the case study considered finding rules for a CA that will be able to calculate the square for a given value $x = 15$. This experiment was conducted for its simplicity in order to perform basic comparison of the EAS and MMAS.

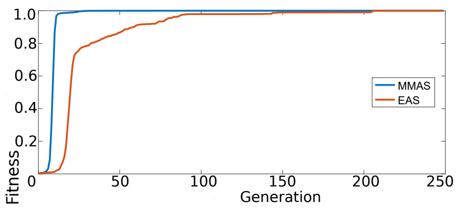
As evident from Fig. 3a, both algorithms fully succeeded in searching for a working solution within the given generation limit. However, a significant difference can be observed in the computational effort. While the median of the number of generations needed to find a working solutions is 963 (compared to 34 in case of the EAS), there was significantly less evaluations needed in the MMAS with respect to the population size. In particular, the median of the number of evaluations for the MMAS is $963 \cdot 750 = 722250$, the EAS needed $34 \cdot 60000 = 2.04$ million evaluations. The statistical comparison of the computational time is shown in Fig. 3b. Finally, we conducted a comparison of the evolution of average fitness for both algorithms and the results are summarised in 3c (note that the generations of the MMAS are re-scaled to the range observed in the EAS). This comparison confirms the initial observation that the performance of the MMAS is significantly better than the EAS since a working solution can, in average, be found in remarkably shorter time.

	Ant algorithm	
	MMAS	EAS
Success rate (%)	100	100
The num. of generations (a median)	963	34
The num. of evaluations (a median)	722250	2.04 million
Computational time (a median in minutes)	34.8	101.5

(a) Mean numbers regarding the computational effort.



(b) Statistical summary of the computational time.



(c) Average fitness development (note that a single generation of the EAS corresponds to 80 generations of the MMAS).

Figure 3: Comparison of the performance of the MMAS and EAS for solving a square calculation in a CA for a given input value $x = 15$.

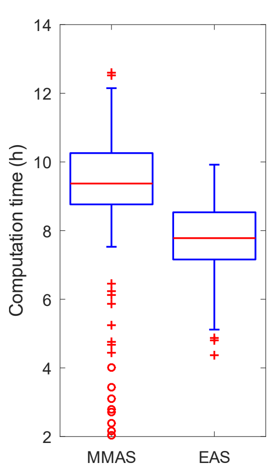
5.2 Calculating the square for x in a range $4 \ldots 9$

Motivated by a previous study published in [1], where generic square calculating CA were discovered, our advanced experiment is focused on finding rules for a CA that will be able to calculate the square for a range of the input values, in particular, for $x = 4, \ldots, 9$. Since it is impossible to evaluate infinite number of possible values, this approach utilises a *training* of the target CA on a reasonable range of the input values and subsequently a *verification* of the resulting CA on larger values. Again, the objective is to evaluate abilities of the EAS and MMAS in solving such a more complex problem.

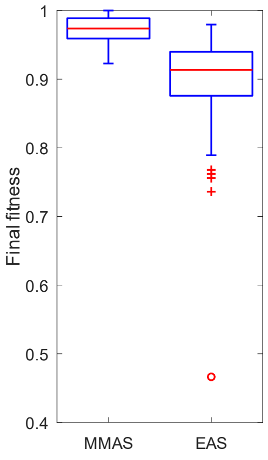
In this case the Elitist Ant System was not able to find any working solution within the given generation limit. The average computational time of a single run is approximately 8 hours which is primarily caused by the most expensive part – performing the CA development for each input value. Therefore, the increase of the generation limit did not represent a reasonable option. As evident from the fitness development (Fig. 4c), at the beginning the EAS is able to optimise the solution but in a later phase of the algorithm, a disadvantage of the EAS appeared – getting stuck in local optima. The proposed MMAS, however, succeeded in 19% of runs and found several working solutions.

A statistical evaluation of both algorithms is shown in Fig. 4a (a comparison of the computational time) and 4b (a comparison of final fitness values). Although the time of computation seems to be better for the EAS, this does not mean that the EAS performs better. It can be observed that these lower values were caused primarily by solution whose evaluation (the CA development) was terminated in advance due to achieving a stable state that, however, did not represent a correct solution. On the other hand, note several short-time runs in case of the MMAS which are those where working solutions were found. The comparison of the final fitness shows a significant advantage of the MMAS that is able to better explore the search space and, in general, find solutions with higher fitness.

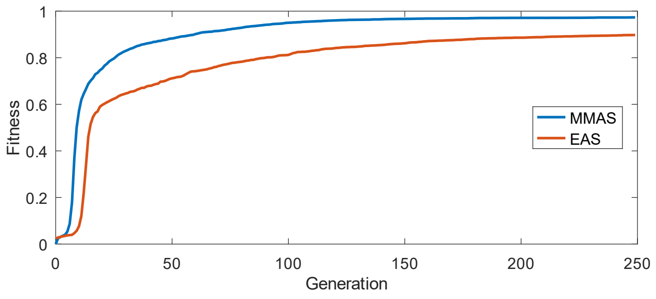
Unfortunately, none of the solutions found by the MMAS was identified as general. More specifically, the resulting CA is able to correctly calculate the square only for the values utilised in the training.



(a) Statistical summary of the computational time.



(b) Statistical summary of the final fitness obtained after terminating the algorithm.



(c) Average fitness development (note that the generations of the MMAS are scaled with respect to the generations of the EAS).

Figure 4: Comparison of the performance of the MMAS and EAS for solving a square calculation in a CA for x in range 4 . . . 9.

5.3 Applying MMAS on solving the square for x in a range 4 . . . 12

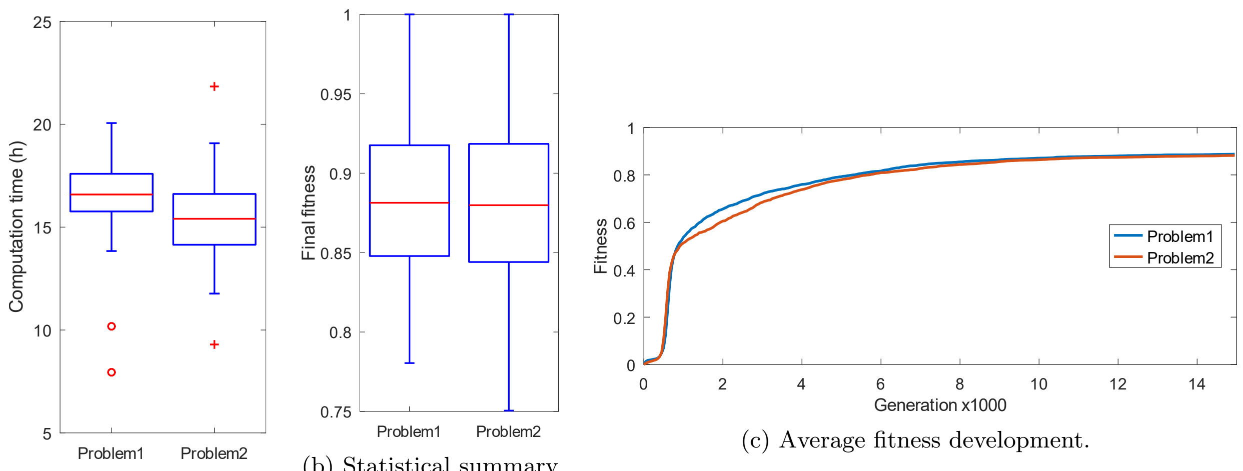
For an advanced evaluation of the MMAS, the most complex experiment conducted in this paper considers finding rules for a CA calculating the square of x in a range 4 . . . 12. The enhancement of the range of training input values is motivated by the aim to obtain a generic solution. In addition to the original encoding of the values in CA (as described in Section 4.3, this will be denoted as problem P#1 herein), the following modified problem setup was considered (denoted as problem P#2, inspired by a similar approach from [22], page 639). The input value is stored in the initial state as a sequence of x cells in state 2, followed by a single cell in state 1. For example, $x = 4$ is encoded as 00022221000. The result will be interpreted as a stable state containing a continuous sequence of x^2 cells with state values ≥ 2, all the other cells possessing states ≤ 1.

These experiments provided two working solutions of problem P#1 a single working solution of problem P#2. As shown in Fig. 5, the MMAS exhibits a similar behaviour in solving these problems. A very low success rate could partially be expected due to a more complex problem requirements.

Although all working solutions are able to calculate the square of values in the given range, none of them works for larger values. This situation can be explained by analysing the corresponding CA development which is, for two selected solutions of P#1 and P#2, shown in Fig. 6. In both cases, the CA development, visualised for different input values, exhibits rather chaotic behaviour instead of a systematic emergence of the results. For example, in case of solving problem P#1, Fig. 6a and 6b shows a development of some "strips", propagating from left to right, that sometimes interact with each other (Fig. 6b). As regards the CA for problem P#2, whose visualisation is shown in Fig. 6c and 6d, a propagating complex structure can be observed (Fig. 6d) that seems to cause a stable state for $x = 6$ but is not able to work for higher input values that were not considered for training. Our observations of both the problems, calculating the square of x which is beyond the training range, indicate that this behaviour can not, in general, achieve a correct result value.

5.4 Discussion

In summary, none of the experiments provided a generic solution for the square calculation in CA. Nevertheless, the results showed that the proposed MMAS is able to design, at an elementary level, complex cellular automata that have probably not been published before. The failure of obtaining generic solutions may have several causes.



(a) Statistical summary of the computational time.

(b) Statistical summary of the final fitness obtained after terminating the algorithm.

(c) Average fitness development.

Figure 5: Comparison of the MMAS applied to the problems P#1 and P#2 of square calculations in CA.

For example, the approach to constructing a complete table-based transition function may be too difficult to be ever able to provide generic solutions. Note that the previous work in this area, published in [1], utilised an advanced transition function representation based on conditional rules. However, the problem is that no method is known for the design of the construction graph for ant algorithms that would be efficient for a given CA representation. Therefore, its solution represent a highly experimental work. Another cause may lie in tuning the ant algorithm itself which, on its own, represents a difficult optimisation problem. We believe, however, that similar experiments may significantly contribute both to improve understanding of the CA design and help to search for more efficient ant algorithms for this task.

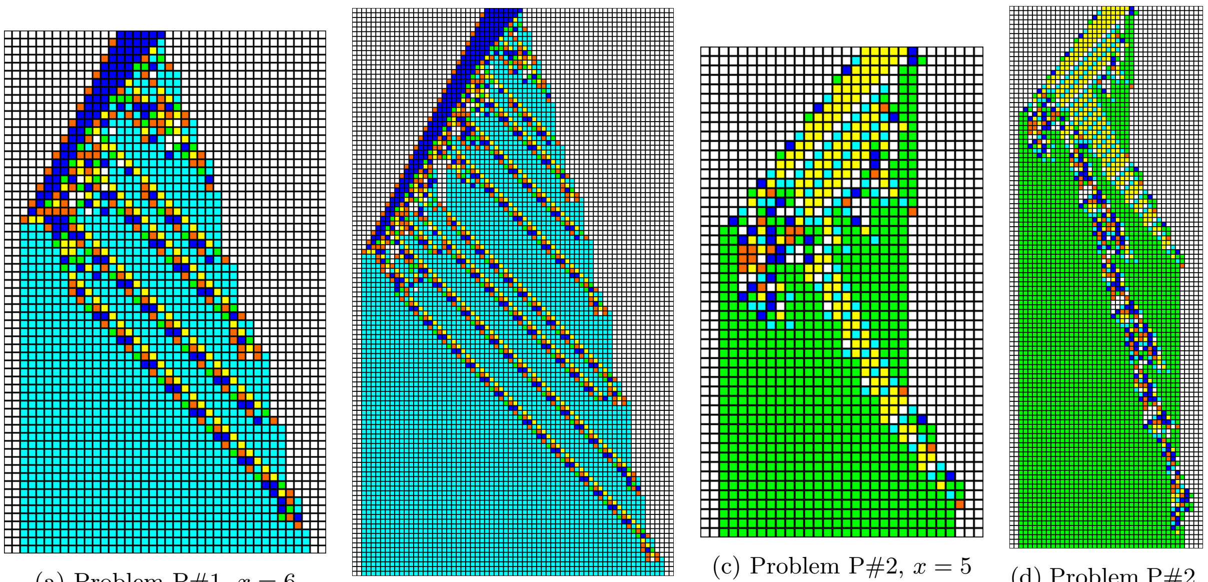
6 Conclusions

A method was presented for the design of cellular automata rules by means of ant algorithms. In particular, Elitist Ant System and a modified MAX-MIN Ant System was applied to search for transition functions of 1D cellular automtata that are able to calculate squares of given input values. The experiments showed that the proposed MAX-MIN Ant System can perform significantly better than the standard variant of Elitist Ant System. In the most advanced case study, the ant algorithm was able to discover cellular automata that calculate the square of a given range of input values. Although no generic solution was achieved, the ant algorithm showed an ability to design a complete set of elementary cellular automata rules that fulfil a required non-trivial specification. We believe that such experiments may significantly contribute to improve the understanding of cellular automata design and help to search for more efficient ant algorithms to solve this task. Therefore, a further tuning of the ant systems, searching for other representations of transition functions and designing appropriate construction graphs represent main ideas of our future research.

Acknowledgement: This work was supported by The Ministry of Education, Youth and Sports of the Czech Republic INTER-COST project Advanced Methods of Nature-Inspired Optimisation and HPC Implementation for the Real-Life Applications – LTC18053, and Large Infrastructures for Research, Experimental Development and Innovations project of the IT4Innovations National Supercomputing Center – LM2015070.

References

[1] Bidlo, M. 2016. On routine evolution of complex cellular automata. *IEEE Transactions on Evolutionary Computation* 20, 5, pp. 742–754.
[2] Codd, E. F. 1968. *Cellular Automata.* Academic Press, New York, USA.
[3] Colorni, A., Dorigo, M., and Maniezzo, V. 1991. Distributed optimization by ant colonies. In *Proceedings of ECAL91 - European Conference on Artificial Life.* Paris, France, pp. 134–142.
[4] Dewdney, A. K. 1990. Computer recreations. *Scientific American* 262, 1, pp. 146–149.
[5] Dorigo, M. and Gambardella, L. M. 1997. Ant colony system: a cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation* 1, 1, pp. 53–66.



(a) Problem P#1, $x = 6$

(b) Problem P#1, $x = 8$

(c) Problem P#2, $x = 5$

(d) Problem P#2, $x = 6$

Figure 6: Demonstration of the CA development for selected solutions.

[6] Dorigo, M., Maniezzo, V., and Colorni, A. 1996. Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 26, 1, pp. 29–41.

[7] Dorigo, M. and Stützle, T. 2004. *Ant Colony Optimization.* MIT Press, USA.

[8] Hanin, C., Omary, F., Elbernoussi, S., Achkoun, K., and Echandouri, B. 2018. A new block cipher system using cellular automata and ant colony optimization (bc-caaco). *International Journal of Information Security and Privacy (IJISP)* 12, 4, pp. 54–67.

[9] Holland, J. H. 1975. *Adaptation in Natural and Artificial Systems.* University of Michigan Press, Ann Arbor, USA.

[10] Ilachinski, A. 2001. *Cellular Automata: A Discrete Universe.* World Scientific, Singapore.

[11] Koza, J. R. 1992. *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* MIT Press, USA.

[12] Karaboga, D. and Basturk, B. 2007. A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (abc) algorithm. *Journal of Global Optimization* 39, 3, pp. 459–471.

[13] Kennedy, J. and Eberhart, R. 1995. Particle swarm optimization. In *Proceedings of ICNN'95 - International Conference on Neural Networks.* IEEE, Vol. 4, pp. 1942–1948.

[14] Liu, X., Li, X., Yeh, A. G. O., He, J., and Tao, J. 2007. Discovery of transition rules for geographical cellular automata by using ant colony optimization. *Science in China Series D: Earth Sciences* 50, 10, pp. 1578–1588.

[15] Martens, D., De Backer, M., Haesen, R., Vanthienen, J., Snoeck, M., and Baesens, B. 2007. Classification with ant colony optimization. *IEEE Transactions on Evolutionary Computation* 11, 5, pp. 651–665.

[16] von Neumann, J. 1966. *The Theory of Self-Reproducing Automata.* A. W. Burks (ed.), University of Illinois Press, USA.

[17] Pourtakdoust, S. H. and Nobahari, H. 2004. An extension of ant colony system to continuous optimization problems. In: *Ant Colony Optimization and Swarm Intelligence.* Springer, Berlin, Heidelberg, pp. 294–301.

[18] Rechenberg, I. 1973. *Evolutionstrategie: Optimierung Technischer Systeme nach Prinzipien der Biologischen Evolution.* Frommann-Holzboog, Stuttgart, Germany [in German].

[19] Sipper, M. 1997. *Evolution of Parallel Cellular Machines – The Cellular Programming Approach*, Lecture Notes in Computer Science, Vol. 1194. Springer, Berlin, Germany.

[20] Stützle, T. and Hoos, H. H. 2000. Max–min ant system. *Future Generation Computer Systems* 16, 8, pp. 889–914.

[21] Thilak, K. D. and Amuthan, A. 2018. Cellular automata-based improved ant colony-based optimization algorithm for mitigating ddos attacks in vanets. *Future Generation Computer Systems* 82, pp. 304–314.

[22] Wolfram, S. 2002. *A New Kind of Science.* Wolfram Media, Champaign IL, USA.

[23] Zhou, K., Ma, S., Zhu, X., Tang, L., and Feng, X. 2010. Improved ant colony algorithm based on cellular automata for obstacle avoidance in robot soccer. In *2010 3rd International Conference on Computer Science and Information Technology.* IEEE, Chengdu, China, vol. 5, pp. 298–302.