# A Simulation-Based Analysis of Interdependent Populations in a Dynamic Ecological Environment

Kristiyan Balabanov, Doina Logofătu, Costin Badica, Florin Leon



# A Simulation-Based Analysis of Interdependent Populations in a Dynamic Ecological Environment

Kristiyan Balabanov[1] ✉, Doina Logofătu[1], Costin Badica[2], and Florin Leon[3]

[1] Frankfurt University of Applied Sciences,
Department of Computer Science and Engineering,
60318 Frankfurt a.M., Germany
balabano@stud.fra-uas.de
[2] University of Craiova,
Department of Computer Sciences and Information Technology,
200285 Craiova, Romania
cbadica@software.ucv.ro
[3] Gheorghe Asachi Technical University of Iaşi
Faculty of Automatic Control and Computer Engineering
700050 Iaşi, Romania
florinleon@gmail.com

**Abstract.** With the ever increasing computational power of computers, simulation-based approaches have become a feasible testing technique that does not require investing valuable resources to create an actual prototype. Thus, design changes can be introduced and design errors can be fixed before it is too late, making simulation a cheaper, safer and often more acceptable from an ethical perspective approach. In our paper we summarize the results from the analysis with the help of a computational simulation of a simple, yet analytically intractable problem scenario from the field of ecology. Our main goal is to confirm that even with a seemingly simple agent-based model and simulation, one could obtain plausible results regarding a system's real life behavior. As a last point we propose a more efficient alternative for analysis, rather than the more expensive simulation.

**Keywords:** Population dynamics · Ecological simulation · Agent-based modeling · Predator-prey relation · Evolution-inspired optimization.

## 1 Introduction

Simulation is a a well-known and wide-spread method of system analysis, and a lot of different branches in industry/science make use of them due to the various advantages that they offer. As an example, it is more economical to test the aerodynamics of a car without the need of an actual prototype [12]. On the other hand, in some projects it is simply the most feasible approach with respect to effort-cost ratio, e.g. to simulate a space rover's mission to another planet [15].

Certain phenomena occur at a rate either too fast or too slow to be efficiently observed in real-time, such as explosions or plant growth. Furthermore, a well-designed simulation can not only save time and money, but also spare lives, for instance when conducting air-bag tests or in any experiments with living organisms. One could refer to the latter as ecological simulation, which has become quite popular recently due to the numerous environmental issues resulting from globalization and industrialization. Despite still existing skepticism, it is evident by now that both have lead to a lot of changes in our environment, mostly negative and often irreversible. Species are becoming endangered or going extinct, which is bound to affect predator/prey chains in the respective ecosystems. With simulation means one could analyze the consequences of the resulting imbalance without involving actual animals and before it is too late.

In this paper we present the results we gathered when simulating a simple marine ecosystem on a micro level. We based our model on the one described in [1], but extended and formalized it according to the well known design of cellular systems and more precisely a *cellular automaton*. The main goals that we set for this project were to further confirm the statement introduced in [1] that even somewhat complex real world phenomena/events/scenarios can be simulated with a relatively simple abstract model, that, nevertheless, exhibits authentic behavior. Our emphasis, however, was on proving the strengths of computational simulation when considering analytically intractable problems. The tests that we present served not only the purpose to verify the accuracy of the extended model, but also to deliver useful insights regarding the problem under consideration, such as input/output relation, without the need of a complex formal description. Furthermore, we acknowledge the pitfalls of simulation-based analysis in terms of computational effort needed and shortly discuss a more efficient approach to optimize the given problem scenario with the use of an *evolutionary programming* technique.

## 2 Problem Description

In its core the problem scenario used for the experiments is a relatively simple one: a finite space populated with objects of certain type, that interact with one another according to predefined rules. A less abstract extension of this scenario would be any ecosystem and the species populating it. In our experiments we modeled a sea ecosystem and three marine species forming a food chain. For the concrete design we used the well-known Wa-Tor world described in A. K. Dewdney's work [3]. It is a torus-shaped world with no landmass, but only one great ocean inhabited by marine species. To simulate an actual ecosystem, each species has a specific behavior according to which it interacts with both the environment and members of other species'. The behavior consists of a set of rules, which reflect the real world equivalents of the respective species' to a certain degree. All rules are defined on a micro level, i. e. only an individual's current state and its direct neighbors are considered. Each rule can be easily modified to reflect any introduced change of both internal (evolution) or external (cataclysmic event)

nature, as it happens in the real world, and then the potential impact on the environment can be observed. Originally Dewdney described a world inhabited by only two species forming a linear predator-prey relationship — sharks (predator) and fish (prey). Their interaction was pivotal for the environment's fate: sharks can survive only if there are enough fish to hunt, and fish can survive only if there are not too many sharks to hunt them to extinction. Fish feed on infinite plankton, which is not modeled for the sake of simplicity. Following the example of [1], we extended Dewdney's idea with an additional third species to create a slightly more complex food chain — whales, which hunt both sharks and fish, but have no enemy. This new species is partially theoretical as it does not depict the actual diet of real world whales.

Such a scenario related to ecology is a perfect example of a problem domain, that is well-nigh impossible to solve using real life resources for two major reasons: a) constructing and populating or isolating an actual ecosystem would be insanely expensive, unless in a very small scale, which might not yield the desired authenticity and/or results; b) experiments with animals in general raise a lot of questions regarding moral principles and are often reproached by modern society. On the other hand, formally specifying a mathematical model of the problem would prove to be as equally difficult simply due to its complexity — thousands of agents, each with its own characteristics, dispersed within the environment, randomly roaming and interacting. In other words, the given problem, although computationally solvable with finite resources in theory, can be seen as analytically intractable in practice. Therefore, developing a computer simulation using a moderately complex model is a viable approach to obtain valuable insights regarding the problem scenario, albeit not necessarily the optimal solution [17,16,5]. In our work we strove to confirm the accuracy of the model presented by [1] and its ability to self-sustain, but also tried to determine whether a relation between the input parameters and the ecosystem's ability to self-sustain exists. Moreover, we searched for an approach to efficiently find optimal input parameter combinations, i. e. such that do not cause an ecosystem collapse (species becoming extinct).

## 3 Implementation of the Problem

To test our ideas we developed a simple and robust software application. As more than 30 years have passed since Dewdney's initial publication, one can easily find various implementations of the Wa-Tor scenario. Proprietary software was excluded from consideration. After a brief evaluation of the *open-source* solutions according to the criteria *maturity* (how well developed is the product so far), *longevity* (what are the prospects of the software being further supported and developed) and *flexibility* (effort needed to integrate/modify the product) as suggested by [15] showed that neither was adequate enough, hence our initiative to design and implement an entirely new set of tools using Java.

### 3.1 Application Architecture and Representation

For the design of our application we chose a classic bio-inspired approach: an abstract cellular system. The idea is derived from biological tissues, where the fundamental unit is the cell. The cell is, indeed, a quite complex structure by itself [10], but when multiple cells cooperate even on a very basic local level, the outcome is a multicellular organism with unmatched capability compared to that of the individual cells building it. The human body is just one of countless examples in nature. This approach has already been widely used in design leading to the generalization of the resulting system type into the so called cellular systems. Essentially, such a system is a finite collection of basic units building a space. The units can be called cells and the space an organism accordingly. Each cell is identified by specific information about itself (e. g. an $n$-tuple of numerical values) and this information at a given time is called a *state*. The dynamics of such a system are expressed with the change in the cells' state depending on various factors. This *state transition* can depend on the current state of the cell, its past state and/or on the states of the surrounding cells. The collection of cells that can directly influence each other's state is called a *neighborhood*. Once the system is running, the cells update their states over time according to a predefined set of state transition rules and always reside in one from a finite set of possible states. A more detailed description of cellular systems is offered by [6, chap. 2].

The most prominent advantage of cellular systems is that they offer a simple modeling approach on a micro level, which however, can still yield plausible insights regarding the global behavior of the model. In the case of our problem, it is much easier to model the behavior of separate individuals based on their direct neighbors, rather than model the behavior of the entire population as one complex object.

From the various kinds known today we adopted the relatively simple, but popular *cellular automaton* (CA), more precisely the 2-dimensional *game of life* CA, as the base architecture of our application. A vivid example is John Conway's Life Game [7,8]. Using Floreano et al.'s [6] decomposition of a cellular automaton into its components and their elaborate description, we modeled the Wa-Tor ecosystem scenario as a cellular automaton as follows:

*Base unit.* The base unit is the cell represented as a square grid tile of certain color in the cellular space.

*Cellular space.* The space is a 2-dimensional lattice of cells forming a rectangular grid with size $500 \times 300$ cells.

*Time variable.* The state transitions in the system unfold along a discrete time axis with *cycle* as its base unit. One state transition happens per cell per cycle.

*State and state set.* The state of each cell consists of the animal type that the cell represents together with its respective attributes tuple. Therefore, each cell

can reside in one of four major states

$$\{Fish,\ Shark,\ Whale,\ Emptycell\}$$

and a multiple of sub-states defined by the numerical values of the respective species' attribute tuple.

*State transition function.* The transition from one state to another is represented by the actions that an individual can perform, such as moving, feeding, reproducing and dying. Both the major state of the neighbor cells and the instance of the attribute tuple of the considered cell influence the transition. It is not deterministic (e. g. a fish surrounded by empty cells could move to any one of it), and where more choices exist, their probability distribution is uniform.

*Neighborhood.* The *von Neumann* neighborhood is implemented, i. e. a cell's state is directly influenced only by neighbors at a *Manhattan distance* of 1 from it (its direct upward, downward, leftward and rightward neighbors).

*Boundary conditions.* For practical reasons the cellular space cannot be infinitely large, hence the need of appropriate boundary conditions to ensure that the system has a homogeneous neighborhood (every cell has the same type of neighborhood). We chose *periodic* boundary conditions by connecting opposite ends of the grid and essentially eliminating the boundaries, i. e. transforming the 2-dimensional grid into a 2-dimensional toroid (torus).

*Initial conditions.* The size of each species' population as well as the initial numerical values for each individual's attribute tuple (equal among all individuals of the same species). Both can be selected by the user before the start of the simulation. The distribution of the individuals over the cellular space is uniform.

*Stopping conditions.* The simulation stops after 3000 cycles (user adjustable).

### 3.2 Defining the Simulation Model

For the purposes of our study we adopted the simplified model proposed by [3] and further described in [1]: the behavior of the animal species consists of the actions moving, feeding, reproducing and dying; an individual is represented as an $n$-tuple of quantifiable attributes. The fish class is the simplest one modeled as the 2-tuple

$$(C_{offs},\ A_{repr}),$$

consisting of the reproduction maturity age and the number of offspring created, which are positive natural numbers. Shark and whale objects are further identified by their life energy (a positive real number), that is constantly depleted and can be replenished only by killing prey, and by the energy gain from eating the prey (a positive natural number), which varies according to the prey type. Unlike fish, predators do not reproduce over a constant period, but do so only if a

certain life energy level has been reached, thus their reproduction rate is directly related to the presence of prey. The shark class is modeled as the 4-tuple

$$(C_{offs},\ E_{repr},\ E_{life},\ E_{eat\_fish})$$

and the whale class as the 5-tuple

$$(C_{offs},\ E_{repr},\ E_{life},\ E_{eat\_fish},\ E_{eat\_shark}),$$

respectively. Each existing individual performs the aforementioned actions as defined in [1] in a specified order and a cycle in the simulation ends when all individuals have 'acted', after which the environment is updated. In terms of cellular automata the actions of an individual/cell can be seen as the state transition of that cell or/and one or more of its neighbors, whereas the various instantiations of the respective attribute tuples as the private information about the given individual/cell.

## 4 Performed Tests and Results Obtained

### 4.1 Model Accuracy

In the previous work on the topic [1] the simulation was run with different combinations of initial population sizes, repeating each combination multiple times to account for the not entirely deterministic behavior of the individuals. The obtained results did show similarities with real world population dynamics [2], e. g. predator overpopulation leads to the prey going extinct, vice versa with prey underpopulation and predators are more vulnerable than prey since they have to roam and search for food. Moreover, by plotting the change in the population sizes over the course of the simulation, it was shown that despite the simplicity of the model, successful initial population size combinations result in the system reaching an *equilibrium* state, and the populations exhibiting dynamics like those described by the Lotka-Volterra-Model [11].

### 4.2 Ecosystem's Capability to Self-Sustain

Since not every initial population size combination is favorable regarding the ecosystem's survival, and some combinations have only a partial success rate, a second series of experiments was aimed to a) prove that there is an integral not empty solution space (initial population size combinations) for which the given model of the simulated ecosystem can reach the so called equilibrium state; and 2) to define its boundaries. For that purpose the system was run automatically over a long period of time with randomly generated initial population sizes from a specified domain for the three species. It was shown that the successful input combinations form a set in 3-dimensional space, which with enough runs would resemble an irregular *solid*. In other words a solution space exists and the ecosystem is with high probability capable to self-sustain given any input from the respective set.

### 4.3 Input/Output Relation

From the results discussed in 4.1 and 4.2 it is clear that the system has a high chance to reach equilibrium for some input combinations, and a very low chance for others, i. e. a relation between the input parameters and the outcome of the simulation presumably exists. This can be proven formally by finding a function $f(p_1, p_2, ..., p_n)$ that maps an instance of the input parameters $p_1, ..., p_2$ to the respective output. As already stated, however, the problem under consideration is in practice analytically intractable, thus such a function cannot be obtained with a reasonable amount of effort. Instead, in our approach we decided to reuse the *empirical* formula described in [1], which is based on the statistical data gained from the initial tests and on the observations made while analyzing it. The function is defined as $S(P_f[n_f], P_s[n_s], P_w[n_w])$, where $S$ is the dependent variable (ecosystem stability), and $P_f[n_f], P_s[n_s], P_w[n_w]$ are the independent variables representing the probability with which the respective initial population sizes of fish, sharks and whales would survive. Moreover, from the already made conclusions it is known that some species are more vulnerable than others, and so their survival should be weighted more, e. g. sharks. As a result the stability of the system is defined as a function of the survival probability of each species, given its initial size, combined with a weighting factor based on the species' vulnerability:

$$S(P_f[n_f], P_s[n_s], P_w[n_w]) = P_f[n_f] \cdot 0.2 + P_s[n_s] \cdot 0.5 + P_w[n_w] \cdot 0.3 \ , \qquad (1)$$

where $P_x[n_x]$ is the survival probability of species $x$ with initial size $n_x$ in the respective test case. Using this approximation formula the ecosystem's stability factor was calculated for various initial predator population sizes within the range [2 , 50000] and two distinct fish population sizes: 5,000 and 50,000. The results are illustrated in Fig. 1 and show an obvious relation between the starting conditions and the simulation outcome. It is easy to see that an increasing predator population decreases the stability factor rapidly, indicated by the colors becoming darker towards the maximum values of the $x$- and $y$-axis. This is especially true for the whale population since the heat map darkens faster vertically (along the $y$-axis), rather than horizontally (along the $x$-axis). Ample quantities of fish prey also contribute for a better ecosystem stability (compare the visibly brighter heat map for the test cases involving an initial fish population of 50,000).

These results correspond to the conclusions made earlier about the accuracy of the model (see section 4.1). Any combination representing a predator overpopulation resp. prey underpopulation is colored in dark. Additionally, the greater part of both heat maps is in blue/purple hue, indicating a value less than 0.3. In other words, in most test cases both predator species, but especially the sharks, went extinct before the stopping condition of the simulation was met. Nevertheless, a comparison with the results regarding the system's ability to self-sustain documented in [1] reveals a partial error in the solution space's bounds. It is our belief that this could be eliminated with a better approximation of the solution

space boundaries (by performing more runs) as suggested in the corresponding section or a light modification of the weighting factors used in Eq. 1.

The last set of tests, that we conducted, were to determine a similar relation between the system's stability factor and the initial numerical values of a species' attribute tuple. For that purpose we repeated all test cases illustrated in Fig. 1 (left), while only changing the instance of the fish species' attribute tuple from (2, 20) to (5, 20), i. e. increasing the maximum number of offspring a fish can spawn from 2 to 5. This yielded a definite increase in the ecosystem's chance to reach equilibrium, as highlighted by the considerably brighter coloring of the heat map even in predator overpopulation/prey underpopulation scenarios (see Fig. 2, right).

### 4.4 Approach to Efficiently Optimize Input Parameters

The final question we would consider during this work is how to optimize the entire set of input parameters, so that the outcome of the simulation is favorable. Up to now only the size of the initial populations was regarded, but when comparing the improvement of the system's stability by changing these as depicted on Fig. 1 with the improvement gained by modifying the attribute tuple of a species (see Fig. 2), one can quickly acknowledge the significantly better results of the latter approach. A major difficulty, however, is again the size of the input space. With sufficient knowledge in combinatorics it is easy to see that using the 3-tuple $(Fish,\ Sharks,\ Whales)$ as input, where each element represents the

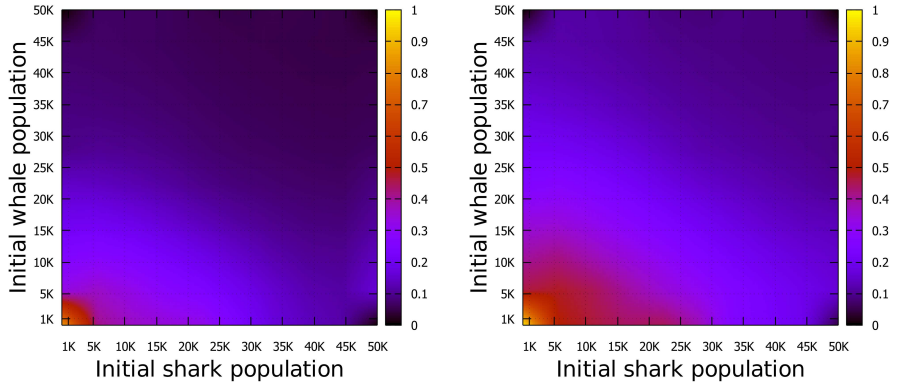


**Fig. 1.** Relation between the initial predator population sizes and the stability of the ecosystem obtained with Eq. 1. Two different fish population sizes have been used in the experiments: 5,000 (left) and 50,000 (right). Darker/colder colors represent lower stability factor. The stability factor ranges from 0.0 (no species survived any of the test cases) to 1.0 (every species survived all of the test cases).

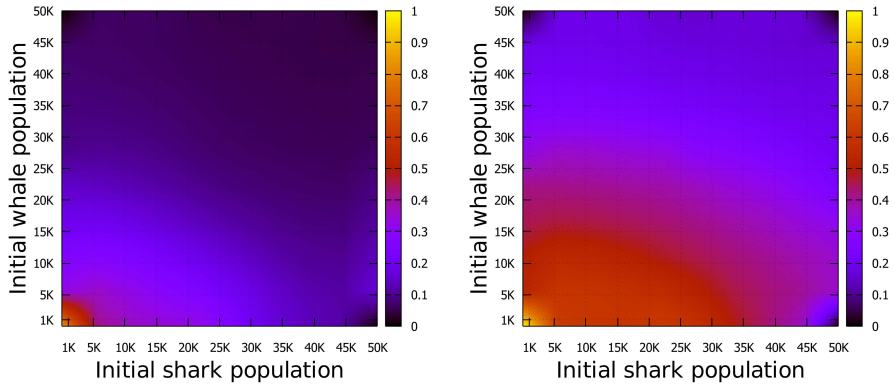


**Fig. 2.** Relation between the attribute 'offspring count' for the fish species and the stability of the ecosystem obtained with Eq. 1. Two different instances of the maximum offspring count have been used in the experiments: 2 (left) and 5 (right).

respective species' population size within the range [10 , 50000], yields a total of

$$44991^3 \approx 50000^3 = 1.25 \cdot 10^{14}$$

possible input combinations. In a similar way we can count the possible instances $I$ for the attribute $n$-tuple of each species with Eq. 2:

$$I = \prod_{i=1}^{n} |A_i|, \tag{2}$$

where $|A_i|$ is the number of distinct numerical values, that attribute $A_i$ can take. For instance the 2-tuple of the fish species could have the following ranges for its elements

$$(C_{offs},\ A_{reproduce}) \Rightarrow ([1\ ,\ 5],\ [1\ ,\ 100]),$$

yielding a total of $I_{fish} = 5 \cdot 100 = 500$ possible attribute instances. Analogously the shark 4-tuple and the whale 5-tuple could be defined as

$$(C_{offs},\ E_{repr},\ E_{life},\ E_{eat\_fish}) \Rightarrow ([1\ ,\ 5],\ [51\ ,\ 100],\ [1\ ,\ 100],\ [1\ ,\ 10])$$

and

$$(C_{offs},\ E_{repr},\ E_{life},\ E_{eat\_fish},\ E_{eat\_shark}) \Rightarrow$$
$$([1\ ,\ 5],\ \ [101\ ,\ 200],\ [1\ ,\ 200],\ [1\ ,\ 10],\ [1\ ,\ 50]),$$

yielding $I_{shark} = 5{\cdot}50{\cdot}100{\cdot}10 = 250000$ and $I_{whale} = 5{\cdot}100{\cdot}200{\cdot}10{\cdot}50 = 50000000$ possible instances respectively. The total count of input parameter instances $I_{total}$ is then

$$I_{total} = I_{fish} \cdot I_{shark} \cdot I_{whale} = 500 \cdot 250000 \cdot 50000000 = 6.25 \cdot 10^{15}\ .$$

Even though this number is not much greater than the number of population size combinations, there are 11 input parameters when summing the tuple elements of all three species, yielding an 11-dimensional input space respectively, unlike the 3-dimensional one for the population sizes. Therefore, applying the graphical solution approaches presented so far in the previous sections would be less than feasible. Moreover, the conclusions made were obtained using excessive testing at the cost of hundreds of hours of computational processing, and could be classified as good approximations at best, which is supported by the small contradiction between the results described in sections 4.2 and 4.3. This raises doubts regarding the efficiency of the simulation-based problem solving should problem complexity increase (e. g. more input parameters).

A more promising approach, that we would like to propose, is to combine the simulation-driven architecture of the cellular automaton as it is with an evolutionary search algorithm to traverse the input space for combinations that yield a stable ecosystem. Essentially, we would transform a simulation problem into an *optimization* one, where a model consisting of objects with specific characteristics and rules for their interaction is known, a desired output range is specified and the point of interest is the unknown set of input instances that yield an output within the desired range.

Inspired by biological evolution, hence their name, *evolutionary algorithms* (EA) have gained significant importance in the recent years, especially in the domain of optimization. Their use can be quite rewarding as an efficient alternative to traditional deterministic methods, more so in very complex and intractable problem scenarios. In the following a short summary of the basics is given, as well as an outline of the modifications needed to incorporate the evolutionary optimization technique in our application. It is our strong belief that the integration effort would not be considerable, but the obtained insights might prove to be of value. For a more comprehensive discussion on the topic of EA basics and application refer to [9,14,4].

*Solution candidate space.* As the name implies this is the set of all possible candidate solutions. It can be indefinitely large, but for practical reasons it has to be bounded. Candidate solutions can be anything from numerical values to entire objects. In the case of our Wa-Tor application, the candidate solution space consists of all attribute combinations for the three species (e. g. $6.25 \cdot 10^{15}$ if the aforementioned example is taken).

*Genetic operators.* These operators are functions that randomly modify the existing collection of candidate solutions to increase diversity. Typical operators are a) *mutation* — changing a random characteristic of a candidate solution; and b) *recombination* — combining two or more solutions to produce new ones; Integrating this functionality in the existing Wa-Tor simulation can be done quite easily in the reproduction action of the animals, e. g. when offspring are spawned their attribute instance can be created by combining that of the parent with those of other existing individuals. Mutation can be implemented similarly as a random change in a spawned offspring's attribute instance inherited by the

parent. Genetic operators modify only the genotype. The phenotype cannot be inherited, but is crucial for the fitness of the respective candidate.

*Solution candidate encoding.* A candidate solution is typically represented as a *phenotype-genotype* pair. Just like in biology the phenotype is comprised of a candidate's visible attributes that change over its lifespan due to interaction or external events. On the other hand the genotype is the entire information about the core structure of a candidate in some encoded form, e. g. DNA in real life and usually a binary string in software applications. In the Wa-Tor scenario the phenotype of an object would be its actual instance in the simulation, whereas an adequate genotype would be the information about the object's attributes received at its creation, that is immutable during the lifespan of the object, but inheritable by its offspring.

*Fitness function.* The idea behind the fitness function is to decrease the population diversity by eliminating poor candidate solutions, but thus improve the overall quality of the candidate solution space (called 'survival of the fittest' or ' natural selection' in the field of biology). The fitness function in the Wa-Tor scenario is the simulated interaction between the individuals. Using the predefined interaction rules such as moving and feeding, individuals may survive or get killed by predators/die out of starvation. Certain genotype instances may have higher chances to prevail than others, e. g. individuals that can give birth to multiple offspring are more likely to pass down and spread their genotype in the respective population.

## 5 Conclusion and Future Work

The main goal of our project was to design and develop an application to test the strengths of a simulation-based analysis. A relatively simple and well-known problem domain was chosen, which, however, poses multiple challenges regarding its formal specification, hence the need of an alternative approach. An ecosystem and various species inhabiting it were modeled in the fashion of Dewdney's Wa-Tor [3]. Even though the created model is a fairly basic one, including only several interaction rules between the organisms within the simulation, and a handful of attributes to characterize them, no mathematical specification of the system can be derived without a significant amount of effort. Via simulation however, it was possible to define very simple rules on a micro level, and still obtain valuable insights regarding the global behavior of the environment and the populations inhabiting it.

We augmented previous work on the topic by performing extensive testing to gather enough data and visualize the relation between the inputs and the output of the system. For that an empirically derived formula (see Eq. 1) was used to calculate the system's output (stability factor) as a function of the probability that a species with a given population size would survive. The formula uses weight coefficients to discriminate between more robust and more vulnerable

species. In Figures 1 and 2 we showed the direct dependence of the ecosystem's stability on the initial population sizes and on a specific species' attribute — the number of created offspring after reproduction. In the final part of the paper the challenges of extensive testing were discussed and a new approach based on evolutionary programming was proposed as a possible more efficient alternative.

Future work would revolve around the expansion of the application to incorporate the described evolutionary technique and allow for the model (species' attributes) to change over the course of the simulation similar to the evolution of organisms in the real world. The potential optimization benefits of this approach are to be evaluated and compared to those of the extensive testing.

## References

1. Balabanov, K., Fietz, R. G., Logofătu, D.: Considerations in Analyzing Ecological Dependent Populations in a Changing Environment. In: Computational Collective Intelligence, pp. 223–232, Volume 10448, Part I (September), Nicosia 2017.
2. Begon, M., Mortimer, M., Thompson, D. J.: Population Ecology: A Unified Study of Animals and Plants, 3rd ed, [Online]: Wiley-Blackwell (1996)
3. Dewdney, A. K.: Sharks and fish Wage an ecological War on the toroidal planet Wa-Tor. In: Scientific American, pp. 14–22, 251 (December), (1984)
4. Eiben, A. E., Smith, J. E.: Introduction to evolutionary computing. 2nd ed. Springer, Heidelberg (2015)
5. Farge, M.: Numerical experimentation: A third way to study nature. In: Frontiers of Computational Science. Proceedings of the International Symposium on Frontiers of Computational Science 2005, pp. 15–30. Springer-Verlag, Berlin (2007)
6. Floreano, D., Mattiussi, C.: Bio-inspired artificial intelligence : theories, methods, and technologies. MIT Press, Cambridge (2008)
7. Gardner, M: The fantastic combinations of John Conway's new solitaire game "life". In: Scientific American, pp. 120–123, 223 (April), (1970)
8. Gardner, M.: On cellular automata, self-reproduction, the Garden of Eden and the game "life". In: Scientific American, pp. 112–117, 224 (February), (1971)
9. Gerdes, I., Klawonn, F., Kruse, R.: Evolutionäre Algorithmen : genetische Algorithmen - Strategien und Optimierungsverfahren - Beispielanwendungen. Vieweg, Wiesbaden (2004)
10. Harold, F. M.: The Way of the Cell. Oxford University Press, Oxford (2001)
11. Hoppensteadt, F.: Predator-prey model. In: Scholarpedia, pp. 1563, 1 (October), (2006)
12. Law, A. M.: Simulation modeling and analysis. McGraw-Hill Higher Education, New York (1997)
13. Logofatu, D., Sobol, G., Stamate, D., Balabanov, K.: A Novel Space Filling based Approach to PSO Algorithms for Autonomous Agents. In: Computational Collective Intelligence, pp. 361–370, Volume 10448, Part I (September), Nicosia 2017.
14. Michalewicz, Z.: Genetic algorithms + data structures : = evolution programs. 3rd ed. Springer, Berlin (2008)
15. Norris, J. S.: Mission-critical development with open source software: lessons learned. In: IEEE Software, pp. 42–49, 21 (January), (2004)
16. Rédei, M.: John von Neumann: Selected letters. RI: American Mathematical Society, Providence (2005)
17. Ulam, S. M.: Adventures of a Mathematician. Scribner, New York (1976)