Spring 2008

Individual-based artificial ecosystems for design and optimization

Shivakar Vulli

Department:

INDIVIDUAL–BASED ARTIFICIAL ECOSYSTEMS FOR DESIGN AND OPTIMIZATION

by

SRINIVASA SHIVAKAR VULLI

A THESIS

Presented to the Faculty of the Graduate School of the

UNIVERSITY OF MISSOURI–ROLLA

In Partial Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE IN MECHANICAL ENGINEERING

2008

Approved by

S. Agarwal, Advisor
K. Krishnamurthy
R. G. Landers

ABSTRACT

Individual-based modeling has gained popularity over the last decade, mainly due to the paradigm's proven ability to address a variety of problems seen in many disciplines, including modeling complex systems from bottom-up, providing relationship between component level and system level parameters, and discovering the emergence of system-level behaviors from simple component level interactions. Availability of computational power to run simulation models with thousands to millions of agents is another driving force in the widespread adoption of individual-based modeling. This thesis proposes an individual-based modeling approach for solving engineering design and optimization problems using artificial ecosystems. The problem to be solved is first "mapped" to the artificial ecosystem's environment and individuals (one or more species). The artificial ecosystems is then allowed to evolve. The optimal solution emerges through the interactions of individuals, which makes this approach attractive for design and optimization in complex systems, where formulation of a global fitness function (as required by conventional evolutionary techniques) is complicated. To demonstrate the proposed approach, the problem of binary texture synthesis is attempted using an artificial predator-prey ecosystem.

ACKNOWLEDGMENT

I would like to express my heartfelt gratitude to Dr. Sanjeev Agarwal, who has been a great mentor, advisor, and above all a role model. Without his valuable insight, guidance and encouragement, this thesis would not have been possible. I am forever indebted to him for instilling into me, a passion for interdisciplinary and multidisciplinary research.

My sincere thanks to Dr. K. Krishnamurthy and Dr. R. G. Landers for agreeing to serve on my committee. They have been most generous in accommodating my tight deadlines into their busy schedules. I would also like to thank Intelligent Systems Center (ISC) for providing financial support to pursue part of this research.

I would like to thank my friends Sarat Sreepathi, for his thought provoking discussions, Shareq Soofi, and Gerard Sequeira, for being wonderful lab mates at ARIA lab and Vamsi Sripathi for his support and feedback.

I would finally like to express my eternal gratitude to my parents for giving me the independence and opportunity to pursue my dreams and always believing I can achieve them.

TABLE OF CONTENTS

Page

LIST OF ILLUSTRATIONS

LIST OF TABLES

1. INTRODUCTION

Ever increasing demand for better capabilities, performance and scalability is driving engineering systems to new complexities. Interconnections and interdependencies among these large-scale systems only adds to the difficulties in designing, modeling and analyzing them. These so called complex systems have been defined by Marashi and Davis [1], as containing 'many components and layers of subsystems with multiple non-linear interconnections that are difficult to recognize, manage and predict'. Almost every engineering and infrastructure related industry today (including electric power, water, and telecommunications) are not only complex systems in their own right, but also highly interdependent on one another, making them subsystems of a much bigger and complex system. Maier [2] calls a system 'which is built from components which are large-scale systems themselves' as a system-of-systems.

Several approaches are in practice for the design and analysis of complex systems[3] such as iterative maps, statistical mechanics, neural networks and system dynamics[4]. One of the newest approach gaining rapid popularity is bottom-up modeling and simulation [5, 6, 7]. In this approach, the complex system is broken up into its constituent subsystems up to the level of granularity required. These subsystems (or subelements) are then designed/modeled using conventional methods available for the problem domain. An interesting inherent characteristic of complex systems is emergence, i.e., the behavior(s) of a complex system cannot be predicted even with the complete knowledge of the behaviors of the subsystems. It is universally agreed that the interactions between the subsystems of a complex system are responsible for emergent behaviors. Therefore to claim a representative design of a complex system, it is necessary to model the interactions between the subsystems with sufficient detail.

With the capability of modeling systems using a collection of autonomous, goal driven, interacting entities called agents, Agent-based modeling[1] appears to be an appropriate choice for modeling complex systems. The availability of computational

[1] Also referred to as individual-based modeling. Both these terms indicate the same bottom-up modeling paradigm using a collections of entities called individuals or agents. Hence, these terms are used interchangeably in this thesis.

power to run simulation models with millions of agents is another reason, agent-based models (ABMs) are enjoying a widespread adoption in a variety of disciplines. However, as a young field, agent-based modeling still has a long way to go before becoming the first choice for complex systems design and analysis. Model validation and efficient communication of results remain major challenges for researchers[8, 9, 10, 11]. The lack of efficient agent-based models for engineering systems design and optimization is another problem which needs to be addressed.

1.1. THE BIG PICTURE

From ant colonies and ecosystems to immune systems and global climate, complex systems can be found everywhere in nature. Continuously evolving and adapting to changes in each other, these inherently robust, natural complex systems are envied by even the best of engineers. Most inventions in the human history have taken inspiration from natural systems or phenomena. Whether it is aviation, tribology or robotics, almost every aspect of modern technology has drawn some form of inspiration from nature. Even in computational sciences, bio-inspired techniques such as artificial neural networks, artificial immune systems, genetic algorithms, ant colony optimization have enjoyed a significant position.

This thesis proposes an individual-based modeling and computational emergence approach to solve design and optimization problems. Inspired by naturally occurring ecosystems, a generalized framework consisting of one or more evolving, interacting species, is developed. Basic building blocks needed for engineering these artificial ecosystems are identified by drawing on the wealth of information available on biological and ecological systems, agent-based modeling and evolutionary techniques. Research issues associated with representation of evolving and non-evolving engineered species and their interactions, representation of the environment and associated constraints, implementation of population dynamics, mating preferences and life cycle dynamics will be addressed. Although the generalized framework formulated here can also be used to enhance the understanding of complex biological, ecological and social systems, this thesis does not delve in these topics.

Application of the proposed framework to a real complex system is beyond the scope of this thesis. Therefore a simple yet classical problem of binary texture

synthesis is attempted as an illustrative application of the framework. Even in the case of this simple problem, signs of emergence can be seen.

1.2. ORGANIZATION OF THE THESIS

A brief overview of the organization of this thesis is as follows. In Section 2, a review of the previous work done in the areas of ecological modeling, individual-based modeling, evolutionary techniques, multi-objective optimization and system-of-systems architecture relevant to this thesis is presented.

In Section 3, the proposed artificial ecosystems framework is discussed. Details of various biological and ecological processes and their individual-based modeling are presented.

In Section 4, several existing agent-based modeling softwares are evaluated using a representative individual-based model.

In Section 5, as individual-based model of a single species environment ecosystem is developed to investigate the population and life-cycle dynamics. Effects of individual level parameters and model assumptions on system level behaviors are analyzed.

In Section 6, an individual-based model of a predator-prey ecosystem is developed to investigate the effects of predation. Effects of individual-level parameters and model assumptions on system level behaviors are also analyzed.

In Section 7, an adaptation of the predator-prey model is employed to solve the problem of binary texture synthesis. Experiments are conducted using this model and the result are presented and discussed.

Finally, this thesis is concluded in Section 8, followed by a discussion on the future directions of research.

1.3. MAJOR CONTRIBUTIONS

The following are regarded as the major contributions of this thesis.

1. The design and description of an individual-based generalized framework inspired by natural ecosystems for solving engineering design and optimization problems.

2. The identification, description and comparison between important ecological processes and their individual level counterparts.

3. The design and implementation of individual-based artificial ecosystems.

4. Demonstrating a set of capabilities of the proposed framework through the use of an illustrative example.

2. RELATED WORK

As mentioned earlier, this thesis derives on the knowledge of agent-based modeling, ecological modeling theories and evolutionary techniques and their application to multi-objective optimization problems. A lot of research effort has been devoted to each of these topic individually, and there exists a huge body of literature on them. In the following sections, previous work on these areas relevant to this thesis, especially over the recent years, is identified and discussed. In Section 2.1, agent-based modeling or individual-based modeling techniques and applications developed over the years are presented and their relative merits and demerits discussed. In Section 2.2, research in the field of ecological modeling especially related to ecosystem dynamics, and individual-based modeling of ecosystems is reviewed. Sections 2.3 and 2.4 a review of current evolutionary techniques and their application to multi-objective optimization problems is presented.

2.1. INDIVIDUAL-BASED MODELING

Individual-based modeling or agent-based modeling refers to the class of analysis tools, in which the system being analyzed is modeled as a collection of autonomous, goal driven, interacting entities called individuals or agents. Von Neumann [12] was probably the first to use this kind of bottom-up modeling approach, which he called Cellular Automata (CA). In CA, each cell's current state depended on its own previous state and its neighbors' states. As noted by Wolfram [13], even with these simple rules and local interactions CAs produce some fascinatingly complex global patterns. Several seminal contributions to individual-based modeling came from a diverse array of disciplines.

Individual-based modeling have been used in social sciences for a long time, albeit sparingly. Schelling [14], demonstrated the relationships between individual motives and group outcomes using social situations of interactive dependence. Granovetter [15], developed threshold models for situations where actors (agents) have two distinct and mutually exclusive behavioral alternatives, the costs and/or benefits of which depend on how many other actors choose which alternative. Another

interesting application was developed by Axelrod [16], who developed an individual-based variation of the prisoner's dilemma to study the circumstances under which a selfish agent would spontaneously cooperate. To his surprise, cooperation arose between agents in unimaginable situations including battles of World War I, battles over trade barriers, strategic alliances between rival businesses, among others. During the last decade, however individual-based modeling has seen an increasing interest in social sciences [7, 5, 6, 17, 18]. Individual-based models (IBMs[2]) have been used for modeling artificial societies [19, 20], anthropology [21, 22], epidemiology [23], human systems [24, 25] such as crowds, traffic, and markets, and urban planning [26, 27].

Another discipline with long standing interest in individual-based modeling is ecology. In his mini-review [8] of 50 odd individual-based animal population models, Grimm identified several advantages of using IBMs for ecological modeling. He also discusses the problems faced and errors made by modelers. DeAngelis et al. [28], in a recent review, identified and discussed IBMs based on ecological and evolutionary processes such as movement, foraging, local competition, modeling of fitness and trait evolution. Grimm et al. [29, 10], argued that patterns[3] are defining characteristics of a system and are therefore possible indicators of essential underlying processes and structures. Consequently, they suggested that, to make bottom-up modeling more rigorous and comprehensive, IBMs should focus on explaining these observed patterns. Applications of individual-based modeling in ecology include studies on population dynamics [30, 31], predator-prey dynamics and co-evolution [32], migration [33, 34, 35] and ecological resource planning [36, 37].

In computer science, agent-based systems[4] stemmed from the research in the fields of distributed artificial intelligence (DAI) and artificial life (Alife). Hewitt and Inman [38] listed a number of limitations of classical DAI which MAS could potentially solve. Additional information on multi-agent systems can be found in Sycara [39], Wooldridge [40], and Weiss [41], who carried out extensive surveys of the architectures and issues in this field along with their applications. Another important milestone in the history of individual-based modeling was Reynolds' BOIDS [42], in which an

[2]IBM will be used as an acronym for Individual-based Model and not Individual-based Modeling

[3]Defined in the work as 'Observations of any kind showing nonrandom structure and therefore containing information on the mechanisms from which they emerge.'

[4]Traditionally called by computer scientists as Multi-Agent Systems (MASs)

individual-based local interaction scheme was used to replicate grouping behaviors in animals.

This increase in popularity of individual-based modeling is mainly attributed to the paradigm's proven ability to address a variety of problems seen in many disciplines. Axtell [6], Bankes [7], Bonabeau [25], Grimm and Railsback [10], and Macal and North [17, 18] have all independently identified the following as merits of individual-based modeling:

- The unsuitability/ inability of competing modeling methodologies such as difference or differential equation-based modeling or system dynamics based modeling to represent individual variation. These approaches often made one or more unrealistic assumptions such as uniform representative cases for the purpose of simplification. In individual-based modeling however such unrealistic assumptions need not be made. Individuals or agents can be made as diverse and heterogeneous as necessary to model real world scenarios accurately. In cases where a system's equations are intractable or provably insoluble, IBMs can shed light on the dynamics of the system as these do not involve solving any equation(s).

- For many of the modern systems, individual-based modeling by its very nature provides a natural description of the system. It is easier to imagine a system as a collection of subsystems performing actions and behaviors, than to imagine it as a set of levels and flows (as in the case of system dynamics approach) or set of difference or differential equations.

- Availability of computational power to run simulations with thousands to millions of agents[43], has made it possible to achieve simulation scales that were not plausible a couple of years ago. With advances made in sensor technologies, it is also possible now to collect data which was previously difficult if not impossible.

- IBMs also provide flexibility. Unlike traditional equation-based models which have to be totally re-solved to incorporate any changes in the assumption made during design, IBMs allow an easy way to test multiple hypothesis to identify the appropriate set.

- The most important advantage of using IBMs is however, their ability to capture emergence. By definition emergent phenomena cannot be reduced to the system constituents, i.e., the system is more than the sum of its components. As traditional top-down approaches model the system as the whole, they can neither identify the relationships between parameters and the emergent behaviors. In individual-based modeling however, the individuals interacting in their environment, capture emergent phenomena from bottom-up. Identifying the relationship between the individual parameters and emergent phenomena is a simple case of rerunning the model a several times for different values of the parameters.

As with any paradigm, several shortcomings of individual-based modeling have been identified and frequently used as arguments against the paradigm. Several similar solutions to these objections have been suggested by researchers from different fields, which are aggregated into the following list.

- Generally, IBMs consists of numerous parameters which makes derivation of governing equations from model results an impossible task, except via the use of mathematics in highly stylized ways. This inability to validate IBMs is usually used as an argument against individual-based modeling. Axtell [6], offers 'computer programs as sufficiency theorems' argument of Newell and Simon [44], as a counter argument which states that if a model A, produces result R, then it a sufficiency theorem for the formal statement R if A. Carley [11], identified the types of validation, and the issues with model validation. One of the validation schemes is to observe the patterns generated by the model and match it against the patterns found in the real world system. This is identical to the pattern-oriented modeling approach proposed by Grimm et al. [29]. The latter also assert that theories of complex systems may never be reducible to simple analytical equations. In which case, individual-based modeling of complex systems may help analyze the underlying theory of these systems using an algorithmic, rather than an analytical approach.

- Another concern frequently expressed by IBM modelers and designers is that, although IBMs may be a more natural representation of the system, they are

difficult to communicate to others [37, 29]. Unlike population-level models, IBMs take into account individual variability, and detailed behaviors of individuals, increasing the number of variable parameters and complexity of the model. This makes the communication of results of an IBM via the familiar language of mathematics, unrealistic. Grimm and Railsback [10], proposed a standard protocol to facilitate communication and replication of IBMs. This protocol was later revised by a consortium of researchers [9] and named the ODD [5] protocol. This protocol is adopted to describe the IBMs presented in this thesis.

- Bankes [7], pointed out that most of the published works, rely on human observers to identify and declare the occurrence of emergent phenomena based on graphical outputs. He argues that unless a formal definition, rule and quantitative tests to detect and validate emergence are discovered, the scientific importance of emergence and individual-based modeling will remain small. No answers to these questions were found in the surveyed literature. Answering these questions is a research in itself and beyond the scope of this thesis. Hence the human observer approach is adopted.

2.2. ECOLOGICAL MODELING

Ecology is defined as the study of systems (called ecosystems) comprising of biological entities (biotic) functioning together with non-living physical matter (abiotic) of the environment. The field of ecology can be broadly classified into

- Behavioral ecology – study of individual behaviors and their effect on individual capability to adapt to its environment.
- Population ecology – study of population dynamics of a single species
- Community ecology – study of interacting populations of multiple species.
- Ecosystem ecology – study of energy and matter interactions between the biotic and abiotic elements of an ecosystem.

[5] ODD stands for Overview, Design Concepts, Details. An online appendix containing significant number of previous IBMs translated to the ODD protocol is available at http://www.ufz.de/oesatools/odd

- Systems ecology – study of ecosystem development and organization.
- Landscape ecology – study of processes and relationships between multiple ecosystems over a very large geographic area.

This thesis concerns itself with population and community ecology. Ecosystems and processes discussed in the following literature survey are for single and multiple species, population dynamics.

The simplest population dynamics model, the exponential model, was proposed by Malthus [45]. In this model population size of a species had three outcomes, exponential decrease, exponential increase or no change. Malthus [46] later observed that while populations grow geometrically, vital resources in their environment remained constant or grew only arithmetically. Thus, population size must eventually exceed the size sustainable by the available resources. This idea was later proposed as the logistic equation by Verhulst in 1838[6]. According to the logistic equation, the rate of population increase depends on the population density. Richards [48] proposed a flexible growth model, which also described the rate of increase as a function of population density. Richards equation, however, has an additional parameter called the shape parameter, which allows the equation to be equivalent to several other popular population dynamics models, including the logistic equation.

For multiple interacting species, models such as predator-prey, host-parasite, plant-herbivore, cooperation, ecological niching, food-web models exist. Several mathematical treatments of these models can be found in [49, 47, 50]. The literature survey presented here concentrates only on the predator-prey ecosystem.

The predator-prey ecosystem consists of two interacting species, of which one (prey) is the food source of another (predator). The predator-prey ecosystem is one of the most widely studies interspecific competition models. The famous Lotka-Volterra (L-V) predator-prey equations were proposed independently by Lotka [51] and Volterra [52]. Nicholson and Bailey [53], proposed a discrete-time host-parasite model. Although this model was based on parasitoid behaviors, it is identical to the predator-prey theory, and is generally referred to as the discrete-time predator-prey model. Further contributions to the predator-prey was made by Holling [54], who

[6] as noted by Kot [47].

established the relationship between predator-prey population dynamics and prey density. For a detailed historical account of predator-prey theory, see Berryman [55]. The applicability of predator-prey model is not confined to the realms of ecology studies. Predator-prey models have been used for studying cooperative strategies [56], robotics [57, 58], and multi-objective optimization [59, 60].

2.3. EVOLUTIONARY TECHNIQUES

Evolutionary computing techniques are the class of optimization algorithms inspired by natural systems and the concept of darwinian evolution. Most of these algorithms consists of a population of individuals, each representing a single solution in the solution space. Using an iterative process, the population is iterated through generations. The attributes (chromosomes) of the the individuals is altered via the use of processes comparable to biological processes such as mating, reproduction and death. These algorithms draw inspiration from darwinian evolution theory, which states that overtime only the fitter genes of a species survive and that these species adapt to survive in their environment. Therefore, the population of individuals which inhabit the simulation after few hundred to few thousand generations, must be highly adapted to the environment and should therefore represent optimal solution to the problem at hand.

Several evolutionary techniques are in use today. These techniques can be broadly classified into evolutionary algorithms and swarm intelligence algorithms. Genetic algorithms (GAs) is the most popular technique in the class of evolutionary algorithms. Although previous work existed [61], Holland [62] is usually accredited with popularizing GAs. The GA proposed by Holland used a population of individuals whose chromosomes were represented using bitstrings. These bitstring chromosomes represented the solutions to the problem. Every iteration (generation) consisted of evaluating each individual using a *fitness function*. Predetermined percentage of top performing (high fitness) individuals are kept alive and the rest killed. Offsprings are generated to fill the deficit in population, by selecting two parents, based on a *selection* criteria and creating a chromosome for the offspring using a *crossover* function. The resulting chromosome is mutated using a small fixed probability. This *mutation*

function[7] is responsible for maintaining diversity of the population. The simulation is then advanced to the next iteration. A large number of variants of Holland's GA exist. For a detailed discussion, on these variants see [63, 61].

Co-evolution is another popular evolutionary technique. Co-evolutionary algorithms consist of multi species whose evolution depends on the evolution of one-another. Several types of co-evolution such as competition, amensalism, mutualism, commensalism and parasitism have been identified [64]. Competitive co-evolutionary algorithms include the predator-prey models, where the prey tries to adapt to survive predators and the predators adapt to catch the evolved prey. This can lead to run-away evolutionary *arms race* called *Red Queen Dynamics*[65, 66, 57, 58]. Cooperative co-evolutionary algorithms (mutualism) on the other hand consists of multiple species which co-exist in an environment. This co-existence is due to some form of mutual benefit to each other. Example of such co-existence is a symbiotic relation, in which one species provides some service to the other and vice-versa. De Jong and Potter [67], proposed an cooperative co-evolutionary approach to designing and learning complex structures. They argued that traditional evolutionary algorithms evaluate solutions to complex problems only on the basis of performance and hence structures involving modularity seldom evolve. In their approach, multiple instances of an evolutionary algorithm are run in parallel, each producing useful substructures/ subcomponents. Representatives from each EA instance are selected and combined to form a global solution whose fitness is used to send feedback to the EA instances, reflecting how well they collaborated with each other.

Unlike evolutionary algorithms, swarm intelligence techniques, draw their inspiration from grouping behaviors in animals. Many animals species form social groups such as flocks in birds, schools in fish, herds in cattle, trails in ants, etc. In these groups, individuals have only local information, i.e., information regarding their immediate vicinity. Any global information[8] is acquired through communication and interaction with other individuals who share their vicinity. Particle Swarm Optimization (PSO) and Ant Colony Optimization (ACO) are the two most popular swarm intelligence techniques. PSO was proposed by Kennedy and Eberhart [68], inspired by

[7]In GA terminology, selection, reproduction, crossover and mutation are called operators.

[8]Any information about the group beyond an individual's immediate vicinity can be considered as global information.

flocking behavior in birds. In PSO, particles representing possible solutions to a given problem are flown through the solution space. Each particle evaluates the fitness of its location based on a *fitness function*. Each particle keeps track of the best location it has found so far and the best location its neighbors have found/communicated. Based on this information, it updates its velocity and position. It is observed that after few iterations the particles start flocking around the global optimal solutions. Numerous variations of this algorithm can be found it literature (see [69, 61]).

ACO derives its inspiration from foraging behavior of ants. Via local interactions, ants form trails connecting food sources and the ant colony. Dorigo [70] proposed the ACO algorithm and later applied it to the traveling salesman problem (TSP) [71]. ACO is generally applied in situations where the problem to be solved can be represented as a graph. Examples of such applications include telecommunication networks [72], shop floor management [73] and data mining [74]. For more information on ACO and its variants, see[9] [75].

2.4. EVOLUTIONARY MULTI-OBJECTIVE OPTIMIZATION

Multi-objective optimization (MOO) involves simultaneously optimizing two or more objectives which are conflicting in general and subject to multiple constraints. Due to their conflicting nature, all the objective functions cannot be simultaneously optimized to their best solution and expect to find a global best solution. Therefore, it is clear that no single absolute solution can be found for MOO problems. Instead, a set of objective vectors (solution) can be obtained such that no other solution can improve on one of the objective functions without worsening another. Such an objective vector is called a *Pareto-optimal set*.

A variety of evolutionary techniques have been applied for MOO problems. This thesis however concentrates on a small subset of the applications using predator-prey models relevant to the current work. For a comprehensive overview of MOO using evolutionary algorithms see[10] [76, 77].

Laumanns et al. [59], were the first to consider a predator-prey model for multi-objective optimization. They considered prey individuals as the possible solutions

[9] Another good resource on ACO is http://iridia.ulb.ac.be/∼mdorigo/ACO/

[10] Also see IEEE CIS Task Force on Multi-Objective Evolutionary Algorithms website at http://www.cs.cinvestav.mx/∼emooworkgroup for EMOO repository.

of the MOO problem. The prey were placed at the vertices of a two-dimensional toroidal grid. Additionally, the prey were considered immobile. The predators randomly moved across this grid killing prey according to one of the optimization criteria. Several predators for each optimization criterion were populated at the start of the simulation run. The predators kill only the worst prey in their immediate neighborhood and the vacated space is immediately replaced by another prey created by discrete recombination of those prey which are part of the recombination neighborhood. Later, Deb [76] identified that Laumann's model suffered from the loss of diversity and poor convergence rate. He proposed modifications such as weight vector assignment to the predators instead of individual objective function, offspring creation through the recombination of the best prey instead of a random one, and predators moving to the best prey location instead of a random walk. Deb reported that these modifications improved the convergence rate, but loss of diversity was still a problem. A revised version was reported by Deb and Rao [78], incorporating elite and diversity preservation mechanisms. Elite-preservation was achieved by accepting new prey only if they performed better than the prey to be killed by the predator. Each prey was assumed to have an influence region defined by a hyper-cube around it on the objective space. Diversity preservation was achieved by not accepting offsprings which were created within the influence zone of any existing prey. Grimm and Schmitt [60], proposed further modifications to the model including adoption of well known self adaptive Evolution Strategy (ES) mechanism [79], and a rotation-independent recombination operator to learn and adapt to the position of the Pareto-set in search space.

3. ARTIFICIAL ECOSYSTEMS

Most current evolutionary techniques discussed in Section 2.3 can be summarized by Figure 3.1. It can be noted that the fitness function and selection mechanism are evaluated external to the evolving population. This is an unnatural way of accomplishing *adaptation* in contrast to the natural systems these algorithms are inspired from. *Survival of the fittest* phenomena which this type of selection mechanism is intended to enforce is an emergent behavior in nature and not something that is explicitly enforced.

Although the fitness mechanism is internal to the ecosystem in the predator-prey coevolutionary techniques discussed earlier, the selected mechanism was rather unusual in that the recombination and reproduction operators created an offspring from the best individuals in the neighborhood decided by the predators.

This thesis proposes a framework which uses closed adaptations of naturally occurring ecosystems to solve design and optimization problems. These engineered ecosystems will be referred to as *artificial ecosystems*. The term closed ecosystem refers to an ecosystem which is self sustaining. Resource exchange (such as immigration and emigration) between the environment and itself is assumed to be non-existent. The general framework presented here is based on an ecological metaphor

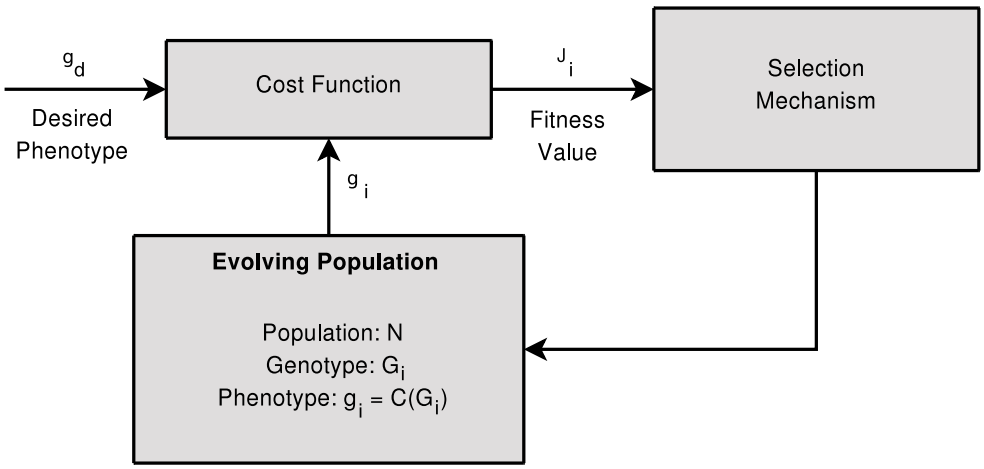


Figure 3.1. Model for a conventional evolutionary technique.

instead of any biological subsystem (like genetics or ant-colony). It is more natural to define the problem into an artificial ecosystem as it allows for a fitness criterion sustainable by the ecosystem to be automatically embedded and enforced internal to the ecosystem.

Given a design or optimization problem[11], a natural ecology model to which the problem can be mapped is selected. When *mapping* the problem to the ecosystem, the design parameter set that needs to be optimized is mapped as adaptable characteristics of one of the biological species. If more that one parameter set needs to be optimized, as in case of MOO problems, more than one species may have adaptable characteristics. Other specified parameters, constraints and information available about the problem are mapped to the environment. Finally, any modifications to the ecosystem that may increase computational efficiency without the loss of generality are applied.

Figure 3.2 shows a schematic model for an artificial ecosystem based on this framework. In the figure, arrows between any two species indicate the interspecific interactions such as predation, parasitoid, grazing, symbiotism and other similar behaviors, which can occur between individuals of different species. Apart from these, intraspecific interactions (not depicted in the figure) which may occur between individuals of the same species such as mate selection, social and territorial dominance can be modeled where required. Effects of the environment on each species are represented by the arrows between the environment and each of the species. These effects can vary from species to species are used to represent the constraints on each parameter set. All these different types of interactions together form the driving forces of the ecosystem which exert the necessary evolutionary pressure on the evolving species' genotypes to adapt optimally. This driving force can be thought of as a fitness function, which would be internal to the ecosystem and evaluated at a local level through the interactions between individuals of various genotypes. Similarly, the selection criterion for population dynamics is not enforced external but internal to the ecosystem through the interactions between individuals of same or different genotypes.

Having mapped the problem to this artificial ecosystem, the resulting ecology is simply allowed to evolve. Inspiration is taken from nature in assuming that the

[11] referred to as problem from here on.

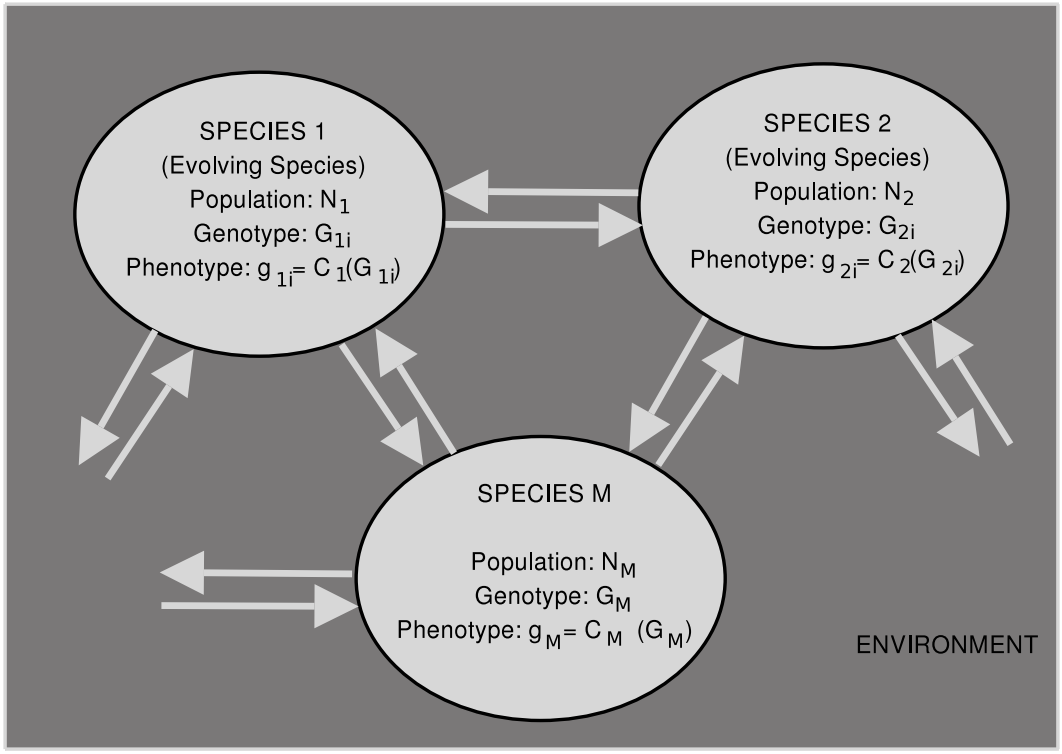


Figure 3.2. Model for artificial ecosystems methodology.

genotypes of the surviving individuals over time would have adapted optimally to the environment and therefore, would have highest levels of fitness.

3.1. INDIVIDUAL-BASED MODELING OF ECOLOGICAL PROCESSES

Modeling ecological and biological processes at individual-level allows explicit inclusion of individual variation in greater detail. This allows the model to capture and demonstrate emergent phenomena observed in natural systems in a more realistic way.

Population dynamics is one of the most important ecological processes. Population dynamics of each species of a closed ecosystem has to exhibit some form of stability. Extinction of even one species, might result in the collapse of the entire ecosystem. Population dynamics of a species in a closed ecosystem is entirely governed by reproduction and mortality processes, that is, by the population growth model of the species. The models of population growth discussed in Section 2.2 are

however described at population level and not individual level. In the case of individual based modeling, population dynamics emerge from individual level reproduction and mortality processes.

As an example, an individual-based version of the famous logistic growth model is described here. Equation (1) gives the logistic equation in its discrete form. The logistic model is density dependent, i.e. the population in the next time step (generation) depends on the population in the current time step.

$$N_{(t+1)} = N_t \cdot e^{\left\{r_0 \cdot \left(1 - \frac{N_t}{K}\right)\right\}} \tag{1}$$

where N_t and $N_{(t+1)}$ are population sizes at time t and $(t+1)$, respectively, K is the carrying capacity, and r_0 is the population growth rate.

Carrying capacity is the maximum population an environment's available resources can sustain. It is clear that the model assumes that the available resources and population growth rate, which is the difference between birth and death rates are constant. These assumptions are not always true.

For the individual-based growth model which can produce population dynamics similar to the logistic equation, each individual is assumed to have the genetic knowledge of the population density at the start of the IBM's execution. Termed here as *genetic density*, this piece of information is passed down from one generation to another. Each individual keeps track of other individuals with in an interaction area (I_{area}). Let P_i be the number of interactions an individual experiences over a time P_{age}. The average population density experienced by an individual is therefore given by $\left\{\frac{P_i/P_{age}}{I_{area}}\right\}$. Let G_d be the genetic density known to the individuals and P_d be the death rate of an individual. Then, if each individual produces number of offsprings given by equation (2) with probability of the birth rate P_b, the population will eventually reach an equilibrium point, for the given G_d, P_b and P_d.

$$N_{offsprings} = \frac{\{G_d \times I_{area}\}}{\left\{\frac{P_i}{P_{age}}\right\}} \tag{2}$$

This equilibrium point, which is equivalent to the carrying capacity, emerges from simple reproduction and mortality rules, rather than being specified. Other

specific reproduction and mortality rules like age dependent mortality, seasonal variation in reproduction, fertility depending on the position in the group [30], etc., can be added when required using simple rules to the individual reproduction and mortality behaviors.

Another important biological process used in the artificial ecosystems methodology is mobility. Mobility is demonstrated by living creatures for a variety of reasons including seasonal migration, foraging for food, escape from predators, search for reproduction site, and search for mate. At an individual-level mobility can be modeled in its simplest form as random walk. At every time step, each individual can move by a fixed distance in any direction. More elaborate rules [10, 42, 80] can be established to achieve emergent patterns such as flocking, herding, schooling, etc. In this thesis only the simple case of random walk model is considered for mobility.

3.2. INDIVIDUAL-BASED MODEL FORMULATION

To facilitate easy communication, replication and peer review of IBMs, Grimm et al.[9] proposed a standard protocol called the ODD protocol. In this section, a brief description of this protocol is presented. This protocol will be used later in this thesis to describe the IBMs developed.

The ODD (Overview, Design Concepts and Details) protocol consists of seven elements as shown in Figure 3.3. A brief description of the seven elements is as follows:

Purpose: This element discusses the model purpose and intention of the modeler. It also serves as the guide for the rest of the formulation.

State variables and scales: Description of the state variables of the low-level entities such as individuals and habitat units are provided in this element. State variables of higher-level entities such as population, metapopulation (community of populations), or landscape (collection of habitat units) are described if any. Finally, the spatial and temporal scales of the model such as size of habitat units, extent of the model world, length of time steps, length of time of the model are discussed.

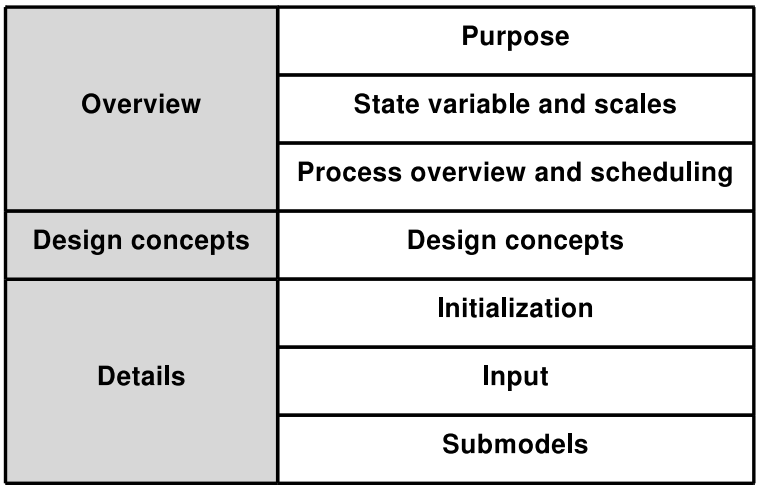


Figure 3.3. The seven elements of the ODD protocol. Reproduced from Grimm [9]

Process overview and scheduling: A verbal overview of individual processes such as reproduction, growth, movement, mortality, used in the model is provided. The scheduling issues such as the order of the process execution, asynchronous or synchronous updates and model ordering of processes which are concurrent in nature are also discussed.

Design concepts: This element consists of a checklist of items that need to be considered when designing an IBM. Item that do not apply to the model being described are simply left out.

Emergence: This describes which system-level phenomena is expected to emerge from individual traits, and which phenomena are simply imposed using rules.

Adaptation: This describes which traits of individuals are adaptable to improve their potential fitness in their environment.

Fitness: This describes how fitness-seeking modeled, if applicable. Fitness-seeking can be achieved in two ways direct fitness-seeking, in which the fitness consequence of a behavior, using a fitness measure is explicitly modeled, and indirect fitness-seeking, in which model specific behaviors are

assumed to contribute directly to fitness-seeking but are difficult to link to fitness directly.

Prediction: This describes how individuals predict future conditions (if any) to decide their current decisions.

Sensing: This describes which internal and state variables the individuals can sense and which variables they are just considered to know.

Interaction: This describes what kinds of interactions among individuals and their environment are modeled.

Stochasticity: This describes whether and why stochasticity is a part of the model.

Collectives: This describes the aggregations (groups) of individuals used in the model and their reason.

Observation: This describes what data is collected from the IBM execution for analyzing it.

Initialization: This element answers questions about the start of the model execution, such as initial values of the state variables and initial conditions of the model, and reasons for their selection.

Input: The input data used and methods to generate or obtain this data are discussed in this element.

Submodels: A detailed description of the processes listed in the "Process overview and scales" is provided in this element.

The reason for using the ODD protocol for all the IBMs in this thesis is to help the reader better understand the latter models due to the familiarity with the structure of the protocol.

4. SOFTWARE EVALUATION

The objective of this section is to evaluate and compare individual-based modeling software available to the research community. A survey and review of existing individual-based modeling platforms was conducted by Railsback et al. [81] in 2006. However, new platforms have been released since, and additional features incorporated into the existing ones. A first hand experience of using the modeling platforms seemed an ideal way to review and compare them.

Table 4.1 lists information about the software platforms evaluated.

Table 4.1. List of software platforms evaluated.

Name	Base Language	Current Website[12]
Ascape	Java	http://ascape.sourceforge.net/
MASON	Java	http://www.cs.gmu.edu/ eclab/projects/mason/
MetaABM	Java, EMF	http://www.metascapeabm.com/
Netlogo	Java	http://ccl.northwestern.edu/netlogo/
Swarm	Obj-C, Java	http://www.swarm.org/
Xholon	Java, UML, XML	http://www.primordion.com/Xholon/

4.1. EVALUATION CRITERIA

Software platforms can be evaluated based on a plethora of criteria depending on the evaluator's need and experience level. Possible criteria (not in any particular order) for individual-based modeling software include the following:

Size and status of the user base and support group – This is probably the most important criteria. A large and active user base indicates both the popularity and usability of the platform. An active support group means additional features requested by the user base can be expected to be implemented in a short

[12] As on 05-March-2008.

time. The size of the support group may not be an issue, but a larger support group is always preferable. A small and active user base and support group indicates that either the software is relatively new or that it is only suitable for a specific category of applications.

Ease of use – It is understandable that some amount of learning is involved at start of using any software, but it is possible to keep that learning curve as smooth as possible by providing sample models and extensive documentation. Ease of use criteria takes into account how difficult it is for an user to learn and use the given software.

Execution speed – Another important criteria when evaluating any software platform (not specifically to individual-based modeling softwares). It would be unwise to conclude that nobody wants to use a slower software. If other features offered by the software are exceptional, then execution speed might not be a dictating criteria. However, with individual-based simulations reaching scales of millions of agents [43], execution speed is becoming an important criteria.

Organization of the software – Organization of the software means how the files and libraries that control the operation of the software are organized. This is mainly of interest to an advanced user who might wish to add to or extend existing features of the software. A logical organization of the software simplifies this process. A high interdependence with in the libraries and subcomponents of the software makes updating/upgrading the software rather difficult and cumbersome.

Scalability – This indicates the capability of the software to solve problems of increasing complexity. A software's performance might be excellent when running simulations with low number of individuals and small number of constraints and interactions, but might perform below average due to a variety of reasons. The use of the word scalability here, also refers to the ability of the software to work in distributed and parallel systems. With the increasing availability of parallel processing systems it is rather important for a software to be able to run a serial version of the simulation on parallel architectures with little or no modification.

Portability – Almost all of scientific research is carried out by groups. Portability means how efficiently a model developed in a software can be communicated to others in the group. Platform independent and web-based languages such as Java perform well in this category. Again this is a situation specific criterion and may not be one of the deciding factors in adoption of a software.

Display capabilities – Though not always important, several platforms include in-built display capabilities. These are particularly useful for beginner and also for debugging purposes.

Programming platform – As noted with scalability, some programming languages are better suited for certain applications and therefore softwares developed on these programming languages are favorable. User competency in the base language of a software also plays an important role in its adoption.

Scheduling mechanisms – Scheduling is an important part of individual-based modeling. Certain processes of a model might have a serious restriction on the order in which they are executed. Hence, the ability of the software to provide scheduling mechanisms for a wide variety of situations is important.

4.2. EVALUATION MODEL

4.2.1. Model Description. A model of single species population dynamics is developed for evaluating platforms listed in Table 4.1. The model consists of 1000 individuals of a single species in a spatially explicit environment of size 141×141. The environment is modeled as a 2-D toroidal grid to eliminate any edge effects. Each individual has four behaviors – *interact*, *move*, *reproduce*, and *death*. Individuals interact with others with in their local environment (called interaction area) and move around their environment randomly (random walk). The life cycle of the individuals is modeled using a constant mortality rate and reproduction according to Equation (2) (Section 3.1). A complete model formulation according to the protocol described in Section 3.2 is provided in the next section. Elements of the ODD protocol which do not apply to the evaluation model are simply left out.

4.2.2. Model Formulation.

Purpose – The purpose of this model is to investigate the emergence of population dynamics of a single species ecosystem from individual biological processes and interactions. Individuals with several behaviors (biological processes) and ability to locally interact with other individuals are setup in a spatially explicit environment.

State variables and scales – The low-level entities in this model are the individuals and the habitat units of the environment. Individuals' state variables include location, age (in time steps), and interactions with others. These variables vary from individual to individual. Other internal state variable which are same for all individuals include *genetic density*, *interaction radius* and *maturity age*. Genetic density is the density of the population at the start of the simulation and is represented as genetic memory, passed from one generation to another. Interaction radius is the radius of the local interaction zone of an individual. Maturity age represents the age after which an individual may reproduce. The environment is 2-D toroidal grid of size 141×141, each location of which is called *habitat unit*, capable of housing one individual at a time. A toroidal grid[13]is assumed to eliminate any edge effects. The state variable of the habitat units is a single boolean value representing whether the habitat unit is occupied or not. The spatial scale of the simulation is confined to the environment (141×141) and the temporal scale of the simulation is 3000 discrete time steps. A time step is completed when all the individuals have executed all their behaviors once. Therefore, time steps may or may not be of same duration when measured in wall clock time.

Process overview and scheduling – Each individual is associated with four behaviors or processes. They are interact, move, reproduce and die. During *interact*, each individual increments an internal counter by the number of other individuals with its interaction radius. Individual movement is controlled by the *move* behavior. At each time step, individuals try to move to a vacant

[13]In this case the right end of the wraps around to the left end, and the top end wraps around to the bottom end forming a torus. This is also known as symmetric boundary condition.

location in their immediate 8-neighborhood. If no vacancy exists, an individual remains in its current location. During *reproduce*, all mature individuals produce offsprings with a certain success rate. If successful, the number of offsprings is inversely proportional to the parent's experienced population density (Equation (2)). Offsprings are spawned one on each available vacant location in the parent's 8-neighborhood. If no locations are available, the offspring is not spawned. A constant death rate is used in the model. During the *die* behavior, each individual draws a number from an uniform random number generator to test it chance of survival. If the number drawn is less than the death rate, the individual dies, else lives for the next time step. Apart form these individual behaviors, a global process, *update age*, is executed at the start of every time step, which increments the age of all individuals by one.

Scheduling is done as follows. At the start of every time step, the update age process is executed first. Then one randomly selected individual at a time executes its behaviors in a predefined order, until the entire population has executed all the behaviors. The behaviors are executed in the order – interact, move, reproduce, and die.

Design concepts –

Emergence: Although individual life cycle (movement, reproduction, and mortality) are described by empirical rules and probabilities, the population dynamics emerge fro the behaviors and interactions of the individuals.

Sensing: Individuals can sense other individuals within their local interaction radius. Also each individual is assumed to know its own age and reproduction capabilities, i.e., the individual does not receive information about its maturity and number of offsprings to produce from any external agency.

Interaction: Interaction between two individual is explicitly modeled, and is used to keep track of the number of other individuals in their interaction radius.

Stochasticity: Individual birth and death events are modeled via probabilities, which add stochasticity to the model. To reduce the effect of stochasticity, each simulation is repeated 10 times, from which respective mean values are reported as results.

Observation: Population size is recorded at the end of each time step.

Initialization – At the start of the simulation, individuals are placed randomly in the environment, with only one individual at any given location. The age and interactions of the initial population are initialized using equations (3), (4), and (5). Table 4.2, lists the parameters used in the evaluation model and their default values.

$$P_{ma} = round\left\{\frac{0.25 \times 100}{P_d}\right\} \tag{3}$$

$$P_a = U\left\{0, 3 \times P_{ma}\right\} \tag{4}$$

$$P_i = round\left\{P_a \times G_d \times I_{area}\right\} \tag{5}$$

where P_{ma} is the maturity age of the individual, P_d is the death rate and $U(x, y)$ is an uniform random number generator, generating values from x to strictly $(y - 1)$ and P_i is the interaction counter of an individual.

Submodels –

Interact: During *interact*, the interaction counter P_i is incremented by the number of individuals, including itself, in the interaction radius (R_i), as shown in Figure 4.1.

Move: During *move*, each individual moves randomly to a vacant location in their 8-neighborhood as shown in Figure 4.2. If no location is available the individual stay in its current location.

Reproduce: During *Reproduce*, each mature adult ($P_a > P_{ma}$) reproduces with a probability of P_b. If successful, the number of offsprings is given by (Equation (2)). However, the actual number of offsprings produced is

Table 4.2. Overview and default values of parameters for the evaluation model.

Parameter	Value
Environment Parameters	
Habitat width (cells) (W)	141
Habitat height (cells) (H)	141
Display width (cells)	564
Display height (cells)	564
Individual Parameters	
Initial population size (N_P)	1000
Initial location	randomly placed, with one individual per location
Interaction radius (cells) (R_i)	5
Interaction area (cells). Number of cells in R_i	81
Birth rate (%) (P_b)	20
Death rate (%) (P_d)	10
Genetic density (G_d)	$\frac{N_p}{W \times H}$
Mobility (cells) (P_m)	1
Maturity age (time steps) (P_{ma})	3 (See eqn. (3))
Offsprings produced ($N_{offsprings}$)	(See eqn. (2))
Initial population age (P_a)	(See eqn. (5))
Initial interactions (P_i)	(See eqn. (5))

less than or equal to the number of vacant locations in the parent's 8-neighborhood. If no locations are available, then further offsprings are not produced until a vacany is available.

Death: There is a constant probability, P_d, of death in each generation (iteration). Each individual draws a random number between 0 and 100 with uniform probability. If the number is less than P_d, the individual dies and is immediately removed from the simulation state.

4.2.3. Implementation. The evaluation model is developed on each of the softwares listed in Table 4.1. A brief description of the softwares and the implementation procedure is discussed in this section.

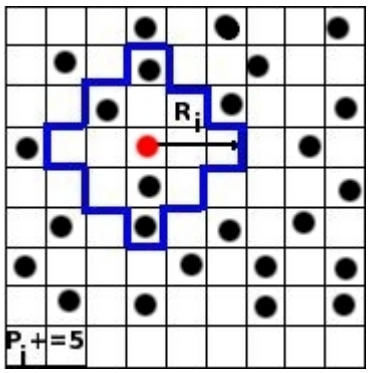


Figure 4.1. *Interact* behavior of an individual. The red dot indicates the individual whose radius R_i creates the interaction area bounded by blue lines. In this case the interaction counter P_i is incremented by 5.

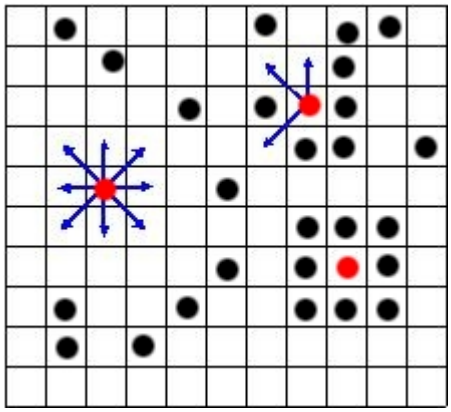


Figure 4.2. *Move* behavior of an individual. Red dot are the individuals who are executing their move behavior. The blue lines indicate the possible locations to move, out of which one location is selected with uniform probability. The individual with no possible locations to move will remain in the current location.

Ascape: Ascape is a open source, free to use software, developed for general-purpose individual-based models. Ascape is written in Java and runs on any platform which supports Java. Models are written in Java using additional classes and functionality provided with Ascape. For users who prefer Integrated Development Environments (IDEs), Ascape integrates easily into the popular Eclipse[14] environment. Ascape provides good visualization capabilities which is useful

[14]Visit http://www.eclipse.org for more information on the Eclipse framework.

for demonstration purposes. Though not extensive, decent documentation on Ascape exists at the project website (refer Table 4.1). Being based on Java, Ascape is relatively easy to use for any one with experience Java or any other Object-Oriented Programming (OOP) language.

Implementation of the evaluation model in Ascape was relatively easy with the use of Eclipse IDE. Java/Ascape provides useful and easy to track error messages which simplfy the implementation process. Figure 4.3 shows the evaluation model running in Ascape.

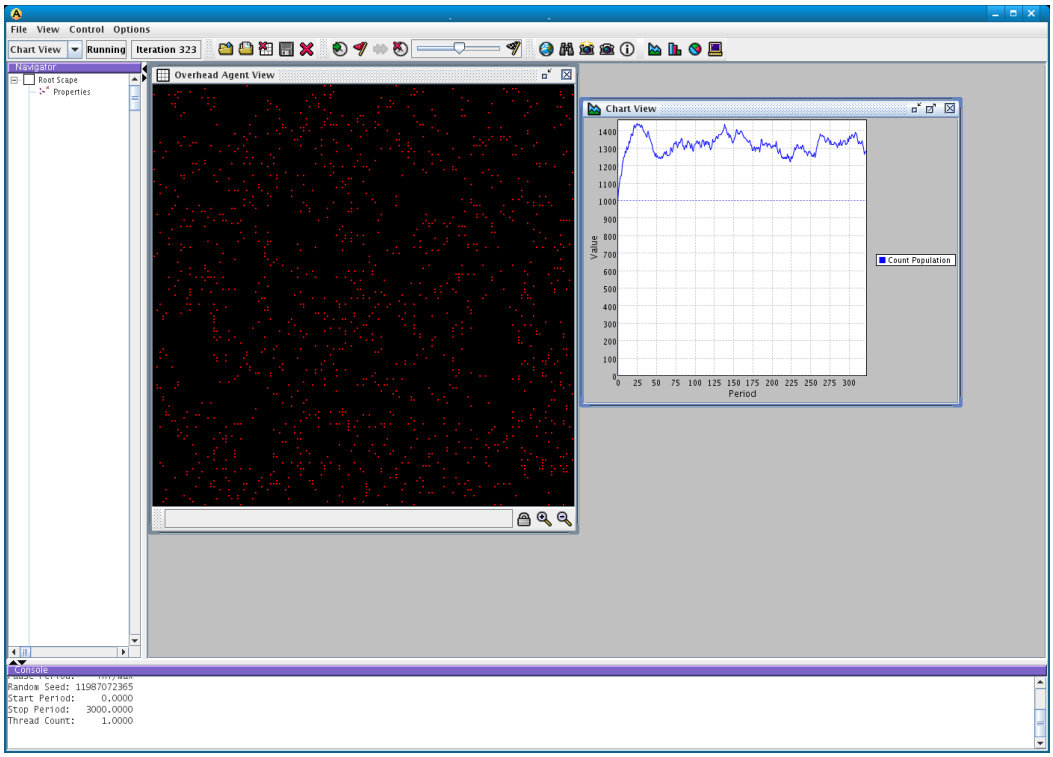


Figure 4.3. Evaluation model running in Ascape. Red dots indicate the locations occupied by individuals and the plot shows the population dynamics.

MASON: MASON is also Java based open source and free to use software for individual-based modeling. MASON was developed with execution speed as the main criteria [81]. Models in MASON typically consists of a layered structure [82] with the lowest layer being the classes describing one or more kinds

of individuals. The next layer, called "*Model Layer*", consists of the rules for scheduling and executing individuals' behaviors. Another decoupled layer, "*Visualization Layer*" controls the display aspects of the model. This kind of layered structure enables easy modification of the model. Very good documentation and large number of code sample are available at the project's website which make MASON easy to learn and use. Implementing the evaluation model in MASON was easy due to the hierarchical structure of the code. Again debugging any errors which occurred was simplified by the helpful error messages and good documentation. Figure 4.4 shows the evaluation model running in MASON.

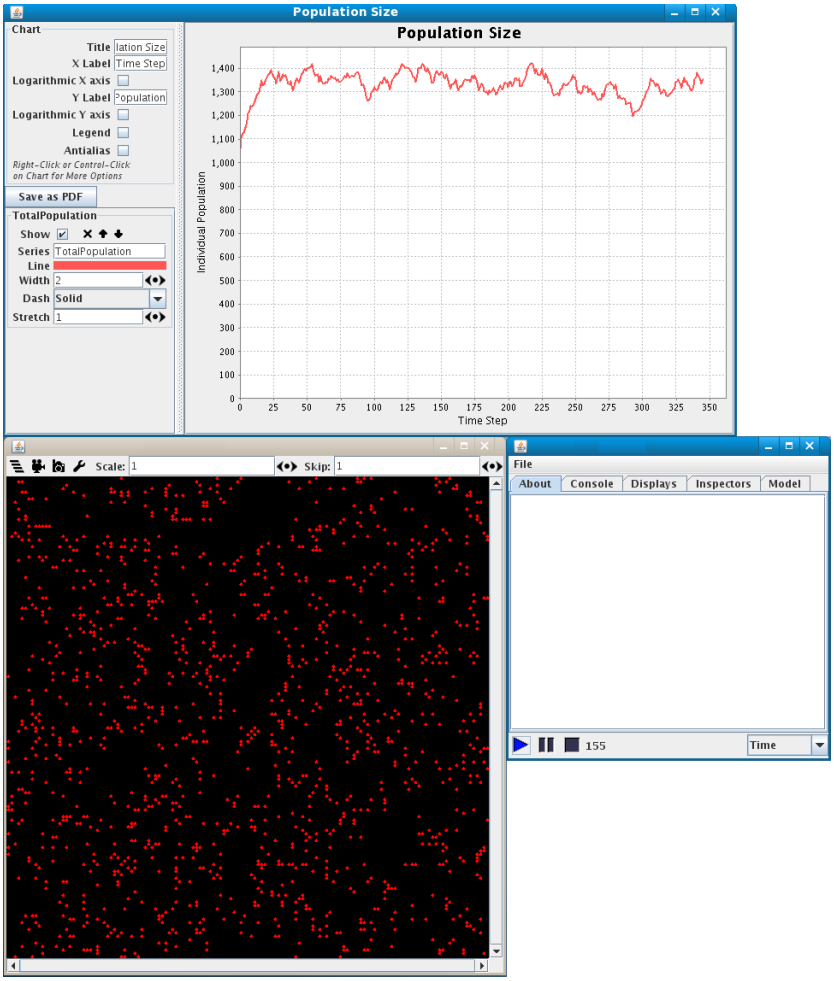


Figure 4.4. Evaluation model running in MASON. Red dots indicate the locations occupied by individuals and the plot shows the population dynamics.

MetaABM: MetaABM is a recently introduced open source and free to use software tool for individual-based modeling and is targeted towards modelers who are not familiar with or not interested in programming. Although metaABM allows Java code snippets to be integrated into the model, model development is mainly carried out using Eclipe Modeling Framework (EMF) based GUI programming. Figure 4.5 shows the MetaABM development environment. The developed model is then run either in Ascape or Repast [15] environments. No good documentation or code samples are provided which makes initial model development rather cumbersome.

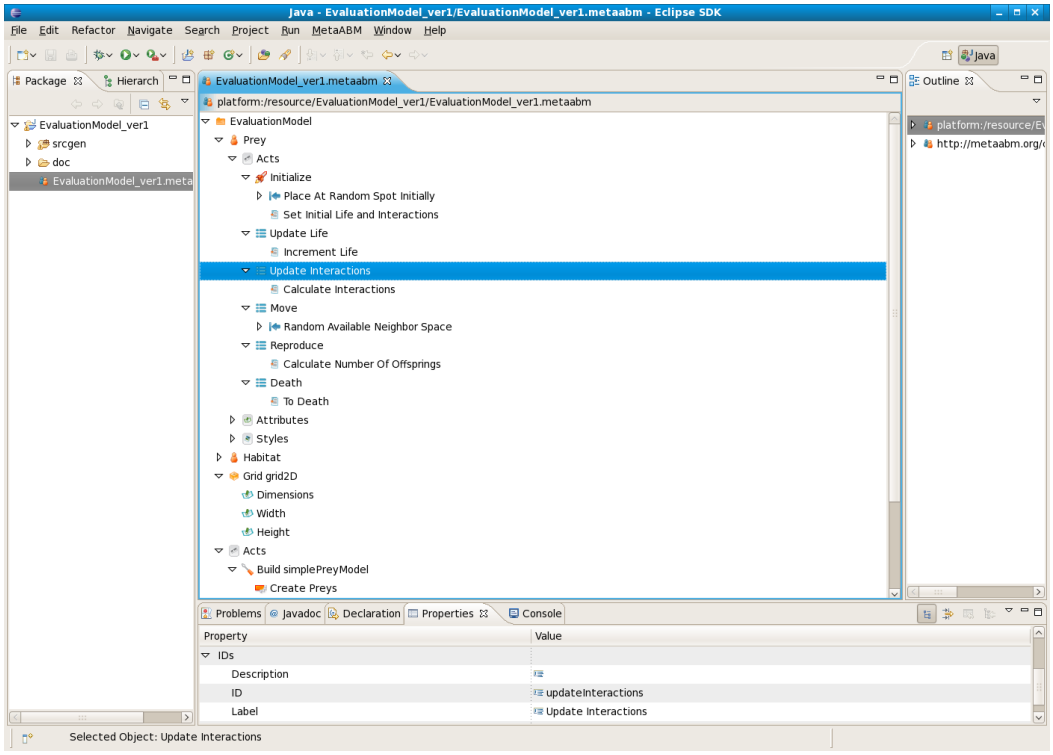


Figure 4.5. MetaABM development environment.

Implementing of the evaluation model was found to be rather difficult even with the easy-to-use GUI programming facility. This can be mainly attributed to the

[15] For more information visit the project's website at http://www.metascapeabm.com.

lack of proper documentation. Also the final model developed is rather complicated, as many GUI programming steps were required to achieve something which could have been done in couple of lines of code. Figure 4.6 shows the evaluation model running in Ascape mode of MetaABM.

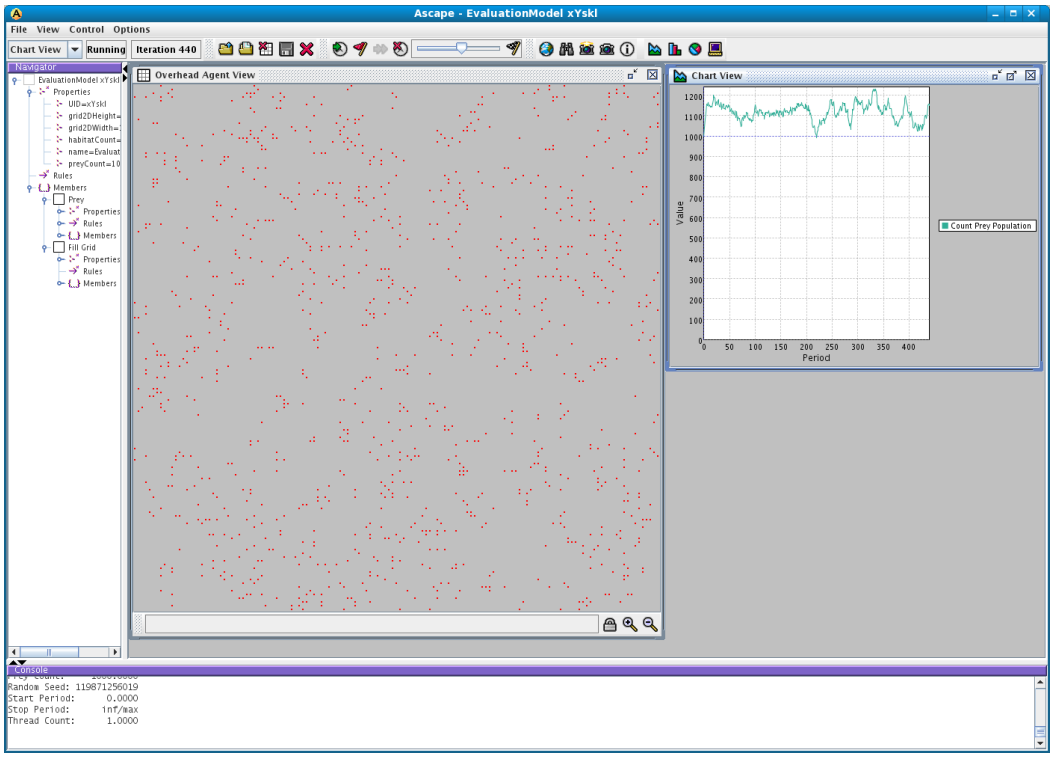


Figure 4.6. Evaluation model running in MetaABM. Red dots indicate the locations occupied by individuals and the plot shows the population dynamics.

NetLogo: Although free to use, NetLogo is the only software in this evaluation which is not open source. However, due to its extreme flexibility, and simplicity it is one of the most used individual-based modeling software. A descendant of the Logo[16] family, NetLogo is written in Java and is available for a large number of platforms. With the largest collection of code samples and extensive documentation NetLogo is ideal for beginners in the field of individual-based

[16] A popular programming language for kids. For more information visit the Logo Foundation website at http://el.media.mit.edu/Logo-foundation.

modeling. An easy to use Logo-like language is used for developing models in NetLogo. The language is very high-level and is very easy to learn. NetLogo also has an in-built GUI display and a large collection of GUI elements like buttons, sliders and charts ready for use.

Implementing of the evaluation model was the simplest in NetLogo. The programming language is both simple to learn and intuitive. Error reporting in NetLogois not only done via easy to track messages, but also via visual means by pointing to the location in the code where the error has occurred. Figure 4.7 shows the evaluation model running in NetLogo.

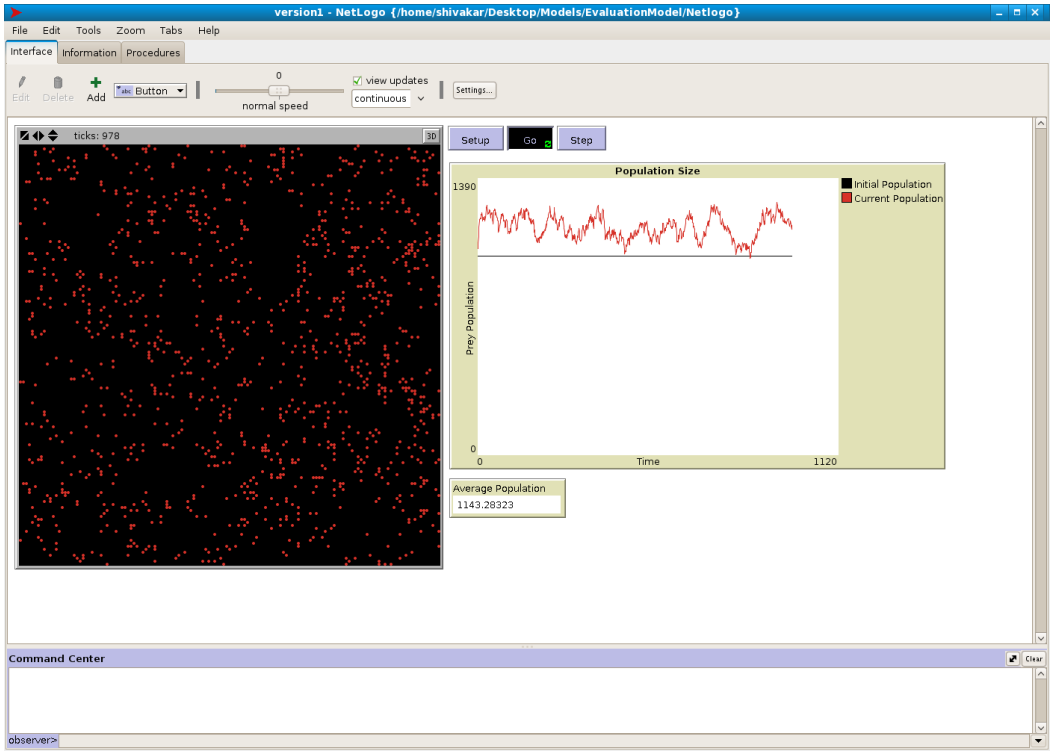


Figure 4.7. Evaluation model running in NetLogo. Red dots indicate the locations occupied by individuals and the plot shows the population dynamics.

Swarm: Swarm is one of the oldest and highly used platform for individual-based modeling. Two versions of swarm, based on Objective-C and Java – both open

source and free to use, are available for use. For the current evaluation, only Objective-C based version is used as the Java based version was reported to be extremely slow [81] and is in a process of complete reimplementation. Good code samples and extensive documentation are available for Swarm. However, Objective-C is not currently a popular programming language and adds to the learning curve for Swarm. Models in Swarm are designed as a hierarchy of *"swarms"*, in which each swarm schedules the lower-level swarms. The top-most swarm, the *Observer Swarm*, is responsible for the visualization elements of the model. *Model Swarm* is the next level swarm which is responsible for scheduling the individuals' behaviors. The lowest-level swarms are the "Agents" which describe the behaviors of the individuals.

Implementing the evaluation model in Swarm was found to be relatively difficult and can be attributed to the lack of familiarity with Objective-C. Due to the concept of weak-typing[17], incomprehensible run-time errors are reported when programming with Objective-C, which cannot be easily tracked without familiarity with the language. Swarm is the only software evaluated, that does not provide a toroidal grid class/structure. Also, the restriction of one individual per habitat unit was placed due to Swarm's inability to handle more than one individual per location. This places serious constraints on which models can be developed using just the functionalities provided by Swarm. Figure 4.8 shows the evaluation model running in Swarm.

Xholon: Xholon is another recently introduced software for individual-based modeling and is both open source and free to use. Xholon is based on a combination of Java and XML[18]. Models are programmed using Java and XML, or imported from UML[19] diagrams. A model in Xholon can be considered as a tree with all the components of model as nodes in the tree. Both individuals and their habitats are hence nodes and behaviors like movement are simply implemented as moving one node of the tree navigating to another node of the tree to interact.

[17] As opposed to strong-typing (as in C++), where the arguments passed to a function are checked for compatible class types during program compilation, weak-typing does not check for the class types at compile time. This results in a run-time exception.

[18] Extended Markup Language.

[19] Unified Modeling Language

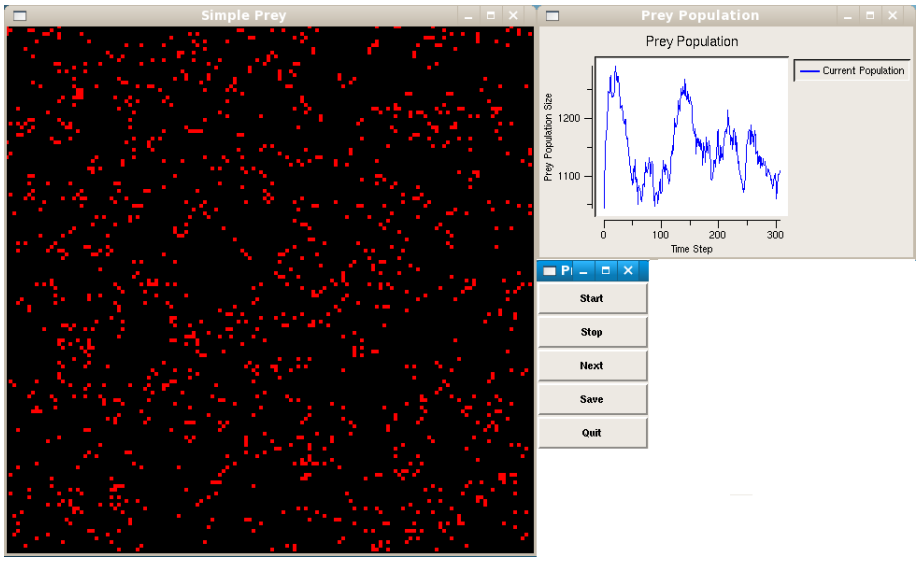


Figure 4.8. Evaluation model running in Swarm. Red dots indicate the locations occupied by individuals and the plot shows the population dynamics.

Navigation of trees is accomplished using XPath[20] syntax, which is a simple and easy to learn. Nodes can also create, delete or move other nodes and subtrees (as allowed by the model). Communication between nodes is accomplished via UML ports and connectors. Although recently introduced, a large collection of code samples are provided for the beginner. However, the documentation is still quite insufficient and incomplete.

Implementing the evaluation model in Xholon produced mixed reactions about the software. Although the lack of proper documentation was hindering, familiarity with the languages involved permitted the development of the model within a reasonable amount of time. Figure 4.9 shows the evaluation model running in Xholon.

4.3. RESULTS AND RECOMMENDATIONS

Based on the development experience and results from the evaluation model, observations for relevant criteria discussed in Section 4.1 are drawn about the software platforms listed in Table 4.1.

[20]XML Path Language

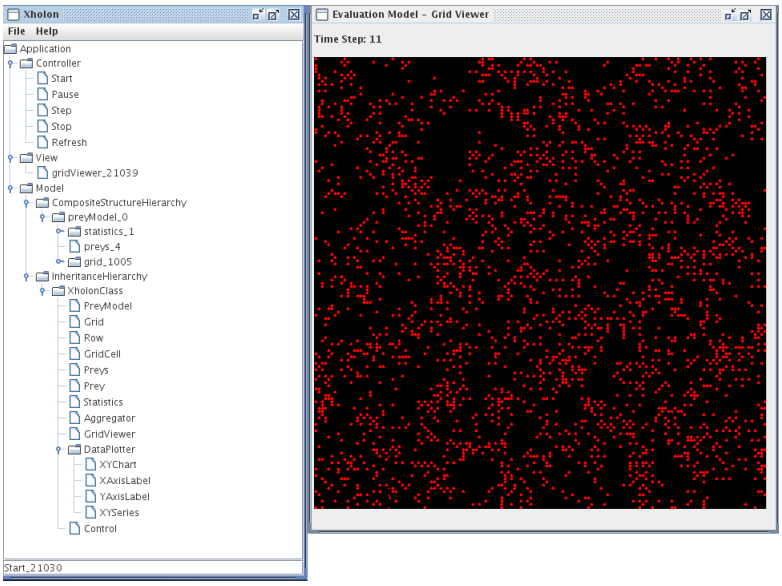


Figure 4.9. Evaluation model running in Xholon. Red dots indicate the locations occupied by individuals.

Size and status of the user base and support group: Swarm and NetLogo are probably the most used and supported of the platforms evaluated. MASON has a very responsive support group, but the size and status of the user base could not be determined. Information about the size of the user base or support group could not be determined for Ascape, Xholon and MetaABM.

Ease of use : NetLogo is the easiest to learn and use among the platforms evaluated. MASON comes in a close second due to the extensive code samples available. Ascape and MetaABM are relatively easy to use when compared to Swarm, which was most difficult to learn, even with extensive documentation and code samples provided. This is attributed to the unfamiliarity with Objective-C, which is used to develop Swarm models. Although Xholon requires experience in a combination of languages, familiarity with these languages made model development easy and relatively simple. However, the evaluation model developed ran much slower (see Table 4.3) than the sample models provided with the software indicating that optimal features of the software were not used while developing the model.

Execution Speed: Table 4.3 shows the time (in sec) taken by each software to run the evaluation. The evaluation model was developed with simple model structure as the main criteria and was not optimized for speed in any of the softwares. Hence, the results should not be construed as representative capability of the software. Having said that, even when no attempt is made to optimize the model for speed, MASON outperformed all the other software by more than 200%.

Table 4.3. Wall clock times (in sec) for running the evaluation model in each software.

Software	With GUI	Without GUI
Ascape	63.6	57.4
MASON	40.6	27
MetaABM	67.6	63.2
Netlogo	181.2	151.6
Swarm	202.4	174.4
Xholon[21]	2840	2580

Scalability: Of all the softwares evaluated, only MASON was found to have the capability to run the model in a distributed processing environment. Although NetLogo has the capability to run distributed simulations via the use of HubNet[22], it is only useful where each instance of the simulation is controlled by a human operator and is not intended for improving speed. MASON on the other hand provides separate classes to parallelize model on a multi-processor system.

Display Capabilities: All the softwares evaluated provided a rich set of visualization capabilities with Swarm and MASON being the ones which made it transparent. In these two platforms, user is provided the option of configuring

[21]Times reported may not be representative of Xholon's capabilites. Sample models provided with Xholon ran much faster than the evaluation model.

[22]For more information on HubNet visit http://ccl.northwestern.edu/netlogo/hubnet.html.

even the GUI elements of the model. Although NetLogo provides a large collection of GUI elements which can be used in a model, configuring them is limited to altering the size, location and color of the elements. In Xholon, adding even simple GUI elements like a chart/plot was found to be difficult and could not be achieved.

Based on the above observations, MASON and NetLogo are adopted for the models developed in this thesis. NetLogo is used for prototyping and MASON for generating results.

5. SINGLE SPECIES POPULATION DYNAMICS (SSPD)

An extension to the single species ecosystem developed in Section 4.2 is presented in this section. Constraints placed on the model due to limitations of software platforms, namely, one individual per habitat unit and inability to spawn offsprings due to non-availability of vacant locations are removed in the current model. The effects of input parameters on the population dynamics are also explored.

5.1. MODEL FORMULATION

The formulation of the SSPD model is exactly the same as the model formulation of the evaluation model discussed in Section 4.2, except for changes in the *move* and *reproduce* behaviors of the individuals. Only these changes are discussed here.

Move: Unlike the evaluation model in which individuals could move only to vacant locations in their 8-neighborhood, in the SSPD model, individuals are free to move to any location (occupied or vacant) in their 8-neighborhood as shown in Figure 5.1.

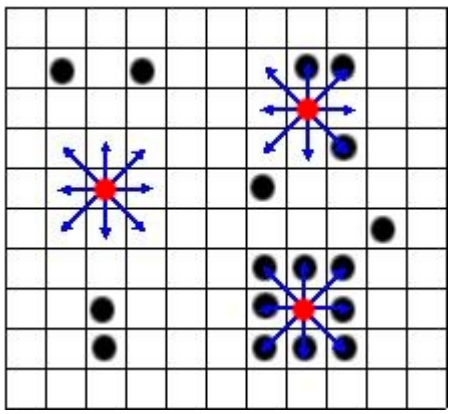


Figure 5.1. *Move* behavior of an individual in SSPD model. Red dots are the individuals who are executing their move behavior. The blue lines indicate the possible locations to move, out of which one location is selected with uniform probability.

Reproduce: During the *reproduce* behavior, individuals spawn offsprings in one of the locations in their 8-neighborhood whether it is occupied or vacant, this way all the offsprings that were intended to be produced are infact produced, giving a more realistic estimate of the population dynamics. Figure 5.2 shows the *reproduce* behavior of the individuals.

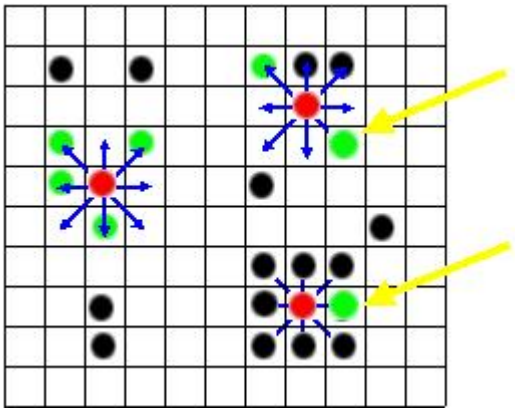


Figure 5.2. *Reproduce* behavior of an individual in SSPD model. Red dots are the individuals who are executing their reproduce behavior and the green dots are the offsprings spawned. The blue lines indicate the possible locations to spawn their offsprings, out of which one location is selected with uniform probability. Locations pointed by the yellow lines are occupied by more than one individual.

5.2. RESULTS AND DISCUSSION

Figure 5.3 shows the trend of population curve during ten runs. Average of the ten runs is reported in Figure 5.4 as the representative trend of the SSPD model. Averaging also eliminates any anomalies which may have occurred due to stochastic birth and death processes.

Figure 5.5 shows the convergence of population towards the equilibrium size determined by a given set of parameters. The plots of the figure were generated by manually keeping the genetic density (G_d)[23] constant for various initial population

[23]Instead of calculating as density of initial population.

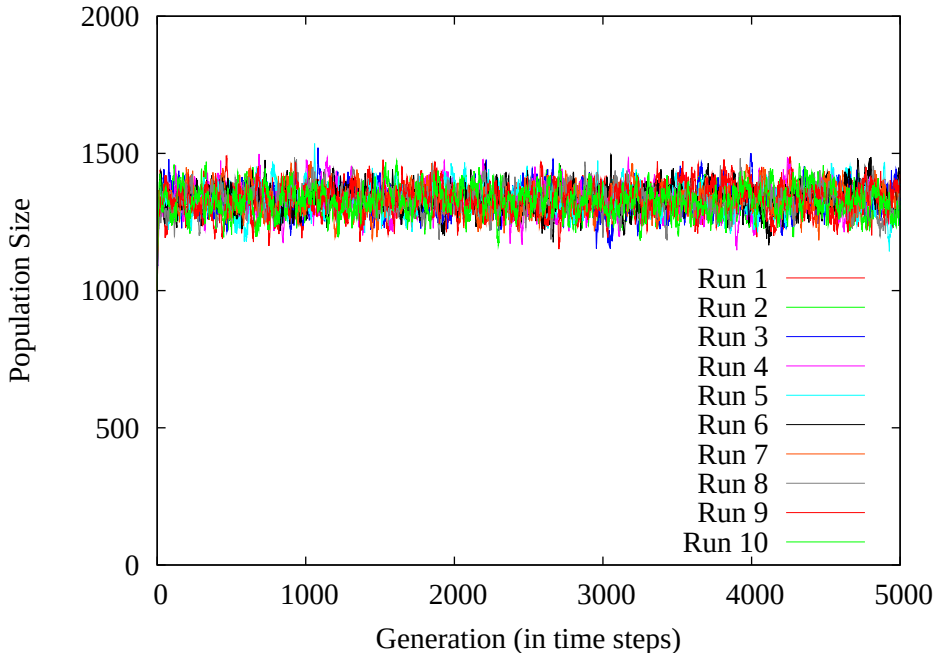


Figure 5.3. Trend of population size during ten runs.

sizes. In each case, the population reaches the stable equilibrium point dictated by the genetic density with 250 time steps and maintains this state for unlimited periods of time.

For testing the effect of individual mobility on average population size, the maximum distance that can be traveled by an individual in a single time step was varied from zero to half the world size (70). Individuals could choose to travel any distance from 0 to the max possible. Also the offsprings could be spawned within this same distance, i.e. anywhere from the current location to the maximum distance that can be traveled. Figure 5.6 demonstrates the effect of individual mobility on average population size, which is approximately equal to the stable population size, for long runs. As can be seen from the figure, except for zero mobility, any other mobility does not have any effect on the average population size. In the case where mobility is zero, locally concentrated clusters increase the average population density experience by an individual and therefore lower number of offsprings are produced (see Equation (2)). Due to this effect, population eventually goes to extinction. Figure 5.7 shows the formation of these local clusters.

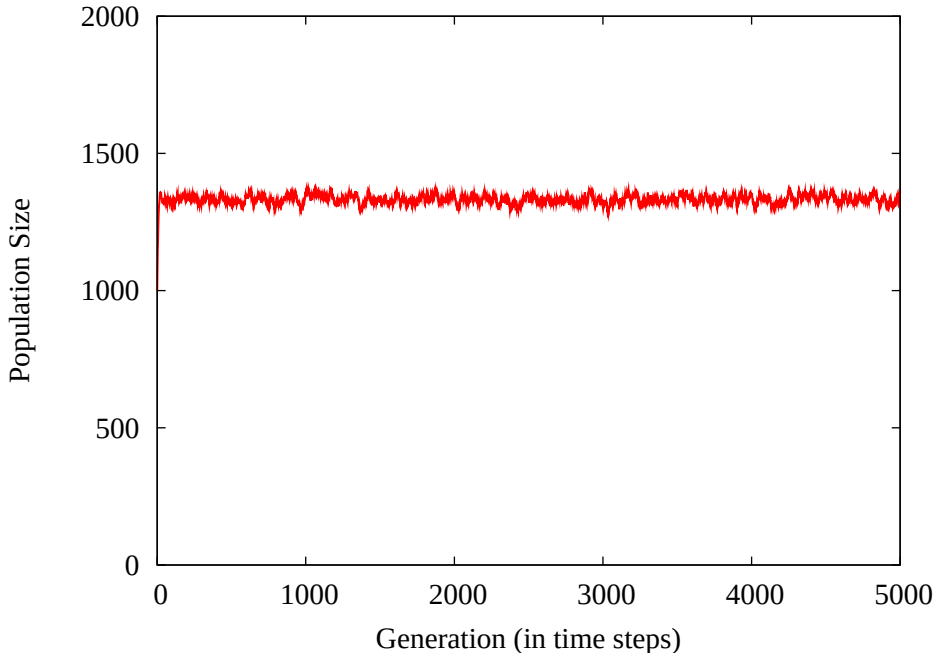


Figure 5.4. Representative trend (average of ten runs) of population size.

For a given set of parameters, both genetic density and growth rate (birth rate death rate), are equivalent to specifying the carrying capacity of an ecosystem (See Equation (1)), i.e., increasing these parameters results in the increase of the equilibrium population size. Figures 5.8 and 5.9 show the effect of genetic density and birth rate on the average population size. Genetic density (G_d) is varied from 0 (no individuals) to 1 (number of individuals equal to the number of habitat units). In Figure 5.9, the death rate is kept constant ($P_d = 10$) and the birth rate is varied from 0 to 100. As can be seen from the figure, for birth rates lower than the death rate (negative growth rate), the population goes to extinction.

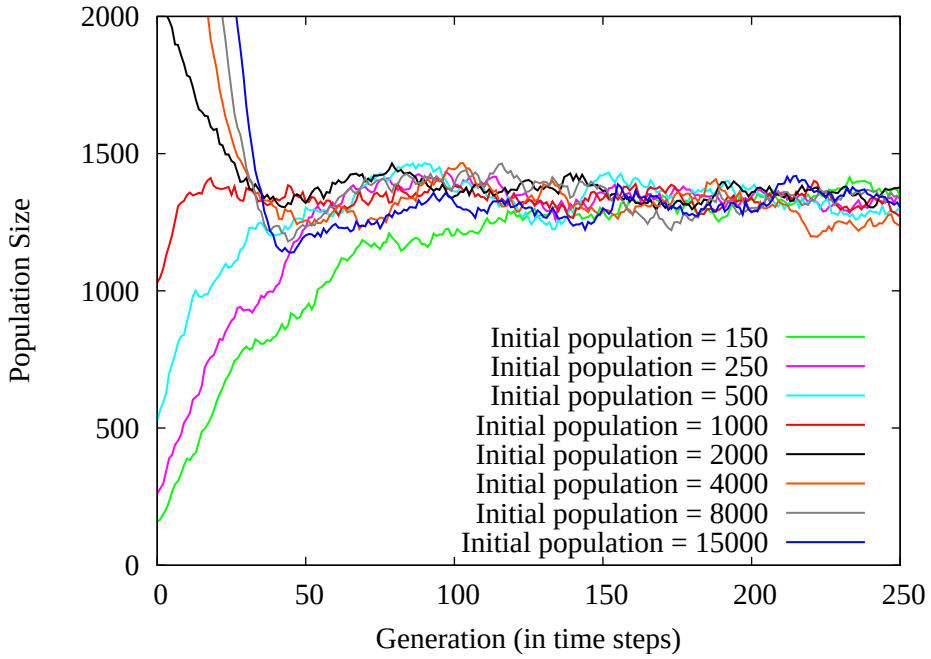


Figure 5.5. Convergence of population size to the equilibrium dictated by a given genetic density. Genetic density is calculated using an initial population of 1000 ($G_d = 0.0503$).

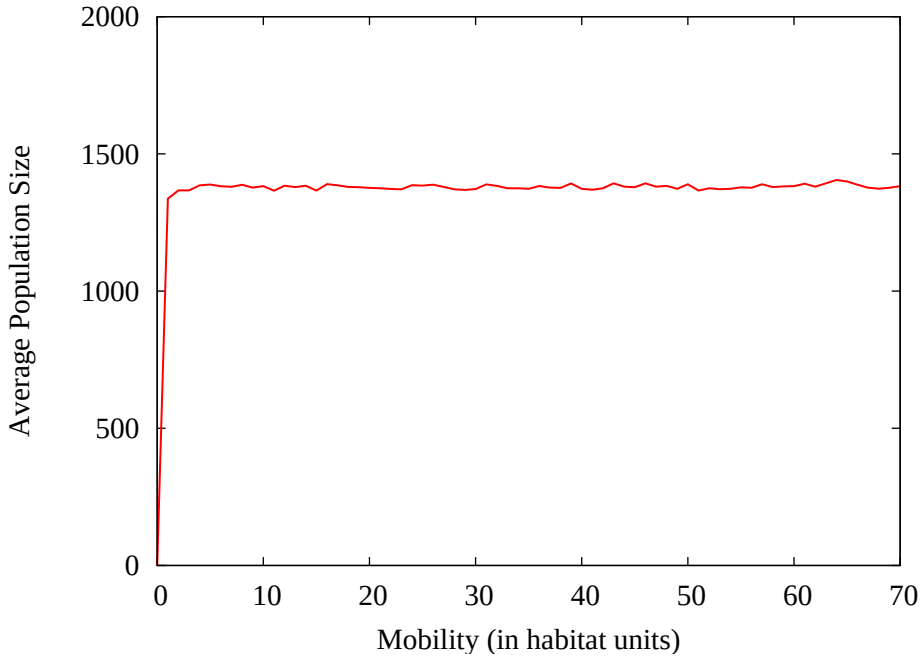


Figure 5.6. Effect of individual mobility on average population size.

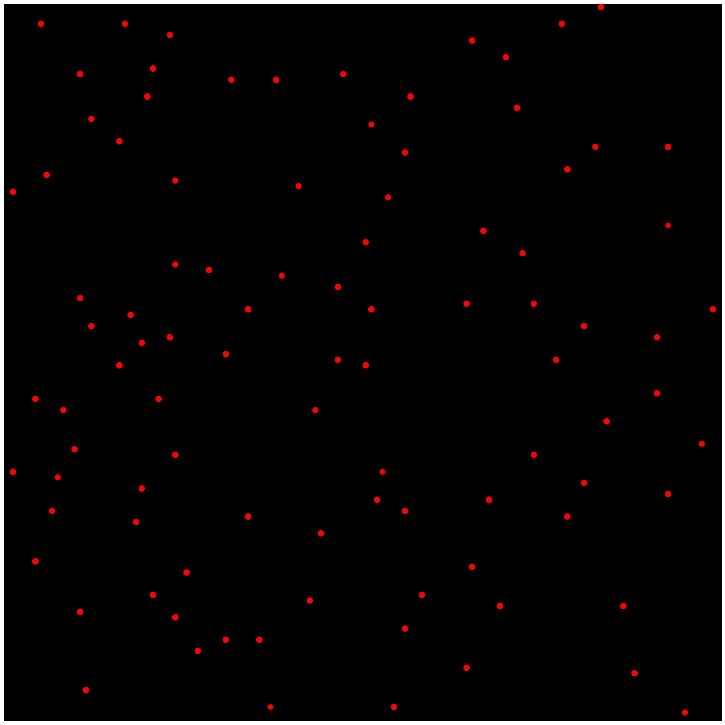


Figure 5.7. Formation of local clusters when the individuals are immobile. Figure reported at time step 500 and the number of individuals is over 700.

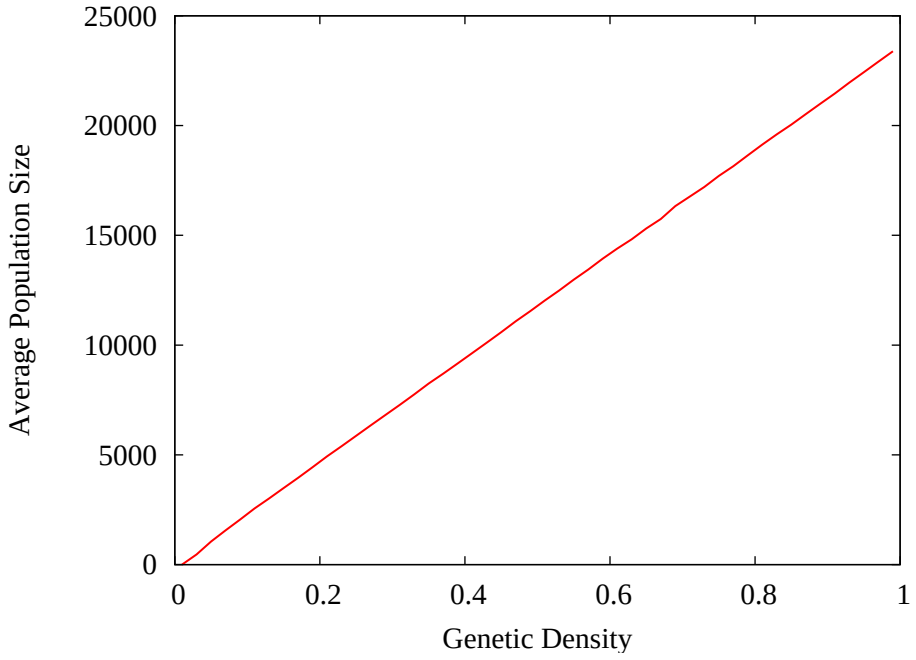


Figure 5.8. Effect of genetic density on average population size.

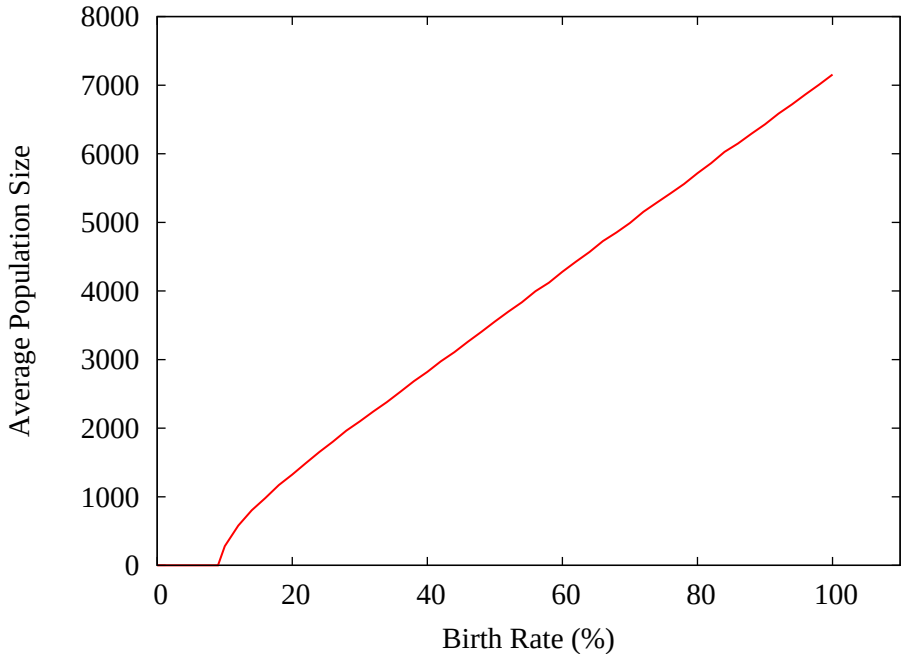


Figure 5.9. Effect of birth rate on average population size.

6. PREDATOR-PREY ECOSYSTEM (PPE)

Predator-prey ecosystem is one of the most widely studied interspecific competition models. It consists of two interacting species with one (prey) being the food source of the other (predator). Lotka [51] and Volterra [52] independently proposed the famous Lotka-Volterra (L-V) equations to model a predator-prey ecosystem. The original L-V equations are given in Equations (6a) and (6b).

$$\frac{dN}{dt} = aN - bNP \tag{6a}$$

$$\frac{dP}{dt} = cNP - dP \tag{6b}$$

where N and P are numbers of prey and predators, a is the prey growth rate, b is predation constant, c is the prey contribution to the predator population, and d is the predator death rate.

The L-V equations consist of two parts, one for the prey and one for the predator population. The prey equation consists of two parts. The first is the growth function of the prey population (positive), and second is the loss of prey population due to predation (negative). The predator equation also consists of two parts. The first is the growth function of the predator population and the second is the loss of predator population due to mortality.

The original L-V equations, used the exponential growth function (usually with a positive malthusian coefficient[24] for the prey population. Therefore, in the absence of predators, the prey population would increase exponentially. To remove this unrealistic effect, the prey population is usually modeled using the logistic model (Equation (7)).

$$\frac{dN}{dt} = aN(1 - N/K) - bNP \tag{7}$$

where K is the carrying capacity of the prey population.

[24]A positive mathusian coefficient or population growth rate results in the case where the population exponentially increases over time.

The predator growth (cNP) is a function of predation rate, i.e. if more prey are eaten (killed) the predator population increases exponentially. This contributes to the decline in prey population, second part of the prey equation (bNP) and is termed as the predator functional response [54].

6.1. MODEL DESCRIPTION

A simple PPE model is developed here, considering a prey population with logistic growth and a constant predator population. The idea behind this is to evaluate and demonstrate the functional response of the predator on the prey population.

The model consists of 1000 prey in a spatially explicit environment of size 128×128. Prey in the current model are comparable to individuals in the SSPD model and have the same four behaviors – *interact*, *move*, *reproduce*, and *death*. The predators are modeled as immortal, immobile and impotent, i.e., the predators cannot move, reproduce or die. Predators are associated with only one behavior – *predation*. The prey population dynamics is modeled using the SSPD model.

6.2. MODEL FORMULATION

Purpose – The purpose of this model is to investigate the predator functional response on a prey population. An interaction-based density-dependent population model developed in Section 5 is used for the prey population. To simplify the model, a constant predator population is used.

State variables and scales – Habitat units, predators, and prey are the low-level entities of this model. Prey state variables include location, age (in time steps), and interactions with other prey. Internal constants of the prey population include – *genetic density*, which is the density of the prey population at the start of the simulation, *interaction radius*, is the radius of the local interaction zone of a prey, and *maturity age*, which represents the age threshold for reproduction. Predators on the other hand, have no state variables. The environment is a 2-D toroidal grid with each location – *habitat unit*, capable of housing any number of individuals. The spatial scale of the simulation is confined to the environment (128×128) and the temporal scale of the simulation is 3000 discrete time steps.

As with the SSPD model, a time step is completed when all the individuals have executed all their behaviors once. Therefore, time steps may or may not be of same duration when measured in wall clock time.

Process overview and scheduling – Prey are associated with four behaviors and predators with only one. Prey behaviors include *interact*, *move*, *reproduce*, and *death*. These processes are exactly similar to the individual behaviors in the SSPD model.

Design Concepts –

Emergence: Although prey life cycle (movement, reproduction, and mortality) and predation are described by empirical rules and probabilities, the population dynamics emerge from the behaviors and interactions of prey among themselves, and the with the predators.

Sensing: Predators can sense prey within their predation radius. Prey can sense other prey within their local interaction radius. Each prey is also assumed to know its own age and reproduction capabilities, i.e., when to reproduce, how many offsprings to spawn is decided by the prey based on its age and interactions.

Interaction: Interaction between a predator and a prey, and between two prey is explicitly modeled. Interaction among prey is used to keep track of the number of other prey in their interaction area. Interaction between predator and prey includes, the predator sensing prey within its predation radius and attempting to kill it.

Stochasticity: Prey birth and death events, and predators' success during predation are modeled via probabilities, which add stochasticity to the model. To reduce the effect of stochasticity, each simulation is repeated 10 times, from which respective mean values are taken as representatives.

Observation: Prey population size is recorded at the end of each time step (generation).

Initialization – Predators are placed at the centers of the four quarters of the environment as shown in Figure 6.1. Prey are randomly placed in the environment. State variables *age*, and *interactions* of the prey are initialized using Equations (3), (4), and (5). Table 6.1, lists the parameters used for the PPE model and their default values.

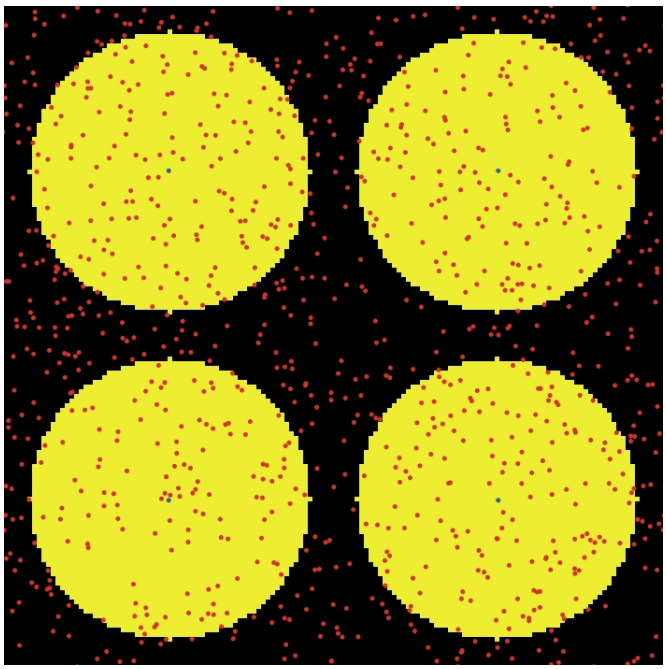


Figure 6.1. PPE model setup. Red dots are the prey, green dots are the predators, and the yellow areas represent predation zones.

Submodels –

Prey Processes:

Interact: During *interact*, the interaction counter P_i is incremented by the number of prey, including itself, within the interaction radius (R_i), as shown in Figure 4.1.

Move: During *move*, each prey moves randomly to another location in their 8-neighborhood as shown in Figure 5.1.

Table 6.1. Overview and default values of the parameters for the PPE model.

Parameter	Value
Environment Parameters	
Habitat Width (cells) (W)	128
Habitat Height (cells) (H)	128
Display width (cells)	512
Display height (cells)	512
Prey Parameters	
Initial number of prey (N_{prey})	1000
Initial prey location	randomly placed
Interaction radius (cells) (R_{prey})	5
Interaction area (cells). Number of cells in R_{prey}	81
Birth rate (%) (P_b)	20
Death rate (%) (P_d)	10
Genetic density (G_d)	$\frac{N_p}{(W \times H)}$
Mobility (cells) (P_m)	1
Maturity age (time steps) (P_{ma}) (See eqn. (3))	3
Offsprings produced (N_{os})	$\frac{\{G_d \times I_{area}\}}{\left\{\frac{P_i}{P_{age}}\right\}}$
Initial prey population age (P_a)	(See eqn. (4))
Initial prey interactions (P_i)	(See eqn. (5))
Predator Parameters	
Initial predator population (N_{pred})	4
Initial predator location	$(\frac{W}{4}, \frac{H}{4})$ $(\frac{W}{4}, \frac{3H}{4})$ $(\frac{3W}{4}, \frac{H}{4})$ $(\frac{3W}{4}, \frac{3H}{4})$
Predation radius (cells) R_{pred}	30
Predation success rate (%) (P_p)	variable, default 10
Predation tries per time step (P_{nt})	variable, default 1

Reproduce: During *Reproduce*, each mature prey reproduces with a probability of P_b. If successful, the number of offsprings is given by (Equation (2)). Each offspring is randomly placed on one of the locations in the parent's 8-neighborhood.

Death: There is a constant probability, P_d, of death in each generation (iteration). Each prey draws a random number between 0 and 100 with uniform probability. If the number is less than P_d, the prey dies and is immediately removed from the simulation state.

Predator Processes:

Predation: During predation, each predator picks P_{nt} number of prey, within its predation radius. The selected prey are killed with a probability of P_p.

6.3. RESULTS AND DISCUSSION

Figure 6.2 shows the effect of predation on average population. For each predation success rate, beyond a particular number of attempts the prey density drops rapidly. Beyond this particular threshold, the prey are killed faster than the offsprings can mature and reproduce.

This analysis is particularly useful for solving design/optimization problems using an artificial predator-prey ecosystem. Values for the predation attempts and predation success rate can be determined from Figure 6.2 depending on the application's need for diversity preservation or faster convergence. Values of predation attempts on the left side of the threshold are good for diversity preservation[25], while the values on the right side are good for faster convergence. Higher values of predation success rates promote faster convergence, where as lower values provide the prey an opportunity to explore the search space, resulting in a diverse set of optimal values.

[25]As required in the case of multi-objective optimization, where more than one optimal solution form the pareto-optimal set.

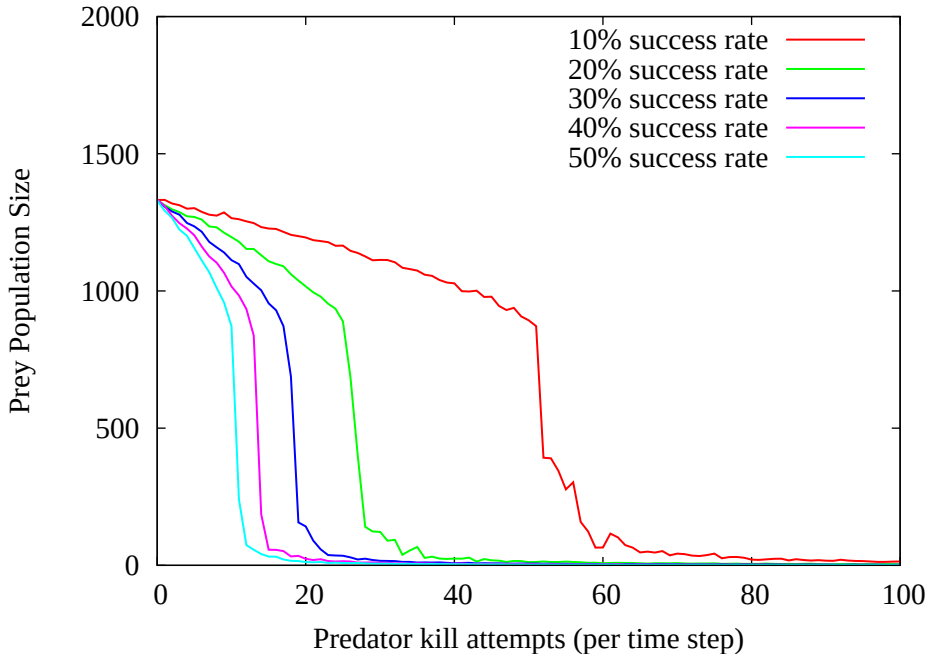


Figure 6.2. Effect of predation on prey population.

7. BINARY TEXTURE SYNTHESIS

As a proof of concept of Artificial Ecosystems methodology discussed in Section 3, the problem of binary texture synthesis is attempted. The problem consists of estimating optimal markov random field (MRF) parameter set capable of regenerating a given random binary texture. An adapted version of the PPE model presented in Section 6 is used for this purpose.

7.1. MRF TEXTURE MODELS

In this section the necessary mathematical representation for binary texture synthesis is introduced.

Consider the image field S to be a $N \times N$ grid. Let $X(i,j)$ be the intensity level at point (i,j) on S. To simplify notation, $X(i,j)$ is written as $X(i)$, for $i = 1, 2, ..., M$ where $M = N^2$. Let Λ be the colorspace from which the intensity of each location on S is drawn. For a binary texture Λ has only two elements, i.e., $\Lambda = \{-1, 1\}$. Let $n(i)$ be the first-order neighborhood [83] of pixel i. S is considered to be a toroidal grid, so that each pixel has exactly four first-order neighbors. Also, this assumption eliminates any *edge effects* on the quality of the results. Unless an image/texture is random noise, the pixel intensity at any location depends on the intensities at other locations. Hence, a conditional joint probability distribution function can be defined for each pixel i as

$$P(X(i)|X(1), X(2), ..., X(i-1), X(i+2), ..., X(M)) \tag{8}$$

Consider, the case where the intensity of a pixel depends only on the intensities of its neighboring pixels. i.e.,

$$P(X(i)|all\ points\ in\ S\ except\ i) = P(X(i)|neighbors\ of\ i) \tag{9}$$

Any joint probability density which satisfies Equation (9)[26] is referred to as a Markov Random Field. Markov Random Fields have been previously used for texture

[26]This property is called Markovianity

synthesis [84, 83, 85]. An anisotropic Ising MRF is used here, which is characterized by the energy function

$$U(x) = -\beta_1 \sum_{\substack{i \\ j}} x_i x_j - \beta_2 \sum_{\substack{i \\ k}} x_i x_k \tag{10}$$

where β_1 and β_2 are the parameters that result in different textures.

Once the parameters are estimated, S is visited site-wise. The intensity of the each site is set to -1 or 1 with probabilities given in Equations (11).

$$P(-1) \propto exp\left\{ -\beta_1 \sum_{\substack{i \\ j}} x_j - \beta_2 \sum_{\substack{i \\ k}} x_k \right\} \tag{11a}$$

$$P(1) \propto exp\left\{ \beta_1 \sum_{\substack{i \\ j}} x_j + \beta_2 \sum_{\substack{i \\ k}} x_k \right\} \tag{11b}$$

where i is the current site, $\sum_{\substack{i \\ j}} x_j$ denotes sum of intensities across all horizontal neighbors of i, and $\sum_{ik} x_k$ denotes sum of intensities across all vertical neighbors of i. The resulting texture is an Ising MRF with parameters β_1 and β_2.

The proposed artificial ecosystems methodology is used to solve the inverse problem. Given an input binary texture, the parameters β_1 and β_2 in Equation (10) are estimated so that the binary texture can be synthesized. Visual inspection is used to verify the output texture.

7.2. PPE MODEL FOR BINARY TEXTURE SYNTHESIS

The ecosystem model developed consists of three components predators, prey and the environment. The texture whose parameters are to be estimated is mapped to the environment as the land cover. The predator species is equipped with the ability to differentiate prey from the background (visual acuity) and kill them. The MRF parameters β_1 and β_2 are mapped as evolvable characteristics of the prey. Based on these parameters, each prey is born with a textured coat, which camouflauges it against the land cover (environment). A prey whose coat parameters are close to those of the environment (original texture), get better camouflage, i.e., better protection

from the predators. Such a prey is said to have adapted to the environment. Due to the evolutionary force from predation, overtime only the prey with coat textures closely resembling the landcover would survive, and hence their parameters can be used to synthesis textures similar to the input texture.

The initial prey population parameters are initialized to random values. The prey pass these parameters (comparable to "genes") to offsprings with a small random mutation at a fixed mutation rate. The predator's seek and kill mechanism (predation) can therefore be thought of as a fitness function, albeit, a local one.

Since the predator does not ave any parameters that need to be evolved, reproduction and death processes for predators do not contribute to the improvement of the system in any way. Prey are given random movement so that probability of th prey staying in a given neighborhood is equal to the probability of leaving the neighborhood. The random movement ensures that only the prey which capture the global characteristics of the input texture, survive. Also due to this random movement of the prey, predator mobility is not necessary for ensuring complete monitoring of the prey population. Therefore, the predators are modeled to be immobile, immortal and impotent, and are placed at strategic locations.

7.3. MODEL FORMULATION

Purpose – The purpose of the model is to investigate the viability of the proposed artificial ecosystems to solve design and optimization problems. A modification of the PPE model developed in Section 6 is used to solve the problem of binary texture synthesis.

State variables and scales – The model consists of two species, predators and prey, and their environment. For each prey, age (in time steps), location in the environment and interactions with other prey are tracked. Each prey is born with certain texture endowing parameters β_1 and β_2 (genes) which determine the texture on its coat. Predators have no state variables. The environment is a 512×512 image of the texture whose parameters are to be estimated. The binary texture is made of 3×3 color cells with each cell representing a potential location for the predators or prey. Each cell is capable of housing more than one

individual. The model operates in discrete time steps. At each step, prey are selected in random order, and their individual processes executed, after which predators are selected in a random order and their individual processes are executed. The simulation state is updated after each individual has completed their process, so that the next individual sees the updated simulation state. Figure 7.1 shows the model setup and examples of prey coats.

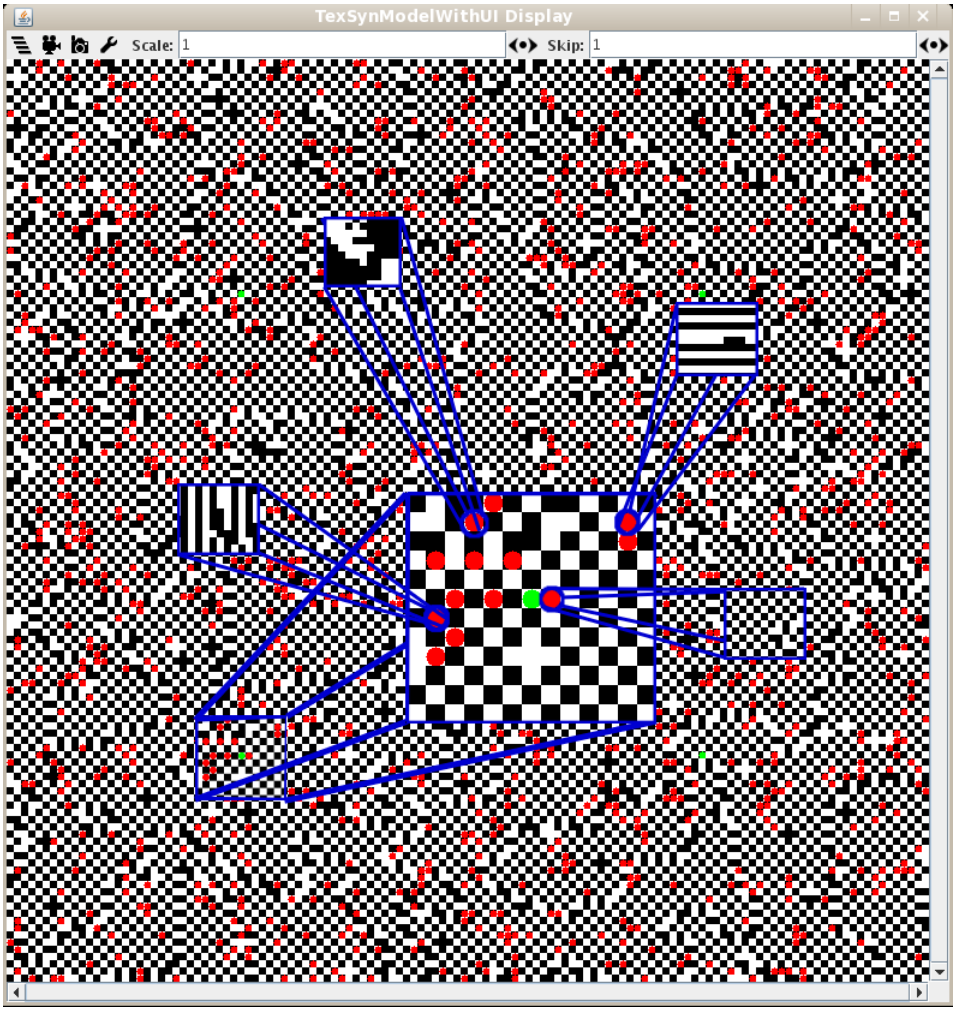


Figure 7.1. Texture synthesis PPE model setup. Red dots indicate prey and green dots indicate predators. Each prey is endowed by genes which project a textured coat. Four such possible texture coat projections are depicted in the image.

Process overview and scheduling – The model proceeds in discrete time steps. Within each time step, each prey is randomly selected and its processes – *interact*, *move*, *reproduce*, and *death* are executed in the same order. Age of all the prey alive at the beginning of a time step is incremented by one. *Interact*, *move* and *death* behaviors of the prey are exactly the same as the ones described in the SSPD and PPE models. In the *reproduce* behavior, the offsprings are produced as in the case of SSPD and PPE models. However, the coat parameters of the offsprings are altered from the parent's coat parameter, by adding a small mutation value at a small mutation rate.

Predators have only one process, *predation*. Predators calculate certain statistics about the local environment and the coats of prey within a distance of *predation radius* from itself. The statistics calculated are given by equations (12), (13), (14) as

$$dc = \frac{\sum_{i=1}^{L_s} \sum_{j=1}^{L_s} x_{ij}}{L_s^2} \tag{12}$$

$$f_x = \frac{\sum_{i=1}^{L_s} \sum_{j=1}^{L_s-1} bool(x_{ij} \; != x_{i(j+1)})}{L_s^2} \tag{13}$$

$$f_y = \frac{\sum_{i=1}^{L_s-1} \sum_{j=1}^{L_s} bool(x_{ij} \; != x_{(i+1)j})}{L_s^2} \tag{14}$$

where L_s is the half of side of the texture image in question. L_s is equal to predation radius for local environment, and is equal to the prey interaction radius for a prey coat. x_{ij} is intensity value at location *(i, j)*, and bool is an boolean function which return 1 if the condition is satisfied, and 0 otherwise.

$$(|E_{dc} - P_{dc}| > T_{dc}) || (|E_{f_x} - P_{f_x}| > T_{fx}) || (|E_{f_y} - P_{f_y}| > T_{fy}) \tag{15}$$

where E_* and P_* represent environment and prey coat statistics, respectively. T_{dc}, T_{fx} and T_{fy} are the thresholds for dc, f_x and f_y, respectively.

Therefore, dc represents the difference in number of white pixels and black pixels, f_x represents the sum of number of changes from white to black and vice versa in each row, and f_y represents the sum of number of changes in each column.

The statistics can be considered as channels of a three channel color space. Any prey whose coat texture is different than the texture of the environment would have different statistics that the environment. If this difference is greater than a given threshold (selected empirically) the prey satisfies equation (15). Such a prey is said to be unadapted to the environment. The predator then randomly selects, one of the unadapted prey and kills it with a probability of *predation success rate*.

Design concepts –

Emergence: Although the prey life cycle (movement, reproduction, and mortality) and predator behaviors (predation) are described by empirical rules and probabilities, the population dynamics, and adaptation of the prey coat to the background texture, emerge from the behaviors and interactions of the individuals.

Sensing: Both the predator and prey can be said to have visual perception. Prey use this type of sensing for interactions with other prey. Predators use visual information for predation. Also each prey is assumed to know its own age and reproduction capabilities.

Interactions: Two types of interactions are explicitly modeled. Interactions between two prey, is used to keep track of number of other prey in vicinity. Predation is the second interaction modeled between a predator and a prey.

Stochasticity: Prey birth and death events, and predation success are modeled via probabilities, which add stochasticity to the model. To obtain more precise prediction values, each simulation is repeated 10 times, from which respective mean values are taken as representatives.

Observation: Prey population size, and number of adapted and unadapted prey (see equation (15)) are recorded at the end of each time step. The

adaptable parameters β_1 and β_2 of the entire prey population is recorded at the start of the simulation and at the end of the simulation.

Initialization – The environment is initialized to the given texture. N_{prey} prey are randomly placed in the environment, and their texture parameters randomly initialized. The age and interactions of the initial prey population are initialized using equations (3), (4), and (5). N_{pred} predators are placed in strategic locations of the environment to maximize predator-prey interactions. See Table 7.1 for an overview and values of the parameters used in the model.

7.4. RESULTS AND DISCUSSION

Figure 7.2 shows the results of texture synthesis. In all the cases, the statistics of the input texture were well captured by the model and there is little if any visual difference between the original and synthesized images. Figures 7.3 and 7.4 are results of experimental runs performed on texture shown in Figure 7.2(e). Figure 7.3 shows the prey population dynamics (average of ten runs) observed during the simulation runs and Figure 7.4 shows the values of prey parameters β_1 and β_2 at the start and end of a simulation run. As an external observer of the ecosystem, it is not possible to estimate the fitness of a prey's coat since the fitness function is internal to the ecosystem. It is however possible to track the number of prey alive. The fitness of the prey population was tracked through the predators. At each time step, the predators report the number of total prey, adapted prey, and unadapted prey in their predation radius. As can be seen from Figure 7.3, the difference between the total prey counted externally and prey count reported by the predators is small. This validates the initial assumptions that a small immobile predator population is adequate to monitor the prey population. From Figure 7.4 it can be seen that although the initial prey population parameters β_1 and β_2 were initialized with a wide range of random values, the final population parameter values clustered into a small parameter space, which can be considered as the solution space of the problem. Interactions between predators and prey are responsible fo adapting the prey parameter values towards the solution space. A major part of the prey population is categorized as unadapted by the predators (Figure 7.3). However, the clustering of parameter values (Figure 7.4)

Table 7.1. Overview and default values of the parameters for the PPE texture synthesis model

Parameter	Value
Environment Parameters	
Habitat Width (cells) (W)	128
Habitat Height (cells) (H)	128
Display width (cells)	512
Display height (cells)	512
Prey Parameters	
Initial number of prey (N_{prey})	1000
Initial prey location	randomly placed
Interaction radius (cells) (R_{prey})	5
Interaction area (cells). Number of cells in R_{prey}	81
Birth rate (%) (P_b)	20
Death rate (%) (P_d)	10
Genetic density (G_d)	$\frac{N_p}{(W \times H)}$
Mobility (cells) (P_m)	1
Mutation rate (%) ($P_{\mu r}$)	10
Mutation (P_μ)	± 0.1
Maturity age (time steps) (P_{ma}) (See eqn. (3))	3
Coat width (cells) (C_w)	$2 \times R_{prey}$
Coat height (cells) (C_w)	$2 \times R_{prey}$
Offsprings produced (N_{os})	$\frac{\{G_d \times I_{area}\}}{\left\{\frac{P_i}{P_{age}}\right\}}$
Initial prey population age (P_a)	(See eqn. (4))
Initial prey interactions (P_i)	(See eqn. (5))
Initial prey coat parameters	
β_1	$U(-3, 3)$
β_2	$U(-3, 3)$
Predator Parameters	
Initial predator population (N_{pred})	4
Initial predator location	$(\frac{W}{4}, \frac{H}{4})$, $(\frac{W}{4}, \frac{3H}{4})$ $(\frac{3W}{4}, \frac{H}{4})$, $(\frac{3W}{4}, \frac{3H}{4})$
Predation radius (cells) R_{pred}	30
Predation success rate (%) (P_p)	50
Texture difference thresholds (T_{dc}, T_{fx}, T_{fy})	
For Texture 7.2a	(0.03, 0.03, 0.03)
For Texture 7.2c	(0.05, 0.05, 0.1)
For Texture 7.2e	(0.05, 0.1, 0.05)

suggests that the parameter values of the unadapted prey must differ from those of the adapted prey by an insignificant amount. To test this hypothesis, we synthesized the output textures in Figure 7.2 using the average parameter values of the final prey population. The synthesized textures are indistinguishable from the original samples, proving the hypothesis. The reported unadapted prey count, could be due to the high mutation and low predation rates used in the simulation. Due to the high mutation rate, there exists a significant probability that an adapted prey with parameters values near the boundary of the solution space could give birth to an unadapted prey. So potentially a cycle could form in which the interactions between predators and prey result in adapted prey and mutation in adapted prey give rise to unadapted prey. The study of this possible emergent phenomenon is however not of interest to the current work and is left as an open problem for further investigation.

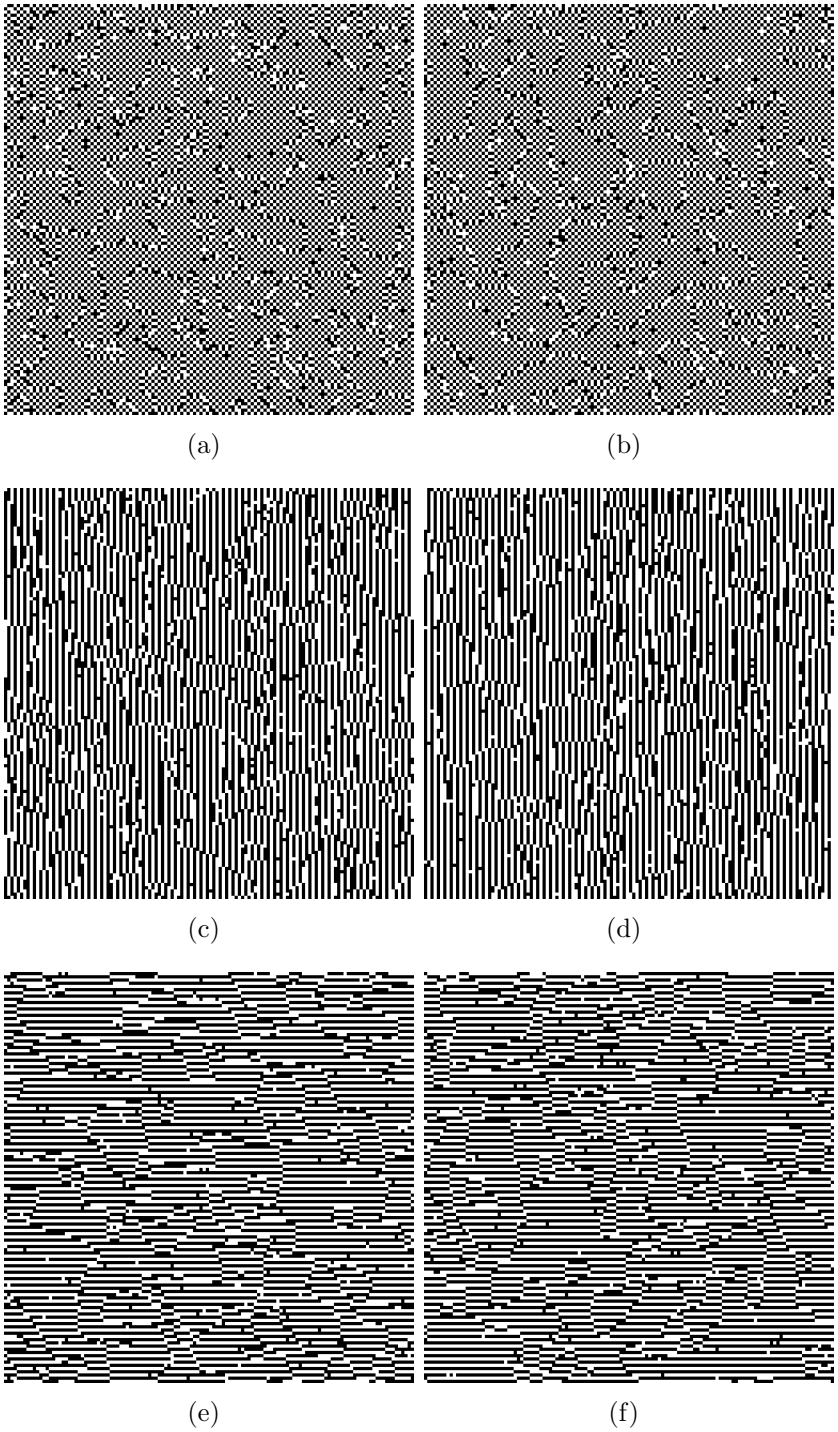


Figure 7.2. Texture synthesis results: *a*, *c*, and *e* are original textures (input); *b*, *d*, and *f* are synthesized textures (output) based on the parameters obtained from adapted population.

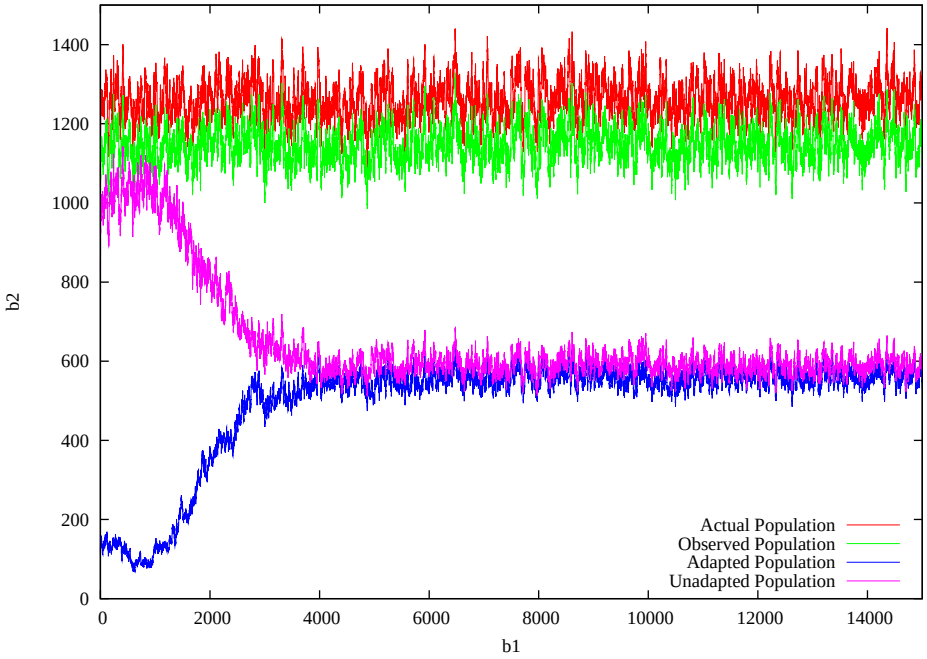
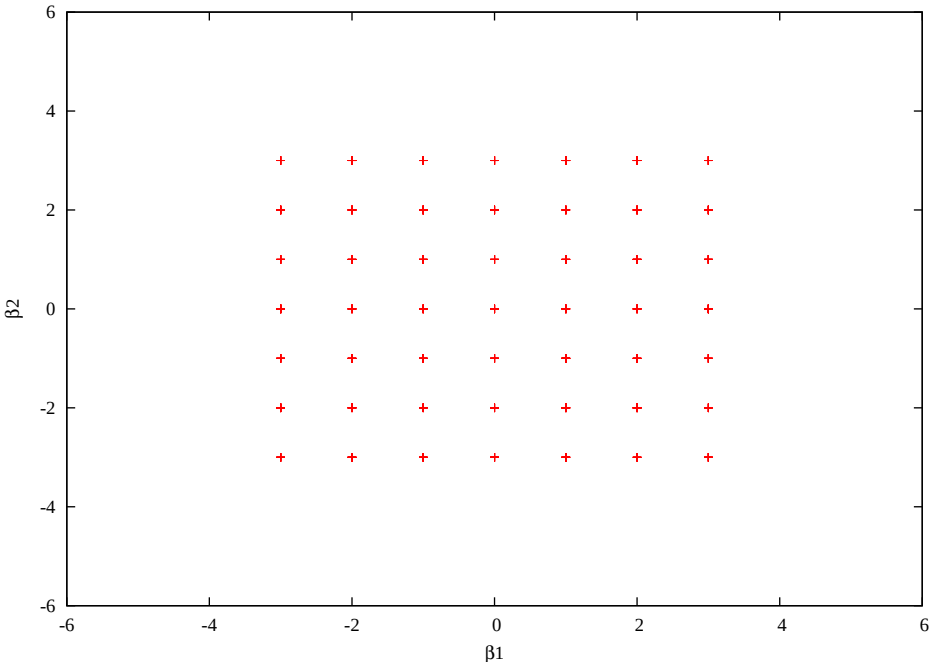
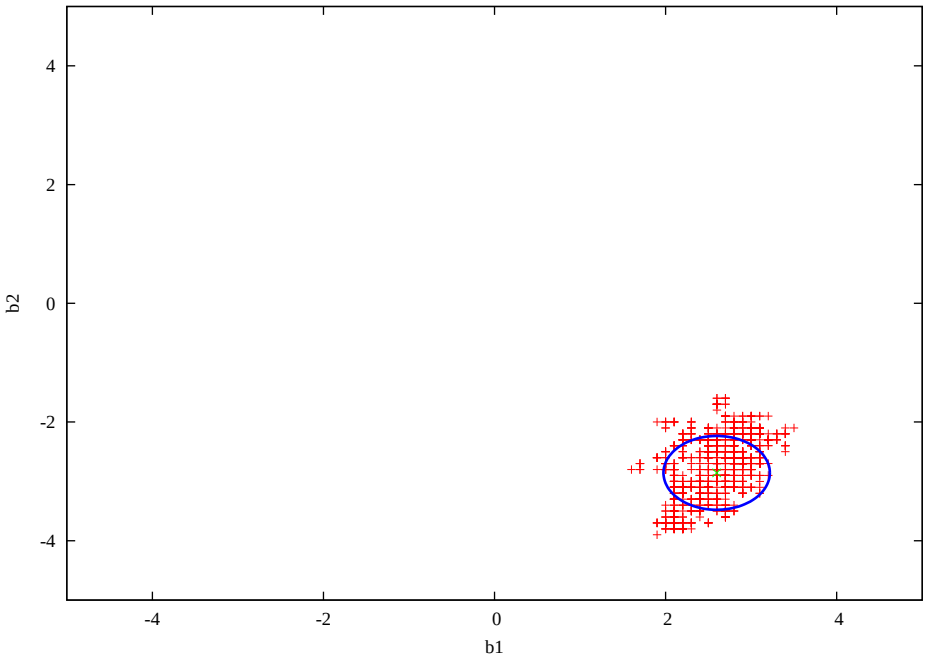


Figure 7.3. Prey population dynamics observed during one simulation run for texture in Figure 7.2(e)



(a)



(b)

Figure 7.4. Prey adaptation: scatter plots of prey parameters for texture in Figure 7.2(e); (a) – initial population and (b) – final population. The green dot indicates the mean of the current run and the radius of the blue circle is the standard deviation of means over 10 runs.

8. CONCLUSIONS AND FUTURE WORK

In this thesis, a novel individual based design and optimization framework, inspired by naturally occurring ecosystems is proposed. This work draws on the knowledge of individual-based modeling, ecological modeling, evolutionary computing, and multi-objective optimization. A comprehensive survey of the literature in these areas, along with an extensive evaluation of available individual-based modeling softwares is conducted. In the later sections, essential details of various population level biological processes were discussed and their individual-based counterparts were developed. Two individual-based template models were developed which could be built upon to solve design and optimization problems.

As a proof of concept, the problem of binary texture synthesis was attempted. The problem was *mapped* to an artificial predator-prey ecosystem. An IBM of this ecosystem was developed and experimental runs were performed. The results demonstrated the paradigm's ability to solve design and optimization problems by synthesizing visually indistinguishable copies of the input texture.

As a part of the future work, the proposed framework's ability to solve complex multi-objective optimization needs to be explored. There is also a need to identify and model ecological processes and individual behaviors, which can then be adapted to be used with the proposed framework.

BIBLIOGRAPHY

[1] Marashi E. and Davis J. P. A Systems Approach for Resolving Complex Issues in a Design Process. In *Workshop on the Complexity in Design and Engineering*, pages 160–170, March 2005.

[2] Maier M. W. Architecting Principles for Systems-of-Systems. *Systems Engineering*, 1(4):267–284, 1998.

[3] Bar-Yam Y. *Dynamics of Complex Systems*. Addison-Wesley, 1997.

[4] Sterman J. D. Systems Dynamics Modeling: Tools for Learning in a Complex World. *California Management Review*, 43(4):8–25, 2001.

[5] Axelrod R. Advancing the Art of Simulation in the Social Sciences. *Japanese Journal for Management Information System*, 12(3):16–22, 2003.

[6] Axtell R. L. Why Agents? On the Varied Motivations for Agent Computing in the Social Sciences. Technical Report Working Paper No. 17, Center on Social and Economic Dynamics, November 2000.

[7] Bankes S. C. Agent-Based Modeling: A Revolution? In *Proceedings of the National Academy of Sciences of the United States of America, Vol. 99, No. 10, Supplement3: Arthur M. Sackler Colloquium of the National Academy of Sciences. Sackler Colloquium on Adaptive Agents, Intelligence, and Emergent Human Organization: Capturing Complexity though Agent-Based Modeling*, pages 7197–7198, 2002.

[8] Grimm V. Ten Years of Individual-Based Modelling in Ecology: What Have We Learned and What Could We Learn in the Future? *Ecological Modelling*, 115:129–148, 1999.

[9] Grimm V., Berger U., Bastiansen F., Eliassen S., Ginot G., Giske G., Goss-Custard J., Grand T., Heinz S. K., Huse G., Huth A., Jepsen J. U., Jorgensen C., Mooij W. M., Muller B., Pe'er G., Piou C., Railsback S. F., Robbins A. M., Robbins M. M., Rossmanith E., Ruger N., Strand E., Souissi S., Stillman R., Vabo R., Visser U., and DeAngelis D. L. A Standard Protocol for Describing Individual-Based and Agent-Based Models. *Ecological Modelling*, 198:115–126, 2006.

[10] Grimm V. and Railsback S. F. *Individual-based Modeling and Ecology*. Princeton Series in Theoretical and Computational Biology. Princeton University Press, 2005.

[11] Carley K. M. Validating Computational Models. Technical report, CASOS, Carnegie Mellon University, 1996.

[12] Von Neumann J. *Theory of Self-Reproducing Automata*. University of Illinois Press, Urbana, Illinois, USA, 1966.

[13] Wolfram S. *A New Kind of Science*. Wolfram Media, Inc., 2002.

[14] Schelling T. C. *Micromotives and Macrobehavior*. W. W. Norton, New York, New York, USA, 1978.

[15] Granovetter M. Threshold Models of Collective Behavior. *The American Journal of Sociology*, 83(6):1420–1443, 1978.

[16] Axelrod R. *Evolution of cooperation*. Basic Books, New York, New York, USA, 1984.

[17] Macal C. M. and North M. J. Tutorial on Agent-Based Modeling and Simulation. In *Proceedings of the 2005 Winter Simulation Conference*, pages 2–15, 2005.

[18] Macal C. M. and North M. J. Tutorial on Agent-Based Modeling and Simulation Part2: How to Model with Agents. In *Proceedings of the 2006 Winter Simulation Conference*, pages 73–83, 2006.

[19] Epstein J. M. and Axtell R. L. *Growing Artificial Societies: Social Science from the Bottom Up*. The MIT Press, November 1996.

[20] Little L. R. and McDonald A. D. Simulations of Agents in Social Networks Harvesting a Resource. *Ecological Modelling*, 204:379–386, 2007.

[21] Kohler T. A., Gumerman G. J., and Reynolds R. G. Simulating Ancient Societies: Computer Modeling is Helping to Unravel the Archaelogical Mysteries of the American Southwest. *Scientific American*, pages 76–83, July 2005.

[22] Christuansen J. H. and Altaweel M. Simulation of Natural and Social Process Interactions: An Example from Bronze Age Mesopotamia. *Social Science Computer Review*, 24(2):209–226, 2006.

[23] Eubank S., Guclu H., Kumar V. S. A., Marathe M. V., Srinivasan A., Toroczkai Z., and Wang N. Modelling Disease Outbreaks in Realistic Urban Social Networks. *Nature*, 429:180–184, 2004.

[24] Helbing D., Farkas I., and Vicsek T. Simulating Dynamical Features of Escape Panic. *Nature*, 407:487–490, 2000.

[25] Bonabeau E. Agent-Based Modeling: Methods and Techniques for Simulating Human Systems. In *Proceedings of National Academy of Sciences, USA, Vol. 99, Suppliment 3*, pages 7280–7287, 2002.

[26] Eubank S. G. What Makes a Simulation Useful? [TRANSIMS]. In *Proceedings of 1999 IEEE International Conference on Systems, Man, and Cybernetics*, volume 4, pages 640–644, 1999.

[27] Reuter H., Holker F., Middelhoff U., Jopp F., Eschenbach C., and Breckling B. The Concepts of Emergent and Collective Properties in Individual-Based Models - Summary and Outlook of the Bornhoved Case Studies. *Ecological Modelling*, 186(4):489–501, 2005.

[28] DeAngelis D. L. and Mooij W. M. Individual-Based Modeling of Ecological and Evolutionary Processes. *Annual Review of Ecology, Evolution, and Systematics*, 36:147–168, 2005.

[29] Grimm V., Revilla E., and Berger U. Pattern-Oriented Modeling of Agent-Based Complex Systems: Lessons from Ecology. *Science*, 310:987–991, 2005.

[30] Pitt W. C., Box P. W., and Knowlton F. F. An Individual-Based Model of Canid Populations: Modelling Territoriality and Social Structure. *Ecological Modelling*, 166:109–121, 2003.

[31] Sumpter D. J. T. and Broomhead D. S. Relating Individual Behavior to Population Dynamics. In *Proceedings: Biological Sciences*, volume 268, pages 925–932, 2001.

[32] Holyoak M., Lawler S. P., and Crowley P. H. Predicting Extinction: Progess with an Individual-Based Model of Protozoan Predators and Prey. *Ecology*, 81(12):3312–3329, 2000.

[33] Huse G. and Giske J. Ecology in Mare Pentium: An Individual-Based Spatio-Temporal Model for Fish with Adapted Behavior. *Fisheries Research*, 37:163–178, 1998.

[34] Huse G., Railsback S., and Ferno A. Modelling Changes in Migration Pattern of Herring: Collective Behavior and Numerical Domination. *Journal of Fish Biology*, 60:571–582, 2002.

[35] Gomez-Mourelo P. From Individual-Based Models to Partial Differential Equations: An Application to the Upstream Movement of Elvers. *Ecological Modelling*, 188(1):93–111, 2005.

[36] Harvey B. C. and Railsback S. F. Elevated Turbidity Reduces Abundance and Biomass of Stream Trout in an Individual-Based Model. Draft manuscript, U. S. Department of Agriculture, Redwood Sciences Laboratory, Arcata, CA, 2004.

[37] Breckling B., Middelhoff U., and Reuter H. Individual-Based Models as Tools for Ecological Theory and Application: Understanding the Emergence of Organisational Properties in Ecological Systems. *Ecological Modelling*, 194:102–113, 2006.

[38] Hewitt C. and Inman J. DAI Betwixt and Between: From "Intelligent Agents" to Open Systems Science. *IEEE Transactions on Systems, Man, and Cybernetics*, 21(6):1409–1419, 1991.

[39] Sycara K. P. Multiagent Systems. *AI Magazine*, 19(2):79–92, 1998.

[40] Wooldridge M. Agent-Based Computing. *Interoperable Communication Networks*, 1(1):71–97, 1998.

[41] Weiss G. *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. The MIT Press, 1999.

[42] Reynolds C. W. Flocks, Herds, and Schools: A Distributed Behavior Model. *Computer Graphics*, 21(4):25–34, 1987.

[43] D'Souza R. M., Lysenko M., and Rahmani K. Sugarscape on Steroids: Simulating over a million agents at interactive rates. *Proceedings of Agent2007 conference*, 2007.

[44] Newell A. and Simon H. A. *Human Problem Solving*. Prentice-Hall, 1972.

[45] Turchin P. *Complex Population Dynamics: A Theoretical/Empirical Synthesis*. Princeton University Press, Princeton, NJ, 2003.

[46] Malthus R. and Gilbert G. *An Essay on the Principle of Population*. Oxford University Press, 1999.

[47] Kot M. *Elements of Mathematical Ecology*. Cambridge University Press, 2001.

[48] Richards F. J. A Flexible Growth Function for Empirical Use. *Journal of Experimental Botany*, 10(29):290–300, 1959.

[49] Hastings A. *Population Biology: Concepts and Models*. Springer-Verlag, New York, New York, USA, 1997.

[50] Rockwood L. L. *Introduction to Population Ecology*. Blackwell Publishing, 2006.

[51] Lotka A. J. *Elements of Physical Biology*. Williams & Wilkins, Baltimore, Maryland, USA, 1925.

[52] Volterra V. Fluctuations in the Abundance of a Species Considered Mathematically. *Nature*, 118:558–560, 1926.

[53] Nicholson A. J. and Bailey V. A. The balance of animal populations. In *Proceedings of the Zoological Society of London*, volume 3, pages 551–598, 1935.

[54] Holling C. S. The Functional Response of Invertebrate Predators to Prey Density. *Memoirs of the Entomological Society of Canada*, 91:293–320, 1966.

[55] Berryman A. A. The Origins and Evolution of Predator-Prey Theory. *Ecology*, 73(5):1530–1535, 1992.

[56] Nishimura S. I. and Ikegami T. Emergence of Collective Strategies in a Prey-Predator Game Model. *Artificial Life*, 3:243–260, 1997.

[57] Nolfi S. and Floreano D. Co-evolving Predator and Prey Robots: Do 'Arms Races' Arise in Artificial Evolution? *Artificial Life*, 4(4):311–335, 1998.

[58] Floreano D. and Nolfi S. God Save the Red Queen! Competition in Co-Evolutionary Robotics. In *2nd Conference on Genetic Programming*, pages 398–406, 1997.

[59] Laumanns M., Rudolph G., and Schwefel H. P. A Spatial Predator-Prey Approach to Multi-Objective Optimization: A Preliminary Study. In *Proceedings of the Fifth Conference on Parallel Problem Solving From Nature*, volume 1498, pages 241–249, 1998.

[60] Grimme C. and Schmitt K. Inside a Predator-Prey Model for Multi-Objective Optimization: A Second Study. In *Proceedings of the Genetic and Evolutionary Computing Conference*, pages 707–714, 2006.

[61] Engelbrecht A. P. *Computational Intelligence: An Introduction*. John Wiley and Sons Inc., 2007.

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

[63] Goldberg D. E. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wiley, Massachusetts, 1989.

[64] Fukuda T. and Kubota N. Learning, Adaptation and Evolution of Intelligent Robotic System. In *Proceedings of IEEE International Symposium on Intelligent Control*, pages 2–7, 1998.

[65] Cliff D. and Miller G. F. Tracking the Red Queen: Measurements of Adaptive Progress in Co-evolutionary Simulations. In *Proceedings of the European Conference on Artificial Life*, pages 200–218, 1995.

[66] Ficici S. G. and Pollack J. B. Challenges in Coevolutionary Learning: Arms Race Dynamics, Open-Endedness, and Mediocre Stable States. In *Proceedings of the Sixth International Conference on Artificial Life*, pages 238–247, 1998.

[67] De Jong K. A. and Potter M. A. Evolving Complex Structures via Cooperative Coevolution. In *Proceedings of the Fourth Annual Conference on Evolutionary Programming*, pages 307–317, 1995.

[68] Kennedy J. and Eberhart R. C. Particle Swarm Optimization. In *Proceedings of the IEEE International Conference on Neural Networks*, pages 1942–1948, 1995.

[69] Kennedy J. and Eberhart R. C. *Swarm Intelligence*. Morgan Kaufmann, 2001.

[70] Dorigo M., Maniezzo V., and Colorni A. Ant System: Optimization by a Colony of Cooperating Agents. *IEEE Transactions on Systems, Man, and Cybernetics–PartB*, 26(1):29–41, 1996.

[71] Dorigo M. and Gambardella L. M. Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem. *IEEE Transactions on Evolutionary Computation*, 1(1):53–66, 1997.

[72] Schoonderwoerd R., Bruten J. L., Holland O. E., and Rothkrantz L. J. M. Ant-Based Load Balancing in Telecommunication Networks. *Adaptive Behavior*, 5(2):169–207, 1996.

[73] Cicirello V. A. and Smith S. F. Ant Colony Control for Autonomous Decentralized Shop Floor Routing. In *Proceedings of the Fifth International Symposium on Autonomous Decentralized Systems*, pages 383–390, 2001.

[74] Handl J. and Meyer B. Improved Ant-Based Clustering and Sorting in a Document Retrieval Interface. In *Proceedings of the Seventh International Conference on Parallel Problem Solving from Nature*, pages 913–923, 2002.

[75] Dorigo M. and Stutzle T. *Ant Colony Optimization*. MIT Press, 2004.

[76] Deb K. *Multi-Objective Optimization using Evolutionary Algorithms*. John Wiley & Sons, 2001.

[77] Zitzler E., Laumanns M., and Bleuler S. A Tutorial on Evolutionary Multiobjective Optimization. In Gandibleux X., Sevaux M., Sorensen K., and T'kindt V., editors, *Metaheuristics for Multiobjective Optimisation*, Lecture Notes in Economics and Mathematical Systems, pages 3–37. Springer-Verlag, 2004.

[78] Deb K. and Udaya B. R. N. Investigating Predator-Prey Algorithms for Multi-Objective Optimization. Technical report, Kanpur Genetic Algorithms Laboratory, Indian Institute of Technology Kanpur, 2005.

[79] Schwefel H. P. *Evolution and Optimum Seeking*. John Wiley & Sons, 1995.

[80] Panait L. and Luke S. Ant Foraging Revisited. In *Proceedings of the Ninth International Conference on the Simulation and Synthesis of Living Systems*, pages 569–574, 2004.

[81] Railsback S. F., Lytinen S. L., and Jackson S. K. Agent-Based Simulation Platforms: Review and Development Recommendations. *Simulation*, 82:609–623, 2006.

[82] Luke S., Cioffi-Revilla C., Panait L., Sullivan K., and Balan G. MASON: A Multiagent Simulation Environment. *Simulation*, 81:517–527, 2005.

[83] Cross G. R. *Markov Random Field Texture Models*. PhD thesis, Michigan State University, 1980.

[84] Perez P. Markov Random Fields and Images. *CWI Quarterly*, 11(4):413–437, 1998.

[85] Cross G. R. and Jain A. K. Markov Random Field Texture Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5(1):25–39, January 1983.

VITA

Srinivasa Shivakar Vulli (Shivakar), was born on 23rd November 1982 in Hyderabad, India. In 2004, he received his Bachelor's degree in Mechanical Engineering from Nagarjuna University, India. He joined the Master of Science program in Mechanical Engineering at Missouri University of Science and Technology (formerly University of Missouri Rolla) in Spring 2006. During the course of his study, he has worked in the capacity of Graduate Research Assistant with Airborne Reconnaissance and Image Analysis (ARIA) laboratory. His research was funded by the Department of Electrical and Computer Engineering and the Intelligent Systems Center at Missouri University of Science and Technology. His areas of interest include individual/agent-based modeling, multi-agent systems, high performance computing, machine vision, machine learning, robotics and evolutionary algorithms.

Shivakar received his Master's degree in Mechanical Engineering from Missouri S&T in May, 2008.