PhD Dissertation

International Doctorate School in Information and Communication Technologies

DISI - University of Trento

FOOD WEB SIMULATION STUDIES ON AQUATIC ECOSYSTEMS

Nerta Gjata

Advisor:

Prof. Corrado Priami

University of Trento

Co-advisor:

Dr. Ferenc Jordan

The Microsoft Research - University of Trento

Centre for Computational Systems Biology

December 2013

Abstract

There is an increasing interest in dynamical food web modeling, and recent advances of computational biology provide new algorithms and tools for modern systems ecology. In this work stochastic individual-based models are used for simulating food web dynamics in two case studies: the Kelian river, in Borneo, Indonesia and a marine ecosystem in Gulf of Guinea. The two cases present effects from human perturbations. In both cases, we constructed food webs, based on real databases. We parameterized the stochastic dynamical model for the system models and performed sensitivity analysis (and community response indicators) in order to quantify the relative importance of system components. The main aims are to understand the importance of functional diversity of aquatic ecosystems and relations between the dynamics of species and the whole community in perturbed ecosystems due to human activities (pollution due to gold mining activity and human settlements in the case of the Kelian river ecosystem and the impact of Fish Aggregation Devices on skipjack tuna communities in the case of the Gulf of Guinea ecosystem). In Kelian river case, our results suggest that invertebrate shredders are indicators of human impact on the river. In downstream sites our results show that the river is more polluted and the relative importance of grazers and shredders decrease. The primary producers disappear downstream as consequence of mine activity and human waste.

In the marine system case study, we analyzed the effects of association between tunas and FADs, and compared this to the behavior of free tuna individuals. The results suggest that skipjack tuna is affected by the use of FADs as fishing strategy on them. Some individual species show more sensibility to variation of population size of tuna.

These two studies contribute to better understand how functional diversity is related to human impact. This kind of approach may help in shaping systems-based conservation and marine fisheries management strategies.

Keywords: food web, aquatic ecosystems, stochastic model, sensitivity analysis

List of Figures

1 Lotka-Volterra predator-prey model 20
2 Predator and prey duplication processes in BlenX model 22
3 Boxes as entities in BlenX. 26
4 Predator-prey interaction in BlenX. 27
5 Simulations obtained running the BlenX code for the Kelian river food web model. 29
6 Water hydrologic cycle 42
7 A simple example of food chain in aquatic ecosystem. 43
8 Indonesia map 46
9 Gold mine at Kelian river. 48
10 Kelian river gold mine once closed. 48
11 Human settlements next to the river. 49
12 Food web of site 1 52
13 Map of Kelian river 56
14 Predator biological processes that may happen in the model 64
15 Food webs of the 6 sites 67
16 Result from simulations of site 1 shows a quasi balanced system 68
17 I_H(M) index for all the functional groups in the 6 sites 70
18 Functional groups with the most interesting I_H(M) indexes 71
19 Graph shows I_H(V) indexes of functional groups in the six sites 73
20 Graph with the functional groups shows the most interesting I_H(V) indexes 74

21 Map of Gulf of Guinea 77
22 Image of skipjack tuna (*Katsuwonus pelamis*) . . 78
23 Skipjacks catches in 2000-2004 78
24 ICCAT report of caught skipjack tuna 79
25 Food Web of the marine ecosystem: skipjack tuna is free from FADs 82
26 Food web of the marine ecosystem: skipjack tuna is associated with FADS 83
27 Food web model image 87
28 Curve trends of marine ecosystem after simulations 94
29 Graph with community importance indexes of marine ecosystem after simulations 96

List of Tables

1 Description of the functional groups in Kelian river 51
2 Site 1: Partial feeding matrix with interaction rates 54
3 Number of trophic groups, reproduction, death and when rates in site 1 55
4 Extracts from BlenX input files 60
5 Results of dynamical measurement in Kelian river ecosystem: I_H(M) index 69
6 Results of dynamical variability index I_H(V) in Kelian river ecosystem 72
7 Major categories for the fish that constituted the dominant phylum 81
8 Population size, birth, death, feeding and fishing rates of case study II 86
9 I_H(M) and I_H(V) indexes for marine dynamic model 95
10 Population size in the six sites 155
11 Birth, death and when parameters of sites 1, 2 and 3 156
12 Birth, death and when parameters of sites 4, 5 and 6 157
13 Partial feeding matrix of site 2 158
14 Partial feeding matrix of site 3 158
15 Partial feeding matrix of site 4 159
16 Partial feeding matrix of site 5 159
17 Partial feeding matrix of site 6 160

Contents

1 Introduction **10**
1.1 The context 10
1.2 The problem 13
1.3 The contribution 14
1.4 Structure of thesis 15
1.5 Publication related to this thesis work 16

2 State of Art **17**
2.1 Introduction 17
2.2 Lotka-Volterra model 19
2.3 Predator-prey BlenX model 21

3 Methods **24**
3.1 Introduction to BlenX 24
3.1.1 Processes in BlenX 30
3.2 Metaheuristic procedure: Scatter search algorithm 30
3.3 Sensitivity analyses 38

4 Aquatic ecosystems **41**
4.1 Introduction 41

5 Case study I **44**
5.1 Human impact in Kelian river, Borneo, Indonesia 44
5.2 Food webs for the six sites of the river 49
5.3 Dataset 53
5.4 Sites of the Kelian river 55
5.5 Dynamic models 58
5.5.1 General description of predator behaviour in BlenX 61

5.5.2 General description of prey behaviour in BlenX 64
5.5.3 General description of food behavior in BlenX 65
5.6 Results 66
5.7 Conclusion 74

6 Case study II **76**
6.1 Fishing by FADs on tunas, Gulf of Guinea 76
6.2 Data set and food webs of marine ecosystem . . . 81
6.3 Stochastic food web model of marine ecosystem . 87
6.4 Results 93
6.5 Conclusions 96

7 Conclusions **99**

A Appendix **115**
A.1 BlenX models for Kelian river 115
A.1.1 Site 1: .prog file 115
A.1.2 Site 1: .types file 119
A.1.3 Site 1: .func file 121
A.2 BlenX model for site 2 122
A.2.1 .prog file 122
A.2.2 .types file 126
A.2.3 .func file 128
A.3 BlenX model for site 3 129
A.3.1 .prog file 129
A.3.2 .types file 133
A.3.3 .func file 135
A.4 BlenX model for site 4 136
A.4.1 .prog file 136

A.4.2 .types file 140
A.4.3 .func file 141
A.5 BlenX model for site 5 142
A.5.1 .prog file 142
A.5.2 .types file 146
A.5.3 .func file 148
A.6 BlenX model for site 6 149
A.6.1 .prog file 149
A.6.2 .types file 153
A.6.3 .func file 154
A.7 Kelian river dataset 155
A.7.1 Population size for the six sites in Kelian river 155
A.7.2 Reproduction and death rates of the six sites in Kelian river 156
A.7.3 The feeding partial matrices of all the sites in Kelian river 158
A.8 BlenX model for skipjack tuna 161

1 Introduction

1.1 The context

Ecology is a term, deriving from Greek (*oikos* means household, *logos* means study), introduced first by German ecologists E. Haeckel in 1869 interpreted as the study of the environment including all relations between organisms and the environment. Nowadays the modern definitions of ecology became more complex from that of Haeckel but the message that ecologists would give of it includes always a common meaning "the emphasis on interactions between living individuals and their abiotic environment, between conspecific individuals, between populations, between different groups of species or between ecosystems" [47]. So ecology is an interdisciplinary field including biology and Earth science.

During 1970 people started to become more sensible to the environmental problems (e.g. climate change, pollution, consumption of natural resources and biodiversity). An example of this increased awareness is the creation, on April 22 1970 of the first "Earth Day".

In the 80s and 90s environmental issues began to receive political attentions and a new discipline (called theoretical ecology) emerged [64]. Its aim is to apply formal methods, as mathematical modelling and computational analysis, to investigate ecological systems. Ecologists goals are focus on the study ecosystems dynamics life processes, interactions and adaptations, energy flow in through living communities, abundance and distribution of organisms and biodiversity.

The complexity of ecosystems directs the ecologist to use food webs as a tool to describe the complex network of interactions

between species [19]. Food webs represent feeding connection (who eats whom) in an ecological community. The species may have trophic interactions between them and with non-trophic and abiotic effects. Their densities vary according to these interactions. The simplest model a model describing the dynamics of ecosystem in which two species interact (predator and prey): this model was introduced by Lotka-Volterra [65] and it is the base for other more complex predator-prey models.
The multiplicity of components and interactions, and the large number of parameters involved in ecological systems pose huge computational problems [65, 16]. Novel algorithms and computational techniques are developed to deal with these difficulties. Recently, the complex network structure of food webs has been studied by means of graph theory, a mathematical tool in which species are represented by nodes and connected by edges representing the direction of energy transfer [13]. Other modelling tools and simulation techniques have been used to obtain more realistic representation of individual and environmental variations, interactions among species and transient dynamics [10].
The models used in ecology can be either deterministic or stochastic. The deterministic approach is the most utilized and diffused and models the average behaviour of a large population of individuals, while the stochastic approach which is less utilized as pose huge computational problems, takes into consideration the noise existing in natural system and allows to model in a more accurate way small populations. Nowadays stochastic approach is becoming more diffuse than in the past thanks to the increased processing power of modern computers [4]. In our thesis work we used a stochastic model to represent the perturbation in the aquatic systems. A stochastic model is useful to supply detailed

predictions on real systems, investigate local interactions, environmental noise and variability. In our work the choice of using the stochastic approach allow us to get information about the role of functional diversity of a species in communities of the river ecosystem and better described the variability of fishing by FADs on different age classes of tuna.
The need to study ecosystems derived from necessity to predict human effects (e.g. pollution, changes to the landscape or hydrological systems, and larger-scale impacts such as global climate change, introduced species, overfishing etc.). The aquatic ecosystems, our case studies (the Kelian river ecosystem in Borneo and the marine ecosystem in the Gulf of Guinea, Atlantic ocean), are considered complex and the linkages between them make difficult to predict the effect of disturbances.
Our main aim in this thesis work is to analyse the human impact on a tropical river and the use of FADs on skipjack tuna fishing building an individual-based model (IBM) [36]. IBM helps to investigate individual-level variability [13], stochasticity and local interactions [5, 50] of ecosystems. For these purposes we use a computational framework based on the process-algebra called BlenX and perform sensitivity analyses in order to quantify which components are the most and least sensitive to the human influence (e.g pollution of the river from gold mine and waste coming from human settlements) and to understand if the massive deployment of those devices FADs is detrimental for tuna's population.

1.2 The problem

Ecosystems have very different nature (e.g. aquatic vs terrestrial), and each of them has its own specific patterns and this makes their study very challenging.
In todays world, ecology is becoming more and more important to understand how to make responsible choices to protect the nature and to minimize the damages for the next generations. Human beings, more of the times, use and waste resources without thinking that tomorrow the same resource may not be available any more in order to be used from next generations (e.g. the water, as our primary need).
This is a problem faced on the case study I of this thesis (see section 5.1 for more details), the pollution problem caused by gold open mine and by human settlements next to Kelian river, in Indonesia. We build a model of the system containing six food webs, modelling the six different sites along the river. The reason of this subdivision is the geographical position of mine and the human settlements regarding to the river. The distribution of the trophic organisms along the river represent some differences from one site to the other because they adapt to new condition of the river or in other cases they may not be present in a specific site.
Another problem concerned to aquatic ecosystem that we modelled in this thesis work is the problem of overfishing. It has critical consequences in biomass level (decreasing of biomass level down to the point where it is no longer possible to catch the fish stocks), growth rate and causes the depletion of resources. The ecosystem condition cannot always face the overfishing and the consequences are dramatic changes in species composition (e.g. some fish species can collapse). In the case study II (see section

6.1 for more details) our aim is to investigate the impact of fish aggregating devices (FADs) on skipjack tuna communities using a network model to simulate the marine food web. We built a scatter search algorithm to fit and balance the parameters used in the model and preformed stochastic simulations and sensitivity analysis on this dynamical system and determined the dependencies between various trophic components.
We can conclude this section quoting an italian expression that represent the negligence that humans shows in dealing with aquatic ecosystems over the years: "out of sight, out of mind".

1.3 The contribution

In this thesis work we study two aquatic ecosystem the river and marine ecosystem. The first is the Kelian river, in Borneo: in this ecosystem the main problem is related to pollution from human activity (the gold mine) and the organic waste coming from human settlements established near to the river. The river was subdivided in six sites and is habited by different trophic groups. We build a model that shows the dynamic behaviour of functional groups of the river.

We then performed some sensitivity analysis studies on the model to predict the consequences of human impact on these trophic groups. We utilize BlenX, in order to run stochastic simulations, do sensitivity analysis and analyze the community effects [51, 46], exploring the functional diversity of species in this tropical ecosystem and the human disturbance on them. For the second case study we study the effects of FADs on skipjack tuna (*Katsuwonus pelamis*) community. We built a stochastic food web model that combines interactions of tuna with FADs and predator-prey behavior with different species based on infor-

mation found in [59]. Having some values which are hypothetical and other real we use a scatter search algorithm in order to find the parameters that best fit the model. Finally we performed sensitivity analyses on this dynamical system in order to determine the dependencies between various trophic components.
This kind of studies allow to understand in case of river ecosystem if the indices of dynamical community importance may help to provide a tool for setting conservation priorities, managing rare species and better understanding ecosystem fragility. In the case of marine ecosystem these studies are useful to understand if the massive deployment of those devices FADs are to be considered as "ecological trap" [54, 38] perturbing tuna's population size and it will help also to give useful guidelines for future sustainable management strategies of tuna fisheries.

1.4 Structure of thesis

- **Chapter 2** describes the state of art of the most recent stochastic models connected to the two cases studies of interest;
- **Chapter 3** describes the methods used to analyze our two case studies, which are the basic strategies to build a BlenX model, a scatter search algorithm to fit the best parameters for the model and the sensitivity analysis approaches to study in order to study the community response to the human influence in aquatic ecosystems;
- **Chapter 4** contains a description of aquatic ecosystems (the river and marine systems);
- **Chapter 5** describes Case study I, the Kelian river in In-

donesia. The chapter contain the data used to build the model and the results of our analysis;

- **Chapter 6** describes Case II, the fishing processes on skipjack tuna in Guinea Gulf. The chapter contains the stochastic model we build to describe the system and the results of our analysis;
- **Chapter 7** summarizes the conclusions of this thesis;

1.5 Publication related to this thesis work

Ferenc Jordn, Nerta Gjata, Shu Mei, Catherine M. Yule 2012. **Simulated food web dynamics along a gradient: Quantifying human influence**. PLoS ONE 7(7): e40280.

2 State of Art

2.1 Introduction

From different studies done by ecologists is commonly accepted that "ecological systems, like terrestrial, freshwater and marine, are complex assemblages of various species with interactions between them" [47]. The complexity of those kind of interactions calls for tools and methods to analyze the dynamics of communities. Network analyses are becoming more and more elaborated to understand not only a static view but the dynamic of the ecosystem. Recently there is a growing interest in dynamic simulations of food webs using dynamic models. The most common approach to model these kind of system is the deterministic one (using ordinary differential equations (ODE)). One of the most used software to perform dynamical simulations and quantify the importance of species and understand the dynamics of biological communities are Ecosim and Ecopath (EwE) [66]. EwE are used to understand the complex marine ecosystems using biomass information for the species and the underlying dynamic model is deterministic. One example of deterministic model is that done with EcoTroph and analyses fishing impacts on marine ecosystems [28]. EcoTroph derives from an existing Ecopath model [29] and models marine ecosystems functioning as flows of biomass from low to high trophic levels (TLs). The model is able to simulate a global change in the fishing pressure. In the model fishing mortality is changed for each fleet segment operating in the ecosystem or for each functional group of the food web having as consequence the increasing number of possible simulations and interactions between fisheries strategies (artisanal and industrial fisheries). EcoTroph through the use of common

stock assessment equations models top-down and bottom-up effects of fishing along the food chain. The spatial, temporal, or environmental variations are not examined.
The traditional models (e.g. deterministic model) usually describe species by means of average population features. When environmental stochastic events or human activities affect ecosystems is important to study the behaviour of each individual in the system [48]. When the population size decreases, the deterministic methodology may fail to capture some important features as genetics (e.g. inbreeding), demographic (e.g. population bottleneck), environmental variabilities [49].
A recent work [51] showed that modeling the ecosystem using a stochastic dynamical approach is more effective in capturing the variation in population size and in getting important insights about the system as understanding extinction and its community-wide effects through the usage of sensitivity analysis. In that work the authors describe the predator-prey interaction based on data from the Prince William Sound, Alaska ecosystem model obtained with Ecosim. The stochastic approach is considered to be more appropriate for this system, due to the fact that small population are involved. The authors use a process-algebra language (called BlenX), for which a simulator, implementing the Gillespie algorithm [15, 14], is available. The results obtained are compared to those of topological analyses and deterministic dynamic (Ecosim) studies done previously. They conclude that the functional group with the greater effect on others is nearshore demersals; the most sensitive species to others is halibut; juvenile herring is the group which shows both largest effect on others and sensitivity to others. Finally from the simulation appears that the most important trophic group

is placed in intermediate trophic level in the food web.

2.2 Lotka-Volterra model

The Lotka-Volterra model [1] is the starting point for most models to describe the interactions of continuously reproducing populations of predator and prey. The model is based on two differential equations, describing a simple two-species predator-prey system, evolving in time to describe the rates of changing in prey population size (a) and that of the predator (b).

Prey equation:

$$\left(\frac{da}{dt}\right) = \alpha a - \beta ab$$

Predator equation:

$$\left(\frac{db}{dt}\right) = \delta ab - \gamma b$$

where the parameters in the equations above have the following interpretation:

- a = density of prey
- b = density of predators
- α = intrinsic rate of prey population increase
- β = predation rate coefficient
- δ = reproduction rate of predators per 1 prey eaten
- γ = predator mortality rate

In the predator-prey system, the predator increases in quantity when there is a great amount of prey and once it crosses the food supply it starts decreasing. The dynamic behaviour of the predator-prey model over time is shown in figure 1.

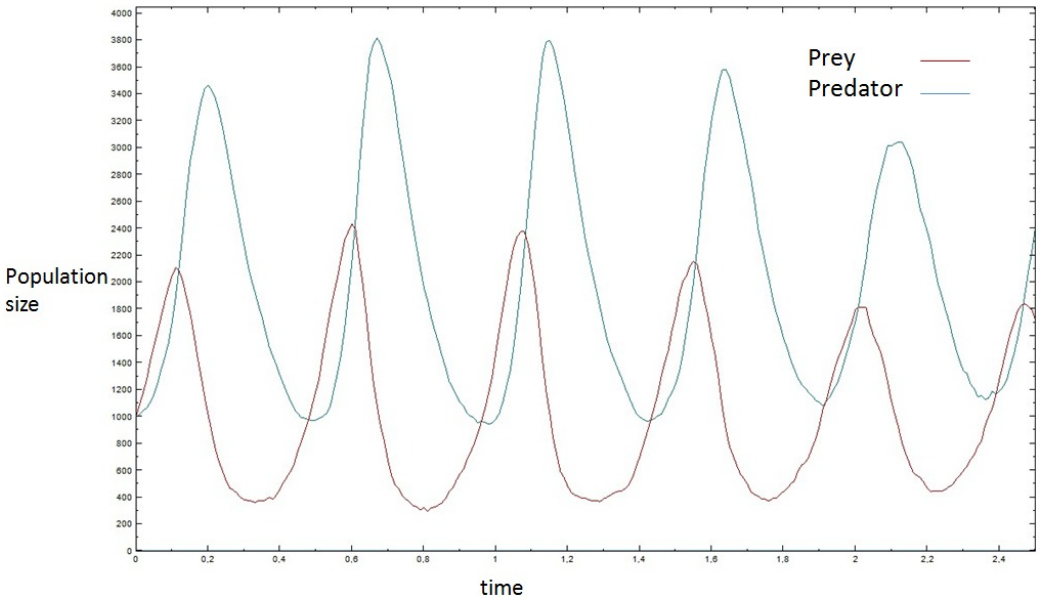


Figure 1: The image is a graphical representation of a simulation run of the Lotka-Volterra model. The plot shows the oscillatory behaviour of the predator (blue line) and the prey (red line) over time. The parameter values used to generate the time series are a=b=1000, α=20, β=δ=0.01 and γ=10.

As the predator population decreases the prey population will have an exponential growth (αa). The food supply for the prey is assumed to be unlimited. These dynamics of increase and decrease of the population size continue in a cycle. Lotka-Volterra model is considered by the ecologists as the basic representation of any interaction between predator-prey like entities in ecology. Over the years this model has been expanded to include more than one predator or prey in the model [41] and it has been implemented in different simulation framework. Another

recent work in stochastic population dynamics is done in [17]. In this work Webworld model introduced first in [7] is used to run simulations in order to study population dynamics making use of a set of equations to describe the dynamic influenced by competition for resources between members of the same species, and between members of different species. It demonstrate the formation of complex food webs which are stable to evolutionary perturbation. Another stochastic, individual based model for food web simulation is that developed in [69] based on the Webworld model [7]. The goal of this last study is to model realistic food webs in order to explore the consequences of a range of behaviour at the individual-level and to model important ecological processes (e.g. predator-evasion, mating strategy etc.).

2.3 Predator-prey BlenX model

The first model done in BlenX (called Lotka-Volterra model) includes a predator, prey and a third species called food (primary producer). Below follows an examples of the three boxes declared in BlenX model.

```
Predator = (t,r).nil + eat!().x!().nil
Prey = food!().x!().nil + eat?().nil
Food = feed?().x!().nil
```

A detailed description of all the features of the BlenX language is described in section 3.1. Here we just want to give an intuitive explanation of the encoding of the simple Lotka-Volterra model in BlenX. The predator and prey reproduce themselves after feeding process (fig. 2).

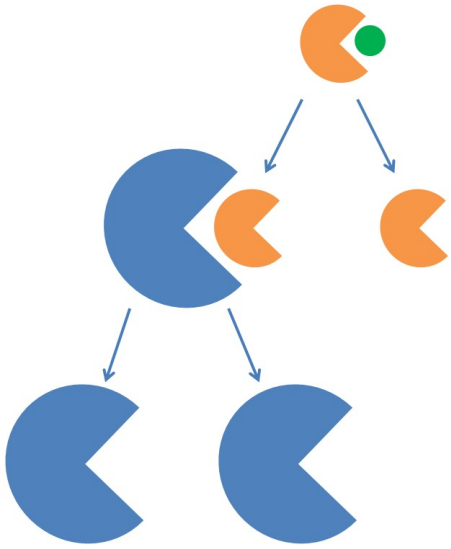


Figure 2: A visual representation of the duplications processes of the prey after eating the food (in orange) and the predator after feeding on prey (in blue).

Applying this same simple strategy on different predators and preys, the authors in [51] built the model of Prince William Sound, Alaska ecosystem with some improvements in order to give a more realistic description of what happens in nature. The model aims to be larger (including, more than one predators, preys and foods) in order to represents a real marine ecosystem. In the first model the reproduction happens after feeding and assumes that each prey is transformed in predator. In the Prince William Sound, Alaska ecosystem model the difference between feeding and reproduction is considered (after feeding action the species in the model has two possibilities: do nothing and keep living its normal life, or start the duplication process with a specific reproduction rate). Another assumption of the model is that food is available to the primary producer (defined as prey in Lotka-Volterra model) at all time. The death process is used

to stabilize the amount of species in the system. Finally, the death rate is not considered only for predator as in the first simple model but it involves all the animals (predator, prey or primary producers *food*). For these reasons this model has been considered in the more realistic model developed in [51].
This model done in BlenX is our starting point to construct the stochastic food webs models of Kelian river in Indonesia and that of skipjack tuna in Gulf of Guinea. In skipjack tuna model we introduced a new external non-living components which are drifting Fish aggregating device (FADs) in order to study their perturbation effect in all the food web. Another difference with the other original model is the fact that we introduced the possibility of tuna to change state from a trapped to a free one. In our model we want to represent in a more realistic way the fishing effects on the marine ecosystem.

3 Methods

3.1 Introduction to BlenX

One of the definitions of model is "A schematic description of a system, theory, or phenomenon that accounts for its known or inferred properties and may be used for further study of its characteristics".
In ecology the model is used to study food webs and its stability. The need to use models in ecology derive from the need to study the complexity of ecosystems related to multiplicity of different levels in food web, interactions and the large number of parameters which may present a problem from computational point of view [17, 65]. For ecological systems new tools are required to represent functional issues [10], to study the structure of food webs, to predict their dynamic behaviour.
Environmental variation caused by climate change, overexploitation of natural resources (e.g. FADs fishing on skipjack tuna) and the destruction and fragmentation of natural habitats (e.g. mining activity in Kelian river) are cause of stochasticity in ecological systems and affect the response of communities to species loss. There is the need to predict and model ecodynamics [3] how ecological communities will respond to these perturbation (e.g. species extinction in terrestrial and aquatic environments in the near future).
The use of individual-based model (IBM) [36] is necessary for the description of individual-level variability [13], stochasticity and local interactions. IBMs are a computational approach for simulating the effects on a system of the individual entity or a group behaviours. IBMs provide a bottom-up (simple behavioural rules generate complex behaviour) study and have the

ability to supply detailed predictions on real systems.
In our work the main aim is to build an IBM based on process algebra computational technique to predict the effect of stochasticity of aquatic ecosystems.
We did this using BlenX, a process algebra-based language supplied of a stochastic simulator (implementing the Gillespie algorithm [30]). It is specifically developed to analyze the interactions between biological entities. The language is inspired by the process calculus Beta-binders [71], which is an extension of pi-calculus [67].
BlenX has been used to model biological systems in which protein interacts in a cell [15, 14] and in those models the rate of interaction of two molecules depends on their concentration and their relative reaction rate. We adapted those concepts to model ecology: the probability of two individuals (predator and prey) to interact depends on predator preference (interaction rate) and on prey/predator density.
The basic metaphor of BlenX is that a biological entity can be represented by a computational device called box. This means that in BlenX each entity is represented by a box with "binder sites" that allow each box to interact with other boxes of the system to perform some biological functions. The different behaviour of each entity is controlled by an internal program that codifies for a set of actions: manage the interactions with the other boxes, modify the process interface and use its binders to handle the internal structure of external processes (fig. 3).

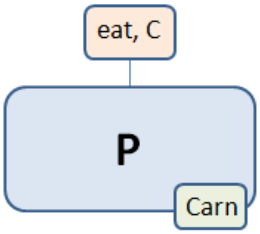


Figure 3: An example of an ecological entity represented by a box. (*eat, C*) represents the interface, *eat* is the interface subject and *C* interface type. *P* is the internal process, *Carn* is the name of the box.

An example of a how to declare a box in BlenX follows below:

```
let Carn : bproc = #(eat,C) [nil]
```

In that code we show a declaration of a very simple box (called *CARN*), with a single binder site (called *eat*) and with an empty internal program.
Each box sends communications through its binders to communicate with other boxes in the system.
In our food webs, boxes represent predators and preys, and the affinity between the different binders represent the strength of the interaction between them. In this way predators transmit signals to different preys depending on their different preference. These affinities can be just real numbers if the reaction that they are accounting for is a basic mass action law, or they can be arbitrary functions if the reaction represents a more complex interaction mechanism. The program inside the box can be used to model reproduction, death and changes in eating habits for each single species in the system.
In figure 4 are shown the boxes of two of the species in the Kelian model (CARN as predator and HERB as prey).

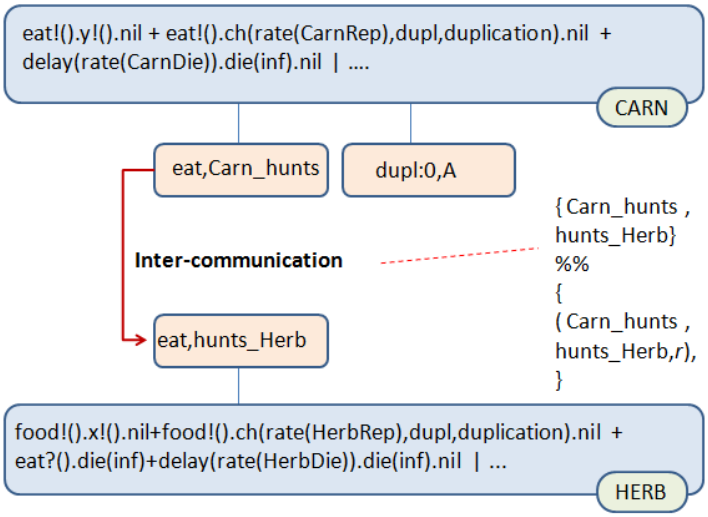


Figure 4: The two boxes of CARN and HERB interact together through the interaction sites with (*eat, carn_hunts*) and (*eat, hunts_herb*) with the affinity rate r. The interaction happens by synchronization of the boxes on the corresponding interfaces through the corresponding *eat!()* and *eat?()* actions. If the first action *(eat()!.y!().nil)* happens interacts with *eat?()* and and then sends an output to the y channel, which is an internal channel used by the CARN box to restart its own internal process (not shown in the figure). In the other box what remains is the *die(inf)* which is an action that represent the death of the box: so with this complex interaction we model the fact that CARN eats HERB. If, instead, the other option of the CARN internal process happens (e.g. *eat!().ch(rate(carnRep),dupl,duplication).nil*) CARN is duplicated.

The box of CARN (carnivore fish) shows two interaction sites (*eat, carn_hunts*) and (*dupl, A*) through which the internal processes communicate with the other box of HERB (herbivores). Through the usage of internal program, we model the fact that a fish eats an herbivore and in response of that different things can happen (at different rates): for example CARN can just eat and re-initiate its own normal behaviour, or it can eat and after that it can reproduce, or it can simply die. All these different

alternative behaviour can be seen in the code as a summation (the process-algebra way of defining alternative paths) of the different processes.
In the first box the internal process is composed by sum and a parallel process that is not shown here. The interfaces (*eat, carn_hunts*) and (*eat, hunts_herb*), are used by functional groups to interact together according to the affinity rate.
The synchronization through the eat channel, will perform a change of one of the box's interfaces: this change will be captured by a global event (here not shown) that will take this single CARN box and generate two CARN boxes: this part is modelling the reproduction of the CARN species. The intercommunication between boxes happens thanks to affinity rate r.
Following the logic explained for the interaction between CARN and HERB, we can encode the interactions of all the other species of the system. All this code it is divided into three text files used as input of the Beta Workbench stochastic simulator [15, 14]. Once simulation are performed, we used another tool, called Plotter, to visualize the results of the simulation. In the software Plotter (which is part of the Beta Workbench) we can observe the results of a stochastic simulation of the model. For example we can have as result a graph with the time represented on the X axis and the population size on the Y axis. It shows the stochastic fluctuation of the organisms in the model (fig. 5).

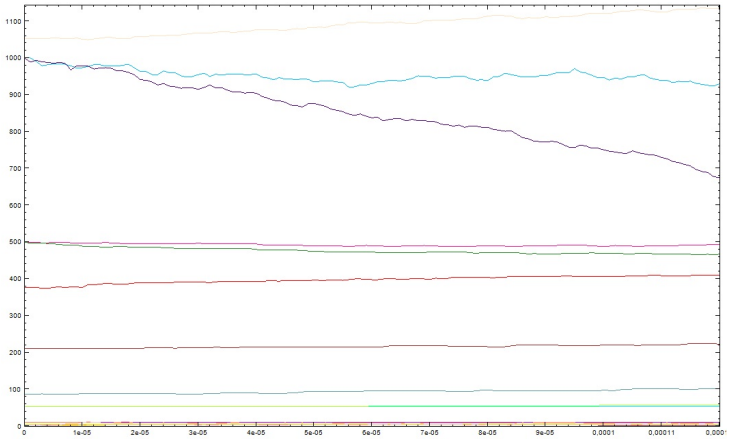


Figure 5: Simulations obtained running the BlenX code for the Kelian river food web model.

Here we described just the base steps of the fod web stochastic model in Kelian river, the all model is available in Appendix A.1.1.
To run BlenX we need to build three files: 1) *.prog* file is called the program file and contains the program structure, i.e. the code for the boxes, with their internal process, the events and the initial settings for the stochastic simulation; 2) *.types* file defines the interaction capabilities of the binders types used by the different boxes coded in the *.prog* files (i.e. predator/prey interactions); 3) *.func* file is an optional file for the declaration of user-defined constants, variables and functions (e.g. reproduction and death rates). In BlenX is possible to use events. They represent statements that are executed with a specified rate and/or when some conditions are satisfied. The event is the composed of a condition *cond* and an action *verb* (the syntax below is an example of event in BlenX).

```
when ( cond ) verb ;
```

The example below is an simple explanation of events, it

means that when entities A and B are greater than two the event "join" will happen at a certain rate *r1*.

```
when(A, B : (|A| > 2 and |B| > 2) : rate(r1)) join (C);
```

Another feature of BlenX that we use in our model is **if-then** statement is used to control the execution flow of the internal code of the box.

3.1.1 Processes in BlenX

Processes in BlenX are presented as sum or as parallel composition of two processes (with + and | respectively). The parallel process, acting as a logic *and*, permits two or more processes to work in parallel or independently. The sum operator acts as a logic *or*, meaning that one or the other process can happen in the model. There is the possibility that the functional group cannot do anything, this situation is described with *nil* process.

3.2 Metaheuristic procedure: Scatter search algorithm

In Systems Biology, as well as in Ecology, efficient and robust methods for parameter estimation are needed. For our work, we are going to use a metaheuristic procedure (based on [73]) for optimization of the marine dynamic model. A metaheuristic is a strategy designed to explore the search space in an optimization problem in order to find near-optimal solutions. Metaheuristics on some class of problems do not guarantee that a global optimal solution will be found but they have a mechanism to avoid be trapped in a local minimum. This procedure can often find good solutions with less computational effort than other algorithms.
There are different of metaheuristics and they are characterized

by the type of search strategy [2]. For instance "single solution" type metaheuristics include simulated annealing and variable neighbourhood search, whereas "population based" type metaheuristics includes evolutionary computation [75], genetic algorithms [34], particle swarm optimization [77] and scatter search [32, 55].
For the marine dynamic model our goal is to find the parameters of the model as reproduction and death rates for the preys which best fit the model considering the real values (reproduction, death, fishing and interaction rates for skipjack tuna). We build the Scatter Search Algorithm for optimization of the dynamic model.
Glover was the first who introduced scatter search (SS) [31] as a heuristic solution methods for integer programming. In their works, the authors choose this algorithm because in case of large number variables it seems to be more reliable. Scatter search strategies were also used to solve a set of over 1000 constrained global optimization problem [63] and proved to give good results in stochastic approaches. Scatter search is a population-based method that uses a reference set to combine its solutions and construct others. The principle of the approach is that useful information about the global optima is stored in a diverse and elite set of solutions (the reference set) and that recombining samples from the set can exploit this information. The strategy involves an iterative process, where a population of diverse and high-quality candidate solutions are partitioned into subsets and linearly recombined to create weighted centroids of sample-based neighbourhood. The results of recombination are refined using an embedded heuristic and assessed in the context of the reference set.

To evaluate the quality of the solutions (in our case the reproduction and death rates) we used an objective function (sometimes referred as *cost function*).
In more details, the scatter search algorithm can be divided into five parts [33]:

- **The diversification generation method** is used to generate a collection of diverse trial solutions. This step focuses on diversification and not on the quality of the resulting solutions. The most effective diversification methods are those able of creating a set of solutions that balance diversification and quality. Better results are produced when the diversification generation step is not purely random and constructs solutions from used diversification measure and objective function. In our implementation we have collected 20 solutions defined as $DIVERSE\_SET\_SIZE = 20$ in python code.

- **The improvement method** transforms trial solutions into one or more enhanced trial solutions with the goal of improving quality, usually measured by the objective function value. The input and output solution may or may not be feasible. The output is a solution that may or may not be better than the original solution. This method is a local search and stops as soon as no improvement is detected in the neighbourhood of the current solution. This step reduces the set size because identical solutions or with same local optimum are merged together. The loop continues until the number (in our code is defined as $MAX\_ITERATIONS = 500$) of improved solutions is reached.

- **The reference step update method** consists on building and maintaining a "reference set" of solutions that are used in the main iterative loop of any scatter search implementation. While there are several implementation options, this element of scatter search is fairly independent from the context of the problem. The first goal of this method is to build the initial reference set of solutions from the population of solutions generated with the diversification method. Subsequent calls of this method serve the purpose of maintaining and updating the reference set. The gap between the two sets is measured with *euclidean distances*. This step typically picks up 10 solutions (as our REF_SET_SIZE in the python code) about which 5 are the best solutions w.r.t. objective function, 5 are the ones that most differs from the solutions in the reference set.

- **The subset generation method** produces subsets of "reference solutions" which become the input to the combination method. The implementation of this method consists of generating all possible pairs of solutions.

- **The solution combination method** transforms given subset of solutions produced by the previous method into one or more combined solution vectors.

Figure 23 contains the pseudocode corresponding of the steps described above for the scatter search algorithm [6].

```
1  # 1. Diversification Generation method
2  InitialSet = ConstructInitialSolution()
3  RefinedSet = []
4
5  # 2. Improvement method
6  For S in InitialSet do:
7      RefinedSet = LocalSearch(S)
8  end
9
10 # 3. Reference Set Update method
11 ReferenceSet = SelectInitialReferenceSet()
12
13 while StopCondition() do:
14
15     # 4. Subset Generation method
16     Subsets = SelectSubset(ReferenceSet)
17     CandidateSet = []
18     For Subset i in Subsets do:
19
20         # 5. Solution Combination method
21         RecombinedCandidates = RecombineMembers()
22         While S in RecombinedCandidates do:
23             CandidateSet = LocalSearch(S)
24         end
25     end
26
27     ReferenceSet = Select(ReferenceSet, CandidateSet,
28      ReferenceSet)
29 end
```

Figure 1: Psudocode of scatter search algorithm: This pseudocode does:
Line 1: method #1 construct the first set of solutions;
Lines from 4 to 6 apply the improvement method calling a local search in loop;
Line 8: the method #3 builds the Reference Set, which is a collection of high quality solutions and diverse solutions Lines from 10 to 21 show the main scatter search loop;
Line 11: the method #4 builds a subset of solutions that become the input of method #5. The subset generation method creates new subsets. A subset is new if it contains at least one new reference solution;
Lines from 13 to 18: the inner while-loop (lines 15 to 17) is executed as long as at least one reference solution is new in the RefSet. If the reference set

contains at least one new solution, the subset generation method builds a list of all the reference solution subsets that will become the input to the combination method;
Line 14: the method #5 transforms the subset of solutions into one or more combined solution vectors

Of the five steps in the scatter search methodology, only four are strictly required. The Improvement Method is usually needed if high quality outcomes are desired, but a scatter search procedure can be implemented without it.
In this work, we implemented a version of the scatter search algorithm in Python. As example to develop the scatter search algorithm we use the work done in [73]. In this project the author faces the problem of the parameter estimation in nonlinear dynamic models of biological systems. He shares few starting data obtained from related study and use the scatter search algorithm trying to minimize the cost function. Our implementation uses the data from the work in [73] to initialize the scatter search algorithm. The idea is to develop a personal scatter search implementation trying to obtain the same results as those of the paper but using a different objective function based on work done in [25].
In our thesis work we developed an implementation of the scatter search algorithm (as explained before) and we adapt it to include our specific models and data. The cost function is computed using the output of the stochastic simulation carried by the BlenX program encoding our model. Below is shown a part of the python code regarding the objective function:

Figure 2: A part of python code containing the objective function.

```
obj_funct_num += 1
```

```
f = open('Therm_iso.temp.func', 'w')
for key, param in enumerate(vector):
    param_num = key + 1
    f.write("let p"+str(param_num)+" : ")
    f.write("const = " + "{:0.8f}".format(param) + ";\n")
f.close()

# running Blenx
runCommand( 'SIM.exe Therm_iso.prog Therm_iso.types
                     Therm_iso.temp.func
                     -f --output=sims/ris' + str(obj_funct_num) )

f = open('ExpPaper.csv', 'r')
for rownum, rowdata in enumerate(f.readlines()):
    if rownum!=0:
        years.append(
            str(int(float(rowdata.strip().split(";")[0])))
        )
f.close()

f = open('sims/ris' + str(obj_funct_num) + '.E.out', 'r')
for row in f.readlines():
    r = row.strip().split("\t")
    year = str(int(float(r[0])))
    if year in years:
        list_data = [ float(r[1]), float(r[2]), float(r[3]),
                      float(r[4]), float(r[5]) ]
        matrix_e_out.append( list_data )
f.close()

f = open('ExpPaper.csv', 'r')
for numrow,numdata in enumerate(f.readlines()):
    if numrow != 0:
        r = numdata.strip().split(";")
        list_data = [ float(r[1]), float(r[2]), float(r[3]),
                      float(r[4]), float(r[5]) ]
        matrix_exp.append( list_data )
f.close()

costo = math.sqrt(
    sum_list(pow_matrix(matrix_diff(matrix_exp,matrix_e_out)))
)
```

At the beginning generate the file *small_tuna.temp.func* with specific values for the reproduction and mortality constants. For example we can generate the following list:

```
let SmallTunaRep : const = 100;
let SmallTunaDie : const = 50;
let SmallTuna_becomeFree : const = 3;
let SmallTunaDie_trapped : const = 10;
let SmallTunaRep_trapped : const = 100;
let VnimbRep : const = 100;
let VnimbDie : const = 30;
...
```

Then we run the BlenX program passing as argument the original files *.prog* and *.types* (containing the model structure) and the *.func* generated in the step before. BlenX produces as output many E.out files that represent the dynamical behaviour over time of the stochastic simulation runs. After we built a matrix from the newest *E.out* file and then compared it with the file ExpPaper.csv, considered reference data, in order to evaluate the cost of the objective function [25]. Below follows an extract of the file *ExpPaper.csv*.

```
time     smalltuna Vnimb epiplfish
0.0003   99        1045   1001
0.0007   50        1040   1010
0.0011   10        1068   1020
0.0022   5         1110   1100
...
```

Evaluations of the cost function will take into account the differences between the data and the simulated traces. The value of the cost function will determine if the generated parameters are going to be used in the next steps of the scatter search algorithm. Finally after several loops and improvements the algorithm returns the best generated parameters. We were looking for these parameters with the aim of find the optimum

values of reproduction and mortality.

3.3 Sensitivity analyses

We expect the results of an ecological system to be close to the equilibrium (the variations more or less have to be constant) during the simulation, even if some extinction events (certain species could die and so disappear forever from the ecosystem) will occur.
Using stochastic, individual-based or event-based simulations we can study the change in the behaviour of the system and we can measure the response of the system to external disturbances. Implementing ecosystem dynamics of Kelian river and the marine one in BlenX, we can do sensitivity analysis for quantifying community importance of species, offering quantitative tools for conservation practice [48].
Sensitivity analysis quantifies the variation in a system's outputs due to variation in parameters that affect the dynamic of the system [37]. In our case is used to study the variation of the change in the network with respect to possible perturbations as based on the work done in [51].
To perform our sensitivity analysis study, we implemented a Python script to the Beta Workbench in order to run batches of stochastic simulations changing the desired parameters. The statistical properties, the mean and the variance, are calculated based on a certain number of reference simulations at time t both in the normal case and in the system where perturbation are introduced. First we define the reference value of population density for species j (A_j) in absence of any perturbation and analyse the effect of species i on the mean population size of species j:

$$A_j = \frac{\sum_{k=1}^{R} a_{k,j}\,(t)}{R}$$

where R is the number of simulations performed and k corresponds to each run; the population size of species j in the undisturbed system ($a_{k,j}$) is recorded at time t. Then we perturb the system by halving each functional group one by one in different runs and the mean values of all components is recorded after time t for the same number R of simulations for each disturbed parameter. The value of population density for species j, after disturbing species i is computed with the following formula

$$A_j\,(i) = \frac{\sum_{k=1}^{R} a_{k,j}\,(i)\,(t)}{R}$$

and the relative response of species j to disturbing species i is calculated with the following expression

$$RR_j\,(i) = \frac{\|A_j - A_j\,(i)\,\|}{A_j}$$

the relative response is normalized over all the living groups (n):

$$NRR_j\,(i) = \frac{RR_j\,(i)}{\sum_{i=1}^{n} RR_j\,(i)}$$

From these response values, we can create a matrix that contains information about the magnitude of the variation of species in column j after disturbing species in row i. The sum along the row and column dimensions provides measure of community importance and community sensitivity, respectively. The community importance of species i considering the mean equals

$$I_H\,(M) = \sum_{j=1}^{n} NRR_j\,(i)$$

and, in case is considered the variance of the R simulations, we provide a community importance metric quantifying the effects on variability of the population dynamics of other groups:

$$I_H(V)_i = \sum_{j=1}^{n} NRR_j(i)$$

The normalized relative response metrics (in mean and variation: I_H, where H stands for the Hurlbert response function, [43]) measure the sensitivity of the system to disturbing component i. These simulation-based values are dynamical measurements of community importance which is strongly needed in conservation biology [61].
In this thesis work the sensitivity analysis helps to quantify the community response to the perturbations inferred in the network. The dynamics of the food webs, in case species i is perturbed we analyze its the effects on species j, may provide information about importance of functional diversity of ecosystems. In our study cases sensitivity analyses shows how human presence affect the population size of the trophic groups in the six sites of the Kelian river and the effects of use of FADs, considered as ecological trap [54, 38], on fishing skipjack tuna in Gulf of Guinea.

4 Aquatic ecosystems

4.1 Introduction

An aquatic ecosystem is a dynamical changing environment. Aquatic ecosystems include oceans, lakes, rivers, streams, estuaries, and wetlands. Within these aquatic ecosystems are living things that depend on the water for survival, such as fish, plants, and micro-organisms. These ecosystems are very fragile and can be easily disturbed by pollution. The two main aquatic ecosystems are the marine one and the one in freshwater. The marine ecosystem is distributed for 71% on Earth surface containing 97% of planet's water. The remaining 2.5 - 2.75% comes from ice, lakes, rivers, groundwater. Only 1% of Earth water is available to humans as fresh water and it can be obtained from surface waters and groundwater. Nowadays freshwater supply is under severe pressure as a result of human activity and natural forces. Figure 2 shows an example how the hydrologic cycle of water functions among the ocean, the atmosphere and land masses. The main processes are evaporation, precipitation, surface run-off and groundwater percolation.

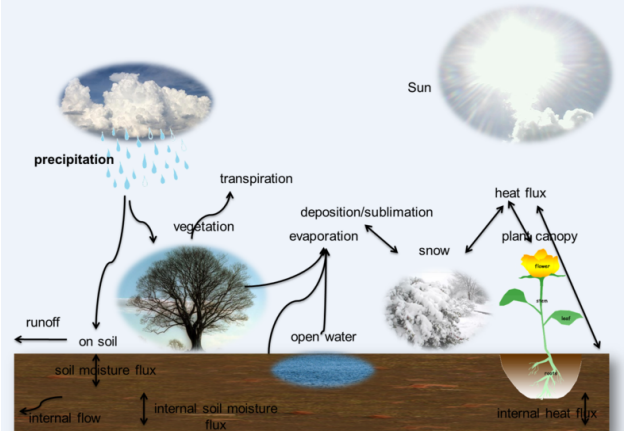


Figure 6: Water hydrologic cycle: water from river to ocean, or from ocean to atmosphere and from atmosphere to groundwater changing state from liquid to gas and after to solid by physical processes; evaporation, condensation, precipitation, infiltration, runoff, and subsurface flow. At the end of all the balance of water on Earth remain constant over time.

There is a law in physics which affirms in a close system "mass can neither be created nor destroyed, although it may be rearranged in space, or the entities associated with it may be changed in form" [42]. This law can be adapted to what happens in water processes too; water is not created new, there is always the same water transformable in vapour (gas state), ice (solid state) and water (liquid state) again. For this reason we must pay more attention on the impact that any human action can have in any stage of the hydrologic cycle. In this thesis work we use aquatic food webs and computational tools to predict the human impact in aquatic ecosystems.
The communities are often dominated by primary producers (autotrophic organisms, plants in terrestrial systems and algae in aquatic systems; they are able to convert inorganic products into organic one in order to be consumed by the heterotrophic organisms called here consumers such as herbivores, carnivores

etc.) that are smaller than the consumers with high growth rates (fig. 7).

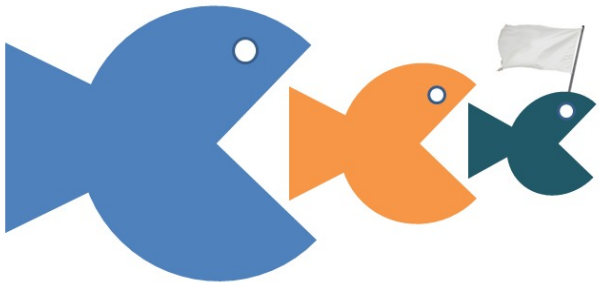


Figure 7: A simple example of food chain in aquatic ecosystem.

Aquatic predators have a lower death rate than the smaller consumers. Primary consumers present a longer lifespans and slower growth rates (e.g. phytoplankton live just a few days, whereas the zooplankton eating the phytoplankton live for several weeks and the fish eating the zooplankton live for several consecutive years). For this reason they are able to accumulate more biomass than the producers they consume.
Using a stochastic model approach on our two case studies we aim to analyze the human effects as pollution and overfishing on the organisms or functional groups that are living in Kelian river, Borneo and on skipjack tuna in the Gulf of Guinea in Atlantic ocean.

5 Case study I

5.1 Human impact in Kelian river, Borneo, Indonesia

Indonesia, an archipelagic country composed of 17.508 islands, is the fourth most populous country in the world with the capital city Jakarta. It is situated along the equator in South East Asia. The country has a strategic position for inter-island and international trade. The land presents a great biodiversity, housing 130 million years old rainforest and many endemic species of plants and animals. A lot of study and analyses are done from the Indonesian Environment Monitor on Pollution, part of East Asian Environment Monitor series. They establish that the economic growth is the reason of soil, air and water pollution and health problem in the archipelago [27]. There are still rudimentary sewerage system and low level of sanitation coverage. The consequences are contamination of surface and groundwater and poor waste management system, 90% of waste are open dumping.
Water pollution, from industrial (e.g. mining), domestic sources (organic waste) and agricultural activity, presents a serious problem regarding the diseases (e.g. diarrhea, hepatitis, etc.) with some effects on oceanic and river fish, coral reefs etc. too. The annual amount of rainfall is large but not enough to prevent the pollution problem. The Clean River Program or Program Kali Bersih (PROKASIH) inaugurated in 1989, aims to prevent in someway the effects of pollution in Indonesia targeting the worst industrial polluters, pointing to reduce their pollution loads by 50% within two years on a voluntary basis. Another result of PROKASIH is the classification of the river water in four categories in order to understand the use:

- water that may be used directly for drinking without treatment
- water to be used for drinking after conventional treatment
- water to be used for fisheries and watering animals
- water to be used for agriculture, municipal supplies, industry, and hydropower

In that work they quantify the maximum amount of different substances allowed to safely use the water for each of the activities listed above [78]. Kelian river, our first case study, situated in Borneo island, as many other rivers in Indonesia is affected by mining activity and human waste coming from their settlements. Borneo (in Indonesian Kalimantan) (fig. 8), is the largest island of Asia, 743.330 km^2, and the third in the world [56]. From geographic point of view it is surrounded by different seas: to the north and north-west by South China Sea, to the north-east by the Sulu Sea, to the east by the Celebes Sea and the Makassar Strait, and to the south by the Java Sea and Karimata Strait. Malay Peninsula and Sumatra are situated to the west of Borneo and Java to the south, Sulawesi To the east, and the Philippines to the north-east. Kapuas layed in West Kalimantan is considered to be the largest river system with a length of 1.143 km. Other rivers are Mahakam in East Kalimantan (980 km long), the Barito in South Kalimantan (880 km long), and Rajang in Sarawak (562.5 km).

Figure 8: Indonesia map showing the Borneo island with Kelian river shown in red.

Kelian river, is a tributary of the Mahakam river situated near the equator in pristine tropical rainforest. The upper reaches of it lays in primary rainforest. The climate is tropical with abundant rainfall, apart on May and August. The width may vary from 15 to 25 m and the maximum and its depth reaches 30 - 40 cm at all sites. In Borneo, the roads are few and rivers are very essential pathways for human habitation and present main water supplies. The biodiversity of the island presents a great variety (e.g. 15.000 species of flowering plants, 3.000 species of trees, 221 species of terrestrial mammals, 420 species of resident birds, 440 freshwater fish species [56]). In 1990s there was a mining boom with consequences of tailing wastes that raised the risk of costly accidents, and contaminating rivers with pollutant. In 1999 a new law concerning to Forest management prohibitions any surface mining in state forest land, regardless of its classification: large-scale, small-scale, artisanal and small-scale mining. From 1997 to 1998 the rainforest was destroyed for industrial reason and from the forest fires by the locals for the crop. Kelian river is an area of interest because of gold mineral. Many streams and rivers pass in Kelian river and drain into

the major river Mahakam, situated in east Kalimantan. The main activity done by the indigenous is cultivation of rice and vegetable crops, once clearing by fires the forest. Some forest are left intact for spiritual reasons. In 1949 a visiting group of Penihing Dayaks visited the island and once they realize that gold was present, the news spread to the surrounding population very quickly and small-scale mining industry soon was established in the area. A small-scale mining industry soon was established in the area. In 1950 there was an ethnically mixed population. In 1970 the Anglo-Austrialan company Rio Tinto Indonesia came to the area. They formed P.T. Kelian Equatorial Mining (K.E.M.) without Indonesian government concession. In 1988 there was demonstration by Borneo communities against K.E.M. operations. The reasons for the demonstration is the air pollution caused by trucks and heavy equipment, the pollution of the river and mass fish deaths because in 1991 an incident caused the falling of 1.200 drums of chemicals in the river. The response of K.E.M. company was by security guards and harassed beaten up and shot at local people [39]. The mine was active from 1991 until 2005 owned for 90% by Anglo-Austrialian company (Rio Tinto) and 10% by Indonesian company. It was the second largest gold mine in Indonesia (fig. 9).

Figure 9: Gold mine at Kelian river.

Once it was closed (fig. 10) in 2005 a lot of problem (e.g. about human rights abuse or environmental pollution left unresolved: one of the main problem is the fact that rainwater accumulated in the mine would cause toxic wastes to enter into local rivers.

Figure 10: Kelian river gold mine once closed.

The mining activity is not the only cause of pollution in the river. Downstream the river there are human settlements (fig. 11) and, as mentioned before, the sewerage system is rudimentary.

Figure 11: The image is an example of human settlements across the Limbag river, in Borneo.

The organic matter coming from human waste is another serious problem for water quality of the river and for the negative effects that has in its fauna .

5.2 Food webs for the six sites of the river

Before starting with the description of food webs in the 6 sites and data used in the dynamic models of Kelian river ecosystem we first introduce some notions about the feeding strategies of functional groups considered in the food webs. Primary producers (autotrophs), placed at the bottom of the food web (e.g. diatoms, green algae and blue-green algae etc. in our river ecosystem), are able to produce biomass from inorganic matters and provide energy, food for the other organisms. Herbivores are animals that feed principally on autotrophs (e.g. plants, algae). They are considered primary consumers in the food web.

It is photosynthesis process that helps herbivores to ingest the carbohydrates produced by a plant (primary producer). Omnivores are species feeding at several trophic levels (in our food web they are omnivorous fish placed at intermediate position). Carnivores are considered meat-eater, their diet consists in consumption of other animals through predation or scavenging and in the food web are the predators placed in the top of the food web usually (in our food web are called top predator and are carnivores fish).
Food webs construction for each of the six sites is done considering some information: a) distribution and diet of macroinvertebrate species or morphospecies; b) diet and presence of fish; c) presence of benthic alga. The networks of the 6 sites are composed from 12 to 15 nodes (trophic groups) linked together as presentation of prey-predator relations. Table 1 shows how functional groups living in the river are classified and the names used in food webs and in the model.

Names	definitions of each functional group
CARN	carnivorous fish
PRED	invertebrate predators
OMNI	omnivorous fish
GRAZ	invertebrate grazers
HERB	herbivorous fish
HEDE	herbivore-detritivore fish
SHRE	invertebrate shredders
COLG	invertebrate collector-gatherers
COLF	invertebrate collectorfilterer
TERR	terrestrial insects
DIAT	diatoms
ALGA	green and blue-green algae
POM	settled and suspended coarse and fine organic particles
LEAF	leaf litter
HUMW	human waste
FILA	filamentous bacteria

Table 1: Description of functional groups in Kelian river and the names used to describe them in the food web model.

Figure 12 shows the food web of site 1. At the bottom of the network are presented five producers providing food of which two are non-living particles LEAF and POM and three are living taxa ALGA, DIAT and TERR. In the next level of the network, we can find six herbivores which feed on producers COLF, COLG, SHRE, HEDE, HERB and GRAZ. Omnivores (OMNI) feed on them and higher predators (PRED and CARN) situated on the top of the food web. Groups as CARN, OMNI, HEDE and HERB are fish [70].

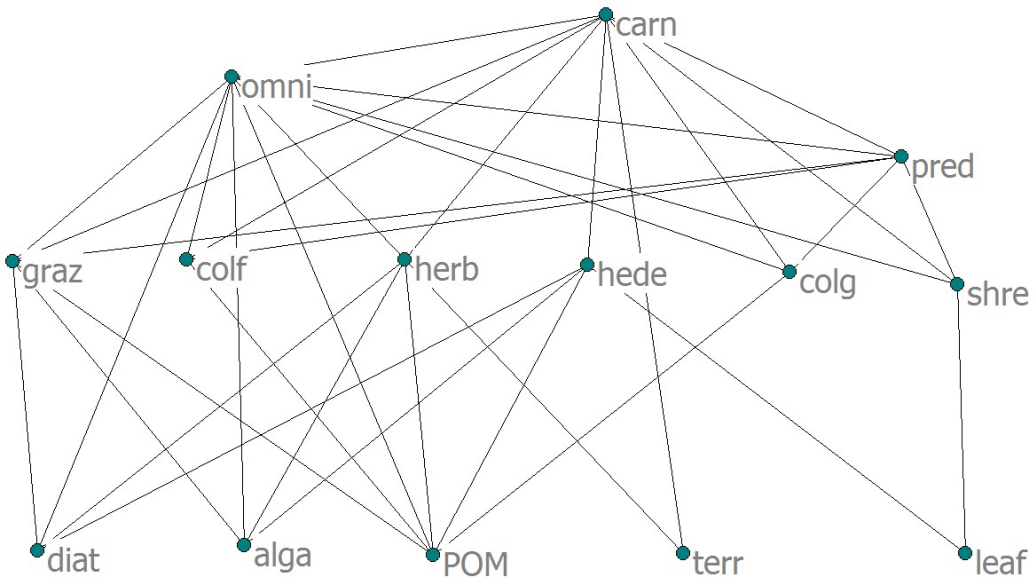


Figure 12: In site 1 the food web shows the interactions between trophic groups in the river ecosystem.

Downstream the river, in site 6 appear HUMW (human waste) and FILA (filamentous bacteria), no present in other sites, placed at the bottom of the food webs. Site 6 is otccupied by human settlement that use to throw their rush into the river, and this is the reason why we find only here HUMW group. FILA (filamentous group) is present in sites 3,4,5 and 6. They are able to metabolize manganese. COLF, GRAZ and SHRE show a sensibility to human impact, for this reason they are missed in sites 4 and 6. Some of them changed diet and became collector-gathers at the polluted sites. Leaf is not present in site 6. The other groups are present in all the sites. The assignment of feeding groups as collectors, grazers, shredders or predators [60] were obtained from the results of gut-content analyses and the use of statistical methods (e.g. Sperman correlation rank) and software (e.g. ANOVA) [81].

5.3 Dataset

In September 1990 (wet season) the organisms that populate the Kelian river were sampled (exactly one year before the activity of the mine). In August 1993 (dry season) after the mine has started its activity [82], then in June 1994 (dry season) and March 1995 (wet season) were done other sampling. The data comes from field data collection from 1991 until 2005 and studies on fish communities [81, 82].
The pollution on the trophic ecology in Kelian River was studied by comparing food webs (on the basis of gut analysis and field and laboratory observations) at six sites paying attention to functional biodiversity of trophic groups. The species are aggregated in functional groups (trophic groups).
For food webs construction we use the information concerned to number of trophic groups (corresponding to the nodes of the network): number of trophic groups in the different sites are for site one 14, for site two 14, site three 15, site four 12, site five 15 and for site six 12 trophic groups. After we used information about the interactions values computed from the inference of prey preference (showed in partial feeding matrix 17) which represent the proportion of food supplies of the predator and reproduction, death and when rates (in Appendix A.7.3 are imported the feeding partial matrices and tables with information about population size, reproduction and death rates concerning to the other sites).

SITE 1	CARN	OMNI	PRED	HEDE	HERB	GRAZ	COLG	COLF	SHRE	TERR	ALGA	POM	DIAT	LEAF
CARN	0	0	0	0	0	0	0	0	0	0	0	0	0	0
OMNI	0.111	0	0	0	0	0	0	0	0	0	0	0	0	0
PRED	0.111	0.111	0	0	0	0	0	0	0	0	0	0	0	0
HEDE	0.111	0	0	0	0	0	0	0	0	0	0	0	0	0
HERB	0.111	0	0	0	0	0	0	0	0	0	0	0	0	0
GRAZ	0.111	0.111	0.25	0	0	0	0	0	0	0	0	0	0	0
COLG	0.111	0.111	0.25	0	0	0	0	0	0	0	0	0	0	0
COLF	0.111	0.111	0.25	0	0	0	0	0	0	0	0	0	0	0
SHRE	0.111	0.111	0.25	0	0	0	0	0	0	0	0	0	0	0
TERR	0.111	0.111	0	0	0	0	0	0	0	0	0	0	0	0
ALGA	0	0.111	0	0.25	0.333	0.07	0	0	0	0	0	0	0	0
POM	0	0.111	0	0.25	0.333	0.734	1	1	0	0	0	0	0	0
DIAT	0	0.111	0	0.25	0.333	0.196	0	0	0	0	0	0	0	0
LEAF	0	0	0	0.25	0	0	0	0	1	0	0	0	0	0

Table 2: Partial feeding matrix showing interaction between predator and prey in site 1; in the columns are the predators and in the rows the preys. The matrix is estimated normalizing (the columns sum to one) the connections between trophic groups by the total intake of each receiving node.

Taking as starting point the values obtained in field samplings the individuals numbers are fitted. In case of ALGA and DIAT the fitting was harder to do.
Since precise birth and death rate are unknown, we used values that fit the qualitative behaviour of the system in time. The table 11 represents information about, population size, reproduction, death and when rates used to construct food webs and in the next step the stochastic model.

	Site 1			
	Num. Indiv.	Rep	Die	When
CARN	5	50	0.5	0
PRED	54	5400	5.4	0
OMNI	10	100	1	0
HERB	10	1000	1	0
HEDE	10	1000	1	0
GRAZ	1054	105400	1.05	0
COLG	377	377000	0.38	0
COLF	212	212000	0.21	0
SHRE	87	87000	0.09	0
ALGA	500	500	500	5000
DIAT	500	500	500	5000
TERR	54	540	5.4	54
LEAF	1000	10000	10000	100000
POM	1000	10000	10000	100000

Table 3: Site 1; table with the information about number of trophic groups, reproduction, death and when rates respectively.

Being a complex model with many parameters, for some data is necessary the approximation. We take a comparative approach, thus the effects of making real differences in measured parameters are quantified in the context of this multi-parameter dynamical model. Finally to have a quasi-balanced behaviour in the model (no mass extinctions and exponential growths) we did some adjustments to the hypothetical values.

5.4 Sites of the Kelian river

The width of the river is between 15 and 25 m, the maximum depth 30-40 cm. The river was sampled at 6 sites in order to study the effects of mining activity and organic waste coming from human settlement in it (fig. 13).

Figure 13: In the map river is subdivided in 6 sites in order to better study the biological communities.

The upper sites were in pristine rainforest but the river became increasingly polluted downstream, largely owing to sedimentation from alluvial gold mining activities.
Below follows the description concerning to the six sites.

- **site 1** is near the pristine rainforest, upstream of the mine and Camp Prampus (miners' houses); composition of species is mainly of diatoms and green algae
- **site 2** next to the open cut mine, is situated upstream the Camp Prampus and downstream with the confluence with Sungai Magerang; the banks are spoiled of vegetation; rocks, sand and mud have finished into the river; blue green algae has the prevalence here, bioindicator of external perturbations

- **site 3** is 100 m downstream of the confluence of Sungai Nakan where is construct a dam to form a polishing pond (discard water); here is where all the waste from P.E. and K.E.M. is accumulated; the flora and the fauna of the site are composed by mainly bacteria, blue green and green algae with the function of entrapment of fine sediments and organic matter
- **site 4** situated below the P.T. K.E.M.; the banks are with sand; filamentous green and blue green algae and fungi live covering the rocks here behaving as a trap sediment and organic matter
- **site 5** situated downstream of Sungai Daunsip spoiled of the vegetation; this is interest area for gold research; there is a decrease in population size and diversity of grazers, shredders and collector-filterers
- **site 6** is the lowest, situated downstream of Sungai Babi and near human settlements; the water is used mainly by local people for drinking, washing and as rubbish deposit; some trophic groups are missing here or are reduced in abundance

Over the years the fauna density and species richness decrease from site 1 to site 6 in correlation with the pollution by the suspended solids and turbidity (e.g. sites 5 and 6 present pollution-tollerant species). A particular attention is directed to benthic invertebrate fauna variation (called shredders in our food webs). These organisms, are usually found in or on the bottom sediments of rivers, streams, and lakes. The study of these organisms is important, being strongly affected by the environment they live in, including sediment composition, water quality,

and hydrological factors that influence the physical habitat, they can be used as bioindicator of water quality. These organisms were abundant in pristine site and less present downstream in correlation with the amount of fine sediment. The information that came from Yule show that pristine sites (1,2) are composed by more complex and richer fauna regarding to sites downstream (5,6). No variation in fish population is observed [70].
The different levels of human disturbance in Kelian river can be studied by comparing the structure of food webs at the 6 sites building a dynamical model, running simulation and doing sensitivity analyses. The analyses of food webs is necessary because the trophic interactions among aquatic organism may reflect the pollution effects.

5.5 Dynamic models

We build a dynamical food web model which helps to analyse the human disturbance along the river. The first simple model describing predator-prey interaction is the model of Lotka-Volterra, and it is used as a base for the predator-prey model published in [51] and described in the State of the Art section 2.2. The Kelian food web model is done based on the previous model done in BlenX. The interactions between species in Lotka-Volterra stochastic model are presented by differential equation, in this work thesis by *rates*. In our food web model of Kelian river, boxes are used to model functional groups, divided in the following categorizes: as *predator* (can be top-predator, CARN from sites 1 to 5 or PRED for site 6; they are placed at the top of the food web); *prey* (here also referred as "intermediate entities" and they can behave as predator and prey at the same time); *food* (primary producers placed at the bottom of the food

web). To build a dynamic model in BlenX we need information about the number of individuals, interaction rates, birth and death rates. The data about number of individuals and interaction rates for the building of the stochastic model for Kelian river are able from field sampling [81]. We took the model in [51] as starting point to describe the stochasticity in Kelian river food webs.
The different states that an animal can be during its life-cycle are:

- eat: an animal eats the prey. In the internal process the action is presented by *eat!().nil* in case of the top-predator or *food!().nil* in case of intermediate entities.
- hunted: a prey is hunted by predator with *eat?().nil* action or with *food?().nil* action in case of the primary producers.
- duplication: intended as biological reproduction at a particular *rate* of a functional group presented by *ch(rate, dupl, duplication)* action in the internal process.
- die: natural death at a particular *rate* of a functional group presented in the model by *delay(rate).die(inf).nil* internal process.

In the model, an alternative path of duplication is possible (at specific rates depending on the species involved): if a box is put inside a *new* event as:

```
when (Alga::rate(algaWhen)) new (1)
```

Using BlenX language and the data obtained from [81], the three files *.prog, .types, .func* necessary to run the model are compiled. The three files are used as input for the BetaWB

simulator and once the simulation are finished running as output files are obtained *res.spec* file with the reactions happening during simulations, *res.E.out* file with the description of individual number and time steps, *res.C.out* and *res.V.out* file with variables computed in case a function is declared in *.func* file. In the table 4 is showed a short description of the three files compiled in order to run the simulation in BlenX.

//.PROG [steps=120, delta=0.01]	Simulation output length−accuracy
TOP PREDATOR let pcarn: pproc= eat!().y!().nil + eat!().ch(rate(carnRep),dupl,duplication).nil + delay(rate(carnDie)).die(inf).nil	Process declaration of the top predator
PREY let pherb: pproc= food!().x!().nil + food!().ch(rate(herbRep),dupl,duplication).nil + eat?().die(inf) + delay(rate(herbDie)).die(inf).nil	Process declaration of the prey
FOOD let palga: pproc = ch(rate(algaRep),dupl,duplication) + food?().die(inf) + delay(rate(algaDie)).die(inf)	Process declaration of the producer
when (Algadup: :inf) split (Alga, Alga);	Duplication event
STARTING run 500 Alga || 5 Carn || 212 Colf	Initial conditions for the simulation
//.TYPES (carn_hunts, hunts_pred, 0.111), (omni_lifes, hunts_shre, 0.111), (pred_lifes, hunts_graz, 0.111), (herb_lifes, diat_lifes, 0.333), (colf_lifes, pom_lifes, 1.00)	Binders of predator−prey interactions and rates

//.FUNC
let algaRep : const = 500;
let algaDie : const = 500;
let algaWhen : const = 5000;
	Definition of the constants

Table 4: Extracts from BlenX input files, ṗrog, ṫypes and ḟunc of the Kelian river stochastic model on the right column and the description of the lines of codes on the right column.

On the top of *.prog* file there are some information about the time of simulation written *[steps=120, delta=0.01]* where steps means number of steps that the simulator will schedule and execute and delta parameter instructs the simulator to record events frequency; *BASERATE : inf* is used as a common basic rate for the actions which do not have an explicit rate set. At the end of the file the *run* command is followed by the initial condition of each functional group. The second description is about file *.types* which imported the predator-prey interactions followed by the feeding rates. In the beginning part of the file, all the binder's types used in the.*prog* file are listed.
Finally, the *.func* file contains all the constants for the rates of death, reproduction and duplication of the boxes coded in the *.prog file*. For the producers are three parameters cited; *nameRepr*, *nameDie* and *nameWhen*. In file *.prog* the rate is used in the condition *when (Food::rate(foodWhen)) new (1)* in case of primary producers and this means that at a certain rate a new *food* is created.
In the next section we describe in more details specific internal program of the boxes in the *.prog file*, used to model the 6 food webs in the Kelian river. In Appendix A.1 we report the full code for the six stochastic food web models of Kelian river regarding all the sites studied in this thesis work.

5.5.1 General description of predator behaviour in BlenX

The functional groups are represented by boxes in file *.prog*. From site 1 to site 5 CARN functional group is the top predator. It has two communication channels called *eat*, and *dupl* and an internal process with parallel operator of subprocesses *pcarn* (explained below):

```
let Carn: bproc= #(eat,carn_hunts),#(dupl:0,A)
[rep y?().pcarn | pcarn];
```

The *rep* operator is used to replicate copies of the process *y?().pcarn*. Below follows the code that is represented by *pcarn* after the parallel (|) symbol:

```
eat!().y!().nil +
eat!().ch(rate(carnRep),dupl,duplication).nil +
delay(rate(carnDie)).die(inf).nil
```

Through the parallel processes can happen an exchange of a message from one of the subprocesses in from *pcarn* and the action *rep y?()*, generating an intra-communication. With *pcarn* begins the description of the process of the predator CARN. The internal process of Carn is defined by three subprocesses linked by *sum (+)* operator:

```
let pcarn: pproc= eat!().y!().nil +
eat!().ch(rate(carnRep),dupl,duplication).nil +
delay(rate(carnDie)).die(inf).nil;
```

The *sum* operator is interpreted as a choice *or*, meaning that one of the three subprocesses can happen: *eat!().y!().nil*, just eat and restarts the internal process through the call on the y channel of the rep BlenX operator (for more details about how this operator works, we refer the reader to [14]); *eat!(). ch(rate(carnRep), dupl, duplication).nil* which represent the eating and at a certain rate called in the model *rate(carnRep)* replicates itself; or *delay(rate(carnDie)).die(inf).nil* at certain rate *rate(carnDie)* the predator dies. The values of the two rates *rate(carnRep)* and *rate(carnDie)* are defined in the *.func* file:

```
let carnRep : const = 50;
let carnDie : const = 0.5;
```

The inter-communication, in ecology is predator-prey interaction, between boxes are described in file *.types* where the first

two are the binders and the third element is the affinity rate. The *eat!()* action needs to be coordinated with an equivalent *eat?()* action in a different box (in this case it will be the prey box) and the rate of this interaction is described in the *.types* file as follows:

```
(carn_hunts, hunts_herb, 0.111)
```

The *eat* channel CARN interacts with the prey HERB with a specific affinity rate (0.111). The *eat* action is executed in two possible ways: 1) *eat!().y!().nil* sends a signal to the prey *eat?().die(inf)* or in case the prey is a primary producer *food?(). die(inf)* which dies (the functional group CARN continues to live its life); 2) *eat!* of box CARN that sends a signal to *eat?* channel of box HERB.
Action *change (ch)* performs modification of the box interface changing the value *A* of the binder *(dupl:0,A)* in *duplication (dupl:0,duplication)* with a certain rate *carnRep*. After this is executed the box Carn will change its state into the flowing Carndup state:

```
let Carndup: bproc= #(eat,carn_hunts),#(dupl:0,duplication)
[rep y?().pcarn];
```

and its fate is controlled by the following split event:

```
when (Carndup: :inf) split (Carn, Carn);
```

Finally having all build the model in BlenX with the *run* command start the simulations.
The figure 14 is a description of predator different paths reacting with the prey.

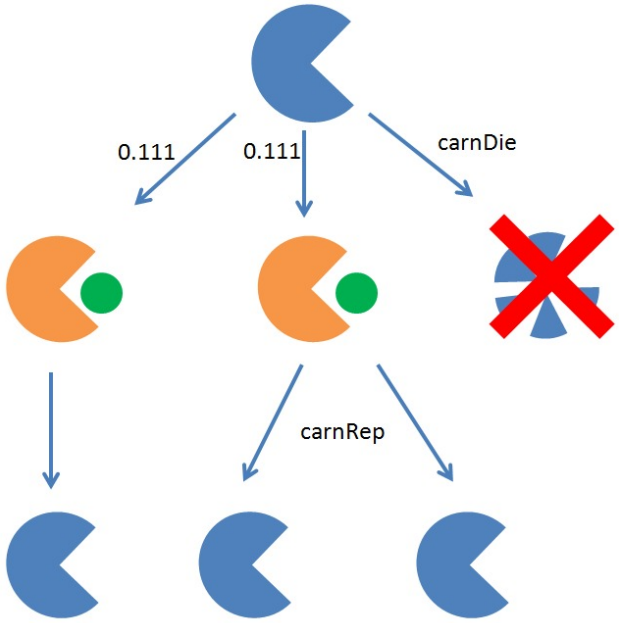


Figure 14: The three alternative destinies of a predator are shown. The results of the two paths on the left, are (one or two) predators back in the initial state. The result of the path on the right is the disappearance of the predator box from the system. For more details about the different steps, see the description in the text.

5.5.2 General description of prey behaviour in BlenX

The box prey (e.g. HERB) is composed by a box with three binders sites (*eat*, *food* and *dupl*) and an internal process as follows:

```
let Herb: bproc = #(eat,hunts_herb),#(food,herb_lifes),
                  #(dupl:0,A)
                  [rep x?().pherb | pherb];
```

The internal process is composed by 4 sub-processes linked by operator *sum (+)*:

```
let pherb: pproc= food!().x!().nil +
food!().ch(rate(herbRep),dupl,duplication).nil +
eat?().die(inf) + delay(rate(herbDie)).die(inf).nil;
```

The action *eat?* creates an inter-communication with the corresponding *eat!* action of the predator (described in section 5.5.1). The action *food!()* of the intermediate group creates an inter-communication with the corresponding *eat!* action of the predator the channel *food?()* of the primary producer or that of another intermediate group and may have two different results: 1) a simple eating and restarting of the initial state of the box, or 2) an eating action followed by a duplication. Replication and natural mortality happen as for the predator, creation of two new boxes for the first case and deletion of a box for the second case.

5.5.3 General description of food behavior in BlenX

In our model, with the word *food*, we describe the primary producers. The box is composed by two binders *food*, *dupl* and an internal process, as follows:

```
let Alga: bproc = #(food,alga_lifes),#(dupl:0,A)
[palga];
```

the inter-communication is executed by internal process as incoming signal and as a result of that, the box is deleted:

```
let palga: pproc= ch(rate(algaRep),dupl,duplication) +
food?().die(inf) + delay(rate(algaDie)).die(inf);
```

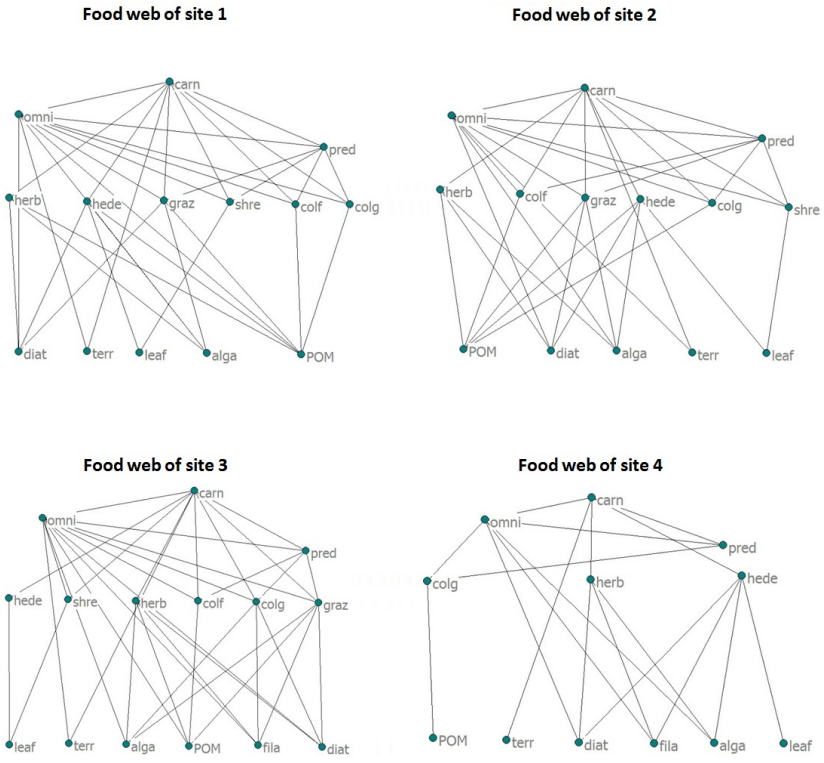
Food does not eat to reproduce as in case of predator and prey, because they are the primary produces, i.e. the base of the food web. This means that they manage to obtain their own food from light energy or chemical energy, without eating. The natural mortality of this box is coded as in the predators and intermediate preys explained before. The primary producer presents a second event apart that of duplication as the other trophic groups (see the example below):

```
when (Alga::rate(algaWhen)) new (1)}
```

which means that at a certain rate *algaWhen* declared in file *.func* a new box of alga will be created.

5.6 Results

With information found in [81] we built 6 food webs (fig. 15) for the sites of the Kelian river. The data obtained for the 6 sites shows that the presence of the individuals differs from site to site.



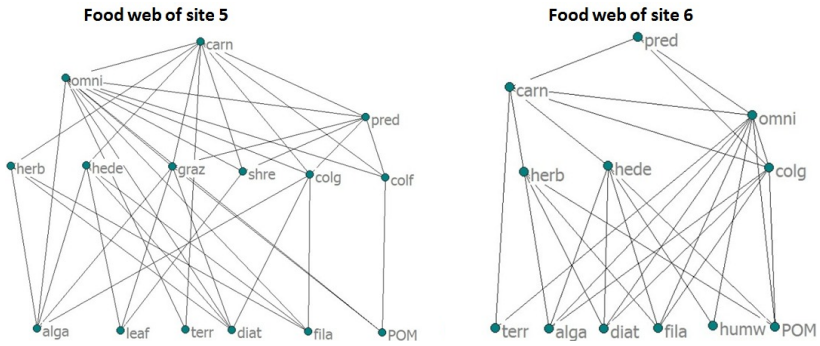


Figure 15: Food webs of the different sites of the Kelian river.

Some functional groups are not present in each site along the river. In sites 1 and 2 the missing species are HUMW and FILA. In sites 3 and 5 is missing only HUMW and in site 4 and 6 GRAZ, COLF, SHRE are eliminated due to pollution effects; HUMW is only in site 4 and LEAF only in site 6. Food webs show structural network differences from one site to the other. The parameters (e.g. reproduction and death rates) also show some differences from site to site. To quantify the functional effects derived from these differences we perform some stochastic simulation studies and sensitivity analysis.
We developed a Python script able to control the stochastic simulator of the Beta Workbench to be able to run batches of simulation runs. After the simulations, we performed some statistical analysis (as explained in Section 3.3) for all functional groups present in each specific site.
Figure 16 shows a plot of a typical simulation run: the curves show a quasi-equilibrium trend in the river ecosystem.

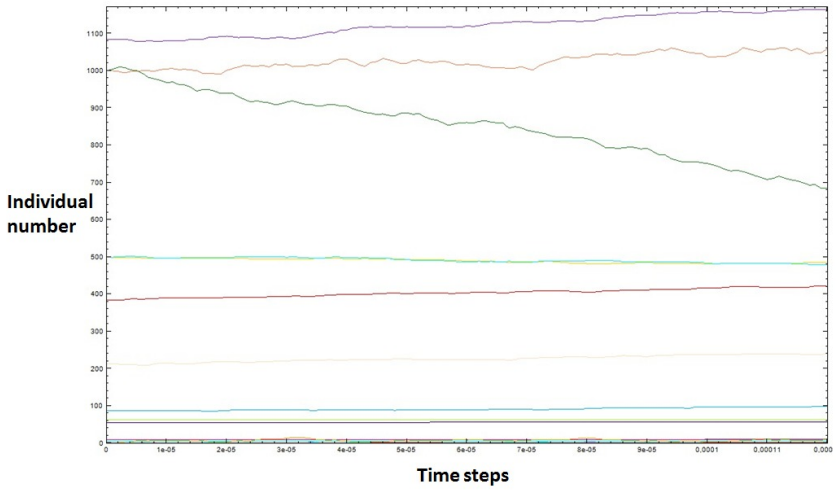


Figure 16: The quasi balanced system of functional groups living in site 1; each curve corresponds to a specific functional group.

Once reference simulation with the unperturbed system has been collected, we perturbed the system halving one by one the functional group. We then computed the same statistical analysis as in the previous case to compare the results. We did sensitivity analyses obtaining dynamical measurements of community importance $I_H(M)$ and $I_H(V)$.
In table 5 are shown the results based on community importance series of the mean $I_H(M)$.

I_H(M)						
	Site 1	Site 2	Site 3	Site 4	Site 5	Site 6
CARN	0.0722	0.0714	0.0676	0.0900	0.0660	0.0859
OMNI	0.0572	0.0662	0.0696	0.0896	0.0692	0.0807
GRAZ	0.0818	0.0878	0.0838	-	0.0688	-
PRED	0.0745	0.0702	0.0618	0.0590	0.0744	0.0931
SHRE	0.0736	0.0690	0.0481	-	0.0547	-
COLF	0.0751	0.0689	0.0574	-	0.0557	-
COLG	0.0741	0.0735	0.0751	0.0874	0.0693	0.0854
HEDE	0.0566	-	0.0560	0.0752	0.0640	0.0653
HERB	0.0707	0.0707	0.0659	0.0897	0.0574	0.0653
TERR	0.0736	0.0759	0.0481	0.0590	0.0749	0.0900
ALGA	0.0780	0.0729	0.0771	0.1078	0.0720	0.0880
POM	0.0687	0.0686	0.0820	0.0893	0.0622	0.0787
LEAF	0.0673	0.0636	0.0694	0.0789	0.0688	-
DIAT	0.0765	0.0758	0.0757	0.0913	0.0759	0.0871
HUMW	-	-	-	-	-	0.0876
FILA	-	-	0.0624	0.0829	0.0666	0.0929

Table 5: The table shows the community importance series I_H(M) of each group in the 6 sites (in blue GRAZ which is more present in sites 2,3; decrease in site 5 and is absent in sites 4 and 6; PRED shows a decrease in the middle of the river and higher value in site 6, SHRE looking to the values we can deduce that shows sensibility to the human impact in the river).

The values obtained are normalized. The grazers (GRAZ) are the most abundant functional group in sites 2 and 3 (with values of 0.0878 and 0.0838); in site 5 (0.0688) they are lower in importance and disappear in sites 4 and 6. From these values we understand that GRAZ are the group with the largest community effect. Invertebrate predators (PRED) are of intermediate importance in site 1 (0.0745) which is a pristine zone and lower in abundance downstream showing intolerance to pollution. In the middle of the river the PRED decreases in importance (0.0618) and in site 6 they are more present showing

an adaptability to human impacts (0.0931). The invertebrate shredders (SHRE) gradually decrease in quantity from site 1 to site 3 (0.0736, 0.0690, 0.0481) and disappear in sites 4 and 6. The shredders abundance is showed to decrease from higher to lower elevations in tropical streams Peninsular Malaysia [21]. Downstream in the polluted sites (e.g. sites 3,4,5 and 6) the primary producers filamentous bacteria (FILA) appear here showing tolerance to pollution.
The graph in figure 17 is constructed from the data in table 5. The curves show the trend of each functional group in all the sites.

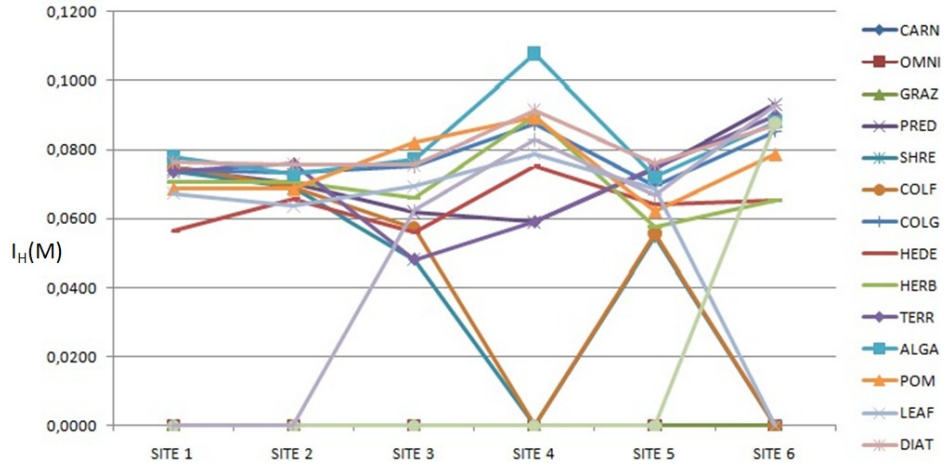


Figure 17: Community importance series of the mean I_H(M) of each trophic group computed for all the six sites. In the axes are represented the six sites and in the ordinate I_H(M) values.

In figure 18 the trend of curves are representative of GRAZ, PRED, SHRE and FILA groups respectively.

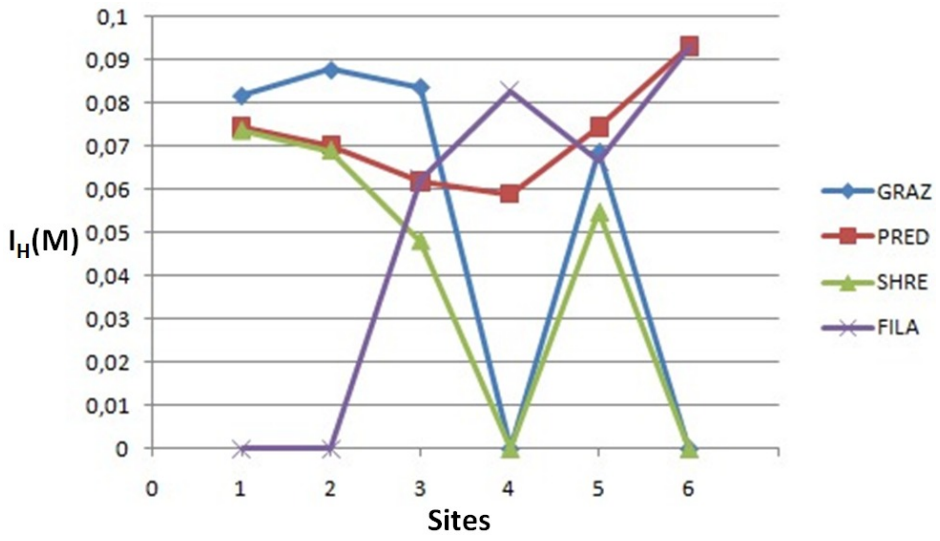


Figure 18: Graph shows the community importance series of the mean $I_H(M)$ for GRAZ, PRED, SHRE and FILA functional groups. In the axes are represented the six sites and in the ordinate $I_H(M)$ values.

Referring to community importance measure of dynamical variability $I_H(V)$ (table 6) GRAZ do not show the same importance as for the $I_H(M)$ index.

I_H (V)						
	Site 1	Site 2	Site 3	Site 4	Site 5	Site 6
CARN	0.0355	0.0713	0.0863	0.0639	0.1294	0.1010
OMNI	0.0524	0.0595	0.0642	0.1271	0.0744	0.1463
GRAZ	0.0661	0.0777	0.0642	-	0.0472	-
PRED	0.0666	0.0800	0.0724	0.0759	0.0509	0.0761
COLF	0.0593	0.0570	0.0726	-	0.0664	-
COLG	0.0723	0.0801	0.0472	0.0797	0.0768	0.0650
HEDE	0.0813	0.0575	0.0579	0.1283	0.0616	0.0672
HERB	0.0755	0.0772	0.0521	0.0755	0.0632	0.0807
SHRE	0.0663	0.0784	0.0654	-	0.0496	-
ALGA	0.0860	0.0653	0.0789	0.0879	0.0666	0.0611
DIAT	0.0903	0.0516	0.0473	0.0666	0.0769	0.1110
LEAF	0,0915	0,0741	0.0746	0.0754	0.0586	-
POM	0.0821	0.0763	0.0702	0.0813	0.0512	0.0690
TERR	0.0748	0.0940	0.0654	0.0759	0.0763	0.0556
HUMW	-	-	-	-	-	0.0543
FILA	-	-	0.0813	0.0625	0.0511	0.1126

Table 6: The table shows I_H(V) index which quantifies community importance based on the influence of dynamical variability of each group in the 6 sites (in blue OMNI which present an increase in abundance in all the sites especially in site 6, LEAF, DIAT, ALGA, POM which decrease in abundance from upstream to downstream the river).

In figure 19 the graph shows the results curves obtained from the I_H(V) index measure, showed in table 6, for all the functional groups in the 6 sites. We can observe that the community sensitivity in terms of dynamical variability is dramatically increasing from site 1, which is a quasi-natural locations, towards sites where the human influence is more predominant.

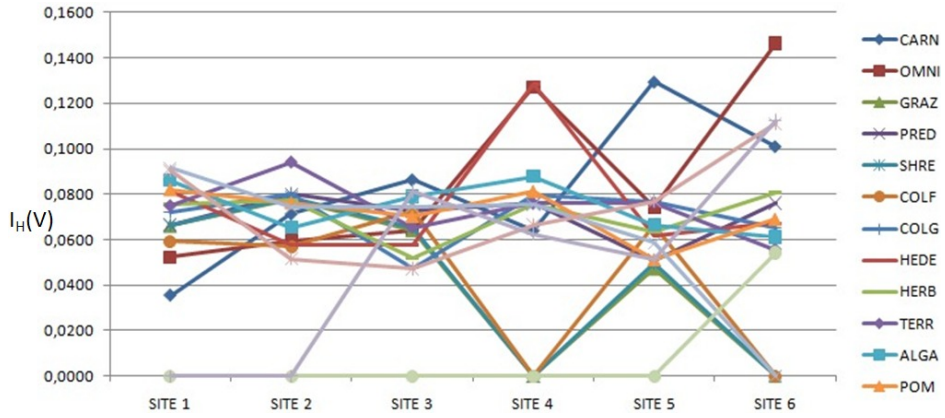


Figure 19: Community importance series of the variance (I_H(V)) of each trophic group computed for the six sites. In the axes are represented the six sites and in the ordinate I_H(V) values.

The primary producers (LEAF, DIAT, ALGA, POM) show decrease in abundance from upstream to downstream the river and especially in sites 2 and 6 where the human impacts are stronger. I_H(V) index suggests that disturbing the primary producers will generate changes in the behaviour of the other groups. From the quantities of omnivores (OMNI) (0.0524, 0.0595, 0.0642, 0.1271, 0.0744, 0.1463) in table 6 is visible an increase in abundance in all the sites and in site 6 they are more present(0.1463). From these values we deduce that the dynamical variability of the human-influenced river ecosystem is more sensitive to changes of the omnivores population (fig. 20).

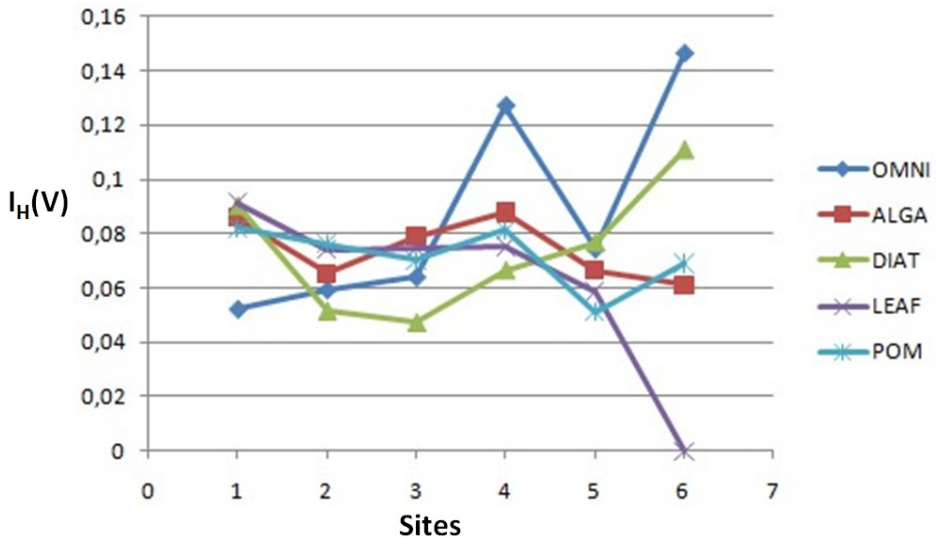


Figure 20: Through the community importance series of the variance $I_H(V)$ in the graph we can observe how the curves of primary producers (LEAF, DIAT, ALGA, POM) decrease in abundance; the omnivores (OMNI) increase in abundance. In the axes are represented the six sites and in the ordinate $I_H(V)$ values.

5.7 Conclusion

The mine activity and human settlements with their discarded materials have affected trophic interactions of the food webs of the river with consequences for the benthic flora and filter-feeding invertebrates.

The goal of the study presented in this chapter was to study the influence of humans on population dynamics of the Kelian river. In order to do that, we build a stochastic model of six food webs (representing different sites along the river). We used the model to run simulations and to perform sensitivity analyses to get some insights in the dynamic behaviour of species in the river.

Using dynamical simulations we aim to analyse the functional diversity of ecosystem at the level of functional groups. From our results we can infer that the invertebrate shredders (SHRE) are indicators of human impact on the river. The role of shredders in food web development is very important [35], because breaking down leaves into smaller particles they supply food for other organisms as collector gatherers and filters. In sites that are located downstream along the river, the vegetation is more sparse, so there is a decrease supply in leaf litter. The primary producers as diatoms (DIAT), algae (ALGA), filamentous bacteria (FILA) disappear downstream likely because of the excess of sediments that is the consequence of mine activity and of the presence of human waste. This effect can be mostly seen in site 6, which is the most polluted site. The grazers (GRAZ) are less important downstream, since other groups as fish omnivores (OMNI) and carnivores (CARN) tolerate better the human influence on the river and their variety is less strong than the one of invertebrate groups.
What we can conclude from this is that in the past before human impacts on the river, the organisms present in the 6 sites must be in similar abundance and diversity. As the years passed, in sites downstream along the river, the community changed its composition due to the effect of pollution disturbance. Our analysis shows that the species that are most affected by the human impact on the river are invertebrates and our analysis confirmed their key role in the aquatic ecosystem that we analyzed [79].

6 Case study II

6.1 Fishing by FADs on tunas, Gulf of Guinea

Tunas are fishes from the Scombridae family. They live in warm salt water and represent a sleek and streamlined body. Their size ranges from 50 cm long and 1.8 kg of weight, to 4.6 m and 684 kg of weight (e.g. atlantic bluefin tuna). They are agile predators, amongst the fastest swimming, with an unique respiratory system that allow them to maintain a body temperature higher than the surrounding water. Tuna species has great commercial importance. In the scientific report of International Seafood Sustainability Foundation (ISSF) on the state of global tuna stocks in 2009 [52] the most important for commercial fishing activity are yellowfin (*Thunnus albacares*), bluefin (*T. thynnus, T. orientalis, and T. macoyii*), skipjack (*Katsuwonus pelamis*), bigeye (*T. obesus*) and albacore (*T. alalunga*). In 2011 the catch of tunas was 4.22 million tonnes; 57% of it was skipjack tuna, 26% yellowfin, 10% bigeye, 5% albacore and bluefin tuna accounts for only 1% of the global catch [45].
In this thesis work the focus is in fishing activity by fishers using or not FADs on skipjack tuna (in the model referred as "small tuna") in Eastern Atlantic Ocean, especially in South Sherbro in the Gulf of Guinea.

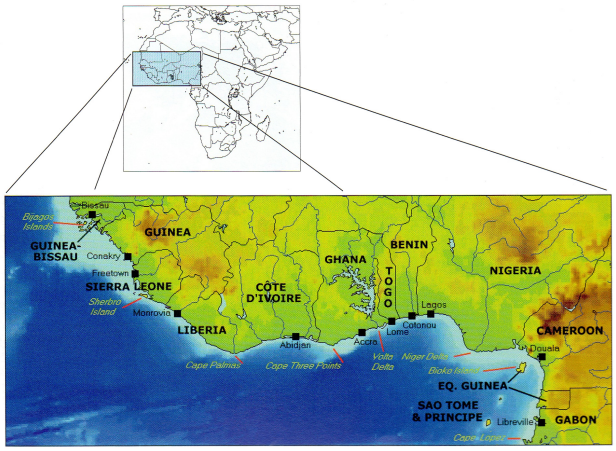


Figure 21: Map of Gulf of Guinea

Skipjack tuna is an epipelagic species inhabiting open waters with temperatures ranging from 15°C to 30°C [26]. Its geographical limits are 55°-60°N and 45°-50°S [44]. During the entire year they inhabit the region of the equator and the region of tropics during the warm season. In the Atlantic Ocean skipjack tuna species distribution interests the eastern Atlantic from Ireland to South Africa, and in the western Atlantic from Canada to northern Argentina. It is an opportunistic predator and its feeding habits are based on fish, crustaceans, cephalopods and molluscs [74]. In the Eastern Atlantic ocean is reported in [44, 59] the diet of skipjack tuna is principally based on *Vinciguerria nimbaria* and cephalopods. This species can live 8 to 12 years [72] and the maximum size reached is 80 cm [11]. Skipjack is the most fecund between the different tuna species: from the age of one it spawns opportunistically throughout the year and in vast sectors of the ocean. For this reason its population is considered sustainable against the current consumption on it. Growth varies according to latitude [38].

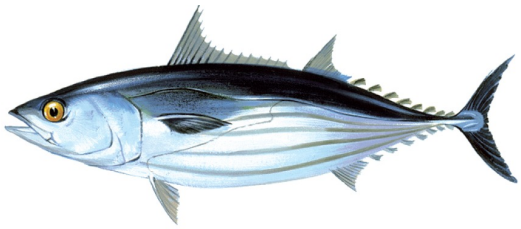


Figure 22: The image represents a drawing of an adult skipjack tuna (*Katsuwonus pelamis*).

The fishing activity on skipjack is done using almost exclusively surface gears throughout the Atlantic, mainly by baitboat and purse seine vessels and a small numbers of them are consequence of incidental longline catches.

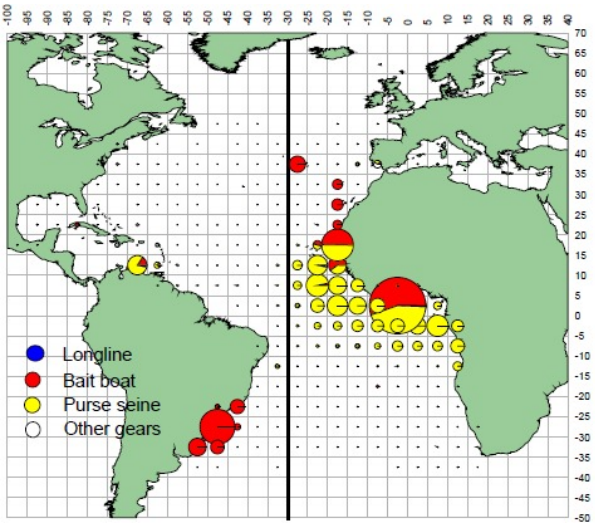


Figure 23: The map is representation of geographical distribution of skipjack catches by principal gears (ICCAT Secretariat [44]).

Since the early 1990s the typical target tropical tuna purse seine fisheries are large yellowfin and bigeye (*Thunnus obesus*) tunas on free-swimming schools, skipjack and juveniles of yellowfin and bigeye associated with artificial drifting fish aggre-

gating devices (FADs) [23, 12]. Tuna stocks catches represents nearly half of all principal market. Nowadays fishing operations on tuna schools associated with drifting FADs became widespread in the Eastern Tropical Atlantic [80]. In the early 1990s, fishing operations on tuna schools associated with drifting FADs became widespread in the Eastern Tropical Atlantic [80]. From ICCAT [44] we can read that the percentage of skipjack tuna catched under FADs reaches 90%, with only a small 10% catched using other methods. The ICCAT report is focused especially on South Gulf of Guinea area more important for fishing made by the use of drifting FADs (see figure 24).

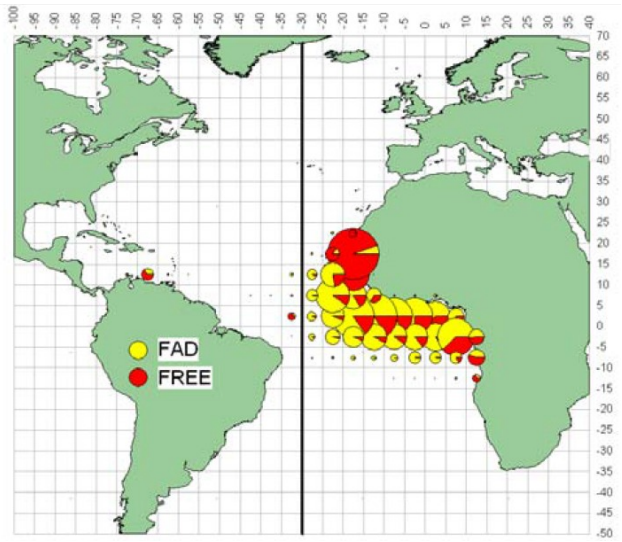


Figure 24: Skipjack tuna catches in free schools and under FADS, 1991-2006.

Skipjacks tend to form schools independently or in association with floating objects (e.g. FADs), marine animals or seamounts. The tendency to associate with floating objects of any kind is not necessarily correlated to trophic purpose. During the night small tunas congregate beneath the object and during

the day they spread out to feed, normally on *V. nimbaria* (in the eastern Atlantic), a species not associated with objects [59]. The hypothesis is that the floating objects affect the dynamic of migration, feeding strategy and population structure especially of small tuna [54]. For this reason the capture of young stocks could have repercussion on population and future breeding potential of tuna population. The catches are based on 80 cm skipjack size (8-10 kg). Countries recording large amounts of skipjack catches include the Maldives, France, Spain, Malaysia, Sri Lanka, and Indonesia. The increasing development of FADs has raised the question of the impact on tuna communities, on the biology (food intake, growth rate, plumpness of the fish) and on the ecology (displacement rate, movement orientation) of skipjack and yellowfin (ecological trap concept) [54, 38]. However, the consequences of this fishing strategy on the skipjack population and on the ecosystem are difficult to assess.
Our aim is to investigate the impact of FADs on skipjack tuna communities using a network model to simulate the marine food web.
We present a stochastic food web model that combines interactions of tuna with FADs and predator-prey behaviour with different species. Tunas associated with FADs change their interaction rates w.r.t. their behaviour as free individuals. We performed stochastic simulations and sensitivity analysis on this dynamical system and determined the dependencies between various trophic components and as last we build the scatter search algorithm for parameter estimation in order to have optimization of the marine dynamic model.
In the following sections we are going to present our model and the results obtained through its analysis. In section 6.2 we in-

troduce information about the data and the skipjack diet [59]. In section 6.3 we describe the dynamic model of marine ecosystem after follows section 6.4 that describe the results obtained, after using for the optimization of parameters, from sensitivity analyses and finally section 6.5 with the conclusions.

6.2 Data set and food webs of marine ecosystem

The information found in [59] helped us to build the food web and the stochastic model of skipjack tuna referring to South Sherbro area (0°-5° N and 10°-20° W) in the Equatorial Atlantic. The trophic flows information in the paper may be considered as a general description of skipjack tuna feeding habits in the Atlantic ocean. The data about skipjack tuna preys are obtained from stomach analyses contents collected in 1995, 1996, 1997, and 1998 from tuna caught during daylight hours. Preys were subdivided in six major categories represented by the fish that constituted the dominant phylum. We use some of them to construct the stochastic model (see table 7).

Names	categories with the dominant phylum
VNIMB	*V. nimbaria*: Photichthyidae
EPIPLFISH	epipelagic fish: Balistidae, Clupeidae, Diretmidae and Exocetidae
CRUST	planktonic crustaceans: Euphausiaceae and natantia Decapods
CEPHAL	cephalopods: squids of the Teuthoidae family
OTHER	other prey: undetermined pulp, tunicates such as Salpidae

Table 7: Description of major categories for skipjack tuna diet preference in the marine stochastic model.

We used these categories to build a small food web, composed by 7 nodes: SMALLTUNA (skipjack tuna), VNIMB, EPIPLFISH, CRUST, CEPHAL, OTHER and Fisher (FADs are

included in the model and interact with skipjack tuna with specific rate; they permit to the Fisher to catch tuna with different rates basing on the cases that they use FADs on tuna or not). We represent the different states of tuna free and trapped so at the end are described two networks based on skipjack tuna diet preference while it is free from FADs (see fig. 25 or when it migrates under FADs (fig.26).

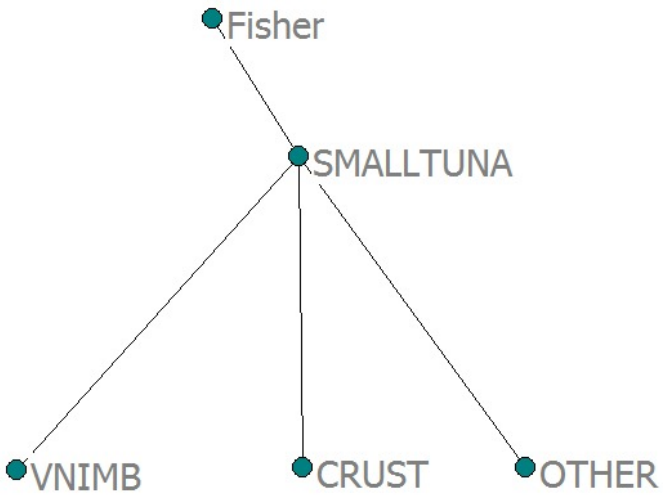


Figure 25: The image represents small tuna as top predator, its preys at the bottom of the food web *V. nimbaria*, crustacean and other organisms and Fisher that catch small tuna at the top of the food web.

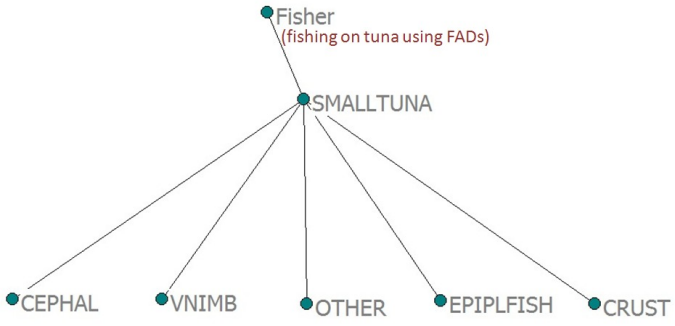


Figure 26: The image represents small tuna as predator, at the bottom of food web its preys *V. nimbaria*, crustacean cephalopods, epipelagic fish, and other organisms and at the top of food web is described fishing operations by Fisher on tuna schools associated with drifting FADs.

The diet preference of skipjack tuna is different related to the fact that it can be in free state or associated with FADs [59]. In the first representation SMALLTUNA is caught by Fisher without the use of FADs, and SMALLTUNA hunts its preys VNIMB, CRUST, other. In figure 26, SMALLTUNA is predator of VNIMB, EPIPLFISH, CEPHAL, CRUST and OTHER. It is caught by Fisher through the use of FADs. In case of skipjack tuna is trapped under FADs in food web is called small_tunaTrapped in the model, in case it is free small_tuna.
Based on data of E. Chassot and to [62] the drifting FADs varying in average between 3.000 and 10.000 at sea depending on the season and the average vessels in Atlantic Ocean are 48. We used data from 2010 because is the current status referring to the period of our work and also because the research on FADs quantitative is very recent. The drifting FADs can include both natural and artificial man-made FADs. The purse seine fisheries are very dynamic and the number of FADs has strongly varied over years (there were already some radar-tracked artificial FADs in the early 1980s). Major technological changes have

occurred over the last decades and deeply modified the way of using and tracking FADs equipped with satellite buoys. The strategy of seeding FADs is very seasonal [57, 58]. During the spawning season, the vessels mainly target schools of large yellowfin and seed less FADs. The information on the number of FADs as today is very sparse. Information is obtained in different ways:

- from the vessel-based quarterly declarations of number active FADs in the Atlantic Ocean for one French fishing company in 2010
- from the vessel-based quarterly declarations of number active FADs in the Indian Ocean for another French fishing company in 2010
- from the fleet-based quarterly declarations of FADs deployed at-sea in the Indian Ocean for the Spanish fleet in 2010,
- some totally empirical estimates for the Ghanaian fleet, for which even the number of active purse seiners is poorly known.

This does not include all drifting FADs for which the buoy is not active anymore (deactivated by the fishermen, exhausted battery, no emission, etc.).
The information about population size are obtained translating biomass (B_i) data to number of individual. Data about the individual number of skipjack tuna in Atlantic ocean are obtained from ICCAT document [44, 8]. To obtain the individual number we use the general mean weight mw_i (expressed in kg) for every individual species, the biomass B_i is converted from tonnes in

kilogram then is divided by the mw_i and all is multiplied by area of Sherbro in South of Guinea Gulf (the average weight of skipjack caught by the European is 2.5 kg with sizes between 30 cm and 65 cm, the annual catch 100 000 ton) and the area.

Data about tuna catches are taken in ICCAT document: they state that tuna caught under FADs are 90% while the remaining 10% is caught with other methods. The data about caught by fisher and entrapment are obtained from [40]. The information regarding proportion of feeding process of skipjack tuna on its preys is obtained from [59] (already cited at the beginning of this section). The reproduction and death rates are obtained from [18] studies. The number of individual for tunas is around 1 000 000. We scaled it down to 100 in order to use this data in the stochastic framework. We did this for the preys and FADs and fisher also. For the other parameters values (reproduction and death rates) needed in the stochastic simulation we used some fitting strategies to align our model to the existing data using a scatter search algorithm to find the best solution. From the data mentioned above we constructed the table 8 containing information about population size, reproduction and death, feeding rates.

	species						
Parameters	small tuna	smalltuna trap.	epiplfish	Vnimb	Cephal	Crust	Other
Population size	100		1000	1000	10000	10000	10000
birth	6.8	5	10	10	51.5	51.5	51.5
death	2	0.4	6.5	6.5	47.5	47.5	47.5
Predator consumption (food web)							
tuna trapped		C_{ss} 0.063	C_{sv} 0.52	C_{scp} 0.22	C_{scr} 0.027	C_{so} 0.17	
tuna free		Cf_{sv} 0.521		Cf_{scr} 0.028	Cf_{so} 0.451		
Fishing strategies							
Fisher	10						
FADs	65						
becoming free rate	0.3						
human catch rate with FADs	0.4						
human catch rate no FADs	0.09						

Table 8: Information of population size of skipjack tuna in case is free or trapped under FADs, preys, Fisher and FADs amount; follows birth and death rates; feeding rates for tuna on its preys in the case tuna is trapped under FADs or in the case tuna is free (if smalltuna is free it feeds on epipelagic fish, *V. nimbaria*, Cephalopods and Crustaceans called in the table C_{ss}, C_{sv}, C_{scp}, C_{so}; if smalltuna is trapped the feeding interactions happen between *V. nimbaria* described as Cf_{sv} ,crustaceans described as Cf_{scr} and other Cf_{so}), and finally the proportion of tuna caught by Fisher using or not FADS.

6.3 Stochastic food web model of marine ecosystem

As in the case study of Kelian river in section 5.5, the stochastic framework that we used is BlenX. For detailed explanation of the language, we refer the reader to section 3.1. In Figure 27, we show an intuitive representation of the main elements of the model: fisher that catches both free and tunas trapped under the FAD (at different rates) and tunas that eat food (at different rates if trapped or free). Preys of tunas are different in the case when they are trapped w.r.t. the free species (see 27).

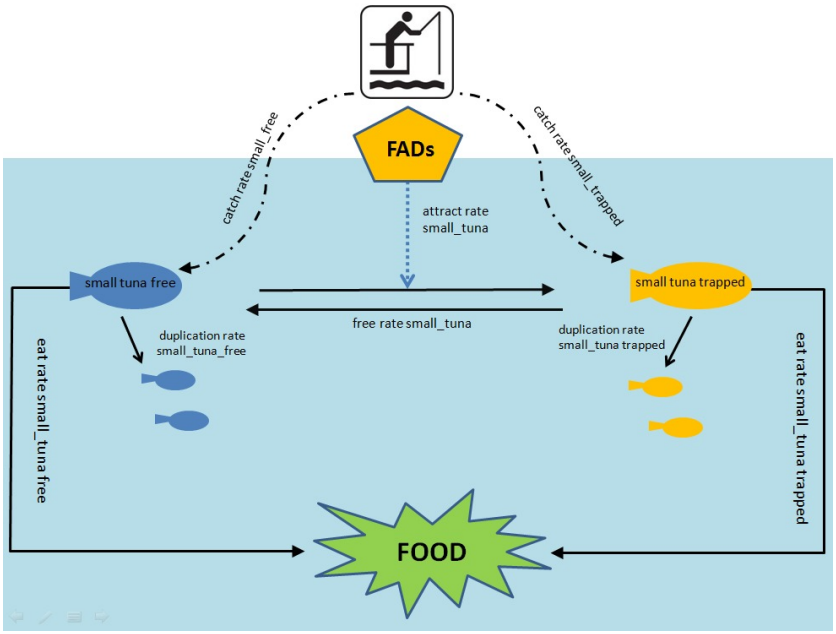


Figure 27: The image represents small tuna as top predator, its preys called FOOD, the possibility to migrate from free to trapped states and vice versa, the possibility of skipjack tuna to be caught from Fisher small_tuna in both cases, when it is free or through the use of FADs.

The natural processes of eating, dying, duplicating and hunting are the same of the ones explained in section 5.5. The

only difference is that small_tuna can be in two different states: trapped under FADs or free from them. Using different rates for the two situations, allows to give to the two states different behaviour and to distinguish the predator-prey interactions in the presence of the free state or in that of trapped.

- trapped: tuna is trapped under FADs when the action on the trapped channel is fired. This triggers a change (*ch*) in the state of the binders of the box to implement the fact that now new rates of interactions will be used by the box to communicate with other boxes in the system (e.g. the trapped box will not be able to communicate with the FAD box any more). All the numerical values for those rates (both of the free and trapped state) are listed in the BlenX *.type* file.

- eating: the eating action is triggered by a firing on the eat channel. After this happens, two alternative sequences are possible: 1) do not do anything specific and go back in the initial state, 2) duplicate with a specific rate. The duplication is guarded by an **if-then** statement that allows to follow those different paths with different rates depending on the state (free/trapped) of the box itself.

- catching: the catching of a fish box happens when a communication happens on the catch channel. This represent the fact that the Fisher catches the tuna, and the tuna box will be deleted from the systems (through a die action).

- natural death: in the case none of the previous actions happens, the tuna box has the chance of following its natural death process, and will die at a specific rate, different for

the case of free/trapped state. This difference is implemented through the usage of if-then statement that check which is the state of the current box.

- dissociation: a trapped tuna can gain back its freedom if the action with the SmallTuna_becomeFree rate happens. In this case all the binders of the box are switched back to the free state and the evolution of the box in time will now use the rates of the free state.

Below are shown parts of the actual code of the BlenX files that defines the model (e.g. the description of internal processes of small_tuna, declaration of boxes, the split event; description of some of its preys when skipjack tuna is under drifting FADs or in case it is free). The full code of the model is available in the Appendix A.8.
The internal process of small_tuna is composed by seven subprocesses linked by *sum (+)* operator described in file *.prog*. Below follows the short part of the code starting with the time steps to run the simulations; follows the declaration of skipjack tuna, after the processes of Fishers and FADs: the first just catch the tuna and the second attract tuna with specific rates described in file *.types* (see section 3.1 for the detail about BlenX functionality).

```
// .PROG FILE

[steps = 1200, delta = 0.001]

//TOP PREDATOR PROCESS

let pSmallTuna: pproc = eat!().y!().nil
+
eat!().(if(not(trapped,SmallTuna\_trappedInFads))
then
```

```
(ch(rate(SmallTunaRep),dupl,duplication).nil)
endif
+
if(trapped,SmallTuna\_trappedInFads)
then
(ch(rate(SmallTunaRep\_trapped),dupl,duplication).nil)
endif
+
catch?().die(inf)
+
trapped?().ch(inf,trapped,SmallTuna_trappedInFads).
ch(inf,catch,SmallTuna_Humancatch_trapped).
ch(inf,eat,SmallTuna_hunts_trapped).y!().nil
+
if(not(trapped,SmallTuna_trappedInFads))
then
(delay(rate(SmallTunaDie)).die(inf).nil)
endif
+
if(trapped,SmallTuna_trappedInFads)
then
(delay(rate(SmallTunaDie_trapped)).die(inf).nil)
endif
+
if(trapped,SmallTuna_trappedInFads)
then
(ch(rate(SmallTuna_becomeFree),trapped,SmallTuna_freeFromFads).
ch(inf,catch,SmallTuna_Humancatch).
ch(inf,eat,SmallTuna_hunts).y!().nil)
endif


//Fisher:
let pFishers: pproc =   catch!().y!().nil;

//FADs
let pFADs: pproc = attract!().y!().nil;
```

In this model is new the use of the **if-then** conditional expression followed by logical operator **not**. The conditional expression are used in order to describe the migration of tuna from free state to entrapment state under drifting FADs. This action is described in the model by the use of parameters in file *.types* and

in the file *.func* and by processes that permit with *change* action happening through communication of binders. The Fisher with the *catch!* action communicates with *catch?* declared in the small_tuna internal process. In the file *.types* are represented the Fisher-tuna interaction for the two different conditions: tuna caught using FADs and tuna in case are free. The FADs with the output channel *attract!*, receive the signal sent from small_tuna input channel *trapped?* with the consequences of the entrapment of tuna under FADs. Fisher and FADs do not duplicate or die, their individual numbers are constant. In files *.types* are described the interaction FADs-tuna to become trapped (see the short code that follows below).

```
// .TYPES FILE

(SmallTuna_freeFromFads,attr_Fish,0.7),

(purseseine_All,SmallTuna_Humancatch,0.09),
(purseseine_All,SmallTuna_Humancatch_trapped,0.4)
```

The preys represent the same internal processes as those described in section 5.5.2. The difference with the model in section 5.5 is the possibility of skipjack tuna to hunt in free state or in trapped state once associated with FADs (in the code the use of binders permit the interaction with the predator small_tuna in both the cases when it is trapped called *SmallTuna_hunts_trapped* and when it is free called *SmallTuna_hunts*). In file *.types* are represented these two different predator-prey interactions.

```
// .TYPES FILE

(SmallTuna_hunts,hunts_Vnimb,0.23),
(SmallTuna_hunts,hunts_Crust,0.3),
```

```
(SmallTuna_hunts_trapped,hunts_Vnimb,0.13),
(SmallTuna_hunts_trapped,hunts_Crust,0.24),
```

The parameter that describe the case that small_tuna can migrate from free state to the trapped under drifting FADs is declared in file *.func*.

```
//.FUNC FILE

let SmallTunaRep : const = 0.8;
let SmallTunaDie : const = 0.6;
let SmallTuna_becomeFree : const = 0.3;
let SmallTunaDie_trapped : const = 0.4;
let SmallTunaRep_trapped : const = 0.7;
```

The split event for duplication and die action for tuna and the preys are the same as described for Kelian model (section 5.5).

```
//SPLIT EVENT

when (SmallTunadup: :inf) split (SmallTuna, SmallTuna)
when (Vnimbdup: :inf) split (Vnimb, Vnimb);
```

The box of *small_tuna* is composed by four binder sites (*eat, dupl, trapped, catch*) and an internal process as follows:

```
//BOX
let SmallTuna: bproc = #(eat,SmallTuna_hunts),
                       #(dupl:0,A),
                       #(trapped,SmallTuna_freeFromFads),
                       #(catch, SmallTuna_Humancatch)
                       [rep y?().pSmallTuna | pSmallTuna];
```

Finally at the bottom of *.prog* file with the **run** command the simulations are executed.

6.4 Results

In this case study we constructed food webs for a marine ecosystem with information of feeding interactions extracted from [59]. The food webs describe the different states of skipjack tuna in two different cases: free or aggregated with drifting FADs. We build a model in which we describe, through different interaction rates, the changes of the tuna from free to trapped and their different behavior with FADs in these two states. The code for the model is reported in Appendix A.8. In order to use the stochastic framework in an effective way we scaled down the prey's population size, the amount of vessels and FADs; we also appropriately scaled reproduction and death rate for the different entities of the model. The stochastic approach allows us to understand the effect of the noise on smaller population's size. The small hypothetical population used here refer to skipjack tuna behavior that uses to aggregate in small classes under a single FAD [9]. The results obtained are referred to a higher geographical scale area, the Gulf of Guinea area.
We used scatter search algorithm for optimization of the dynamic model (for more detail see section 3.2). Figure 28 shows a typical time series obtained simulating the BlenX model reported in Appendix A.8. Below follows the plot obtained by the simulations done in BlenX.

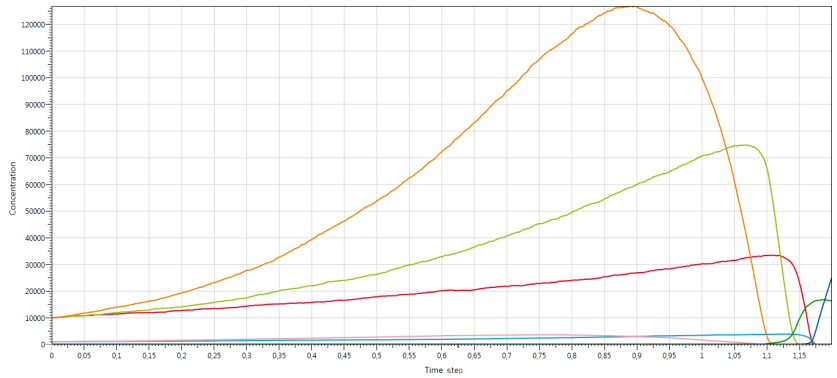


Figure 28: The plot is obtained from dynamical simulations of the BlenX model of skypjack tuna. We found in the graph the lines positioned as: in pink VNIMB (fourth line), OTHER in orange (first line), CEPHAL in red (third line), CRUST in green (second line), SMALLTUNA in green (the line is positioned near to the axes), EPIPLFISH in blue (fifth line)

In a first moment after the simulations, SMALLTUNA population size does not grow up. The reason is that in the model SMALLTUNA eats and after this process with a specific rate can reproduce. Growing up the population size of the preys SMALLTUNA amount increase too.
In order to further analyze the behavior of the model under different conditions, we connected a Python script to the Beta Workbench, to run different simulations and perform a sensitivity analysis study (for more detail see section 3.3). We run simulations with different initial values and calculate the statistical properties (mean and variance) based on a certain number of simulations (reference). Afterwards, we perturbed the system by halving of a given species [51]. With dynamical simulations and sensitivity analysis the main aim is to quantify the community response to the changes happened in the networks as we did for the river ecosystem (section 5.6). Table 9 summarizes the results of the sensitivity analysis through the use of I_H(M)

and I_H(V) indexes.

	I_H(M)	I_H(V)
SMALLTUNA	0.104	0.105
VNIMB	0.118	0.115
EPIPLFISH	0.138	0.156
CEPHAL	0.34	0.135
CRUST	0.197	0.365
OTHER	0.103	0.125

Table 9: The table shows the community importance series I_H(M) and I_H(V) of each individual species. The individual species which are hunted with a high feeding rate both in case tuna is under FADs or free show lower I_H(M) and I_H(V) indexes comparing to the individual species which are hunted only in case tuna is aggregated around FADs.

The results based on community importance series of the mean I_H(M) show that Cephalopods (CEPHAL 0.34) are in leading position (small tuna in trapped state feeds mainly in *V. nimbaria* and cephalopods [58, 53]).
Other prey (OTHER) shows low I_H(M) index (0.103) probably affected by the high pressure of predation of SMALLTUNA on them (skipjack tuna hunts them in both the states: free and trapped). The skipjack tuna has a low I_H(M) index (SMALLTUNA 0.104) probably due to the effects of fishing strategies: fisher catch them using or not the drifting FADs. Other prey (OTHER), *V. nimbaria* (VNIMB) and crustaceans (CRUST) are hunted from tuna both when tuna is free and trapped [59, 58]. Epipelagic fish and crustaceans show average importance (indexes 0.138 and 0.197) and are hunted by predator tuna during the trapped state [59].
Observing the results obtained from the community importance measures focusing on dynamical variability I_H(V) the dominant role is of Crustaceans (CRUST 0.365). VNIMB and OTHER

(0.115 and 0.125) show low $I_H(V)$ indexes after SMALLTUNA (0.105). Cephalopods are of average importance with values of 0.135.
In figure 29 is shown the graph obtained from $I_H(M)$ and $I_H(V)$ indexes for all the individual species.

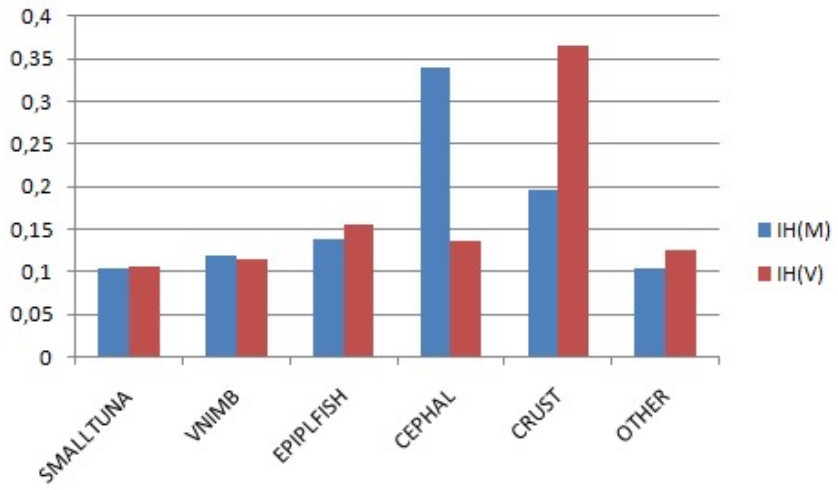


Figure 29: The plot is obtained from dynamical simulations for marine ecosystems. In blue is represented the $I_H(M)$ index and the red the $I_H(V)$ one; in axes are the species and in ordinate the indexes values. SMALLTUNA for both the community importance measures shows the lowest positions in the graph; CEPHAL presents the largest community effect $I_H(M)$ and CRUST the largest $I_H(V)$; EPIPLFISH has a middle importance for the $I_H(V)$ and $I_H(M)$ indexes.

6.5 Conclusions

The purse seine started fishing on drifting FADs in the early nineties. This method, based on the behaviour of tuna to aggregate around natural logs, developed worldwide catching yearly over one million tons of tunas [22]. Tunas that use to aggregate around FADs are small size (less than 70 cm, e.g. skipjack

tuna). The eastern part of the Gulf of Guinea, the area that we treat in this work thesis, is enriched by seasonal coastal upwellings from July to September and debris are deposited here. Recently is observed decrease in size of tunas caught with FADs in the Atlantic ocean. The hypothesis is that drifting FADs affect negatively the growth and the natural mortality of juvenile tunas under FADs [54].
Our results suggest that SMALLTUNA is a clear indicator of effects of fishing strategy by FADs. Tuna in the food web is the predator of the other individual species, which in less presence of tuna increase in abundance. The population size of some individual species as *V. nimbaria* and OTHER prey which are hunted from tuna in both its states (free or trapped) are more sensible to the variability of population size in the model. This probably depends from the diet preference of tuna on them [59]. From [58, 53] work we know that *V. nimbaria* is the main prey hunted of all small tunas, deductible from interactions rates 0.521 and 0.52 described in file *.types* too (see Appenidx A.8).
Cephalopods present the largest community effect I_H(M) perturbing them the community gives a strong response. These species are considered important components of most marine food webs and in some case they may play an indirect role in facilitating prey capture to secondary predators [76].
Observing the high I_H(V) index of crustaceans which quantifies community importance based on the influence on dynamical variability means that these species tolerate better the changes of population size of small tuna related to the effects of fishing strategies. These category is frequently present in diet preference of tuna but in low concentration [59].
The indices of dynamical community importance may help in bi-

ology conservation and may allow to understand that the massive deployment of FADs is detrimental for tuna's population and may help to give guidelines for future sustainable management strategies of tuna fisheries.

7 Conclusions

To analyze the dynamics of complex systems we used food webs and ecological networks models. Food webs represent interaction between species and their study is useful because it allows to investigate the structure of the ecosystem and its dynamical behavior upon internal/external changes. Models are essential tools to study the stability of food webs and to help explaining and predict their behavior under different conditions. Population-dynamical food-web models describe the trend of species densities change in a community over time according to the trophic interactions (as well as non-trophic and abiotic effects e.g. the use of FADs by fisher to catch skipjack tuna in the Gulf of Guinea) [68, 20].
In our thesis work building stochastic models and we performed sensitivity analyses on aquatic ecosystems, strongly impacted by humans, we aim to explore the role of functional diversity in communities highlighting the importance of the species in the food web. We use a process algebra-based language (called BlenX) for modelling to investigate natural interactions e.g. predator-prey interaction (ecological entities are represented by boxes and the interactions by affinity relation), simulating parallel and concurrent interactions with a modular approach. This stochastic model helps on studying ecological interactions especially in presence of small population size where studying environmental noise and variability can be of major importance [48].
In our two case studies performing sensitivity analyses we measured after disturbance average and variance responses in order to quantify community importance of species. In the first case study we investigate the different roles of trophic groups in the six sites of Kelian river. We found out that some trophic groups

are more vulnerable to human influence as invertebrate shredders (SHRE), considered as indicator of disturbance of human presence disappearing in the most affected sites (4 and 6), primary producers and grazers decrease in abundance downstream, while other as carnivore and omnivore fish show to tolerate pollution [46].
In the second case study the use of stochastic model is used to analyze the effects of fishing with FADs on juvenile tuna (skipjack tuna) on Gulf of Guinea. The effects of FADs on the fishing consequences are difficult to evaluate precisely [24] and from some recent studies emerges that aggregation of juvenile tunas under these devices would be detrimental for their biological characteristics (e.g. the use of FADs may alter the natural movements, may affect negatively the growth and the natural mortality of small tuna and may migrate less to the productive coastal areas) [54, 53]. With sensitivity analyse we investigated the community importance of tuna and the effects of FADs on them. We used the scatter search algorithm for parameter estimation that better fit the marine dynamic model. From the results obtain we understand that SmallTuna population size is affected from teh use of FADs.
In this thesis work we used stochastic simulations and sensitivity analyses in order to study individuals and local, parallel processes on the aquatic ecosystems. We investigated dynamic community importance of particular trophic groups trying to give a tool that may contribute to quantitative conservation biology and give useful guidelines for future sustainable management strategies of tuna fisheries.
Interesting future development of this work involve a more complex stochastic model (already existing) of tuna fishing through

the use of FADs. From the work cited in Section 6.3 the growth from small tuna to large individual may be included in the model to better model what happens in the real aquatic system. It seems that small tuna move under FADs considering them as refuge, while large tuna use them for trophic reason [59]. The stochastic model may help to understand these different behavior of small and large tuna related to FADs.

Acknowledgements

As the end of the thesis and my PhD period has come I would like to do some acknowledgements to the place of work COSBI and people that make this possible as my supervisors Corrado Priami and Ferenc Jordan.
Here I met people that have a very important role in my PhD formation and realization, apart my supervisors, and so a very special thank goes to them: but the most special goes to Alida Palmisano, I found in her not only a good colleague but also a special friend. With a lot of patient she assisted me in every moment and gave a lot of great advice in my work and helped in correcting my thesis.
I would to thank Alessandro Romanel apart his vocal talent for giving me some advice in BlenX permitting me to proceed with my work, the same Michele Forlin and Davide Prandi. I am very grateful to these kind people who help me a lot during my PhD period although they left COSBI the first year of my PhD. I would to thank Marco Scotti that gave to me some helpful advice for the final exam. Thank you to other ex COSBI people as Roberto Larcher, Roberto Valentini, Lorenzo Dematté, Tommaso Mazza, Irene Preti, Bianca Baldacci, Michele Di Cosmo, Chiara Damiani and the old PhD students that began with me hoping they are getting well.

A thank goes to the people that I knew at Ifremer institute, France, where I spend my period aboard permitting and assisting me to work on skipjack tuna and the effect of FADs on them: Sibylle Dueri, Christian Mullon, Frédéric Menard.

Thank you to my master's degree supervisor Fausto Tinti for the master thesis who did a great work with me finishing it (I did not say to you that time how grateful was to you) and thank you for counting on you once again.

At the end of these acknowledgements a very special thank goes to my parents that without them it would be hard to finish in time this thesis work, to my partner and above all to my wonderful daughter that made me happy and laugh every moment.

Per concludere "tutto incerto a questo mondo" cit. della persona pi saggia, generosa e alla quale devo molto.

References

[1] Alan A Berryman. The orgins and evolution of predator-prey theory. *Ecology*, pages 1530–1535, 1992.

[2] Leonora Bianchi, Marco Dorigo, Luca Maria Gambardella, and Walter J Gutjahr. A survey on metaheuristics for stochastic combinatorial optimization. *Natural Computing: an international journal*, 8(2):239–287, 2009.

[3] Ottar N Bjørnstad, Jean-Marc Fromentin, Nils Chr Stenseth, and Jakob Gjøsæter. Cycles and trends in cod populations. *Proceedings of the National Academy of Sciences*, 96(9):5066–5071, 1999.

[4] Daniel I Bolnick, Priyanga Amarasekare, Márcio S Araújo, Reinhard Bürger, Jonathan M Levine, Mark Novak, Volker HW Rudolf, Sebastian J Schreiber, Mark C Urban, and David A Vasseur. Why intraspecific trait variation matters in community ecology. *Trends in ecology & evolution*, 26(4):183–192, 2011.

[5] Ulrich Brose, Eric L Berlow, and Neo D Martinez. Scaling up keystone effects from simple to complex ecological networks. *Ecology Letters*, 8(12):1317–1325, 2005.

[6] Jason Brownlee. *Clever algorithms: nature-inspired programming recipes*. Jason Brownlee, 2011.

[7] Guido Caldarelli, Paul G Higgs, and Alan J McKane. Modelling coevolution in multispecies communities. *arXiv preprint adap-org/9801003*, 1998.

[8] Manuela Capello, Marc Soria, Pascal Cotel, Gaël Potin, Laurent Dagorn, and Pierre Fréon. The heterogeneous spatial and temporal patterns of behavior of small pelagic fish in an array of fish aggregating devices (fads). *Journal of Experimental Marine Biology and Ecology*, 430:56–62, 2012.

[9] José J Castro, José A Santiago, and Ana T Santana-Ortega. A general theory on fish aggregation to floating objects: an alternative to the meeting point hypothesis. *Reviews in fish biology and fisheries*, 11(3):255–277, 2001.

[10] James S Clark. Beyond neutral science. *Trends in Ecology & Evolution*, 24(1):8–15, 2009.

[11] Bruce B Collette, Cornelia E Nauen, et al. *FAO species catalogue. Volume 2. Scombrids of the world. An annotated and illustrated catalogue of tunas, mackerels, bonitos and related species known to date.* Number 125. 1983.

[12] Laurent Dagorn, Kim N Holland, Victor Restrepo, and Gala Moreno. Is it good or bad to fish with fads? what are the real impacts of the use of drifting fads on pelagic marine ecosystems? *Fish and Fisheries*, 2012.

[13] Donald L DeAngelis and Wolf M Mooij. Individual-based modeling of ecological and evolutionary processes. *Annual Review of Ecology, Evolution, and Systematics*, pages 147–168, 2005.

[14] Lorenzo Dematté, Roberto Larcher, Alida Palmisano, Corrado Priami, and Alessandro Romanel. Programming biology in blenx. In *Systems Biology for Signaling Networks*, pages 777–820. Springer, 2010.

[15] Lorenzo Dematté, Corrado Priami, and Alessandro Romanel. The beta workbench: a computational tool to study the dynamics of biological systems. *Briefings in Bioinformatics*, 9(5):437–449, 2008.

[16] Barbara Drossel, Paul G Higgs, and Alan J McKane. The influence of predator–prey population dynamics on the long-term evolution of food web structure. *Journal of Theoretical Biology*, 208(1):91–107, 2001.

[17] Barbara Drossel, Paul G Higgs, and Alan J McKane. The influence of predator–prey population dynamics on the long-term evolution of food web structure. *Journal of Theoretical Biology*, 208(1):91–107, 2001.

[18] Sibylle Dueri, Blaise Faugeras, and Olivier Maury. Modelling the skipjack tuna dynamics in the indian ocean with apecosm-e: Part 1. model formulation. *Ecological Modelling*, 245:41–54, 2012.

[19] Jennifer A Dunne. The network structure of food webs. *Ecological networks: linking structure to dynamics in food webs*, pages 27–86, 2006.

[20] Jennifer A Dunne. The network structure of food webs. *Ecological networks: linking structure to dynamics in food webs*, pages 27–86, 2006.

[21] Alan Feest, Timothy D Aldred, and Katrin Jedamzik. Biodiversity quality: a paradigm for biodiversity. *Ecological Indicators*, 10(6):1077–1082, 2010.

[22] Alain Fonteneau, Javier Ariz, Daniel Gaertner, Viveca Nordstrom, and Pilar Pallares. Observed changes in the

species composition of tuna schools in the gulf of guinea between 1981 and 1999, in relation with the fish aggregating device fishery. *Aquatic Living Resources*, 13(4):253–257, 2000.

[23] Alain Fonteneau, Emmanuel Chassot, and Nathalie Bodin. Global spatio-temporal patterns in tropical tuna purse seine fisheries on drifting fish aggregating devices (dfads): Taking a historical perspective to inform current challenges. *Aquatic Living Resources*, 26(1):37–48, 2013.

[24] Alain Fonteneau, Didier Gascuel, and Pilar Pallares. Vingt-cino ans d'evaluation des ressources thonieres de l'atlantique quelques reflexions methodologiques. *Collective volume of scientific papers-international commission for the conservation of atalntic tunas*, 50:523–562, 1998.

[25] Michele Forlin. *Knowledge discovery for stochastic models of biological systems*. PhD thesis, University of Trento, 2010.

[26] Rainer Froese and Daniel Pauly. Fishbase, 2010.

[27] Ryo Fujikura. *Supporting Pollution Controls and Sustainable Environmental Monitoring*. PhD thesis, Tokyo University of Agriculture and Technology, 2004.

[28] L Gasche, D Gascuel, L Shannon, and Y-J Shin. Global assessment of the fishing impacts on the southern benguela ecosystem using an ecotroph modelling approach. *Journal of Marine Systems*, 90(1):1–12, 2012.

[29] Didier Gascuel and Daniel Pauly. Ecotroph: modelling marine ecosystem functioning and impact of fishing. *Ecological Modelling*, 220(21):2885–2898, 2009.

[30] Daniel T Gillespie. Exact stochastic simulation of coupled chemical reactions. *The journal of physical chemistry*, 81(25):2340–2361, 1977.

[31] Fred Glover. Heuristics for integer programming using surrogate constraints. *Decision Sciences*, 8(1):156–166, 1977.

[32] Fred Glover, Manuel Laguna, and Rafael Martí. Fundamentals of scatter search and path relinking. *Control and cybernetics*, 39(3):653–684, 2000.

[33] Fred Glover, Manuel Laguna, and Rafael Martí. Scatter search. In *Advances in evolutionary computing*, pages 519–537. Springer, 2003.

[34] David Goldberg. Genetic algorithms in optimization, search and machine learning. *Addison Wesley, New York. Eiben AE, Smith JE (2003) Introduction to Evolutionary Computing. Springer. Jacq J, Roux C (1995) Registration of non-segmented images using a genetic algorithm. Lecture notes in computer science*, 905:205–211, 1989.

[35] Nicholas J Gotelli and Aaron M Ellison. Food-web models predict species abundances in response to habitat change. *PLoS biology*, 4(10):e324, 2006.

[36] Volker Grimm, Uta Berger, Finn Bastiansen, Sigrunn Eliassen, Vincent Ginot, Jarl Giske, John Goss-Custard,

Tamara Grand, Simone K Heinz, Geir Huse, et al. A standard protocol for describing individual-based and agent-based models. *Ecological modelling*, 198(1):115–126, 2006.

[37] Rudiyanto Gunawan, Yang Cao, Linda Petzold, and Francis J Doyle III. Sensitivity analysis of discrete stochastic systems. *Biophysical Journal*, 88(4):2530–2540, 2005.

[38] Jean-Pierre Hallier, Daniel Gaertner, et al. Drifting fish aggregation devices could act as an ecological trap for tropical tuna species. *Marine Ecology Progress Series*, 353:255–264, 2008.

[39] Ralph Hamann. Kelian equatorial mining, indonesia: mine closure. *Putting Partnerships to Work: Strategic Alliances for Development between Gover*, 1(74):138–150, 2004.

[40] John Hampton. Natural mortality rates in tropical tunas: size really does matter. *Canadian Journal of Fisheries and Aquatic Sciences*, 57(5):1002–1010, 2000.

[41] MP Hassell and HN Comins. Discrete time models for two-species competition. *Theoretical Population Biology*, 9(2):202–221, 1976.

[42] FREDERIC L Holmes. Antoine lavoisier and hans krebs: Two styles of scientific creativity. *Creative people at work. Oxford University Press, New York*, pages 44–68, 1989.

[43] Stuart H Hulbert. Functional importance vs keystoneness: reformulating some questions in theoretical biocenology. *Australian Journal of Ecology*, 22(4):369–382, 1997.

[44] IEO. Iccat manual, chapter 2.1.3:skipjack tuna. 2006.

[45] D.C. USA. International Seafood Sustainability Foundation, Washington. Issf tuna stock status update - 2013 (2). *ISSF Technical Report 2013-04A. year=2013*.

[46] Ferenc Jordán, Nerta Gjata, Shu Mei, and Catherine M Yule. Simulating food web dynamics along a gradient: Quantifying human influence. *PloS one*, 7(7):e40280, 2012.

[47] Ferenc Jordán and István Scheuring. Network ecology: topological constraints on ecosystem dynamics. *Physics of Life Reviews*, 1(3):139–172, 2004.

[48] Ferenc Jordán, Marco Scotti, and Corrado Priami. Process algebra-based computational tools in ecological modelling. *Ecological Complexity*, 8(4):357–363, 2011.

[49] Russell Lande. Genetics and demography in biological conservation. *Science(Washington)*, 241(4872):1455–1460, 1988.

[50] Simon A Levin. Ecosystems and the biosphere as complex adaptive systems. *Ecosystems*, 1(5):431–436, 1998.

[51] Carmen Maria Livi, Ferenc Jordán, Paola Lecca, and Thomas A Okey. Identifying key species in ecosystems with stochastic sensitivity analysis. *Ecological Modelling*, 222(14):2542–2551, 2011.

[52] Jacek Majkowski. *Global fishery resources of tuna and tuna-like species*. Number 483. Food & Agriculture Org., 2007.

[53] Emile Marchal and Anne Lebourges. Acoustic evidence for unusual diel behaviour of a mesopelagic fish (vinciguerria nimbaria) exploited by tuna. *ICES Journal of Marine Science: Journal du Conseil*, 53(2):443–447, 1996.

[54] Francis Marsac, Alain Fonteneau, and Frédéric Ménard. Drifting fads used in tuna fisheries: an ecological trap? In *Pêche thonière et dispositifs de concentration de poissons, Caribbean-Martinique, 15-19 Oct 1999*, 2000.

[55] Rafael Marti, Manuel Laguna, and Fred Glover. Principles of scatter search. *European Journal of Operational Research*, 169(2):359–372, 2006.

[56] E Meijaard and V Nijman. Primate hotspots on borneo: predictive value for general biodiversity and the effects of taxonomy. *Conservation Biology*, 17(3):725–732, 2003.

[57] F Ménard, A Hervé, and A Fonteneau. An area of high seasonal concentrations of tunas: 2-4 n 10-20°w. the site of the picolo programme. 1998.

[58] Frédéric Ménard, Alain Fonteneau, Daniel Gaertner, V Nordstrom, Bernard Stéquert, and Emile Marchal. Exploitation of small tunas by a purse-seine fishery with fish aggregating devices and their feeding ecology in an eastern tropical atlantic ecosystem. *ICES Journal of Marine Science: Journal du Conseil*, 57(3):525–530, 2000.

[59] Frédéric Ménard, Bernard Stéquert, Alex Rubin, Miguel Herrera, and Émile Marchal. Food consumption of tuna in the equatorial atlantic ocean: Fad-associated versus unassociated schools. *Aquatic living resources*, 13(4):233–240, 2000.

[60] Richard W Merritt and Kenneth W Cummins. *An introduction to the aquatic insects of North America*. Kendall Hunt, 1996.

[61] L Scott Mills, Michael E Soulé, and Daniel F Doak. The keystone-species concept in ecology and conservation. *BioScience*, 43(4):219–224, 1993.

[62] Gala Moreno, Laurent Dagorn, Gorka Sancho, and David Itano. Fish behaviour from fishers' knowledge: the case study of tropical tuna around drifting fish aggregating devices (dfads). *Canadian Journal of Fisheries and Aquatic Sciences*, 64(11):1517–1528, 2007.

[63] Arnold Neumaier, Oleg Shcherbina, Waltraud Huyer, and Tamás Vinkó. A comparison of complete global optimization solvers. *Mathematical programming*, 103(2):335–356, 2005.

[64] Eugene Pleasants Odum and Gary W Barrett. Fundamentals of ecology. 1971.

[65] Mercedes Pascual. Computational ecology: from the complex to the simple and back. *PLoS computational biology*, 1(2):e18, 2005.

[66] Daniel Pauly, Villy Christensen, and Carl Walters. Ecopath, ecosim, and ecospace as tools for evaluating ecosystem impact of fisheries. *ICES Journal of Marine Science: Journal du Conseil*, 57(3):697–706, 2000.

[67] Andrew Phillips and Luca Cardelli. A correct abstract machine for the stochastic pi-calculus. *Electronic Notes in Theoretical Computer Science*, 2004.

[68] Stuart L Pimm. *Food webs*. Springer, 1982.

[69] Craig R Powell and Richard P Boland. The effects of stochastic population dynamics on food web structure. *Journal of theoretical biology*, 257(1):170–180, 2009.

[70] J. H. Powell and R. E. Powell. Assessment of aquatic impacts associated with gold mining operations in the prampus region of the kelian river system, east kalimantan, indonesia. a review 19901993. 1993.

[71] Corrado Priami and Paola Quaglia. Beta binders for biological interactions. In *Computational Methods in Systems Biology*, pages 20–33. Springer, 2005.

[72] Froese R. and D. Pauly. Fishbase. world wide web electronic publication. www.fishbase.org, version (06/2006). 2006.

[73] Maria Rodriguez-Fernandez, Jose A Egea, and Julio R Banga. Novel metaheuristic for parameter estimation in nonlinear dynamic biological systems. *BMC bioinformatics*, 7(1):483, 2006.

[74] C Roger and E Marchal. Mise en evidence de conditions favorisant labondance des albacores, thunnus albacares, et des listaos, katsuwonus pelamis, dans latlantique equatorial est. *Collect. Vol. Sci. Pap, ICCAT*, 42(2):237–248, 1994.

[75] Thomas P. Runarsson and Xin Yao. Stochastic ranking for constrained evolutionary optimization. *Evolutionary Computation, IEEE Transactions on*, 4(3):284–294, 2000.

[76] MJ Smale. Cephalopods as prey. iv. fishes. *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences*, 351(1343):1067–1081, 1996.

[77] S Suresh, PB Sujit, and AK Rao. Particle swarm optimization approach for multi-objective composite box-beam design. *Composite Structures*, 81(4):598–605, 2007.

[78] Thomas E Walton, Priya Mathur, Toru Uemachi, et al. Indonesia environment monitor 2003. *Special Focus: Reducing Pollution. Washington, DC, Jakarta: The World Bank*, 2003.

[79] Edward O Wilson. The little things that run the world (the importance and conservation of invertebrates), 1987.

[80] A Xavier, A Delgado, A Fonteneau, C Gonzales, F Pilar, and P Pallares. Logs and tunas in the eastern tropical atlantic: A review of present knowledge and uncertainties. *Inter-American tropical tuna commission, La Jolla, CA*, 1992.

[81] Catherine M Yule, Luz Boyero, and Richard Marchant. Effects of sediment pollution on food webs in a tropical river (borneo, indonesia). *Marine and Freshwater Research*, 61(2):204–213, 2010.

[82] CM Yule. The impact of sediment pollution on the benthic invertebrate fauna of the kelian river, east kalimantan, indonesia. *Tropical limnology.*, 3:61–75, 1995.

A Appendix

A.1 BlenX models for Kelian river

A.1.1 Site 1: .prog file

Figure 3: The stochastic model done in BlenX of site 1; .prog file.

```
[steps=120]
<<BASERATE : inf>>

// ---------- PROCESS ----------

let palga: pproc= ch(rate(algaRep),dupl,duplication) +
    food?().die(inf) + delay(rate(algaDie)).die(inf);
let pcarn: pproc= eat!().y!().nil +
    eat!().ch(rate(carnRep),dupl,duplication).nil +
    delay(rate(carnDie)).die(inf).nil;
let pcolf: pproc=
    food!().x!().nil+food!().ch(rate(colfRep),dupl,duplication).nil
    + eat?().die(inf)+delay(rate(colfDie)).die(inf).nil;
let pcolg: pproc=
    food!().x!().nil+food!().ch(rate(colgRep),dupl,duplication).nil
    + eat?().die(inf)+delay(rate(colgDie)).die(inf).nil;
let pdiat: pproc= ch(rate(diatRep),dupl,duplication) +
    food?().die(inf) + delay(rate(diatDie)).die(inf);
let phede: pproc=
    food!().x!().nil+food!().ch(rate(hedeRep),dupl,duplication).nil
    + eat?().die(inf)+delay(rate(hedeDie)).die(inf).nil;
let pherb: pproc=
    food!().x!().nil+food!().ch(rate(herbRep),dupl,duplication).nil
    + eat?().die(inf)+delay(rate(herbDie)).die(inf).nil;
let pgraz: pproc= food!().x!().nil+food!().ch(rate(grazRep)
    ,dupl,duplication).nil +
    eat?().die(inf)+delay(rate(grazDie)).die(inf).nil;
let pleaf: pproc= ch(rate(leafRep),dupl,duplication) +
    food?().die(inf) + delay(rate(leafDie)).die(inf);
let pomni: pproc=
    food!().x!().nil+food!().ch(rate(omniRep),dupl,duplication).nil
    + eat?().die(inf)+delay(rate(omniDie)).die(inf).nil;
let pPOM: pproc= ch(rate(POMRep),dupl,duplication) +
    food?().die(inf) + delay(rate(POMDie)).die(inf);
let ppred: pproc=
    food!().x!().nil+food!().ch(rate(predRep),dupl,duplication).nil
```

```
   + eat?().die(inf)+delay(rate(predDie)).die(inf).nil;
let pshre: pproc=
   food!().x!().nil+food!().ch(rate(shreRep),dupl,duplication).nil
   + eat?().die(inf)+delay(rate(shreDie)).die(inf).nil;
let pterr: pproc= ch(rate(terrRep),dupl,duplication) +
   food?().die(inf) + delay(rate(terrDie)).die(inf);

// ---------- BOXES ----------

// top predator
let Alga: bproc = #(food,alga_lifes),#(dupl:0,A) [palga];
let Carn: bproc=#(eat,carn_hunts),#(dupl:0,A) [rep y?().pcarn |
   pcarn];
let Colf: bproc=
   #(eat,hunts_colf),#(food,colf_lifes),#(dupl:0,A) [rep
   x?().pcolf | pcolf];
let Colg : bproc =
   #(eat,hunts_colg),#(food,colg_lifes),#(dupl:0,A) [rep
   x?().pcolg | pcolg];
let Diat: bproc = #(food,diat_lifes),#(dupl:0,A) [pdiat];
let Graz: bproc =
   #(eat,hunts_graz),#(food,graz_lifes),#(dupl:0,A) [rep
   x?().pgraz| pgraz];
let Hede : bproc =
   #(eat,hunts_hede),#(food,hede_lifes),#(dupl:0,A) [rep
   x?().phede | phede];
let Herb: bproc =
   #(eat,hunts_herb),#(food,herb_lifes),#(dupl:0,A)[rep x?().
   pherb | pherb];
let Leaf: bproc = #(food,leaf_lifes),#(dupl:0,A) [pleaf];
let Omni: bproc =
   #(eat,hunts_omni),#(food,omni_lifes),#(dupl:0,A) [rep
   x?().pomni| pomni];
let POM: bproc = #(food,pom_lifes),#(dupl:0,A) [pPOM];
let Pred: bproc=
   #(eat,hunts_pred),#(food,pred_lifes),#(dupl:0,A) [rep
   x?().ppred| ppred];
let Shre: bproc =
   #(eat,hunts_shre),#(food,shre_lifes),#(dupl:0,A) [rep
   x?().pshre| pshre];
let Terr: bproc = #(food,terr_lifes),#(dupl:0,A) [pterr];

let Humw: bproc=#(food,humw_lifes),#(dupl:0,A) [nil];
let Fila: bproc=#(food,fila_lifes),#(dupl:0,A) [nil];
```

```
// ---------------- DUPLICATION ----------------------

let Algadup: bproc=#(food,alga_lifes),#(dupl:0,duplication)
    [nil];
let Carndup: bproc=#(eat,carn_hunts),#(dupl:0,duplication) [rep
    y?().pcarn];
let Colfdup: bproc =
    #(eat,hunts_colf),#(food,colf_lifes),#(dupl:0,duplication)
    [rep x?().pcolf];
let Colgdup: bproc =
    #(eat,hunts_colg),#(food,colg_lifes),#(dupl:0,duplication)
    [rep x?().pcolg];
let Diatdup: bproc=#(food,diat_lifes),#(dupl:0,duplication)
    [nil];
let Grazdup: bproc =
    #(eat,hunts_graz),#(food,graz_lifes),#(dupl:0,duplication)
    [rep x?().pgraz];
let Hededup: bproc =
    #(eat,hunts_hede),#(food,hede_lifes),#(dupl:0, duplication)
    [rep x?().phede];
let Herbdup: bproc =
    #(eat,hunts_herb),#(food,herb_lifes),#(dupl:0, duplication)
    [rep x?().pherb];
let Leafdup: bproc=#(food,leaf_lifes),#(dupl:0,duplication)
    [nil];
let Omnidup: bproc =
    #(eat,hunts_omni),#(food,omni_lifes),#(dupl:0, duplication)
    [rep x?().pomni];
let POMdup: bproc=#(food,pom_lifes),#(dupl:0,duplication) [nil];
let Preddup: bproc =
    #(eat,hunts_pred),#(food,pred_lifes),#(dupl:0,duplication)
    [rep x?().ppred];
let Shredup: bproc =
    #(eat,hunts_shre),#(food,shre_lifes),#(dupl:0, duplication)
    [rep x?().pshre];
let Terrdup: bproc=#(food,terr_lifes),#(dupl:0,duplication)
    [nil];

let Filadup: bproc=#(food,fila_lifes),#(dupl:0,duplication)
    [nil];
let Humwdup: bproc=#(food,humw_lifes),#(dupl:0,duplication)
    [nil];

// ---------------- CONDITIONS ----------------------
```

```
// Duplication
when (Algadup: :inf) split (Alga, Alga);
when (Carndup: :inf) split (Carn, Carn);
when (Colfdup: :inf) split (Colf, Colf);
when (Colgdup: :inf) split (Colg, Colg);
when (Diatdup: :inf) split (Diat, Diat);
when (Hededup: :inf) split (Hede, Hede);
when (Herbdup: :inf) split (Herb, Herb);
when (Grazdup: :inf) split (Graz, Graz);
when (Leafdup: :inf) split (Leaf, Leaf);
when (Omnidup: :inf) split (Omni, Omni);
when (POMdup: :inf) split (POM, POM);
when (Preddup: :inf) split (Pred, Pred);
when (Shredup: :inf) split (Shre, Shre);
when (Terrdup: :inf) split (Terr, Terr);

when (Filadup: :inf) split (Fila, Fila);
when (Humwdup: :inf) split (Humw, Humw);

// Prey
when (Alga::rate(algaWhen)) new (1);
when (Diat::rate(diatWhen)) new (1);
when (Leaf::rate(leafWhen)) new (1);
when (POM::rate(POMWhen)) new (1);
when (Terr::rate(terrWhen)) new (1);

// ---------- STARTING ----------

run 500 Alga || 5 Carn || 212 Colf || 385 Colg || 500 Diat ||
    1085 Graz || 10 Hede || 10 Herb || 1000 Leaf || 10 Omni ||
    1000 POM || 63 Pred || 87 Shre || 54 Terr || 0 Humw || 0 Fila
```

A.1.2 Site 1: .types file

Figure 4: The stochastic model done in BlenX of site 1; .types file.

```
{duplication, A, carn_hunts, hunts_pred, pred_lifes,
    hunts_graz, graz_lifes, hunts_colg, colg_lifes, hunts_omni,
    omni_lifes, hunts_herb, herb_lifes, hunts_colf, colf_lifes,
    hunts_shre, shre_lifes, hunts_hede, hede_lifes, alga_lifes,
    diat_lifes, terr_lifes, leaf_lifes, pom_lifes, humw_lifes,
    fila_lifes}
%%
{
//------------TOP PREDATOR-------------

        (carn_hunts, hunts_pred, 0.111),
        (carn_hunts, hunts_graz, 0.111),
        (carn_hunts, hunts_colf, 0.111),
        (carn_hunts, hunts_colg, 0.111),
        (carn_hunts, hunts_omni, 0.111),
        (carn_hunts, hunts_shre, 0.111),
        (carn_hunts, terr_lifes, 0.111),
        (carn_hunts, hunts_herb, 0.111),
        (carn_hunts, hunts_hede, 0.111),

//------------PREY & PREDATOR-------------

        (omni_lifes, hunts_pred, 0.111),
        (omni_lifes, hunts_colg, 0.111),
        (omni_lifes, hunts_graz, 0.111),
        (omni_lifes, hunts_colf, 0.111),
        (omni_lifes, hunts_shre, 0.111),
        (omni_lifes, terr_lifes, 0.111),
        (omni_lifes, alga_lifes, 0.111),
        (omni_lifes, pom_lifes, 0.111),
        (omni_lifes, diat_lifes, 0.111),
//-----------------------------------------
        (pred_lifes, hunts_graz, 0.25),
        (pred_lifes, hunts_colg, 0.25),
        (pred_lifes, hunts_colf, 0.25),
        (pred_lifes, hunts_shre, 0.25),
//-----------------------------------------
        (hede_lifes, diat_lifes, 0.25),
        (hede_lifes, pom_lifes, 0.25),
        (hede_lifes, leaf_lifes, 0.25),
        (hede_lifes, alga_lifes, 0.25),
```

```
//-------------------------------------------------
        (herb_lifes, diat_lifes, 0.333),
        (herb_lifes, alga_lifes, 0.333),
        (herb_lifes, pom_lifes, 0.333),
//-------------------------------------------------
        (graz_lifes, alga_lifes, 0.07),
        (graz_lifes, diat_lifes, 0.196),
        (graz_lifes, pom_lifes, 0.734),
//-------------------------------------------------
        (colg_lifes, pom_lifes, 1.00),
//-------------------------------------------------
        (colf_lifes, pom_lifes, 1.00),
//-------------------------------------------------
        (shre_lifes, leaf_lifes, 1.00)

}
```

A.1.3 Site 1: .func file

Figure 5: The stochastic model done in BlenX of site 1; .func file.

```
let carnRep : const = 50;
let carnDie : const = 0.5;
let predRep : const = 6300;
let predDie : const = 6.3;
let omniRep : const = 100;
let omniDie : const = 1;
let herbRep : const = 1000;
let herbDie : const = 1;
let hedeRep : const = 1000;
let hedeDie : const = 1;
let grazRep : const = 108500;
let grazDie : const = 1.085;
let colgRep : const = 385000;
let colgDie : const = 0.385;
let colfRep : const = 214000;
let colfDie : const = 0.214;
let shreRep : const = 114000;
let shreDie : const = 0.114;
let algaRep : const = 500;
let algaDie : const = 500;
let algaWhen : const = 5000;
let diatRep : const = 500;
let diatDie : const = 500;
let diatWhen : const = 5000;
let terrRep : const = 540;
let terrDie : const = 5.4;
let terrWhen : const = 54;
let leafRep : const = 10000;
let leafDie : const = 10000;
let leafWhen : const = 100000;
let POMRep : const = 10000;
let POMDie : const = 10000;
let POMWhen : const = 100000;
```

A.2 BlenX model for site 2

A.2.1 .prog file

Figure 6: The stochastic model done in BlenX of site 2; .prog file.

```
[steps=120]
<<BASERATE : inf>>

// ---------- PROCESS ----------

let palga: pproc= ch(rate(algaRep),dupl,duplication) +
    food?().die(inf) + delay(rate(algaDie)).die(inf);
let pcarn: pproc= eat!().y!().nil +
    eat!().ch(rate(carnRep),dupl,duplication).nil +
    delay(rate(carnDie)).die(inf).nil;
let pcolf: pproc=
    food!().x!().nil+food!().ch(rate(colfRep),dupl,duplication).nil
    + eat?().die(inf)+delay(rate(colfDie)).die(inf).nil;
let pcolg: pproc=
    food!().x!().nil+food!().ch(rate(colgRep),dupl,duplication).nil
    + eat?().die(inf)+delay(rate(colgDie)).die(inf).nil;
let pdiat: pproc= ch(rate(diatRep),dupl,duplication) +
    food?().die(inf) + delay(rate(diatDie)).die(inf);
let pgraz: pproc= food!().x!().nil+food!().ch(rate(grazRep)
    ,dupl,duplication).nil +
    eat?().die(inf)+delay(rate(grazDie)).die(inf).nil;
let phede: pproc=
    food!().x!().nil+food!().ch(rate(hedeRep),dupl,duplication).nil
    + eat?().die(inf)+delay(rate(hedeDie)).die(inf).nil;
let pherb: pproc=
    food!().x!().nil+food!().ch(rate(herbRep),dupl,duplication).nil
    + eat?().die(inf)+delay(rate(herbDie)).die(inf).nil;
let pleaf: pproc= ch(rate(leafRep),dupl,duplication) +
    food?().die(inf) + delay(rate(leafDie)).die(inf);
let pomni: pproc=
    food!().x!().nil+food!().ch(rate(omniRep),dupl,duplication).nil
    + eat?().die(inf)+delay(rate(omniDie)).die(inf).nil;
let pPOM: pproc= ch(rate(POMRep),dupl,duplication) +
    food?().die(inf) + delay(rate(POMDie)).die(inf);
let ppred: pproc=
    food!().x!().nil+food!().ch(rate(predRep),dupl,duplication).nil
    + eat?().die(inf)+delay(rate(predDie)).die(inf).nil;
let pshre: pproc=
    food!().x!().nil+food!().ch(rate(shreRep),dupl,duplication).nil
```

```
   + eat?().die(inf)+delay(rate(shreDie)).die(inf).nil;
let pterr: pproc= ch(rate(terrRep),dupl,duplication) +
   food?().die(inf) + delay(rate(terrDie)).die(inf);

// ---------- BOXES ----------

let Alga: bproc = #(food,alga_lifes),#(dupl:0,A) [ palga ];
let Carn: bproc=#(eat,carn_hunts),#(dupl:0,A)[ rep y?().pcarn |
   pcarn ];
let Colf: bproc=
   #(eat,hunts_colf),#(food,colf_lifes),#(dupl:0,A) [ rep
   x?().pcolf | pcolf];
let Colg : bproc =
   #(eat,hunts_colg),#(food,colg_lifes),#(dupl:0,A) [ rep
   x?().pcolg | pcolg];
let Diat: bproc = #(food,diat_lifes),#(dupl:0,A) [ pdiat ];
let Graz: bproc =
   #(eat,hunts_graz),#(food,graz_lifes),#(dupl:0,A) [ rep
   x?().pgraz| pgraz ];
let Hede : bproc =
   #(eat,hunts_hede),#(food,hede_lifes),#(dupl:0,A) [ rep
   x?().phede | phede ];
let Herb: bproc =
   #(eat,hunts_herb),#(food,herb_lifes),#(dupl:0,A) [ rep x?().
   pherb | pherb];
let Leaf: bproc = #(food,leaf_lifes),#(dupl:0,A) [ pleaf ];
let Omni: bproc =
   #(eat,hunts_omni),#(food,omni_lifes),#(dupl:0,A) [ rep
   x?().pomni| pomni ];
let POM: bproc = #(food,pom_lifes),#(dupl:0,A) [ pPOM ];
let Pred: bproc=
   #(eat,hunts_pred),#(food,pred_lifes),#(dupl:0,A) [ rep
   x?().ppred| ppred];     // [((rep
   x?().food!().x!().nil+food!().ch(0.2166,dupl,duplication).nil
   + eat?().die(inf)+delay(0.01).die(inf).nil;)
   |food!().x!().nil+food!().ch(0.2166,dupl,duplication).nil +
   eat?().die(inf)+delay(0.01).die(inf).nil)]
let Shre: bproc =
   #(eat,hunts_shre),#(food,shre_lifes),#(dupl:0,A) [ rep
   x?().pshre| pshre ];
let Terr: bproc = #(food,terr_lifes),#(dupl:0,A) [ pterr ];

let Fila: bproc=#(food,fila_lifes),#(dupl:0,A) [nil];
let Humw: bproc=#(food,humw_lifes),#(dupl:0,A) [nil];
```

```
// ---------------- DUPLICATION ----------------------

let Algadup: bproc=#(food,alga_lifes),#(dupl:0,duplication) [
    nil ];
// Top predator
let Carndup: bproc=#(eat,carn_hunts),#(dupl:0,duplication) [
    rep y?().pcarn ];
let Colfdup: bproc =
    #(eat,hunts_colf),#(food,colf_lifes),#(dupl:0,duplication) [
    rep x?().pcolf ];
let Colgdup: bproc =
    #(eat,hunts_colg),#(food,colg_lifes),#(dupl:0,duplication) [
    rep x?().pcolg ];
let Diatdup: bproc=#(food,diat_lifes),#(dupl:0,duplication) [
    nil ];
let Grazdup: bproc =
    #(eat,hunts_graz),#(food,graz_lifes),#(dupl:0,duplication) [
    rep x?().pgraz];
let Hededup: bproc =
    #(eat,hunts_hede),#(food,hede_lifes),#(dupl:0, duplication)
    [ rep x?().phede ];
let Herbdup: bproc =
    #(eat,hunts_herb),#(food,herb_lifes),#(dupl:0, duplication)
    [ rep x?().pherb ];
let Leafdup: bproc=#(food,leaf_lifes),#(dupl:0,duplication) [
    nil ];
let Omnidup: bproc =
    #(eat,hunts_omni),#(food,omni_lifes),#(dupl:0, duplication)
    [ rep x?().pomni];
let POMdup: bproc=#(food,pom_lifes),#(dupl:0,duplication) [ nil
    ];
let Preddup: bproc =
    #(eat,hunts_pred),#(food,pred_lifes),#(dupl:0,duplication) [
    rep x?().ppred];
let Shredup: bproc =
    #(eat,hunts_shre),#(food,shre_lifes),#(dupl:0, duplication)
    [ rep x?().pshre ];
let Terrdup: bproc=#(food,terr_lifes),#(dupl:0,duplication) [
    nil ];

let Filadup: bproc=#(food,fila_lifes),#(dupl:0,duplication) [
    nil ];
let Humwdup: bproc=#(food,humw_lifes),#(dupl:0,duplication) [
    nil ];
```

```
// ———————— CONDITIONS ————————————————

// Duplication
when (Algadup: :inf) split (Alga, Alga);
when (Carndup: :inf) split (Carn, Carn);
when (Colfdup: :inf) split (Colf, Colf);
when (Colgdup: :inf) split (Colg, Colg);
when (Diatdup: :inf) split (Diat, Diat);
when (Grazdup: :inf) split (Graz, Graz);
when (Hededup: :inf) split (Hede, Hede);
when (Herbdup: :inf) split (Herb, Herb);
when (Leafdup: :inf) split (Leaf, Leaf);
when (Omnidup: :inf) split (Omni, Omni);
when (POMdup: :inf) split (POM, POM);
when (Preddup: :inf) split (Pred, Pred);
when (Shredup: :inf) split (Shre, Shre);
when (Terrdup: :inf) split (Terr, Terr);

when (Filadup: :inf) split (Fila, Fila);
when (Humwdup: :inf) split (Humw, Humw);

when (Alga::rate(algaWhen)) new (1);
when (Diat::rate(diatWhen)) new (1);
when (Leaf::rate(leafWhen)) new (1);
when (POM::rate(POMWhen)) new (1);
when (Terr::rate(terrWhen)) new (1);

// ———————— STARTING ————————————————

run 500 Alga || 5 Carn || 200 Colf || 383 Colg || 500 Diat ||
    1288 Graz || 10 Hede || 10 Herb || 1000 Leaf || 10 Omni ||
    1000 POM || 66 Pred || 27 Shre || 54 Terr || 0 Humw || 0 Fila
```

A.2.2 .types file

Figure 7: The stochastic model done in BlenX of site 2; .types file.

```
{duplication, A, carn_hunts, hunts_pred, pred_lifes,
    hunts_graz, graz_lifes, hunts_colg, colg_lifes, hunts_omni,
    omni_lifes, hunts_herb, herb_lifes, hunts_colf, colf_lifes,
    hunts_shre, shre_lifes, hunts_hede, hede_lifes, alga_lifes,
    diat_lifes, terr_lifes, leaf_lifes, pom_lifes, humw_lifes,
    fila_lifes}
%%
{
//------------TOP PREDATOR-----------------

        (carn_hunts, hunts_pred, 0.111),
        (carn_hunts, hunts_graz, 0.111),
        (carn_hunts, hunts_colf, 0.111),
        (carn_hunts, hunts_colg, 0.111),
        (carn_hunts, hunts_omni, 0.111),
        (carn_hunts, hunts_shre, 0.111),
        (carn_hunts, terr_lifes, 0.111),
        (carn_hunts, hunts_herb, 0.111),
        (carn_hunts, hunts_hede, 0.111),

//------------PREY & PREDATOR--------------

        (omni_lifes, hunts_pred, 0.125),
        (omni_lifes, hunts_colg, 0.125),
        (omni_lifes, hunts_graz, 0.125),
        (omni_lifes, hunts_colf, 0.125),
        (omni_lifes, hunts_shre, 0.125),
        (omni_lifes, terr_lifes, 0.125),
        (omni_lifes, alga_lifes, 0.125),
        (omni_lifes, diat_lifes, 0.125),
//-----------------------------------------
        (pred_lifes, hunts_graz, 0.25),
        (pred_lifes, hunts_colg, 0.25),
        (pred_lifes, hunts_colf, 0.25),
        (pred_lifes, hunts_shre, 0.25),
//-----------------------------------------
        (hede_lifes, diat_lifes, 0.25),
        (hede_lifes, pom_lifes, 0.25),
        (hede_lifes, leaf_lifes, 0.25),
        (hede_lifes, alga_lifes, 0.25),
//-----------------------------------------
```

```
        (herb_lifes, diat_lifes, 0.333),
        (herb_lifes, alga_lifes, 0.333),
        (herb_lifes, pom_lifes, 0.333),
//-------------------------------------------
        (graz_lifes, alga_lifes, 0.07),
        (graz_lifes, diat_lifes, 0.196),
        (graz_lifes, pom_lifes, 0.734),
//-------------------------------------------
        (colg_lifes, pom_lifes, 1.00),
//-------------------------------------------
        (colf_lifes, pom_lifes, 1.00),
//-------------------------------------------
        (shre_lifes, leaf_lifes, 1.00)

}
```

A.2.3 .func file

Figure 8: The stochastic model done in BlenX of site 2; .func file.

```
let carnRep : const = 50;
let carnDie : const = 0.5;
let predRep : const = 6600;
let predDie : const = 6.6;
let omniRep : const = 100;
let omniDie : const = 1;
let herbRep : const = 1000;
let herbDie : const = 1;
let hedeRep : const = 1000;
let hedeDie : const = 1;
let grazRep : const = 128800;
let grazDie : const = 1.288;
let colgRep : const = 383000;
let colgDie : const = 0.383;
let colfRep : const = 200000;
let colfDie : const = 0.200;
let shreRep : const = 27000;
let shreDie : const = 0.027;
let algaRep : const = 500;
let algaDie : const = 500;
let algaWhen : const = 5000;
let diatRep : const = 500;
let diatDie : const = 500;
let diatWhen : const = 5000;
let terrRep : const = 540;
let terrDie : const = 5.4;
let terrWhen : const = 54;
let leafRep : const = 10000;
let leafDie : const = 10000;
let leafWhen : const = 100000;
let POMRep : const = 10000;
let POMDie : const = 10000;
let POMWhen : const = 100000;
```

A.3 BlenX model for site 3

A.3.1 .prog file

Figure 9: The stochastic model done in BlenX of site 3; .prog file.

```
[steps=120]
<<BASERATE : inf>>

// ---------- PROCESS ----------

let palga: pproc= ch(rate(algaRep),dupl,duplication) +
    food?().die(inf) + delay(rate(algaDie)).die(inf);
let pcarn: pproc= eat!().y!().nil +
    eat!().ch(rate(carnRep),dupl,duplication).nil +
    delay(rate(carnDie)).die(inf).nil;
let pcolf: pproc=
    food!().x!().nil+food!().ch(rate(colfRep),dupl,duplication).nil
    + eat?().die(inf)+delay(rate(colfDie)).die(inf).nil;
let pcolg: pproc=
    food!().x!().nil+food!().ch(rate(colgRep),dupl,duplication).nil
    + eat?().die(inf)+delay(rate(colgDie)).die(inf).nil;
let pdiat: pproc= ch(rate(diatRep),dupl,duplication) +
    food?().die(inf) + delay(rate(diatDie)).die(inf);
let pFila: pproc= ch(rate(FilaRep),dupl,duplication) +
    food?().die(inf) + delay(rate(FilaDie)).die(inf);
let pgraz: pproc= food!().x!().nil+food!().ch(rate(grazRep)
    ,dupl,duplication).nil +
    eat?().die(inf)+delay(rate(grazDie)).die(inf).nil;
let phede: pproc=
    food!().x!().nil+food!().ch(rate(hedeRep),dupl,duplication).nil
    + eat?().die(inf)+delay(rate(hedeDie)).die(inf).nil;
let pherb: pproc=
    food!().x!().nil+food!().ch(rate(herbRep),dupl,duplication).nil
    + eat?().die(inf)+delay(rate(herbDie)).die(inf).nil;
let pleaf: pproc= ch(rate(leafRep),dupl,duplication) +
    food?().die(inf) + delay(rate(leafDie)).die(inf);
let pomni: pproc=
    food!().x!().nil+food!().ch(rate(omniRep),dupl,duplication).nil
    + eat?().die(inf)+delay(rate(omniDie)).die(inf).nil;
let pPOM: pproc= ch(rate(POMRep),dupl,duplication) +
    food?().die(inf) + delay(rate(POMDie)).die(inf);
let ppred: pproc=
    food!().x!().nil+food!().ch(rate(predRep),dupl,duplication).nil
    + eat?().die(inf)+delay(rate(predDie)).die(inf).nil;
```

```
let pshre: pproc=
   food!().x!().nil+food!().ch(rate(shreRep),dupl,duplication).nil
   + eat?().die(inf)+delay(rate(shreDie)).die(inf).nil;
let pterr: pproc= ch(rate(terrRep),dupl,duplication) +
   food?().die(inf) + delay(rate(terrDie)).die(inf);

// ---------------- BOXES ----------------------

let Alga: bproc = #(food,alga_lifes),#(dupl:0,A) [ palga ];
let Carn: bproc=#(eat,carn_hunts),#(dupl:0,A)[ rep y?().pcarn |
   pcarn ];
let Colf: bproc=
   #(eat,hunts_colf),#(food,colf_lifes),#(dupl:0,A) [ rep
   x?().pcolf | pcolf];
let Colg : bproc =
   #(eat,hunts_colg),#(food,colg_lifes),#(dupl:0,A) [ rep
   x?().pcolg | pcolg];
let Diat: bproc = #(food,diat_lifes),#(dupl:0,A) [ pdiat ];
let Fila: bproc=#(food,fila_lifes),#(dupl:0,A) [ pFila ];
let Graz: bproc =
   #(eat,hunts_graz),#(food,graz_lifes),#(dupl:0,A) [ rep
   x?().pgraz| pgraz ];
let Hede : bproc =
   #(eat,hunts_hede),#(food,hede_lifes),#(dupl:0,A) [ rep
   x?().phede | phede ];
let Herb: bproc =
   #(eat,hunts_herb),#(food,herb_lifes),#(dupl:0,A) [ rep x?().
   pherb | pherb];
let Leaf: bproc = #(food,leaf_lifes),#(dupl:0,A) [ pleaf ];
let Omni: bproc =
   #(eat,hunts_omni),#(food,omni_lifes),#(dupl:0,A) [ rep
   x?().pomni| pomni ];
let POM: bproc = #(food,pom_lifes),#(dupl:0,A) [ pPOM ];
let Pred: bproc=
   #(eat,hunts_pred),#(food,pred_lifes),#(dupl:0,A) [ rep
   x?().ppred| ppred];
let Shre: bproc =
   #(eat,hunts_shre),#(food,shre_lifes),#(dupl:0,A) [ rep
   x?().pshre| pshre ];
let Terr: bproc = #(food,terr_lifes),#(dupl:0,A) [ pterr ];

let Humw: bproc=#(food,humw_lifes),#(dupl:0,A) [nil];

// ---------------- DUPLICATION ----------------------
```

```
let Algadup: bproc=#(food,alga_lifes),#(dupl:0,duplication) [
    nil ];
let Carndup: bproc=#(eat,carn_hunts),#(dupl:0,duplication) [
    rep y?().pcarn ];
let Colfdup: bproc =
    #(eat,hunts_colf),#(food,colf_lifes),#(dupl:0,duplication) [
    rep x?().pcolf ];
let Colgdup: bproc =
    #(eat,hunts_colg),#(food,colg_lifes),#(dupl:0,duplication) [
    rep x?().pcolg ];
let Diatdup: bproc=#(food,diat_lifes),#(dupl:0,duplication) [
    nil ];
let Filadup: bproc=#(food,fila_lifes),#(dupl:0,duplication) [
    nil ];
let Grazdup: bproc =
    #(eat,hunts_graz),#(food,graz_lifes),#(dupl:0,duplication) [
    rep x?().pgraz];
let Hededup: bproc =
    #(eat,hunts_hede),#(food,hede_lifes),#(dupl:0, duplication)
    [ rep x?().phede ];
let Herbdup: bproc =
    #(eat,hunts_herb),#(food,herb_lifes),#(dupl:0, duplication)
    [ rep x?().pherb ];
let Leafdup: bproc=#(food,leaf_lifes),#(dupl:0,duplication) [
    nil ];
let Omnidup: bproc =
    #(eat,hunts_omni),#(food,omni_lifes),#(dupl:0, duplication)
    [ rep x?().pomni];
let POMdup: bproc=#(food,pom_lifes),#(dupl:0,duplication) [ nil
    ];
let Preddup: bproc =
    #(eat,hunts_pred),#(food,pred_lifes),#(dupl:0,duplication) [
    rep x?().ppred];
let Shredup: bproc =
    #(eat,hunts_shre),#(food,shre_lifes),#(dupl:0, duplication)
    [ rep x?().pshre ];
let Terrdup: bproc=#(food,terr_lifes),#(dupl:0,duplication) [
    nil ];

let Humwdup: bproc=#(food,humw_lifes),#(dupl:0,duplication) [
    nil ];

// ---------- CONDITIONS ----------

// Duplication
```

```
when (Algadup: :inf) split (Alga, Alga);
when (Carndup: :inf) split (Carn, Carn);
when (Colfdup: :inf) split (Colf, Colf);
when (Colgdup: :inf) split (Colg, Colg);
when (Diatdup: :inf) split (Diat, Diat);
when (Filadup: :inf) split (Fila, Fila);
when (Grazdup: :inf) split (Graz, Graz);
when (Hededup: :inf) split (Hede, Hede);
when (Herbdup: :inf) split (Herb, Herb);
when (Leafdup: :inf) split (Leaf, Leaf);
when (Omnidup: :inf) split (Omni, Omni);
when (POMdup: :inf) split (POM, POM);
when (Preddup: :inf) split (Pred, Pred);
when (Shredup: :inf) split (Shre, Shre);
when (Terrdup: :inf) split (Terr, Terr);

when (Humwdup: :inf) split (Humw, Humw);


when (Alga::rate(algaWhen)) new (1);
when (Diat::rate(diatWhen)) new (1);
when (Fila::rate(POMWhen)) new (1);
when (Leaf::rate(leafWhen)) new (1);
when (POM::rate(POMWhen)) new (1);
when (Terr::rate(terrWhen)) new (1);


// ---------- STARTING ----------

run 500 Alga || 5 Carn || 12 Colf || 62 Colg || 500 Diat ||
    1000 Fila || 592 Graz || 10 Hede || 10 Herb || 1000 Leaf ||
    10 Omni || 1000 POM || 14 Pred || 18 Shre || 54 Terr || 0
    Humw
```

A.3.2 .types file

Figure 10: The stochastic model done in BlenX of site 3; .types file.

```
{duplication, A, carn_hunts, hunts_pred, pred_lifes,
    hunts_graz, graz_lifes, hunts_colg, colg_lifes, hunts_omni,
    omni_lifes, hunts_herb, herb_lifes, hunts_colf, colf_lifes,
    hunts_shre, shre_lifes, hunts_hede, hede_lifes, alga_lifes,
    diat_lifes, terr_lifes, leaf_lifes, pom_lifes, humw_lifes,
    fila_lifes}
%%
{
//------------TOP PREDATOR-------------

        (carn_hunts, hunts_pred, 0.125),
        (carn_hunts, hunts_graz, 0.125),
        (carn_hunts, hunts_colf, 0.125),
        (carn_hunts, hunts_omni, 0.125),
        (carn_hunts, hunts_shre, 0.125),
        (carn_hunts, terr_lifes, 0.125),
        (carn_hunts, hunts_herb, 0.125),
        (carn_hunts, hunts_hede, 0.125),

    (carn_hunts, hunts_colg, 0.111),


//------------PREY & PREDATOR-------------

        (omni_lifes, hunts_pred, 0.111),

        (omni_lifes, hunts_colg, 0.001),

        (omni_lifes, hunts_graz, 0.111),
        (omni_lifes, hunts_colf, 0.111),
        (omni_lifes, hunts_shre, 0.111),
        (omni_lifes, terr_lifes, 0.111),
        (omni_lifes, alga_lifes, 0.111),
        (omni_lifes, pom_lifes, 0.111),
        (omni_lifes, diat_lifes, 0.111),
        (omni_lifes, fila_lifes, 0.111),
//-----------------------------------------------
        (pred_lifes, hunts_graz, 0.5),
        (pred_lifes, hunts_colf, 0.5),

//-----------------------------------------------
```

```
        (hede_lifes, leaf_lifes, 1.00),
//-------------------------------------------------------
        (herb_lifes, diat_lifes, 0.25),
        (herb_lifes, alga_lifes, 0.25),
        (herb_lifes, pom_lifes, 0.25),
        (herb_lifes, fila_lifes, 0.25),
//-------------------------------------------------------
        (graz_lifes, alga_lifes, 0.25),
        (graz_lifes, diat_lifes, 0.25),
        (graz_lifes, fila_lifes, 0.25),
        (graz_lifes, pom_lifes, 0.25),
//-------------------------------------------------------
        (colf_lifes, pom_lifes, 1.00),
//-------------------------------------------------------
        (colg_lifes, alga_lifes, 0.02),
        (colg_lifes, diat_lifes, 0.6),
        (colg_lifes, fila_lifes, 0.38),
//-------------------------------------------------------
        (shre_lifes, leaf_lifes, 1.00)

}
```

A.3.3 .func file

Figure 11: The stochastic model done in BlenX of site 3; .func file.

```
let carnRep : const = 50;
let carnDie : const = 0.5;
let predRep : const = 140;
let predDie : const = 1.4;
let omniRep : const = 100;
let omniDie : const = 1;
let herbRep : const = 1000;
let herbDie : const = 1;
let hedeRep : const = 1000;
let hedeDie : const = 1;
let grazRep : const = 59200;
let grazDie : const = 0.592;
let colfRep : const = 12000;
let colfDie : const = 0.12;
let colgRep : const = 62000;
let colgDie : const = 0.062;
let shreRep : const = 18000;
let shreDie : const = 0.018;
let algaRep : const = 500;
let algaDie : const = 500;
let algaWhen : const = 5000;
let diatRep : const = 500;
let diatDie : const = 500;
let diatWhen : const = 5000;
let terrRep : const = 540;
let terrDie : const = 5.4;
let terrWhen : const = 54;
let leafRep : const = 10000;
let leafDie : const = 10000;
let leafWhen : const = 100000;
let POMRep : const = 10000;
let POMDie : const = 10000;
let POMWhen : const = 100000;
let FilaRep : const = 10000;
let FilaDie : const = 10000;
let FilaWhen : const = 100000;
```

A.4 BlenX model for site 4

A.4.1 .prog file

Figure 12: The stochastic model done in BlenX of site 4; .prog file.

```
[steps=120]
<<BASERATE : inf>>

// ---------- PROCESS ----------

let palga: pproc= ch(rate(algaRep),dupl,duplication) +
    food?().die(inf) + delay(rate(algaDie)).die(inf);
let pcarn: pproc= eat!().y!().nil +
    eat!().ch(rate(carnRep),dupl,duplication).nil +
    delay(rate(carnDie)).die(inf).nil;
let pcolg: pproc=
    food!().x!().nil+food!().ch(rate(colgRep),dupl,duplication).nil
    + eat?().die(inf)+delay(rate(colgDie)).die(inf).nil;
let pdiat: pproc= ch(rate(diatRep),dupl,duplication) +
    food?().die(inf) + delay(rate(diatDie)).die(inf);
let pfila: pproc= ch(rate(FilaRep),dupl,duplication) +
    food?().die(inf) + delay(rate(FilaDie)).die(inf);
let phede: pproc=
    food!().x!().nil+food!().ch(rate(hedeRep),dupl,duplication).nil
    + eat?().die(inf)+delay(rate(hedeDie)).die(inf).nil;
let pherb: pproc=
    food!().x!().nil+food!().ch(rate(herbRep),dupl,duplication).nil
    + eat?().die(inf)+delay(rate(herbDie)).die(inf).nil;
let pleaf: pproc= ch(rate(leafRep),dupl,duplication) +
    food?().die(inf) + delay(rate(leafDie)).die(inf);
let pomni: pproc=
    food!().x!().nil+food!().ch(rate(omniRep),dupl,duplication).nil
    + eat?().die(inf)+delay(rate(omniDie)).die(inf).nil;
let pPOM: pproc= ch(rate(POMRep),dupl,duplication) +
    food?().die(inf) + delay(rate(POMDie)).die(inf);
let ppred: pproc=
    food!().x!().nil+food!().ch(rate(predRep),dupl,duplication).nil
    + eat?().die(inf)+delay(rate(predDie)).die(inf).nil;
let pterr: pproc= ch(rate(terrRep),dupl,duplication) +
    food?().die(inf) + delay(rate(terrDie)).die(inf);

// ---------- BOXES ----------

let Alga: bproc = #(food,alga_lifes),#(dupl:0,A) [ palga ];
```

```
let Carn: bproc=#(eat,carn_hunts),#(dupl:0,A)[ rep y?().pcarn |
    pcarn ];
let Colg : bproc =
    #(eat,hunts_colg),#(food,colg_lifes),#(dupl:0,A) [ rep
    x?().pcolg | pcolg];
let Diat: bproc = #(food,diat_lifes),#(dupl:0,A) [ pdiat ];
let Fila: bproc = #(food,fila_lifes),#(dupl:0,A) [ pfila ];
let Hede : bproc =
    #(eat,hunts_hede),#(food,hede_lifes),#(dupl:0,A) [ rep
    x?().phede | phede ];
let Herb: bproc =
    #(eat,hunts_herb),#(food,herb_lifes),#(dupl:0,A) [ rep x?().
    pherb | pherb];
let Leaf: bproc = #(food,leaf_lifes),#(dupl:0,A) [ pleaf ];
let Omni: bproc =
    #(eat,hunts_omni),#(food,omni_lifes),#(dupl:0,A) [ rep
    x?().pomni| pomni ];
let Pred: bproc=
    #(eat,hunts_pred),#(food,pred_lifes),#(dupl:0,A) [ rep
    x?().ppred| ppred];
let POM: bproc = #(food,pom_lifes),#(dupl:0,A) [ pPOM ];
let Terr: bproc = #(food,terr_lifes),#(dupl:0,A) [ pterr ];


let Graz: bproc =
    #(eat,hunts_graz),#(food,graz_lifes),#(dupl:0,A) [ nil ];
let Colf: bproc=
    #(eat,hunts_colf),#(food,colf_lifes),#(dupl:0,A) [  nil ];
let Shre: bproc =
    #(eat,hunts_shre),#(food,shre_lifes),#(dupl:0,A) [ nil ];
let Humw: bproc=#(food,humw_lifes),#(dupl:0,A) [nil];

// ---------- DUPLICATION ----------

let Algadup: bproc=#(food,alga_lifes),#(dupl:0,duplication) [
    nil ];
let Carndup: bproc=#(eat,carn_hunts),#(dupl:0,duplication) [
    rep y?().pcarn ];
let Colgdup: bproc =
    #(eat,hunts_colg),#(food,colg_lifes),#(dupl:0,duplication) [
    rep x?().pcolg ];
let Diatdup: bproc=#(food,diat_lifes),#(dupl:0,duplication) [
    nil ];
let Hededup: bproc =
    #(eat,hunts_hede),#(food,hede_lifes),#(dupl:0, duplication)
```

```
    [ rep x?().phede ];
let Herbdup: bproc =
    #(eat,hunts_herb),#(food,herb_lifes),#(dupl:0, duplication)
    [ rep x?().pherb ];
let Filadup: bproc=#(food,fila_lifes),#(dupl:0,duplication) [
    nil ];
let Leafdup: bproc=#(food,leaf_lifes),#(dupl:0,duplication) [
    nil ];
let Omnidup: bproc =
    #(eat,hunts_omni),#(food,omni_lifes),#(dupl:0, duplication)
    [ rep x?().pomni ];
let POMdup: bproc=#(food,pom_lifes),#(dupl:0,duplication) [ nil
    ];
let Preddup: bproc =
    #(eat,hunts_pred),#(food,pred_lifes),#(dupl:0,duplication) [
    rep x?().ppred ];
let Terrdup: bproc=#(food,terr_lifes),#(dupl:0,duplication) [
    nil ];

let Grazdup: bproc =
    #(eat,hunts_graz),#(food,graz_lifes),#(dupl:0,duplication) [
    nil ];
let Colfdup: bproc =
    #(eat,hunts_colf),#(food,colf_lifes),#(dupl:0,duplication) [
    nil ];
let Shredup: bproc =
    #(eat,hunts_shre),#(food,shre_lifes),#(dupl:0, duplication)
    [ nil ];
let Humwdup: bproc=#(food,humw_lifes),#(dupl:0,duplication) [
    nil ];

// ---------- CONDITIONS ----------

// Duplication
when (Algadup: :inf) split (Alga, Alga);
when (Carndup: :inf) split (Carn, Carn);
when (Colgdup: :inf) split (Colg, Colg);
when (Diatdup: :inf) split (Diat, Diat);
when (Herbdup: :inf) split (Herb, Herb);
when (Hededup: :inf) split (Hede, Hede);
when (Leafdup: :inf) split (Leaf, Leaf);
when (Omnidup: :inf) split (Omni, Omni);
when (POMdup: :inf) split (POM, POM);
when (Preddup: :inf) split (Pred, Pred);
when (Terrdup: :inf) split (Terr, Terr);
```

```
when (Filadup: :inf) split (Fila, Fila);
when (Grazdup: :inf) split (Graz, Graz);
when (Colfdup: :inf) split (Colf, Colf);
when (Shredup: :inf) split (Shre, Shre);
when (Humwdup: :inf) split (Humw, Humw);

when (Alga::rate(algaWhen)) new (1);
when (Diat::rate(diatWhen)) new (1);
when (Fila::rate(POMWhen)) new (1);
when (Leaf::rate(leafWhen)) new (1);
when (POM::rate(POMWhen)) new (1);
when (Terr::rate(terrWhen)) new (1);

// ---------- STARTING ----------

run 500 Alga || 5 Carn || 68 Colg || 500 Diat || 1000 Fila ||
    10 Hede || 10 Herb || 1000 Leaf || 10 Omni || 1000 POM || 8
    Pred || 54 Terr || 0 Graz || 0 Colf || 0 Shre || 0 Humw
```

A.4.2 .types file

Figure 13: The stochastic model done in BlenX of site 4; .types file.

```
{duplication, A, carn_hunts, hunts_pred, pred_lifes,
    hunts_graz, graz_lifes, hunts_colg, colg_lifes, hunts_omni,
    omni_lifes, hunts_herb, herb_lifes, hunts_colf, colf_lifes,
    hunts_shre, shre_lifes, hunts_hede, hede_lifes, alga_lifes,
    diat_lifes, terr_lifes, leaf_lifes, pom_lifes, humw_lifes,
    fila_lifes}
%%
{
//------------TOP PREDATOR-------------

        (carn_hunts, hunts_pred, 0.25),
        (carn_hunts, hunts_omni, 0.25),
        (carn_hunts, hunts_herb, 0.25),
        (carn_hunts, hunts_hede, 0.25),
//------------PREY & PREDATOR-------------

        (omni_lifes, hunts_pred, 0.2),
        (omni_lifes, hunts_colg, 0.2),
        (omni_lifes, alga_lifes, 0.2),
        (omni_lifes, diat_lifes, 0.2),
        (omni_lifes, fila_lifes, 0.2),
//---------------------------------------
        (pred_lifes, hunts_colg, 1.00),
//---------------------------------------
        (hede_lifes, diat_lifes, 0.25),
        (hede_lifes, leaf_lifes, 0.25),
        (hede_lifes, alga_lifes, 0.25),
        (hede_lifes, fila_lifes, 0.25),
//---------------------------------------
        (herb_lifes, diat_lifes, 0.333),
        (herb_lifes, alga_lifes, 0.333),
        (herb_lifes, fila_lifes, 0.333),
//---------------------------------------
        (colg_lifes, pom_lifes, 1.00)

}
```

A.4.3 .func file

Figure 14: The stochastic model done in BlenX of site 4; .func file.

```
let carnRep : const = 50;
let carnDie : const = 0.5;
let predRep : const = 800;
let predDie : const = 0.8;
let omniRep : const = 100;
let omniDie : const = 1;
let herbRep : const = 1000;
let herbDie : const = 1;
let hedeRep : const = 1000;
let hedeDie : const = 1;
let colgRep : const = 68000;
let colgDie : const = 0.68;
let algaRep : const = 500;
let algaDie : const = 500;
let algaWhen : const = 5000;
let diatRep : const = 500;
let diatDie : const = 500;
let diatWhen : const = 5000;
let terrRep : const = 540;
let terrDie : const = 5.4;
let terrWhen : const = 54;
let leafRep : const = 10000;
let leafDie : const = 10000;
let leafWhen : const = 100000;
let POMRep : const = 10000;
let POMDie : const = 10000;
let POMWhen : const = 100000;
let FilaRep : const = 10000;
let FilaDie : const = 10000;
let FilaWhen : const = 100000;
```

A.5 BlenX model for site 5

A.5.1 .prog file

Figure 15: The stochastic model done in BlenX of site 5; .prog file.

```
[steps=120]
<<BASERATE : inf>>

// ---------- PROCESS ----------

let palga: pproc= ch(rate(algaRep),dupl,duplication) +
    food?().die(inf) + delay(rate(algaDie)).die(inf);
let pcarn: pproc= eat!().y!().nil +
    eat!().ch(rate(carnRep),dupl,duplication).nil +
    delay(rate(carnDie)).die(inf).nil;
let pcolf: pproc=
    food!().x!().nil+food!().ch(rate(colfRep),dupl,duplication).nil
   + eat?().die(inf)+delay(rate(colfDie)).die(inf).nil;
let pcolg: pproc=
    food!().x!().nil+food!().ch(rate(colgRep),dupl,duplication).nil
   + eat?().die(inf)+delay(rate(colgDie)).die(inf).nil;
let pdiat: pproc= ch(rate(diatRep),dupl,duplication) +
    food?().die(inf) + delay(rate(diatDie)).die(inf);
let pFila: pproc= ch(rate(FilaRep),dupl,duplication) +
    food?().die(inf) + delay(rate(FilaDie)).die(inf);
let pgraz: pproc= food!().x!().nil+food!().ch(rate(grazRep)
    ,dupl,duplication).nil +
    eat?().die(inf)+delay(rate(grazDie)).die(inf).nil;
let phede: pproc=
    food!().x!().nil+food!().ch(rate(hedeRep),dupl,duplication).nil
   + eat?().die(inf)+delay(rate(hedeDie)).die(inf).nil;
let pherb: pproc=
    food!().x!().nil+food!().ch(rate(herbRep),dupl,duplication).nil
   + eat?().die(inf)+delay(rate(herbDie)).die(inf).nil;
let pleaf: pproc= ch(rate(leafRep),dupl,duplication) +
    food?().die(inf) + delay(rate(leafDie)).die(inf);
let pomni: pproc=
    food!().x!().nil+food!().ch(rate(omniRep),dupl,duplication).nil
   + eat?().die(inf)+delay(rate(omniDie)).die(inf).nil;
let pPOM: pproc= ch(rate(POMRep),dupl,duplication) +
    food?().die(inf) + delay(rate(POMDie)).die(inf);
let ppred: pproc=
    food!().x!().nil+food!().ch(rate(predRep),dupl,duplication).nil
   + eat?().die(inf)+delay(rate(predDie)).die(inf).nil;
```

```
let pshre: pproc=
   food!().x!().nil+food!().ch(rate(shreRep),dupl,duplication).nil
   + eat?().die(inf)+delay(rate(shreDie)).die(inf).nil;
let pterr: pproc= ch(rate(terrRep),dupl,duplication) +
   food?().die(inf) + delay(rate(terrDie)).die(inf);


// ---------------- BOXES ----------------------

let Alga: bproc = #(food,alga_lifes),#(dupl:0,A) [ palga ];
let Carn: bproc=#(eat,carn_hunts),#(dupl:0,A)[ rep y?().pcarn |
   pcarn ];
let Colf: bproc=
   #(eat,hunts_colf),#(food,colf_lifes),#(dupl:0,A) [ rep
   x?().pcolf | pcolf];
let Colg : bproc =
   #(eat,hunts_colg),#(food,colg_lifes),#(dupl:0,A) [ rep
   x?().pcolg | pcolg];
let Diat: bproc = #(food,diat_lifes),#(dupl:0,A) [ pdiat ];
let Fila: bproc=#(food,fila_lifes),#(dupl:0,A) [ pFila ];
let Graz: bproc =
   #(eat,hunts_graz),#(food,graz_lifes),#(dupl:0,A) [ rep
   x?().pgraz| pgraz ];
let Hede : bproc =
   #(eat,hunts_hede),#(food,hede_lifes),#(dupl:0,A) [ rep
   x?().phede | phede ];
let Herb: bproc =
   #(eat,hunts_herb),#(food,herb_lifes),#(dupl:0,A) [ rep x?().
   pherb | pherb];
let Leaf: bproc = #(food,leaf_lifes),#(dupl:0,A) [ pleaf ];
let Omni: bproc =
   #(eat,hunts_omni),#(food,omni_lifes),#(dupl:0,A) [ rep
   x?().pomni| pomni ];
let POM: bproc = #(food,pom_lifes),#(dupl:0,A) [ pPOM ];
let Pred: bproc=
   #(eat,hunts_pred),#(food,pred_lifes),#(dupl:0,A) [ rep
   x?().ppred| ppred];
let Shre: bproc =
   #(eat,hunts_shre),#(food,shre_lifes),#(dupl:0,A) [ rep
   x?().pshre| pshre ];
let Terr: bproc = #(food,terr_lifes),#(dupl:0,A) [ pterr ];

let Humw: bproc=#(food,humw_lifes),#(dupl:0,A) [nil];

// ---------------- DUPLICATION ----------------------
```

```
let Algadup: bproc=#(food,alga_lifes),#(dupl:0,duplication) [
    nil ];
// Top predator
let Carndup: bproc=#(eat,carn_hunts),#(dupl:0,duplication) [
    rep y?().pcarn ];
let Colfdup: bproc =
    #(eat,hunts_colf),#(food,colf_lifes),#(dupl:0,duplication) [
    rep x?().pcolf ];
let Colgdup: bproc =
    #(eat,hunts_colg),#(food,colg_lifes),#(dupl:0,duplication) [
    rep x?().pcolg ];
let Diatdup: bproc=#(food,diat_lifes),#(dupl:0,duplication) [
    nil ];
let Filadup: bproc=#(food,fila_lifes),#(dupl:0,duplication) [
    nil ];
let Grazdup: bproc =
    #(eat,hunts_graz),#(food,graz_lifes),#(dupl:0,duplication) [
    rep x?().pgraz];
let Hededup: bproc =
    #(eat,hunts_hede),#(food,hede_lifes),#(dupl:0, duplication)
    [ rep x?().phede ];
let Herbdup: bproc =
    #(eat,hunts_herb),#(food,herb_lifes),#(dupl:0, duplication)
    [ rep x?().pherb ];
let Leafdup: bproc=#(food,leaf_lifes),#(dupl:0,duplication) [
    nil ];
let Omnidup: bproc =
    #(eat,hunts_omni),#(food,omni_lifes),#(dupl:0, duplication)
    [ rep x?().pomni];
let POMdup: bproc=#(food,pom_lifes),#(dupl:0,duplication) [ nil
    ];
let Preddup: bproc =
    #(eat,hunts_pred),#(food,pred_lifes),#(dupl:0,duplication) [
    rep x?().ppred];
let Shredup: bproc =
    #(eat,hunts_shre),#(food,shre_lifes),#(dupl:0, duplication)
    [ rep x?().pshre ];
let Terrdup: bproc=#(food,terr_lifes),#(dupl:0,duplication) [
    nil ];

let Humwdup: bproc=#(food,humw_lifes),#(dupl:0,duplication) [
    nil ];

// ---------- CONDITIONS ----------
```

```
when (Algadup: :inf) split (Alga, Alga);
// Duplication
when (Carndup: :inf) split (Carn, Carn);
when (Colfdup: :inf) split (Colf, Colf);
when (Colgdup: :inf) split (Colg, Colg);
when (Diatdup: :inf) split (Diat, Diat);
when (Filadup: :inf) split (Fila, Fila);
when (Grazdup: :inf) split (Graz, Graz);
when (Hededup: :inf) split (Hede, Hede);
when (Herbdup: :inf) split (Herb, Herb);
when (Leafdup: :inf) split (Leaf, Leaf);
when (Omnidup: :inf) split (Omni, Omni);
when (POMdup: :inf) split (POM, POM);
when (Preddup: :inf) split (Pred, Pred);
when (Shredup: :inf) split (Shre, Shre);
when (Terrdup: :inf) split (Terr, Terr);

when (Humwdup: :inf) split (Humw, Humw);


when (Alga::rate(algaWhen)) new (1);
when (Diat::rate(diatWhen)) new (1);
when (Fila::rate(FilaWhen)) new (1);
when (Leaf::rate(leafWhen)) new (1);
when (POM::rate(POMWhen)) new (1);
when (Terr::rate(terrWhen)) new (1);


// ---------------- STARTING ----------------------

run 500 Alga || 5 Carn || 4 Colf || 62 Colg || 500 Diat || 1000
    Fila || 39 Graz || 10 Hede || 10 Herb || 1000 Leaf || 10
    Omni || 1000 POM || 23 Pred || 4 Shre || 54 Terr || 0 Humw
```

A.5.2 .types file

Figure 16: The stochastic model done in BlenX of site 5; .types file.

```
{duplication, A, carn_hunts, hunts_pred, pred_lifes,
    hunts_graz, graz_lifes, hunts_colg, colg_lifes, hunts_omni,
    omni_lifes, hunts_herb, herb_lifes, hunts_colf, colf_lifes,
    hunts_shre, shre_lifes, hunts_hede, hede_lifes, alga_lifes,
    diat_lifes, terr_lifes, leaf_lifes, pom_lifes, humw_lifes,
    fila_lifes}
%%
{
//------------TOP PREDATOR-------------

        (carn_hunts, hunts_pred, 0.111),
        (carn_hunts, hunts_omni, 0.111),
        (carn_hunts, hunts_graz, 0.111),
        (carn_hunts, hunts_colf, 0.111),
        (carn_hunts, hunts_colg, 0.111),
        (carn_hunts, hunts_shre, 0.111),
        (carn_hunts, terr_lifes, 0.111),
        (carn_hunts, hunts_herb, 0.111),
        (carn_hunts, hunts_hede, 0.111),

//------------PREY & PREDATOR-------------

        (omni_lifes, hunts_pred, 0.1),
        (omni_lifes, hunts_colg, 0.1),
        (omni_lifes, hunts_graz, 0.1),
        (omni_lifes, hunts_colf, 0.1),
        (omni_lifes, hunts_shre, 0.1),
        (omni_lifes, terr_lifes, 0.1),
        (omni_lifes, alga_lifes, 0.1),
        (omni_lifes, pom_lifes, 0.1),
        (omni_lifes, diat_lifes, 0.1),
        (omni_lifes, fila_lifes, 0.1),
        (pred_lifes, hunts_graz, 0.25),
        (pred_lifes, hunts_colg, 0.25),
        (pred_lifes, hunts_colf, 0.25),
        (pred_lifes, hunts_shre, 0.25),
        (hede_lifes, diat_lifes, 0.25),
        (hede_lifes, leaf_lifes, 0.25),
        (hede_lifes, alga_lifes, 0.25),
        (hede_lifes, fila_lifes, 0.25),
        (herb_lifes, diat_lifes, 0.333),
```

```
        (herb_lifes, alga_lifes, 0.333),
        (herb_lifes, fila_lifes, 0.333),
        (graz_lifes, alga_lifes, 0.07),
        (graz_lifes, diat_lifes, 0.196),
        (graz_lifes, pom_lifes, 0.734),
        (colg_lifes, alga_lifes, 0.02),
        (colg_lifes, diat_lifes, 0.6),
        (colg_lifes, fila_lifes, 0.38),
        (colf_lifes, pom_lifes, 1.00),
        (shre_lifes, leaf_lifes, 1.00)

}
```

A.5.3 .func file

Figure 17: The stochastic model done in BlenX of site 5; .func file.

```
let carnRep : const = 50;
let carnDie : const = 0.5;
let predRep : const = 2300;
let predDie : const = 2.3;
let omniRep : const = 100;
let omniDie : const = 1;
let herbRep : const = 1000;
let herbDie : const = 1;
let hedeRep : const = 1000;
let hedeDie : const = 1;
let grazRep : const = 3900;
let grazDie : const = 0.039;
let colgRep : const = 62000;
let colgDie : const = 0.062;
let colfRep : const = 4000;
let colfDie : const = 0.004;
let shreRep : const = 4000;
let shreDie : const = 0.004;
let algaRep : const = 500;
let algaDie : const = 500;
let algaWhen : const = 5000;
let diatRep : const = 500;
let diatDie : const = 500;
let diatWhen : const = 5000;
let terrRep : const = 540;
let terrDie : const = 5.4;
let terrWhen : const = 54;
let leafRep : const = 10000;
let leafDie : const = 10000;
let leafWhen : const = 100000;
let POMRep : const = 10000;
let POMDie : const = 10000;
let POMWhen : const = 100000;
let FilaRep : const = 10000;
let FilaDie : const = 10000;
let FilaWhen : const = 100000;
```

A.6 BlenX model for site 6

A.6.1 .prog file

Figure 18: The stochastic model done in BlenX of site 6; .prog file.

```
[steps=120]
<<BASERATE : inf>>

// ---------------- PROCESS ----------------------
let palga: pproc= ch(rate(algaRep),dupl,duplication) +
    food?().die(inf) + delay(rate(algaDie)).die(inf);
let pcarn: pproc=
    food!().x!().nil+food!().ch(rate(carnRep),dupl,duplication).nil
    + eat?().die(inf)+delay(rate(carnDie)).die(inf).nil;
let pcolg: pproc=
    food!().x!().nil+food!().ch(rate(colgRep),dupl,duplication).nil
    + eat?().die(inf)+delay(rate(colgDie)).die(inf).nil;
let pdiat: pproc= ch(rate(diatRep),dupl,duplication) +
    food?().die(inf) + delay(rate(diatDie)).die(inf);
let pfila: pproc= ch(rate(FilaRep),dupl,duplication) +
    food?().die(inf) + delay(rate(FilaDie)).die(inf);
let phede: pproc=
    food!().x!().nil+food!().ch(rate(hedeRep),dupl,duplication).nil
    + eat?().die(inf)+delay(rate(hedeDie)).die(inf).nil;
let pherb: pproc=
    food!().x!().nil+food!().ch(rate(herbRep),dupl,duplication).nil
    + eat?().die(inf)+delay(rate(herbDie)).die(inf).nil;
let phumw: pproc= ch(rate(HumwRep),dupl,duplication) +
    food?().die(inf) + delay(rate(HumwDie)).die(inf);
let pomni: pproc=
    food!().x!().nil+food!().ch(rate(omniRep),dupl,duplication).nil
    + eat?().die(inf)+delay(rate(omniDie)).die(inf).nil;
let pPOM: pproc= ch(rate(POMRep),dupl,duplication) +
    food?().die(inf) + delay(rate(POMDie)).die(inf);
let ppred: pproc= eat!().y!().nil +
    eat!().ch(rate(predRep),dupl,duplication).nil +
    delay(rate(predDie)).die(inf).nil;
let pterr: pproc= ch(rate(terrRep),dupl,duplication) +
    food?().die(inf) + delay(rate(terrDie)).die(inf);


// ---------------- BOXES ----------------------

let Alga: bproc = #(food,alga_lifes),#(dupl:0,A) [ palga ];
```

```
let Carn: bproc=
   #(eat,hunts_carn),#(food,carn_lifes),#(dupl:0,A) [ rep
   x?().pcarn| pcarn];
let Colg : bproc =
   #(eat,hunts_colg),#(food,colg_lifes),#(dupl:0,A) [ rep
   x?().pcolg | pcolg];
let Diat: bproc = #(food,diat_lifes),#(dupl:0,A) [ pdiat ];
let Fila: bproc = #(food,fila_lifes),#(dupl:0,A) [ pfila ];
let Hede : bproc =
   #(eat,hunts_hede),#(food,hede_lifes),#(dupl:0,A) [ rep
   x?().phede | phede ];
let Herb: bproc =
   #(eat,hunts_herb),#(food,herb_lifes),#(dupl:0,A) [ rep x?().
   pherb | pherb];
let Humw: bproc= #(food,humw_lifes),#(dupl:0,A) [ phumw ];
let Omni: bproc =
   #(eat,hunts_omni),#(food,omni_lifes),#(dupl:0,A) [ rep
   x?().pomni | pomni ];
let POM: bproc = #(food,pom_lifes),#(dupl:0,A) [ pPOM ];
let Pred: bproc=#(eat,pred_hunts),#(dupl:0,A)[ rep y?().ppred |
   ppred ];
let Terr: bproc = #(food,terr_lifes),#(dupl:0,A) [ pterr ];

let Graz: bproc =
   #(eat,hunts_graz),#(food,graz_lifes),#(dupl:0,A) [ nil ];
let Colf: bproc=
   #(eat,hunts_colf),#(food,colf_lifes),#(dupl:0,A) [ nil ];
let Shre: bproc =
   #(eat,hunts_shre),#(food,shre_lifes),#(dupl:0,A) [ nil ];
let Leaf: bproc= #(food,fila_lifes),#(dupl:0,A) [nil];

// ---------- DUPLICATION ----------

let Algadup: bproc=#(food,alga_lifes),#(dupl:0,duplication) [
   nil ];
let Carndup: bproc =
   #(eat,hunts_carn),#(food,carn_lifes),#(dupl:0,duplication) [
   rep x?().pcarn];
let Colgdup: bproc =
   #(eat,hunts_colg),#(food,colg_lifes),#(dupl:0,duplication) [
   rep x?().pcolg ];
let Diatdup: bproc=#(food,diat_lifes),#(dupl:0,duplication) [
   nil ];
let Filadup: bproc=#(food,fila_lifes),#(dupl:0,duplication) [
   nil ];
```

```
let Hededup: bproc =
    #(eat,hunts_hede),#(food,hede_lifes),#(dupl:0, duplication)
    [ rep x?().phede ];
let Herbdup: bproc =
    #(eat,hunts_herb),#(food,herb_lifes),#(dupl:0, duplication)
    [ rep x?().pherb ];
let Humwdup: bproc=#(food,humw_lifes),#(dupl:0,duplication) [
    nil ];
let Omnidup: bproc =
    #(eat,hunts_omni),#(food,omni_lifes),#(dupl:0, duplication)
    [ rep x?().pomni];
let POMdup: bproc=#(food,pom_lifes),#(dupl:0,duplication) [ nil
    ];
let Preddup: bproc=#(eat,pred_hunts),#(dupl:0,duplication) [
    rep y?().ppred ];
let Terrdup: bproc=#(food,terr_lifes),#(dupl:0,duplication) [
    nil ];

let Grazdup: bproc =
    #(eat,hunts_graz),#(food,graz_lifes),#(dupl:0,duplication) [
    nil ];
let Colfdup: bproc =
    #(eat,hunts_colf),#(food,colf_lifes),#(dupl:0,duplication) [
    nil ];
let Shredup: bproc =
    #(eat,hunts_shre),#(food,shre_lifes),#(dupl:0, duplication)
    [ nil ];
let Leafdup: bproc=#(food,leaf_lifes),#(dupl:0,duplication) [
    nil ];

// ---------- CONDITIONS ----------
// Duplication
when (Algadup: :inf) split (Alga, Alga);
when (Carndup: :inf) split (Carn, Carn);
when (Colgdup: :inf) split (Colg, Colg);
when (Diatdup: :inf) split (Diat, Diat);
when (Filadup: :inf) split (Fila, Fila);
when (Hededup: :inf) split (Hede, Hede);
when (Herbdup: :inf) split (Herb, Herb);
when (Humwdup: :inf) split (Humw, Humw);
when (Omnidup: :inf) split (Omni, Omni);
when (POMdup: :inf) split (POM, POM);
when (Preddup: :inf) split (Pred, Pred);
when (Terrdup: :inf) split (Terr, Terr);
```

```
when (Grazdup: :inf) split (Graz, Graz);
when (Colfdup: :inf) split (Colf, Colf);
when (Shredup: :inf) split (Shre, Shre);
when (Leafdup: :inf) split (Leaf, Leaf);

when (Alga::rate(algaWhen)) new (1);
when (Diat::rate(diatWhen)) new (1);
when (Fila::rate(FilaWhen)) new (1);
when (Humw::rate(HumwWhen)) new (1);
when (POM::rate(POMWhen)) new (1);
when (Terr::rate(terrWhen)) new (1);


// ---------------- STARTING ----------------------

run 500 Alga || 5 Carn || 21 Colg || 500 Diat || 1000 Fila ||
    10 Hede || 10 Herb || 1000 Humw || 10 Omni || 1000 POM || 1
    Pred || 54 Terr || 0 Graz || 0 Colf || 0 Shre || 0 Leaf
```

A.6.2 .types file

Figure 19: The stochastic model done in BlenX of site 6; .types file.

```
{duplication , A, pred_hunts , hunts_carn , carn_lifes ,
    hunts_colg , colg_lifes , hunts_omni , omni_lifes , hunts_herb ,
    herb_lifes , hunts_hede , hede_lifes , alga_lifes , diat_lifes ,
    terr_lifes , pom_lifes , humw_lifes , fila_lifes , hunts_graz ,
    graz_lifes , hunts_colf , colf_lifes , hunts_shre , shre_lifes ,
    leaf_lifes}
%%
{
 // TOP PREDATOR

        (pred_hunts , hunts_carn , 0.333),
        (pred_hunts , hunts_colg , 0.333),
        (pred_hunts , hunts_omni , 0.333),

// INTERMEDATE SPECES
        (carn_lifes , hunts_colg , 0.111),
        (carn_lifes , hunts_omni , 0.111),
        (carn_lifes , terr_lifes , 0.111),
        (carn_lifes , hunts_herb , 0.111),
        (carn_lifes , hunts_hede , 0.111),
        (colg_lifes , alga_lifes , 0.02),
        (colg_lifes , diat_lifes , 0.6),
        (colg_lifes , fila_lifes , 0.19),
        (colg_lifes , pom_lifes , 0.19),
        (omni_lifes , hunts_colg , 0.14),
        (omni_lifes , terr_lifes , 0.14),
        (omni_lifes , alga_lifes , 0.14),
        (omni_lifes , pom_lifes , 0.14),
        (omni_lifes , diat_lifes , 0.14),
        (omni_lifes , humw_lifes , 0.14),
        (omni_lifes , fila_lifes , 0.14),
        (herb_lifes , diat_lifes , 0.25),
        (herb_lifes , alga_lifes , 0.25),
        (herb_lifes , pom_lifes , 0.25),
        (herb_lifes , fila_lifes , 0.25),
        (hede_lifes , diat_lifes , 0.2),
        (hede_lifes , pom_lifes , 0.2),
        (hede_lifes , alga_lifes , 0.2),
        (hede_lifes , humw_lifes , 0.2),
        (hede_lifes , fila_lifes , 0.2)
}
```

A.6.3 .func file

Figure 20: The stochastic model done in BlenX of site 6; .func file.

```
let predRep : const = 5400;
let predDie : const = 0.54;
let carnRep : const = 50;
let carnDie : const = 0.5;
let colgRep : const = 2100;
let colgDie : const = 0.021;
let omniRep : const = 10;
let omniDie : const = 0.1;
let herbRep : const = 100;
let herbDie : const = 0.1;
let hedeRep : const = 100;
let hedeDie : const = 0.1;
let algaRep : const = 500;
let algaDie : const = 500;
let algaWhen : const = 5000;
let diatRep : const = 500;
let diatDie : const = 500;
let diatWhen : const = 5000;
let terrRep : const = 54;
let terrDie : const = 5.4;
let terrWhen : const = 54;
let POMRep : const = 10000;
let POMDie : const = 10000;
let POMWhen : const = 100000;
let HumwRep : const = 500;
let HumwDie : const = 500;
let HumwWhen : const = 5000;
let FilaRep : const = 500;
let FilaDie : const = 500;
let FilaWhen : const = 5000;
```

A.7 Kelian river dataset

A.7.1 Population size for the six sites in Kelian river

	Site 1	Site 2	Site 3	Site 4	Site 5	Site 6
carn	5	5	5	5	5	5
pred	54	66	14	8	23	2
terr	54	54	54	-	54	54
graz	1054	1288	592	-	39	-
colg	377	383	-	68	62	21
omni	10	10	10	10	10	10
herb	10	10	10	10	10	10
colf	212	200	12	-	4	-
shre	87	27	18	-	4	-
hede	10	10	10	10	10	10
alga	500	500	500	500	500	500
diat	500	500	500	500	500	500
POM	1000	1000	1000	1000	1000	1000
leaf	1000	1000	1000	1000	1000	-
fila	-	-	1000	1000	1000	1000
humw	-	-	-	-	-	1000

Table 10: In the table are reported data about population size in the six sites of the Kelian river.

A.7.2 Reproduction and death rates of the six sites in Kelian river

	Site 1			Site 2			Site 3		
	Rep	Die	When	Rep	Die	When	Rep	Die	When
carn	50	0.50	-	50	0.50	-	50	0.50	-
pred	5400	5.40	-	5400	5.40	-	5400	5.40	-
omni	100	1.00	-	100	1.00	-	100	1.00	-
herb	1000	1.00	-	1000	1.00	-	1000	1.00	-
hede	1000	1.00	-	1000	1.00	-	1000	1.00	-
graz	105400	1.05	-	105400	1.05	-	105400	1.05	-
colg	377000	0.38	-	377000	0.38	-	-	-	-
colf	212000	0.21	-	212000	0.21	-	212000	0.21	-
shre	87000	0.09	-	87000	0.09	-	87000	0.09	-
alga	500	500.00	5000	500	500.00	5000	500	500.00	5000
diat	500	500.00	5000	500	500.00	5000	500	500.00	5000
terr	540	5.40	54	540	5.40	54	540	5.40	54
leaf	10000	10000.00	100000	10000	10000.00	100000	10000	10000.00	100000
POM	10000	10000.00	100000	10000	10000.00	100000	10000	10000.00	100000
humw	-	-	-	-	-	-	-	-	-
fila	-	-	-	-	-	-	500	500.00	5000

Table 11: The table shows the parameters of reproduction, death and when for all the trophic groups in the sites 1, 2 and 3.

	Site 4			Site 5			Site 6		
	Rep	Die	When	Rep	Die	When	Rep	Die	When
carn	50	0.50	-	50	0.50	-	50	0.50	-
pred	5400	5.40	-	5400	5.40	-	5400	5.40	-
omni	100	1.00	-	100	1.00	-	100	1.00	-
herb	1000	1.00	-	1000	1.00	-	1000	1.00	-
hede	1000	1.00	-	1000	1.00	-	1000	1.00	-
graz	-	-	-	105400	1.05	-	-	-	-
colg	377000	0.38	-	377000	0.38	-	377000	0.38	-
colf	-	-	-	212000	0.21	-	-	-	-
shre	-	-	-	87000	0.09	-	-	-	-
alga	500	500.00	5000	500	500.00	5000	500	500.00	5000
diat	500	500.00	5000	500	500.00	5000	500	500.00	5000
terr	-	-	-	540	5.40	54	540	5.40	54
leaf	10000	10000.00	100000	10000	10000.00	100000	-	-	-
POM	10000	10000.00	100000	10000	10000.00	100000	10000	10000.00	100000
humw	-	-	-	-	-	-	500	500.00	5000
fila	500	500.00	5000	500	500.00	5000	500	500.00	5000

Table 12: The table shows the parameters of reproduction, death and when for all the trophic groups in sites 4, 5 and 6.

A.7.3 The feeding partial matrices of all the sites in Kelian river

SITE 1	CARN	OMNI	PRED	HEDE	HERB	GRAZ	COLG	COLF	SHRE	TERR	ALGA	POM	DIAT	LEAF
CARN	0	0	0	0	0	0	0	0	0	0	0	0	0	0
OMNI	0.111	0	0	0	0	0	0	0	0	0	0	0	0	0
PRED	0.111	0.125	0	0	0	0	0	0	0	0	0	0	0	0
HEDE	0.111	0	0	0	0	0	0	0	0	0	0	0	0	0
HERB	0.111	0	0	0	0	0	0	0	0	0	0	0	0	0
GRAZ	0.111	0.125	0.25	0	0	0	0	0	0	0	0	0	0	0
COLG	0.111	0.125	0.25	0	0	0	0	0	0	0	0	0	0	0
COLF	0.111	0.125	0.25	0	0	0	0	0	0	0	0	0	0	0
SHRE	0.111	0.125	0.25	0	0	0	0	0	0	0	0	0	0	0
TERR	0.111	0.125	0	0	0	0	0	0	0	0	0	0	0	0
ALGA	0	0.125	0	0.25	0.333	0.07	0	0	0	0	0	0	0	0
POM	0	0	0	0.25	0.333	0.734	1	1	0	0	0	0	0	0
DIAT	0	0.125	0	0.25	0.333	0.196	0	0	0	0	0	0	0	0
LEAF	0	0	0	0.25	0	0	0	0	1	0	0	0	0	0

Table 13: Site 2, partial feeding matrix, in the columns are presented the predator and in the rows the preys. The matrix is estimated normalizing (the columns sum to one) the connections between trophic groups by the total intake of each receiving node.

SITE 1	CARN	OMNI	PRED	HEDE	HERB	GRAZ	COLG	COLF	SHRE	TERR	ALGA	POM	DIAT	LEAF	FILA
CARN	0	0	0	0	0	0	0	0	0.125	0	0	0	0	0	0
OMNI	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PRED	0.125	0	0	0	0	0	0	0	0	0	0	0	0	0	0
HEDE	0.125	0	0	0	0	0	0	0	0	0	0	0	0	0	0
HERB	0.125	0	0	0	0	0	0	0	0	0	0	0	0	0	0
GRAZ	0.125	0.111	0.5	0	0	0	0	0	0	0	0	0	0	0	0
COLG	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
COLF	0.125	0.111	0.5	0	0	0	0	0	0	0	0	0	0	0	0
SHRE	0.125	0.111	0	0	0	0	0	0	0	0	0	0	0	0	0
TERR	0.125	0.111	0	0	0	0	0	0	0	0	0	0	0	0	0
ALGA	0	0.111	0	0	0.25	0.25	0	0	0	0	0	0	0	0	0
POM	0	0.111	0	0	0.25	0.25	0	1	0	0	0	0	0	0	0
DIAT	0	0.111	0	0	0.25	0.25	0	0	0	0	0	0	0	0	0
LEAF	0	0.111	0	1	0	0	0	0	1	0	0	0	0	0	0
FILA	0	0.111	0	0	0.25	0.25	0	0	0	0	0	0	0	0	0

Table 14: Site 3, partial feeding matrix, in the columns are presented the predator and in the rows the preys. The matrix is estimated normalizing (the columns sum to one) the connections between trophic groups by the total intake of each receiving node.

SITE 1	CARN	OMNI	PRED	HEDE	HERB	COLG	TERR	ALGA	POM	DIAT	LEAF	FILA
CARN	0	0	0	0	0	0	0	0	0	0	0	0
OMNI	0.25	0	0	0	0	0	0	0	0	0	0	0
PRED	0.25	0.2	0	0	0	0	0	0	0	0	0	0
HEDE	0.25	0	0	0	0	0	0	0	0	0	0	0
HERB	0.25	0	0	0	0	0	0	0	0	0	0	0
COLG	0	0.2	1	0	0	0	0	0	0	0	0	0
TERR	0	0	0	0	0	0	0	0	0	0	0	0
ALGA	0	0.2	0	0.25	0.333	0	0	0	0	0	0	0
POM	0	0	0	0	0	1	0	0	0	0	0	0
DIAT	0	0.2	0	0.25	0.333	0	0	0	0	0	0	0
LEAF	0	0	0	0.25	0	0	0	0	0	0	0	0
FILA	0	0.2	0	0.25	0.333	0	0	0	0	0	0	0

Table 15: Site 4, partial feeding matrix, in the columns are presented the predator and in the rows the preys. The matrix is estimated normalizing (the columns sum to one) the connections between trophic groups by the total intake of each receiving node.

SITE 1	CARN	OMNI	PRED	HEDE	HERB	GRAZ	COLG	COLF	SHRE	TERR	ALGA	POM	DIAT	LEAF	FILA
CARN	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
OMNI	0.111	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PRED	0.111	0.1	0	0	0	0	0	0	0	0	0	0	0	0	0
HEDE	0.111	0	0	0	0	0	0	0	0	0	0	0	0	0	0
HERB	0.111	0	0	0	0	0	0	0	0	0	0	0	0	0	0
GRAZ	0.111	0.1	0.25	0	0	0	0	0	0	0	0	0	0	0	0
COLG	0.111	0.1	0.25	0	0	0	0	0	0	0	0	0	0	0	0
COLF	0.111	0.1	0.25	0	0	0	0	0	0	0	0	0	0	0	0
SHRE	0.111	0.1	0.25	0	0	0	0	0	0	0	0	0	0	0	0
TERR	0.112	0.1	0	0	0	0	0	0	0	0	0	0	0	0	0
ALGA	0	0.1	0	0.25	0.333	0.07	0.02	0	0	0	0	0	0	0	0
POM	0	0.1	0	0	0	0.584	0	1	0	0	0	0	0	0	0
DIAT	0	0.1	0	0.25	0.333	0.196	0.6	0	0	0	0	0	0	0	0
LEAF	0	0	0	0.25	0	0.15	0	0	1	0	0	0	0	0	0
FILA	0	0.1	0	0.25	0.333	0	0.38	0	0	0	0	0	0	0	0

Table 16: Site 5, partial feeding matrix, in the columns are presented the predator and in the rows the preys. The matrix is estimated normalizing (the columns sum to one) the connections between trophic groups by the total intake of each receiving node.

SITE 1	CARN	OMNI	PRED	HEDE	HERB	COLG	TERR	ALGA	POM	DIAT	HUMW	FILA
CARN	0	0	0.333	0	0	0	0	0	0	0	0	0
OMNI	0.112	0	0.333	0	0	0	0	0	0	0	0	0
PRED	0	0	0	0	0	0	0	0	0	0	0	0
HEDE	0.222	0	0	0	0	0	0	0	0	0	0	0
HERB	0.222	0	0	0	0	0	0	0	0	0	0	0
COLG	0.222	0.145	0.333	0	0	0	0	0	0	0	0	0
TERR	0.222	0.145	0	0	0	0	0	0	0	0	0	0
ALGA	0	0.145	0	0.2	0.25	0.02	0	0	0	0	0	0
POM	0	0.145	0	0.2	0.25	0.19	0	0	0	0	0	0
DIAT	0	0.14	0	0.2	0.25	0.6	0	0	0	0	0	0
HUMW	0	0.14	0	0.2	0	0	0	0	0	0	0	0
FILA	0	0.14	0	0.2	0.25	0.19	0	0	0	0	0	0

Table 17: Site 6, partial feeding matrix, in the columns are presented the predator and in the rows the preys. The matrix is estimated normalizing (the columns sum to one) the connections between trophic groups by the total intake of each receiving node.

A.8 BlenX model for skipjack tuna

Figure 21: The stochastic model done in BlenX skipjack tuna model; .prog file.

```
[steps=1200, delta=0.01]
<<y : inf>>


//Small_Tuna
let pSmallTuna: pproc = eat!().y!().nil

                +

                eat!().(if(not(trapped,SmallTuna_trappedInFads))
                    then
                (ch(rate(SmallTunaRep),dupl,duplication).nil)
                    endif

                +

                if(trapped,SmallTuna_trappedInFads) then
                (ch(rate(SmallTunaRep_trapped),dupl,duplication).nil)
                    endif)

                +

                catch?().die(inf)

                 +

                trapped?().ch(inf,trapped,SmallTuna_trappedInFads).
                ch(inf,catch,SmallTuna_Humancatch_trapped).
                ch(inf,eat,SmallTuna_hunts_trapped).y!().nil
                +

                if(not(trapped,SmallTuna_trappedInFads)) then

                    (delay(rate(SmallTunaDie)).die(inf).nil)
                    endif

                +
                if(trapped,SmallTuna_trappedInFads) then
```

```
                    (delay(rate(SmallTunaDie_trapped)).die(inf).nil)
                    endif

                +

                if(trapped,SmallTuna_trappedInFads) then
                (ch(rate(SmallTuna_becomeFree),trapped,SmallTuna_freeFromFads).
                ch(inf,catch,SmallTuna_Humancatch).
                ch(inf,eat,SmallTuna_hunts).y!().nil) endif;


// small_tuna preys


let pVnimb: pproc = ch(rate(VnimbRep),dupl,duplication).nil +
                        eat?().die(inf) +
                        delay(rate(VnimbDie)).die(inf).nil;


let pepiplfish: pproc = ch(rate(
   epiplfishRep),dupl,duplication).nil +
                        eat?().die(inf) +
                        delay(rate(epiplfishDie)).die(inf).nil;


let pCephal: pproc = ch(rate( CephalRep),dupl,duplication).nil +
                        eat?().die(inf) +
                        delay(rate(CephalDie)).die(inf).nil;


let pCrust: pproc = ch(rate( CrustRep),dupl,duplication).nil +
                        eat?().die(inf) +
                        delay(rate(CrustDie)).die(inf).nil;


let pOther: pproc = ch(rate( OtherRep),dupl,duplication).nil +
                        eat?().die(inf) +
                        delay(rate(OtherDie)).die(inf).nil;



// Fisher:
let pFishers: pproc = catch!().y!().nil;
```

```
// FADs
let pFADs: pproc = attract!().y!().nil;


// BOXES
let SmallTuna: bproc = #(eat,SmallTuna_hunts),
                       #(dupl:0,A),
                       #(trapped,SmallTuna_freeFromFads),
                       #(catch, SmallTuna_Humancatch)
                       [rep y?().pSmallTuna | pSmallTuna];

let Vnimb: bproc =     #(eat,hunts_Vnimb),
                       #(dupl:0,A)
                       [pVnimb];

let epiplfish: bproc = #(eat,hunts_epiplfish),
                       #(dupl:0,A)
                       [pepiplfish];

let Cephal: bproc = #(eat,hunts_Cephal),
                       #(dupl:0,A)
                       [pCephal];

let Crust: bproc = #(eat,hunts_Crust),
                       #(dupl:0,A)
                       [pCrust];

let Other: bproc = #(eat,hunts_Other),
                       #(dupl:0,A)
                       [pOther];

let Fishers: bproc = #(catch,purseseine_All),
                       #(dupl:0,A)
                       [rep y?().pFishers|pFishers];

let FADs: bproc = #(attract,attr_Fish)
                       [rep y?().pFADs|pFADs];


let SmallTuna_Trapped: bproc = #(eat,SmallTuna_hunts_trapped),
                       #(dupl:0,A),
                       #(trapped,SmallTuna_trappedInFads),
                       #(catch, SmallTuna_Humancatch_trapped)
                       [rep y?().pSmallTuna | pSmallTuna];
```

```
let SmallTunadup: bproc = #(eat,SmallTuna_hunts),
                          #(dupl:0,duplication),
                          #(trapped,SmallTuna_freeFromFads),
                          #(catch, SmallTuna_Humancatch)
                          [rep y?().pSmallTuna];

let Vnimbdup: bproc = #(eat,hunts_Vnimb),
                          #(dupl:0,duplication) [nil];

let epiplfishdup: bproc = #(eat,hunts_epiplfish),
                          #(dupl:0,duplication)
                          [nil];

let Cephaldup: bproc = #(eat,hunts_Cephal),
                          #(dupl:0,duplication)
                          [nil];

let Crustdup: bproc = #(eat,hunts_Crust),
                          #(dupl:0,duplication)
                          [nil];

let Otherdup: bproc = #(eat,hunts_Other),
                          #(dupl:0,duplication)
                          [nil];


let SmallTunadup_Trapped: bproc =
                          #(eat,SmallTuna_hunts_trapped),
                          #(dupl:0,duplication),
                          #(trapped,SmallTuna_trappedInFads),
                          #(catch, SmallTuna_Humancatch_trapped)
                          [rep y?().pSmallTuna];

when (SmallTunadup: :inf) split (SmallTuna, SmallTuna);
when (SmallTunadup_Trapped: :inf) split (SmallTuna_Trapped,
   SmallTuna_Trapped);
when (Vnimbdup: :inf) split (Vnimb, Vnimb);
when (epiplfishdup: :inf) split (epiplfish, epiplfish);
when (Cephaldup: :inf) split (Cephal, Cephal);
when (Crustdup: :inf) split (Crust, Crust);
when (Otherdup: :inf) split (Other, Other);

// STARTING SIMULATIONS
run 100 SmallTuna || 1000 Vnimb || 1000 epiplfish || 10000
   Cephal || 10000 Crust || 10000 Other || 10 Fishers || 65 FADs
```

Figure 22: The stochastic model done in BlenX skipjack tuna model; .types file.

```
{duplication ,
A,

SmallTuna_freeFromFads ,
SmallTuna_trappedInFads ,

SmallTuna_hunts ,
SmallTuna_Humancatch ,

SmallTuna_hunts_trapped ,
SmallTuna_Humancatch_trapped ,

purseseine_All ,

hunts_Vnimb ,

attr_Fish ,

hunts_epiplfish ,
hunts_Cephal ,
hunts_Crust ,
hunts_Other
}
%%
{
(SmallTuna_hunts ,hunts_Vnimb ,0.521) ,
(SmallTuna_hunts ,hunts_Crust ,0.028) ,
(SmallTuna_hunts ,hunts_Other ,0.451) ,


(SmallTuna_freeFromFads ,attr_Fish ,0.7) ,

(SmallTuna_hunts_trapped ,hunts_Vnimb ,0.52) ,
(SmallTuna_hunts_trapped ,hunts_Crust ,0.027) ,
(SmallTuna_hunts_trapped ,hunts_Cephal ,0.22) ,
(SmallTuna_hunts_trapped ,hunts_epiplfish ,0.063) ,
(SmallTuna_hunts_trapped ,hunts_Other ,0.17) ,


(purseseine_All ,SmallTuna_Humancatch ,0.09) ,
(purseseine_All ,SmallTuna_Humancatch_trapped ,0.4)
}
```

Figure 23: The stochastic model done in BlenX skipjack tuna model; .func file.

```
let SmallTunaRep : const = 6;
let SmallTunaDie : const = 3;
let SmallTuna_becomeFree : const = 0.3;
let SmallTunaDie_trapped : const = 1.4;
let SmallTunaRep_trapped : const = 4;
let VnimbRep : const = 9;
let VnimbDie : const = 6.5;
let epiplfishRep : const = 7.9;
let epiplfishDie : const = 6.5;
let CephalRep : const = 48.5;
let CephalDie : const = 47.5;
let CrustRep : const = 49.5;
let CrustDie : const = 47.5;
let OtherRep : const = 51.5;
let OtherDie : const = 47.5;
```