New Computational Tools for Analysis of Biological Processes with Application to Environmental Planning

Thesis submitted in accordance with the requirements of the University of Liverpool for the degree of Doctor in Philosophy in the

Networks and Distributed Computing Group
Algorithms Section
Department of Computer Science

by
Daniyah Ali Aloqalaa
May 12, 2019

Abstract

This thesis studies two biological processes in which we develop and use efficient computer science tools such as conductance, maximum flow, optimisation and apply them to the two biological processes. More specifically, the project aims to introduce novel ways to model and analysis a biological process, called invasion process, as well as examine and develop algorithms for solving problems related to the process. The invasion process is a biological process in which many species are invading their ranges to new ranges, in response to global climate change and land use change. We first propose a suitable way to model and estimate the duration of the invasion process in real heterogeneous landscapes using graph sparsification approach. Then, we propose an innovative method to estimate the duration of the invasion process in real landscapes using network flow theory. Furthermore, we propose a new algorithmic method of changing landscapes by combining network flow methodology with optimisation technique, in order to improve the invasion process. Following that, we compare the proposed methods/algorithms with other known approaches and evaluate them using existing real heterogeneous landscapes. The project also aims to study and understand another biological aspect, which is the standard genetic code (SGC). The SGC is the set of rules by which information encoded in genetic material (DNA or RNA sequences) is translated into proteins (amino acid sequences) by living cells. It was proposed that the structure of the SGC evolved to minimise harmful consequences of mutations and transnational errors. To study this problem, we first introduce a new general methodology to describe the general structure of the SGC. This methodology comes from graph theory and allows us to consider multiple factors in the graph, such as mutations types – transition and transversion. Then, we develop a clustering algorithm to find optimal codes according to the conductance; the optimal code is the code that has a robust structure against mutations of different types. Finally, we compare the structure of the resultant optimal codes from implementing the developed clustering algorithm with the structure of the SGC.

Acknowledgements

I am deeply grateful to my supervisor, Prof. Dariusz Kowalski, whose guidance, knowledge and patience added greatly to my experience as a Ph.D. student. Throughout my academic and personal journey during my time as your student, you have always been generous in sharing your time, insights and great experience with me. Thank you for always finding ways to motivate me and inspire me to keep doing research, and for believing in me ever since I met you.

I am especially thankful to my second supervisor, Prof. Prudence Wong, with whom I also worked closely during the course of my Ph.D.. Thank you for acquainting me to collaborators, for guiding me and sharing with me your extensive knowledge, your insight and your creative thinking.

Many thanks also go to my academic advisors, Prof. Leszek Gąsieniec and Dr. Russell Martin, for their time and feedback during my studies. I must also acknowledge all of my co-authors and collaborators for their contributions to the papers we worked on together. I found those experiences both enjoyable and educating.

My four years at the University of Liverpool were funded by Qassim University, Kingdom of Saudi Arabia. I very much appreciate this financial support, as it allowed me to focus on my studies and personal interests. Special thanks also goes to Information Technology Department at the College of Computer, Qassim University.

I would like to offer special thanks to my Dad, who, although no longer with us, always believed in my ability to be successful in the academic area — you are gone but your belief in me has made this journey possible.

Finally, thanks to my Mum, my brothers and sisters — you have been a great support and encouragement throughout my life. The utmost thanks go to my husband for his constant support and patience.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Computational biology is a branch of biology involving the application of computers and computer science to the understanding and modelling the structures of biological processes. The main concern of this thesis is modelling, analysing, and solving related problems to two biological processes. In particular, this thesis aims to develop and use efficient tools in computer science such as conductance, maximum flow, and optimisation technique with the goal of computer modelling, analysis of two biological processes. The first process we study is called an invasion process in which we aim to protect species against rapid climate change in landscapes. The second process we study is the structure of the genetic code in which we aim to protect the genetic code structure against mutations. Below are the formal contexts of the biological processes we consider in this thesis and their related problems.

A high risk arises due to climate and land use changes, as it can put a threat and cause the extinction of various species. In the report provided by Professor Sir John Lawton which is titled as "Making Space for Nature: A review of England's Wildlife Sites and Ecological Network" - submitted to the Secretary of State, the Department for Environment, Food and Rural Affairs in 2010 guided to recognition in higher policy levels as England lacked in a coherent and resilient ecological network. Since then a lot of research has been done by ecologists and others to understand how species response to climate change. More importantly, they attempt to intervene to increase the chances for species to "invade" landscapes of a more suitable environment to survive. One of the main methodologies is to use mathematical/computational model to capture the characteristics of environment and species so that it is possible to use some existing mathematical tools to solve related formula and explain the invasion process. Nevertheless, it is important not only to understand the

process but to be able to suggest how to modify or adapt some landscapes to assist the process. This is a challenging problem and the existing approaches involve solving very hard computational problems in the sense that any solution would consume longer hours to operate.

This thesis targets this problem and proposes new and novel ways to model the problem by application of a powerful computational technique termed as network (or graph) theory. In a more evolved manner, we adapt some existing solutions to our proposed new models. The most crucial point is that we propose new solutions that are much more efficient (less computational intensive) than the current ones. It is interesting that the biological problem we address has a flip side. Instead of finding a way to assist species to invade a certain environment, it is equally important to disrupt the invasion in other situations. As for example disease and pest control in farming play an important role where the invasion of farmland by pests can be reduced by stopping the diseases and pests. By the same models we propose, we can understand what makes it favourable for the diseases and pests to invade and hence a better disruption process can be proposed which will later benefit both communities.

In this thesis, we also target another biological problem which is related to studying the optimality of the general structure of the standard genetic code. The standard genetic code is the set of rules by which genetic information is translated into proteins, from codons, i.e., triplets of nucleotides, to amino acids. The questions about the origin and the main factor responsible for the present structure of the code are still under a hot debate. Various methodologies have been used to study the features of the code and assess the level of its potential optimality. In this thesis, we aim to introduce a new general approach to evaluate the quality of the genetic code structure. This methodology also comes from graph theory and allows us to find some new optimal structures of the genetic codes in order to minimise harmful consequences of mutations and transnational errors.

1.1 Background

1.1.1 Biological relevance

As per the reviews provided by Lawton (2010) in [81] there has been a high-level of policy recognition that England lacked in a coherent and resilient ecological network and that the level of fragmentation of habitat is high (particularly in lowland areas). There is a

major concern that climate and land use change could threat and cause the extinction of many species [68, 73, 98]. Species are observed to respond to these threats by shifting their geographic distributions [25]. However, the abilities of species to shift can be limited as the habitats lacked in resources that are required for their colonisation [66, 72, 107, 113]. A high risk will evolve towards species extinction if species shift too slowly or in case if they are unable to shift at all [108]. Therefore, to maintain ecosystem function and ecosystem services in a changing climate, it becomes important to facilitate the adaptation of species, especially by enabling them to shift to new locations with more suitable climate and landscape environment [69].

Policymakers and nature conservation organisations are increasingly aiming to restore habitat in a coordinated way at large spatial scales through a variety of land management mechanisms [64, 80]. Hodgson et al. [68, 69] showed that the spatial arrangement of restored habitat could make an enormous difference on the speed of range shifting achieved by species. Corridors and stepping stones are considered to be important tools while other spatial patterns may possess better properties to speed up shift and colonisation [69]. However, there is still a lack of systematic and efficient approaches to determine "automatically" the best way to increase the speed of range shifts.

This invasion process can also be viewed from the flip side where one would aim to control spread of disease or pests. Studying how these species invade gives clues and provides ideas on how to stop them from affecting crops and hence economic output of a farm. In contrast to enhancing colonisation by finding corridors or stepping stones that can help the invasion, here it would be to find the "bottleneck" of invasion process and to alter the characteristics of these areas such that it no longer favours spreading of the disease or pests. It is viable to capture both sides of the picture into the same problem. In this thesis, we model the invasion process as an abstract computational problem. Providing a solution to this computational problem will shed light on further understanding of both biological problems.

1.1.2 Some existing biological approaches

The previous researches on dynamics of range expansion focuses on homogenous landscapes [106, 110] but more recent research has considered fragmented and heterogeneous landscapes [55, 69]. Different models and techniques have been developed like individual-based [46], using dispersal kernels [110], Bayesian techniques [104], integro-difference mod-

elling methods, electric circuit theory [32, 69], Markov model (Bocedi et al. [17] gave a good review).

The process of dispersal can be seen as three key stages: emigration, transfer and settlement [29, 30], each of which has its own mechanism. This involves the study of reproduction probability and the risk of population extinction. Nevertheless, there is still a lack of theoretical understanding of how behaviour of species at these three stages affect the rate of range shifts [17]. It is desirable to understand more about the relationships between emigration, transfer and settlement rules and the rate of population spread. A platform, called RangeShifter [18], has been developed recently attempting to model eco-evolutionary dynamics and it is based on known models [30] of the three stages.

1.1.3 Key thesis questions

Invasion process. While a lot of efforts focus on modelling the population dynamics process, it is unclear how one might intervene or improve the process.

- Will improvement/modification in a certain landscape be helpful?
- How can the best locations be determined for intervention if we have a limited amount of resources to spend?

Genetic code. In order to study and understand the features of the genetic code, a lot of various methodologies have been used, however to the best of our knowledge, there is no study that shows how to use the graph representation to understand the genetic code and evaluate the quality of its structure.

- How resilient is the genetic code to mutations of different types?

1.1.4 Computer science relevance

Network flow and rumor spreading

In this thesis we aim to give a new model for the invasion process using graph theory and in particular network flow, from which we will develop an efficient and effective software tool to ease the study of the process. Network flow is a fundamental problem studied in Computer Science and has a wide variety of applications. Initially, some explanation of the fundamental concepts is provided and then in the next section, the relationships between the concepts and the invasion process is described.

In its simplest form, a network [93] is a collection of points (called *vertices* or *nodes*) joined together in pairs by lines (called *edges*). There are various systems which can be represented as networks as for example metabolic network representing metabolic reactions among metabolite, the neural network representing synapse among neurons. The properties and structure representing a graph often give a lot of information about the system the graph that it represented. For fulfilling the purpose of this thesis, the following concepts are of particular value: *independent paths*, *connectivity*, *minimum cut set*, and *maximum flow*. Roughly speaking these concepts measure the "bottleneck" of connection among the vertices.[1]

The concept of maximum flow can be explained as follows. Assuming an edge has "capacity" that measures the amount of traffic can go through it, then we can ask the question of how much traffic in total can go through the whole network. The problem of finding the maximum flow has been studied for a long time period and there are well established algorithms to find such flow efficiently [31]. Network flow has been employed in studying connectivity in landscapes in the form of dispersal corridors [98]. To the best of our knowledge, there is no theoretical evidence of how to efficiently use network flow for estimating characteristics of invasion processes in a way allowing optimisation of these characteristics by intervening the system within a bounded quantity of resources.

Measuring the time of similar processes on networks, such as broadcast or rumor spreading, has been studied before and this is closely related to our work here to measure the expected time of the invasion process. Rumor spreading is a fundamental communication process aiming to disseminate the source message to all nodes in a network [74]. The notion of *conductance* was used to measure how well the vertices in a network are connected [24] and hence is useful for determining how long the rumor spreading process takes [26]. In this thesis, we identify the relationship of *conductance* with the invasion process and develop a model to capture this relationship.

[1]For any pair of vertices, a set of paths connecting them is *edge-independent* (resp. *vertex-independent*) if they do not share any common edges (resp. vertex). The number of independent paths between a pair of vertices is called the *connectivity* of the vertices. Connectivity can also be visualised in terms of "bottleneck" between vertices. Formally speaking, the idea of bottleneck is related to the notion of cut sets. A set of vertices is a *cut set* if its removal will disconnect some pair of vertices, and a *minimum cut set* is the smallest such cut set.

1.2 Thesis aims and objectives

The primary aim of this thesis is to: *combine the network flow approach with optimisation technique and apply it to the population invasion problem in order to improve the invasion process.*

This thesis also intends to represent the general structure of the genetic code by a graph and include information about mutation types (transition and transversion) into the graph representation, in order to find an optimal genetic code graph partition such that the code is robust against mutations.

These above aims raise the following core thesis objectives:

- Propose a suitable way to model the invasion process in a given real landscape using graph sparsification approach.
- Propose an innovative model for the invasion process based on network flow theory.
- Propose a new algorithmic method of changing the network, hence the landscape, by combining network flow with optimisation technique, to improve the invasion process.
- Evaluate the proposed methods/algorithms that mentioned above using existing real heterogeneous landscapes.
- Develop an appropriate clustering algorithm to find an optimal structure of the genetic code with respect to the minimum average code conductance.

1.3 Summary of our results

In this section, we provide a short overview of the problems we considered and the results we derived.

1.3.1 Modelling and estimating duration of the invasion process using graph sparsification approach

In this work we propose three formulas that estimate the duration of the invasion process in heterogeneous landscapes. We also propose a new simulation method, called *R-local*, to model the invasion process using a tool of landscape network sparsification to efficiently approximate the duration of the process. More specifically, we aim to simplify the structure

of large landscapes using the concept of sparsification in order to substantially decrease the time required to compute a good estimate of the invasion time in these landscapes. We propose a formula that estimate parameter R for efficient *R-local* simulations. We then compare how good our proposed *R-local* simulation method is with respect to the previously used *full* simulation method (state-of-the-art method), by applying these methods to real heterogeneous landscapes in Great Britain. We also examine how the duration of the invasion process is affected by different factors, such as dispersal coefficient, landscape quality and landscape size. We finally evaluate the proposed *R-local* simulation method by performing *full* and *R-local* simulations on 36 landscape, which selected randomly. The extensive evaluations that have been carried out show that the proposed simulation method guaranties high simulation accuracy and at the same time, a huge saving on computational resources such as time and memory compared to the previously used simulation method.

A preliminary version of this work was published in [6] and an extension of it [3] is under review for publication in the Journal of Biological Systems at the time of the write-up of this thesis. The details of this part of our work can be found in Chapter 3.

1.3.2 Modelling and estimating duration of the invasion process using network flow theory

From computational perspective, estimating the invasion time by running simulations is very time consuming, as the *full* simulation method (state-of-the-art method) is based on a Markov Chain of exponential number of states with respect to the landscape size; therefore, in practice, this simulation method is not suitable especially in case of frequent environmental changes or for environmental planning. Following that, we propose a new way to estimate the time of the invasion process using a powerful computational approach based on conductance and network flow theory. More specifically, in this part of out work we give a new formula for estimating the invasion time using a combination of network flow methodologies, and prove asymptotic bounds on the quality of the obtained approximation. The proposed approach is analysed mathematically and applied to real heterogeneous landscapes of Great Britain to estimate the duration of the process; the theoretical bounds obtained are compared with simulation results. The evaluations of the proposed approach demonstrate its accuracy and efficiency in approximating the invasion time.

This part of our work was published in [4] and presented in details in Chapter 4.

1.3.3 Manipulating/Improving landscapes to minimise the duration of the invasion process

In this work we aim to manipulate landscapes within some pre-defined budget in order to minimise the duration of the invasion process. We first use our new method in [4] (which will be presented later in Chapter 4), that models and approximates the time taken by species to invade landscapes and reach the new areas of suitable environment. Based on this, we propose and test a new method that can help to compute the best locations in landscapes in order to restore habitat which leads to minimising the expected time taken by species to invade landscapes and reach targets. Following that, we compare our new optimisation method with two baseline methods by implementing them on real heterogeneous landscapes of Great Britain to produce improved landscapes. On these improved landscapes (i.e., the produced landscapes by our new optimisation method and the two baseline methods), we run simulations to compute the duration of the invasion process and then find out how the duration is influenced by landscapes' modifications. We finally show that the new optimisation method outperforms the competitive baseline methods in terms of proposing landscapes' modifications that minimise the expected time of the invasion process.

This part of our work was published in [5] and presented in details in Chapter 5.

1.3.4 The impact of the transversion/transition ratio on the optimal genetic code graph partition

The standard genetic code (SGC) is a system of rules ascribing 20 amino acids and stop translation signal to 64 codons, i.e triplets of nucleotides. It was proposed that the structure of the SGC evolved to minimise harmful consequences of mutations and transnational errors. To study this problem, we describe the SGC structure by a graph, in which codons are vertices and edges correspond to single nucleotide mutations that may occur between the codons. We also introduce weights (W) for mutation types to distinguish transversions from transitions. Transition, in genetics and molecular biology, refers to a point mutation in deoxyribonucleic acid (DNA) that changes a purine nucleotide to another purine (i.e., Adenine [A]↔Guanine [G]), or a pyrimidine nucleotide to another pyrimidine (i.e., Cytosine [C]↔Thymine [T]). Transversion, in molecular biology, refers to a point mutation, where a purine is changed for a pyrimidine, or vice versa (i.e., A↔C, A↔T, G↔C, G↔T). Using the graph representation, the SGC is a partition of the set of vertices into 21 disjoint subsets. In this case, the question about the potential robustness of the genetic code to the mutations

can be reformulated into the optimal graph clustering task. To investigate this problem, we apply an appropriate clustering algorithm, which searched for the codes characterised by the minimum average W-conductance calculated from the set W-conductance of codon groups. Our algorithm found three optimal codes for various ranges of the applied weights. The average W-conductance of the SGC was the most similar to that of the optimal codes in the range of weights corresponding to the observed transversion/transition ratio in natural mutational pressures. However, it should be noted that the optimisation of the SGC was not as perfect as the optimal codes. It implies that the evolution of the SGC was driven not only by the selection for the robustness against mutations or mistranslations but also other factors, e.g., subsequent addition of amino acids to the code according to the expansion of amino acid metabolic pathways.

This part of our work was published in [2] and presented in details in Chapter 6.

1.4 Thesis outline

The material covered in this thesis has been arranged in the following way:

Chapter 2

This chapter introduces notation used throughout this thesis, the general model of the invasion process, and the dataset used. It also describes the state-of-the-art simulation method, called *full* simulation, which is used to compare with the proposed methods/algorithms in Chapters 3-5.

Chapter 3

This chapter introduces a new model of invasion process using network sparsification in order to efficiently estimate the duration of the process. Here, we aim to simplify the structure of large landscapes using the concept of sparsification in order to substantially decrease the computational resources (time and memory), which required to compute a good estimate of the invasion time in these landscapes. The work presented in this chapter was published in [6].

Chapter 4

This chapter presents a novel approach to measure the invasion time in real landscapes based on combinations of conductance and network flow theory. The work presented in this chapter was published in [4].

Chapter 5

This chapter introduces a new optimisation method that propose modifications on landscapes within a bounded quantity of resources in order to minimise the invasion time. The new optimisation method is based on the developed method in Chapter 4, which approximates the invasion time in real landscapes. The work presented in this chapter was published in [5].

Chapter 6

The work presented in this chapter focuses on studying the general structure of the genetic code. In particular, in this chapter we introduce a new clustering algorithm, using a methodology adapted from graph theory, to find some optimal structures of the genetic code in terms of robustness against single nucleotide mutations. The work presented in this chapter was published in [2].

Chapter 7

The final chapter summarises the results of this thesis.

Each of the Chapters 3-6 concludes with a set of open problems and/or possible further research questions that may be of interest to the reader.

1.5 Author's publications included in this thesis

Table 1.1: The author's publications and co-authors throughout the duration of her Ph.D. studies.

Title	*Authors*	*Appeared*
The Impact of Landscape Sparsification on Modelling and Analysis of the Invasion Process [6]	D. A. Aloqalaa, J. A. Hodgson, and P. W.H. Wong	SEA 2017[2]
The Impact of Landscape Sparsification on Modelling and Analysis of the Invasion Process [3] (journal version)	D. A. Aloqalaa, J. A. Hodgson, D. R. Kowalski, and P. W.H. Wong	JBS (under review)[3]
Estimating Invasion Time in Real Landscapes [4]	D. A. Aloqalaa, J. A. Hodgson, D. R. Kowalski, and P. W.H. Wong	ICCBB 2018[4]
Testing Methods to Minimise Range-shifting Time with Conservation Actions [5]	D. A. Aloqalaa, J. A. Hodgson, D. R. Kowalski, and P. W.H. Wong	ICBBT 2019[5]
The impact of the transversion/transition ratio on the optimal genetic code graph partition [2]	D. A. Aloqalaa, D. R. Kowalski, Paweł Błażej, Małgorzata Wnetrzak, Dorota Mackiewicz and Paweł Mackiewicz	BIOINFORMATICS 2019[6]

[2]SEA 2017: the 16$^{\text{th}}$ International Symposium on Experimental Algorithms, 2017

[3]JBS: Journal of Biological Systems

[4]ICCBB 2018: the 2$^{\text{nd}}$ International Conference on Computational Biology and Bioinformatics, 2018

[5]ICBBT 2019: the 11$^{\text{th}}$ International Conference on Bioinformatics and Biomedical Technology, 2019

[6]BIOINFORMATICS 2019: the 10$^{\text{th}}$ International Conference on Bioinformatics Models, Methods and Algorithms, 2019

Chapter 2

General Model of Invasion and Preliminaries

This Chapter introduces the notation used throughout Chapters 3-5 and the general model of the invasion process. It also describes the state-of-the-art of simulation method in [69], called a *full* method, which is used later to compare with the proposed methods in Chapters 3-5. We then presents the dataset used in Chapters 3-5. For convenience, we finally present all notations used in these chapters in Table 2.4.

2.1 Notation and technical preliminaries

We are given a two-dimensional rectangular grid landscape of height H (rows) and width W (columns) as an input, which we represent as a directed landscape graph $G = (V, E)$. Denote by $n = |V|$ the number of vertices (patches) in the landscape graph, by $m = |E|$ the total number of directed edges between vertices, and $deg(v)$ be the out-degree of a vertex $v \in V$. Let $q(v)$ denote the quality of a patch v, where the quality is a number between zero and one given as an input. Denote by Q the sum of quality over all vertices in a landscape graph G and it is defined as: $Q(G) = \sum_{v \in V} q(v)$. We distinguish two sets of patches, S and T, where S denotes the set of populated source patches and T denotes the set of unpopulated target patches. Both S and T have non-zeros values of quality. In addition, we define the maximum and minimum quality as $q_{max} = \max\{q(v) : v \in V\}$ and $q_{min} = \min\{q(v) > 0 : v \in V\}$, respectively. We also define *R-local* graph $Loc(G, R) = (V', E')$ as a subgraph of G that contains all the vertices of the landscape graph G (i.e., $V' = V$) and a set of edges $E' \subseteq E$

such that for any $(u, v) \in E$, $(u, v) \in E'$ if the Euclidean distance between vertices v and u is at most R. For a defined R-*local* graph $Loc(G, R)$, we define d_{min} as the minimum distance d such that $Loc(G, d)$ connects S to T, i.e., d_{min} is the smallest value of d such that for every vertex $v \in T$ there is a path from some vertex $u \in S$ in graph $Loc(G, d)$.

For any subset of vertices $X \subseteq V$, vol(X) denotes the *volume* of X and it is defined as $\sum_{v \in X} \sum_{u \in V, (v,u) \in E} p(v, u)$, where $p(v, u)$ is the weight of edge (v, u). We define the maximum and minimum vertex degree as follows: $deg_{max} = \max\{deg(v) : v \in V\}$ and $deg_{min} = \min\{deg(v) : v \in V\}$. For any set of vertices $X \subseteq V$ and any vertex v, we denote by $deg_X(v)$ the degree of v restricted to set X of out-neighbours, i.e., $deg_X(v)$ is the number of vertices in X adjacent (i.e., connected by an edge) to node v in the graph. For any two sets of vertices $X, Y \subseteq V$ such that $X \cap Y = \emptyset$, we define $E(X, Y)$ to be the set of edges from vertices in X to vertices in Y. For any set of vertices $X \subseteq V$, we define the complement of X, denoted by X^c, as $\{v \in V : v \notin X\}$.

We use the formula of colonisation probability proposed by Hodgson et al. [69] to define the *transition probability* $p(v, u)$ between patches v and u as

$$p(v, u) = q(v) \cdot \frac{\exp\left(-\alpha d(v, u)\right)}{\left(\frac{2\pi}{\alpha^2}\right) - 1} \ , \tag{2.1}$$

where $\alpha > 0$ is the *dispersal coefficient* which is dependent on the species and is assumed to be the same for all patches and $d(v, u)$ is the Euclidean distance between patches v and u.

2.2 Invasion model using *Full* simulation method

In our work we simulate the behaviour of the invasion process by building an invasion model (a simulator) that uses the original approach in [69], which we call the *full* method, to compute the number of rounds needed for invasion. The input parameters to the simulator are: a two-dimensional array that represents a given real landscape and stores qualities of patches, a source vector containing indices of populated patches, a target vector containing indices of unpopulated target patches, and *dispersal coefficient* α which is a positive number (see Table 2.1). For a given landscape, the simulator constructs a two-dimensional array of size equal to the one of the given landscape. Each cell in the constructed array corresponds to a patch in the landscape and can take only two values, zero or one, where zero means the cell is unpopulated while one means it is populated. At the beginning of the invasion process, only the source populated patches take value of one and others take value of zero.

The simulator returns an estimated duration needed (number of rounds) to invade landscape and reach targets by the use of real probabilities for each pair of patches v, u, in which v is populated and u is not. We use *transition probabilities* (Equation (2.1)) between patches v and u to decide whether patch v populates patch u or not.

More formal description of the structure of the *full* simulation method is given in Algorithm 1. The generic structure of the *full* method contains inputs (as mentioned above), outputs (cf., output variables in Table 2.1), and COUNT ROUNDS function. The COUNT ROUNDS function counts the number of rounds required for successful invasion and to compute the real time execution for each simulation (i.e., the time taken by a computer system to run or execute simulation). The function includes nested loops of three levels. The main loop (lines 7-18) counts the number of rounds to populate target patches. The second level loop (lines 9-18) is for all populated patches that are trying to populate unpopulated patches. The inner level loop (lines 12-18) is for all unpopulated patches. Each unpopulated patch becomes populated if the *transition probability* (Equation (2.1)) between the populated and unpopulated patches is greater than a generated random number between zero and one (lines 15-18). We consider only populating a patch with non-zero quality because patch has zero value means "no habitat" and thus the species could not reproduce there. The COUNT ROUNDS function terminates when all (or any of) non-zero target patches become populated and returns the number of rounds needed for successful invasion as well as the execution time of the simulation.

Table 2.1: Input parameters and output variables for Algorithm 1.

Input parameters:
1. G: 2-dimensional array stores qualities of patches in a given real landscape
2. S: vector containing indices (i, j) of initial populated patches (source patches)
3. T: vector containing indices (i, j) of unpopulated target patches
4. α: given number > 0
Output variables:
1. Number of rounds needed for successful invasion (invasion time)
2. Execution time of simulation

Algorithm 1 Modelling invasion process using *full* simulation method (G, S, T, α)

```
1: function COUNT ROUNDS(G, S, T, α)
2:     start ← record start execution time of simulation
3:     Create 2-dimensional array B having size equal to array G ← 0
4:     for each populated patch (i, j) in vector S do
5:         B(i, j) ← 1
6:     Rounds ← 0                          ▷ counter to count rounds for successful invasion
7:     while all (or any of) target patch in T is unpopulated do
8:         Rounds ← Rounds+1
9:         for i ← 0 to number of rows in G do          ▷ loop for all populated patches
10:            for j ← 0 to number of columns in G do
11:                if patch B(i, j) is populated then
12:                    for z ← 0 to number of rows in G do ▷ loop for all unpopulated patches
13:                        for l ← 0 to number of columns in G do
14:                            if patch B(z, l) is unpopulated and G(z, l) ≠ 0 then
15:                                p ← Transition probability between B(i, j) and B(z, l)
16:                                w ← Generate a random number between 0 and 1
17:                                if w < p then
18:                                    Populate patch B(z, l)
19:    end ← Record end execution time of simulation
20:    ExecutionTime ← end - start
21:    return Rounds, ExecutionTime
```

This *full* simulation method is time consuming, and what is more, it is difficult to analyse and stop at proper time as it is based on a Markov Chain of exponential size wrt the size of the landscape. Therefore, we aim to develop new efficient methods in Chapters 3 and 4 to estimate the number of rounds of the invasion process in a more efficient way.

2.3 The dataset used in Chapters 3-5

For evaluation purposes of the proposed methods (algorithms) in Chapters 3-5, the dataset of the 1km resolution raster version of the Land Cover Map 2007 (LCM2007) for Great Britain [91] is used. To examine the proposed methods (algorithms), different sized landscapes from different maps of the aggregate classes are extracted. The aggregate classes data contain one tiff file for each land use class as in Table 2.2, some of these files are used to extract landscapes.

Table 2.2: Aggregate land classes [91].

Aggregate class	Aggregate class number
Broadleaf woodland	1
Coniferous woodland	2
Arable	3
Improved grassland	4
Semi-natural grassland	5
Mountain, heath, bog	6
Saltwater	7
Freshwater	8
Coastal	9
Built-up areas and gardens	10

Each map consists of 1300 rows/height (pixels) and 700 columns/width (pixels) and each 1km^2 pixel provides the percentage cover of a particular land cover at LCM2007 Class level [91]. We consider the percentage cover at each patch in such an extracted landscape as the quality of each patch. For examination purposes, from percentage values we formed three groups of landscape qualities, namely: low quality, medium quality and high quality to represent the quality of the extracted landscape. If the average of all patches qualities in such an extracted landscape is between 0% and 5%, 5% and 25%, 25% and 100%, then the extracted landscape is of low quality, medium quality and high quality, respectively (Table 2.3).

Table 2.3: Types of landscape quality.

Landscape quality	Average of qualities
Low quality	0-5%
Medium quality	5-25%
High quality	25-100%

2.4 Summary of notations used in Chapters 3-5

Table 2.4: Summary of notations used in Chapters 3-5.

	Description	Support
α	Dispersal coefficient	$\alpha \in \mathbb{R}^+$
G	A landscape graph $G = (V, E)$ where vertices each represents a patch and each patch is connected by a directed edge E to each other patches	
n	Number of patches in a landscape graph G	
m	Number of edges in a landscape graph G	
H	Number of rows in a landscape graph G	$H \in \mathbb{R}^+$
W	Number of columns in a landscape graph G	$W \in \mathbb{R}^+$
S	A vector containing indices of initial populated patches (source patches)	
T	A vector containing indices of unpopulated target patches	
$q(v)$	Quality of patch v, which is a given number between 0 and 1	$q(v) \in \{0, 1\}$
$q_{min}(G)$	Minimum quality in a landscape graph G	$q_{min}(G) \in \mathbb{R}\ 0 < q_{min}(G) \leq 1$
$q_{max}(G)$	Maximum quality in a landscape graph G	$q_{max}(G) \in \mathbb{R}\ 0 < q_{max}(G) \leq 1$
$Q(G)$	The sum of quality over all vertices in a landscape graph G and it is defined as: $Q(G) = \sum_{v \in V} q(v)$	
$R(G)$	A local distance needed to create a landscape subgraph $Loc(G, R)$ of a landscape graph G	$R(G) \in \mathbb{R}^+$

$Loc(G, R)$	A subgraph $Loc(G, R) = (V', E')$ of G that contains all vertices of G (i.e., $V' = V$) and a set of edges $E' \subseteq E$ such that for any $(u, v) \in E$, $(u, v) \in E'$ if the Euclidean distance between vertices v and u is at most R	
$d_{min}(G)$	Minimum distance d such that every vertex in T is reachable from some vertex in S in graph $Loc(G, d)$	$d_{min}(G) \in \mathbb{R}^+$
$d(v, u)$	Euclidean distance between patch v and u	$d(v, u) \in \mathbb{R}^+$
$p(v, u)$	*Transition probability* between patch v and u using Equation (2.1)	$p(v, u) \in \mathbb{R}\ 0 \leq p(v, u) \leq 1$
$deg(v)$	The out-degree of a vertex $v \in V$	
deg_{min}	Minimum vertex degree and it is defined as $\min\{deg(v) : v \in V\}$	
deg_{max}	Maximum vertex degree and it is defined as $\max\{deg(v) : v \in V\}$	
$deg_X(v)$	The degree of vertex v restricted to set X of out-neighbours	$X \subseteq V$
$\text{vol}(X)$	The *volume* of X and it is defined as $\sum_{v \in X} \sum_{u \in V, (v,u) \in E} p(v, u)$	$X \subseteq V$
$E(X, Y)$	The set of edges from vertices in X to vertices in Y	$X, Y \subseteq V$
X^c	The complement of X and it is defined as $\{v \in V : v \notin X\}$	$X \subseteq V$

Chapter 3

Modelling Invasion Process using Graph Sparsification Approach

We present here a new proposed simulation method of the invasion process, called *R-local* simulation method, using a tool of landscape network sparsification to efficiently estimate a duration of the process (invasion time). More specifically, we aim to simplify the structure of large landscapes using the concept of sparsification in order to substantially decrease the time required to compute a good estimate of the invasion time in these landscapes. We compare the proposed method (*R-local*) with the state-of-the-art method (*full* simulations) presented in Chapter 2, which are based on the concept of sparse and dense networks, respectively. Additionally, we show the results of applying these two methods to real heterogeneous landscapes in Great Britain to compute the total estimated time taken by species to invade landscapes and reach targets. We finally evaluate the proposed *R-local* simulation method and show that our new method approximates the duration of the invasion process to high accuracy (more than 90% accuracy) using a substantially reduced computation time [1] (as high as 75 times faster) and memory.

[1]The time spent by a computer system to run or execute simulations.

3.1 Overview

3.1.1 The problem and our model

Climate and land use changes are two threats that cause the extinction of numerous species [68, 73, 98, 113]. It was observed that species are responding to climate change by shifting their geographical area [25, 112], however the ability of their population to shift depends on the availability of suitable habitat to shift and colonise [68, 72, 98, 107]. Species become under a high risk of extinction if they are shifting very slowly or do not have the ability to shift [108]. Therefore, to maintain the functioning of an ecosystem and services in a changing climate, it becomes an important need to facilitate the adaptation of species, especially by enabling them to shift to new locations with more suitable climate [69]. It is an urgent need for policymakers and nature conservation organisations to find out whether and how they can facilitate range shifts [69]. However, it is not easy to make decisions that can facilitate range shifts in fragmented landscapes in the sense that any decision would need to be taken under running/testing many scenarios and each scenario would take very long time to run.

In this work, our focus is to build a new sparse computational model for the invasion process using the network modelling approach. To model the invasion process we consider the following scenario. For a given landscape, we create a landscape network, where each node represents a patch of habitat (henceforth patch) in the landscape. We distinguish two sets of patches: the source patches represent initially populated patches in which species are located, and the target patches represent the target locations for the invasion process. The invasion process is to populate (some) target patches and we aim to estimate the time needed for achieving this. A stochastic model from [69] has been implemented in the simulation, which is based on the probability of a patch to be invaded expressed by a formula depending on various characteristics (distance, quality of patch, etc.) of all other patches in the network. Such simulation is computationally expensive, especially when the number of patches is large.

In this chapter, we propose to approximate invasion time by exploiting network sparsification. Recall that in the protocol implementing *full* invasion method (presented in Chapter 2), in each round of the invasion process each populated patch tries to populate (independently) all other unpopulated patches in the landscape. Here, we propose a more general *R-local* invasion method, and associated protocol, such that in each round, every

patch only tries to populate other unpopulated patches within a local distance R. The *full* invasion protocol can be seen as an R-*local* invasion protocol with R equal to the diameter of the landscape. With a smaller R, the R-*local* invasion process is expected to take less computation time and memory in each round of computation, while it may take more rounds (duration of the invasion process) to populate the target patches. If the local distance R is chosen properly, the R-*local* invasion protocol can compute a comparable (to the *full* method) duration of invasion process, while the computation time can be substantially reduced.

Technically speaking, the work presented in this chapter combines ideas from probability and random processes [60, 78, 88] with some use of network (graph) foundations [93].

3.1.2 Related work and challenges

Related work

A number of empirical and theoretical studies have been devoted to show that spatial arrangement of habitats is an important factor that affects the invasion time (named as speed of advance in [68, 69]) to new landscapes with more suitable climate [68, 69]. Different simulation methods have been used in these studies to observe the invasion time in a fragmented landscape in the United Kingdom. Hodgson et al. [68] used a modified version of the Incidence Function Metapopulation model (IFM [63]), while in [69] four different metrics have been used to predict the invasion time. In more details, these metrics are conductivity, maximum flow, reciprocal of the length of the shortest path, and reciprocal of the length of the multiple shortest paths. The authors illustrated that the conductivity metric, which is derived from electric circuit theory, predicted the invasion time better than other candidate metrics. While there is a correlation between conductivity and invasion time, however, it is difficult to say how to scale conductivity to predict the invasion time with high accuracy.

From computer science perspective, the problem which is most related to the invasion process is rumor spreading in graphs. It is a popular communication process for disseminating information to all nodes in networks [74]. The most popular and simple rumor spreading protocols are: Push, in which a node with information sends it *to* a random neighbour, and Pull, in which a node without information requests it *from* a random neighbour. The time of rumor spreading was estimated using various graph-theoretical measures, including graph conductance, node expansion and others (cf., [77]). While the rumor spreading processes

can be seen as the invasion process, the invasion process studied in this work differs from PUSH and PULL processes in many ways. One of the main differences is that the PUSH and PULL processes have been analysed in undirected unweighted graphs, while the graphs are directed and weighted (different weights on edges) in the case of invasion. Another difference is that in the PUSH and PULL processes actions are made independently at nodes, while in the invasion process — at edges.

Challenges

Whilst it has been shown in [68, 69] the benefits of using different tools such as habitat corridors and stepping stones to speed up shifting, however, it is still difficult for conservationists to make decisions that can facilitate range shifts in large landscapes, therefore there is a need for a tool efficiently computing the invasion time of the original and modified landscapes. Minimising computation time is especially important because the ultimate aim is for a decision-making tool that can tune the arrangement of the modified landscape to find scenarios where a small addition of habitat leads to a large decrease in invasion time. Running many scenarios with different permutations of habitat could require excessive computation times even with moderately-sized landscapes. Furthermore, planning for climate change requires the consideration of large landscapes (e.g., temperature isoclines are expected to shift at several km per decade).

The main challenge in this part of our work is:

> *how to estimate the duration of the invasion process especially in large landscapes in a fast way with high accuracy.*

We first exploit the concept of network sparsification in the context of modelling and analysis the invasion process, in a way where each node in the network is connected only with nodes of some small distance R. Furthermore, the second related challenge is to determine the best choice of the local distance R such that we get a good trade-off between quality of estimate the invasion time and actual computation time.

3.1.3 Contributions and organisation of the chapter

Contributions

Our contribution is three-fold. We firstly propose three formulas that estimate the duration of the invasion process based on the formula of the *transition probability* in Equation (2.1).

These formulas estimate well the invasion time for three types of invasion, based only on some global parameters of a given landscape: first success (Equation (3.1)), majority success (Equation (3.2)), and all successes (Equation (3.3)). The first success invasion measures time until *any* of target patches gets populated, the majority success continues until *majority* of target patches are invaded, and finally the all successes process measures invasion time when all target patches get populated. Importantly, we also propose a formula that estimates parameter R for efficient R-*local* simulation, cf., Equation (3.4).

Secondly, we compare how good our proposed R-*local* simulation method is with respect to the previously used *full* simulation method, by applying these methods to real heterogeneous landscapes in Great Britain. Further, we illustrate how to determine the local distance R systematically to reach a good trade-off between quality of estimate the invasion time and computation time (execution time of simulation). We then extrapolate, by simulation, constant c in the theoretical formula for R in Equation (3.4), in order to make it even more accurate. We define three objective functions to interpolate constant c in Equation (3.4), namely: the Euclidean distance objective function, the absolute objective function and the min-max objective function. For the selected best method of extrapolation, the obtained constants vary between 0.6 and 0.75 depending on scenarios; the small variation shows that our theoretical formula is quite accurate even without more careful extrapolation.

Finally, we validate our proposed R-*local* method by performing simulation on 36 different real landscapes. The extensive experimental validation with real data illustrates the effectiveness and accurateness of the proposed R-*local* protocol, run for value R computed based on our theoretical formula (Equation (3.4)), in estimating the duration of invasion process in moderately-sized landscapes in a relatively short computation time, with respect to the previously used *full* method. More specifically, we show that the R-*local* method approximates the invasion time to more than 90% accuracy and speeds the computation time up to 75 times faster than the *full* method even for relatively small landscapes of 1000 patches, and is growing rapidly with dispersal parameter α (to be introduced later) and with the landscape size (for low and medium quality landscapes, while surprisingly the speedup remains stable, with small fluctuations, for high quality landscapes).

Organisation of the chapter

Section 3.2 presents the types of successful invasions we consider in this chapter. Section 3.3 describes the proposed R-*local* simulation method. Section 3.4 introduces the considered

landscapes used to study and examine the proposed method in this chapter. Section 3.5 derives formulas to estimate the duration of invasion process and illustrates methodology to compute the local distance R properly. Section 3.6 focuses on interpolating a constant in the formula of R and presents the results for the considered dataset. Section 3.7 describes the experimental framework used to evaluate the proposed *R-local* method and its results. Section 3.8 illustrates the improvement in memory and computation time using the proposed method. Finally, Section 3.9 discusses some possible future works.

3.2 Invasion types

In this chapter, we consider three types of successful invasions, namely: *first* success, *majority* success and *all* successes invasion. For a given landscape graph, distinguished source and target patches, we firstly define "all non-zero target patches" as the total number of target patches that are non-zeros in quality, and "majority of all non-zero target patches" as the number of patches being more than half of the total number of non-zero quality target patches. Following that, the *first* success invasion measures the estimated invasion time to populate any of the non-zero target patches. The *majority* success is the estimated invasion time to populate the majority (more than 50%) of all non-zero target patches. Finally, *all* successes is defined as the estimated invasion time to populate all non-zero target patches.

3.3 Invasion model using R-*local* simulation method

The invasion model using *R-local* simulation method can be seen as a modified version of the invasion model using *full* simulation method (presented in Chapter 2). In the *R-local* simulation method, we define a local distance R to populate patches within the defined R instead of all patches in the whole landscape, which is the case in the *full* simulation method (in other words, R equals to the diameter of the landscape in the *full* method). More formal description of the structure of the *R-local* method, which is similar to the structure of the *full* method, is given in Algorithm 2. The generic structure of the *R-local* method contains inputs parameters, outputs variables (as mentioned in Table 2.1 in Chapter 2), and COUNT ROUNDS function. The COUNT ROUNDS function counts the number of rounds required for *first*, *majority* and *all* successes and to compute the real time execution for each simulation. The function includes nested loops of three levels. The main loop (lines 9-24) counts the number of rounds to populate target patches. The second level loop (lines

11-24) is for all populated patches that are trying to populate unpopulated patches. Each populated patch in the landscape only tries to populate every other unpopulated patch within local distance R around the populated patch. The inner level loop (lines 14-24) is for all unpopulated patches. This inner loop runs only for unpopulated patches that are of distance at most R from the populated patch (*from populated patch to R*, lines 14 and 15 in Algorithm 2). Each unpopulated patch becomes populated if the *transition probability* (Equation (2.1)) between the populated and unpopulated patches is greater than a generated random number between zero and one (lines 19-22). We consider only populating a patch with non-zero quality. Each time when an unpopulated target patch becomes populated, the algorithm checks if the total number of non-zero patches at target is equal to *one* or *majority* or *all* non-zero targets' number, and the number of rounds is recorded accordingly. The COUNT ROUNDS function terminates when all non-zero target patches become populated and returns the number of rounds (invasion time) needed for each type of the successful invasions as well as the execution time of simulation.

3.4 The studied landscapes

As mentioned in Chapter 2, the dataset of the 1km resolution raster version of the Land Cover Map 2007 (LCM2007) for Great Britain [91] is used for evaluation purposes. To examine and evaluate the proposed *R-local* simulation method, three landscapes from different maps of the aggregate classes are extracted. Recall that the average of all patches qualities is between 0% and 5%, 5% and 25%, 25% and 100% for low, medium and high quality, respectively. For each quality type, we extract a rectangular landscape that consists of 5 rows (height) and 300 columns (width) (see Figure 3.1). Landscapes of low and medium qualities were extracted from semi-natural grassland GB map, while the one of high quality from an improved grassland GB map. On these extracted rectangular landscapes, we assume that all patches at the first column of each landscape are occupied and the goal is to populate patches at the target columns (col. 10, 20, 30, etc.). Each landscape is extracted according to the following criteria:

1. The qualities of all source occupied patches are non-zeros.

2. At each target column, at least one of the patches is non-zero in quality.

In each of these extracted landscapes, the first column contains the source patches while columns with numbers $10, 20, 30, \ldots, 300$ contain intermediate target patches — within each

Algorithm 2 Modelling invasion process using *R-local* simulation method (G, S, T, α)

```
1: function COUNT ROUNDS(G, S, T, α)
2:     start ← record start execution time of simulation
3:     R ← Use Equation (3.4)
4:     Create 2-dimensional array B having size equal to array G ← 0
5:     for each populated patch (i, j) in vector S do
6:         B(i, j) ← 1
7:     Rounds ← 0                    ▷ counter to count rounds for successful invasions
8:     while any of target patch in T is unpopulated do
9:         Rounds ← Rounds+1
10:        for i ← 0 to number of rows in G do          ▷ loop for all populated patches
11:            for j ← 0 to number of columns in G do
12:                if patch B(i, j) is populated then
13:                    for z ← i − R to i + R do        ▷ loop for all unpopulated patches
14:                        for l ← j − R to j + R do
15:                            if patch B(z, l) is unpopulated & G(z, l) ≠ 0 then
16:                                dist ← Euclidean distance between B(i, j), B(z, l)
17:                                if 0 < dist ≤ R then
18:                                    p ← Probability between B(i, j), B(z, l) using Eq. (2.1)
19:                                    w ← Generate a random number between 0 & 1
20:                                    if w < p then
21:                                        Populate patch B(z, l)
22:                                        if B(z, l) is at the target column then
23:                                            Check the invasion type
24:    end ← Record end execution time of simulation
25:    ExecutionTime ← end - start
26:    return Rounds, ExecutionTime
```

simulated invasion process, we monitor and record time in which patches in these columns are populated.

3.5 The method — theoretical part

3.5.1 New formulas estimating duration of the invasion process

Based on the formula of the *transition probability* (Equation (2.1)), we propose three new estimating times of invasions in such a landscape. For a given landscape graph G, the estimated time of invasion from source S to target T contains at most $\lceil \frac{H}{d_{min(G)}} \rceil + \lceil \frac{W}{d_{min(G)}} \rceil$

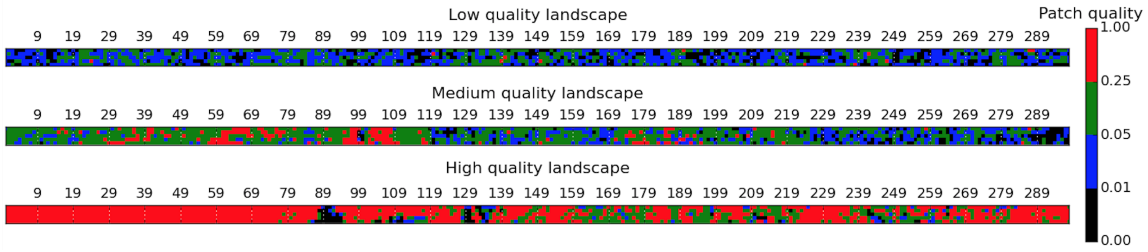


Figure 3.1: The studied landscapes; three landscapes of size 5×300 and of low, medium and high quality extracted from LCM2007 GB (aggregate classes) maps. In each landscape, the colour corresponds to the quality of each patch; black, blue, green, and red corresponds to zero, low (0.01-0.05), medium (0.05-0.25), and high quality (0.25-1), respectively.

hops which is not more than $2 + \frac{H+W}{d_{min}(G)}$ hops.[2] According to the formula of the *transition probability* (Equation (2.1)) between patches, the probability of a single hop in the landscape graph G is at least $q_{min}(G) \cdot \frac{\exp(-\alpha \cdot d_{min}(G))}{\left(\frac{2\pi}{\alpha^2}\right)-1}$. Thus, the expected number of rounds for each hop is the inverse of its probability, i.e., $\frac{\left(\frac{2\pi}{\alpha^2}\right)-1}{q_{min}(G) \cdot \exp(-\alpha \cdot d_{min}(G))}$. Since the number of hops in the estimated time of invasion from source S to target T is $\frac{H+W}{d_{min}(G)}$, the total expected time of invasion for the *first* success is at most

$$\frac{H+W}{d_{min}(G)} \cdot \frac{\left(\frac{2\pi}{\alpha^2}\right)-1}{q_{min}(G) \cdot \exp\left(-\alpha \cdot d_{min}(G)\right)} \cdot c \ , \tag{3.1}$$

where c is a small constant to be determined by simulations. Consequently, the total expected time of invasion for the *majority* success is at most

$$\left[\frac{H+W}{d_{min}(G)} \cdot \frac{\left(\frac{2\pi}{\alpha^2}\right)-1}{q_{min}(G) \cdot \exp\left(-\alpha \cdot d_{min}(G)\right)} + \left(\frac{H}{2}-1\right) \cdot \frac{\left(\frac{2\pi}{\alpha^2}\right)-1}{q_{min}(G) \cdot \exp\left(-\alpha \cdot d_{min}(G)\right)}\right] \cdot c \ , \tag{3.2}$$

[2]The main formula is because the maximum geometric distance from S to T is at most $H+W$, for each pair of source and target nodes we could find a shortest path such that some initial geometric edges are vertical and the remaining ones are horizontal, and finally, on each such shortest geometric path from S to T in each of these two segments (vertical and horizontal) we could replace d_{min} consecutive edges by a single edge in graph $Loc(G, d_{min})$ (except the last part of each segment, which could be shorter than d_{min} but we still need to replace it by an edge, this is why there are ceilings in the formula).

where $\left(\frac{H}{2}-1\right)$ is an upper bound on the total number of *majority* of target patches decreased by one and c is again a small constant to be interpolated by simulations. Therefore, the total expected time of invasion for *all* successes is at most

$$\left\lceil \frac{H+W}{d_{min}(G)} \cdot \frac{\left(\frac{2\pi}{\alpha^2}\right)-1}{q_{min}(G)\cdot \exp\left(-\alpha \cdot d_{min}(G)\right)} + (H-1)\cdot \frac{\left(\frac{2\pi}{\alpha^2}\right)-1}{q_{min}(G)\cdot \exp\left(-\alpha \cdot d_{min}(G)\right)} \right\rceil \cdot c \ , \tag{3.3}$$

where $(H-1)$ is an upper bound on the total number of *all* target patches decreased by one and c is a small constant to be interpolated by simulations.

3.5.2 Estimation of parameter R for accurate and efficient R-*local* simulations

It is expected that the number of rounds required for the R-*local* method is larger. In our simulation, we aim to find the local distance R that allows the following accuracy:

$$\frac{\text{FULL}}{\text{R-LOCAL}} = \frac{\text{average number of rounds using the } \textit{full} \text{ method}}{\text{average number of rounds using the } R\textit{-local} \text{ method}} \geq 90\% \ .$$

As a starting point, we run simulation using both *full* and R-*local* methods with some expected local distances R. It has been observed that the required local distance R for the FULL/R-LOCAL ratio to be at least 90% in the *first* success is greater than or equal to the needed local distance R for *majority* and *all* successes. Based on this observation, we propose an equation that computes the local distance R for a given landscape graph G based on the expected time of invasion for the *first* success. Observe that the probability of a single hop in a given landscape graph G is less than the inverse of the total expected time of the invasion process for the *first* success:

$$q_{min}(G)\cdot \frac{\exp\left(-\alpha \cdot d_{min}(G)\right)}{\left(\frac{2\pi}{\alpha^2}\right)-1} \ll \frac{1}{\text{expected time of invasion for } \textit{first} \text{ success}} \ .$$

Therefore, the following holds:

$$q_{max}(G)\cdot \frac{\exp\left(-\alpha \cdot R(G)\right)}{\left(\frac{2\pi}{\alpha^2}\right)-1} \ll \frac{d_{min}(G)}{H+W}\cdot q_{min}(G)\cdot \frac{\exp\left(-\alpha \cdot d_{min}(G)\right)}{\left(\frac{2\pi}{\alpha^2}\right)-1}$$

$$\exp\left(-\alpha \cdot R(G)\right) \ll \frac{d_{min}(G) \cdot \bar{q}(G)}{H+W} \cdot \exp\left(-\alpha \cdot d_{min}(G)\right) \ .$$

$$-\alpha \left(R(G) - d_{min}(G)\right) < \ln\left(\frac{d_{min}(G) \cdot \bar{q}(G)}{H+W}\right) \ .$$

Finally, we get the best (smallest) local distance R needed to create a landscape subgraph $Loc(G, R)$, for a given landscape graph G, and guarantees the FULL/R-LOCAL accuracy is:

$$R(G) \approx \left[\frac{1}{\alpha} \cdot \ln\left(\frac{H+W}{d_{min}(G) \cdot \bar{q}(G)}\right) + d_{min}(G)\right] \cdot c \ , \tag{3.4}$$

where $\bar{q}(G) = \frac{q_{min}(G)}{q_{max}(G)}$ and c is a small constant to be determined by simulations.

Equation (3.4) is the core theoretical finding, for which we interpolate the constant c in Section 3.6 by performing simulations and validate the theory in Section 3.7.

3.6 The method — interpolating constant c in Equation (3.4) based on simulations

3.6.1 Methodology

The simulations in this part of our work are directed at four goals. The first is to monitor the behaviour of *full* and *R-local* methods and compare results obtained by each method. The second is to investigate what values of local distance R would allow the FULL/R-LOCAL accuracy. Based on the results obtained from the simulation, the third goal is to predict an equation for the local distance R, depending on landscape size, dispersal coefficient α, and landscape quality. Finally, we aim to combine results from simulation and the predicted equation to compare them and conduct an independent validation based on the value obtained from the proposed equation on the local distance R.

For each prefix $5 \times 10, 5 \times 20, 5 \times 30, \ldots, 5 \times 300$ in each of the extracted rectangular landscapes in Figure 3.1, we run the *full* simulation and, simultaneously, the *R-local* simulation with some predicted local distances R; ideally, we aimed to find the value of R that satisfies the desired FULL/R-LOCAL accuracy, while the local distance $R-1$ does not satisfy it.

We consider four values for the dispersal coefficient α: 0.25, 0.5, 1 and 2. For each prefix and for each dispersal coefficient α, we run *full* and *R-local* simulation 100 times and compute the average number of rounds (estimated time of invasion) for *first*, *majority*, and

all successes.

Additionally, we define the following three objective functions in order to interpolate constant c in Equation (3.4). In these functions, we use two terminologies: "approx R" and "opt R" to express the local distance R using Equation (3.4) and simulation such that it is the smallest distance satisfying the FULL/R-LOCAL accuracy, respectively. We call it "opt R" because this value of R fulfils our goal of accuracy.

The Euclidean distance objective function (ED). This function chooses the constant c such that it minimises the sum over all prefixes z of the square difference between the approx R and opt R:

$$c_{ED} = \underset{c \in \mathbb{R}^+}{\operatorname{argmin}} \left\{ \sum_{j=1}^{z} (approxR_j \cdot c - optR_j)^2 \right\}.$$

The absolute objective function (AB). This objective function chooses the constant c such that it minimises the sum over all prefixes z of the absolute difference between the approx R and opt R:

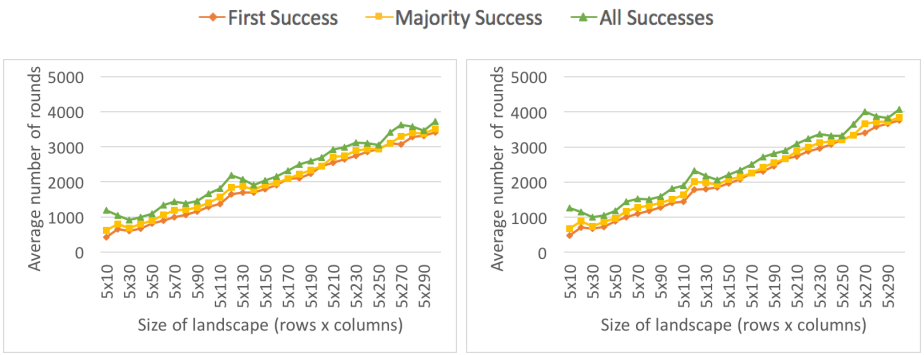
$$c_{AB} = \underset{c \in \mathbb{R}^+}{\operatorname{argmin}} \left\{ \sum_{j=1}^{z} |approxR_j \cdot c - optR_j| \right\}.$$

The min-max (MM) objective function. The min-max (MM) objective function chooses the constant c such that it minimises the approx R to be greater than or equal to opt R for all prefixes z:

$$c_{MM} = \underset{c \in \mathbb{R}^+}{\operatorname{argmin}} \left\{ \max_{1 \le j \le z} (approxR_j \cdot c - optR_j) \ge 0 \right\}.$$

3.6.2 Simulation results

The estimated time of invasion (i.e., average number of rounds over 100 independent repetitions) for each prefix 5×10, 5×20, 5×30, ..., 5×300 in each studied landscape using *full* and *R-local* simulation methods is presented in Figures 3.2-3.4. These simulation results show that the FULL/R-LOCAL accuracy of at least 90% is satisfied in all scenarios, as shown in Figures 3.5-3.7.



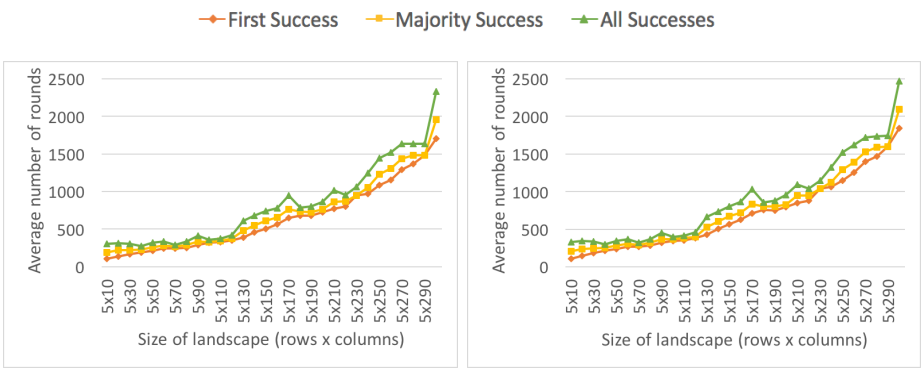
(a) $\alpha = 0.25$, *full* method (b) $\alpha = 0.25$, *R-local* method

(c) $\alpha = 0.5$, *full* method (d) $\alpha = 0.5$, *R-local* method

(e) $\alpha = 1$, *full* method (f) $\alpha = 1$, *R-local* method

(g) $\alpha = 2$, *full* method (h) $\alpha = 2$, *R-local* method

Figure 3.2: The average number of rounds needed for *first*, *majority* and *all* successes for each prefix in landscape of size 5×300 and of **low** quality when the dispersal coefficient α = 0.25, 0.5, 1, 2 using *full* and *R-local* methods.



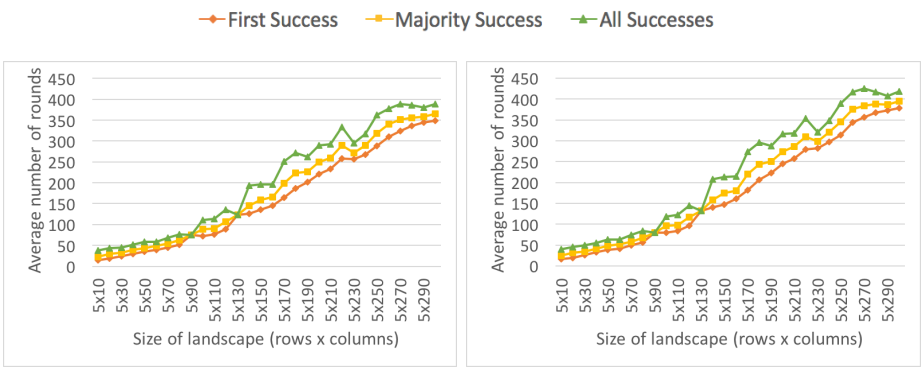
(a) $\alpha = 0.25$, *full* method (b) $\alpha = 0.25$, *R-local* method

(c) $\alpha = 0.5$, *full* method (d) $\alpha = 0.5$, *R-local* method

(e) $\alpha = 1$, *full* method (f) $\alpha = 1$, *R-local* method

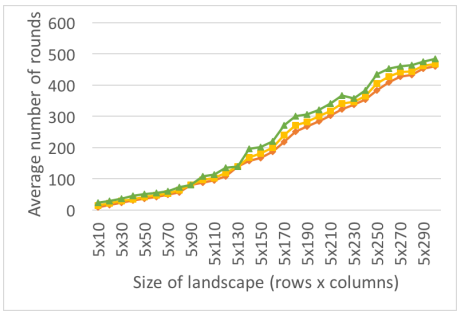
(g) $\alpha = 2$, *full* method (h) $\alpha = 2$, *R-local* method

Figure 3.3: The average number of rounds needed for *first*, *majority* and *all* successes for each prefix in landscape of size 5×300 and of **medium** quality when the dispersal coefficient $\alpha = 0.25, 0.5, 1, 2$ using *full* and *R-local* methods.

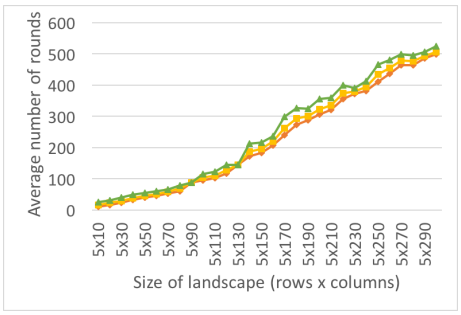


(a) $\alpha = 0.25$, *full* method

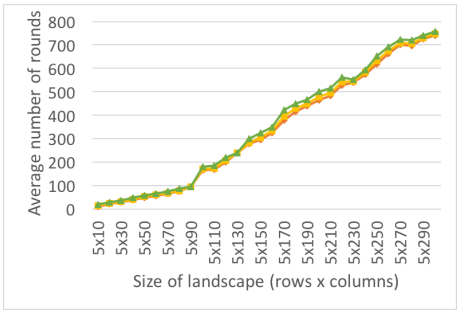
(b) $\alpha = 0.25$, *R-local* method



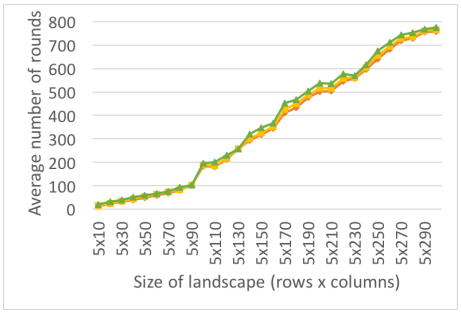
(c) $\alpha = 0.5$, *full* method



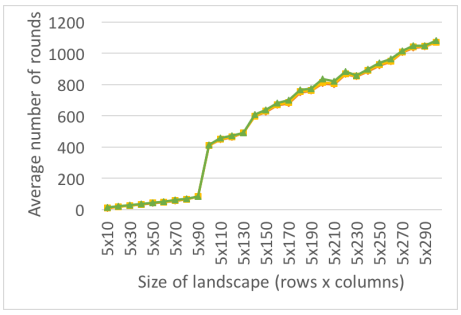
(d) $\alpha = 0.5$, *R-local* method



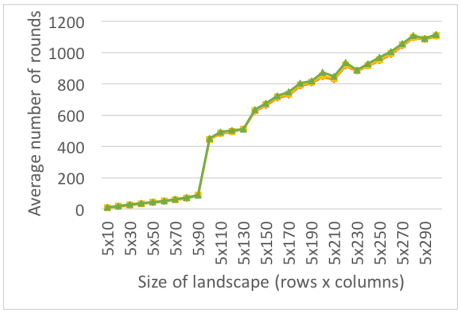
(e) $\alpha = 1$, *full* method



(f) $\alpha = 1$, *R-local* method

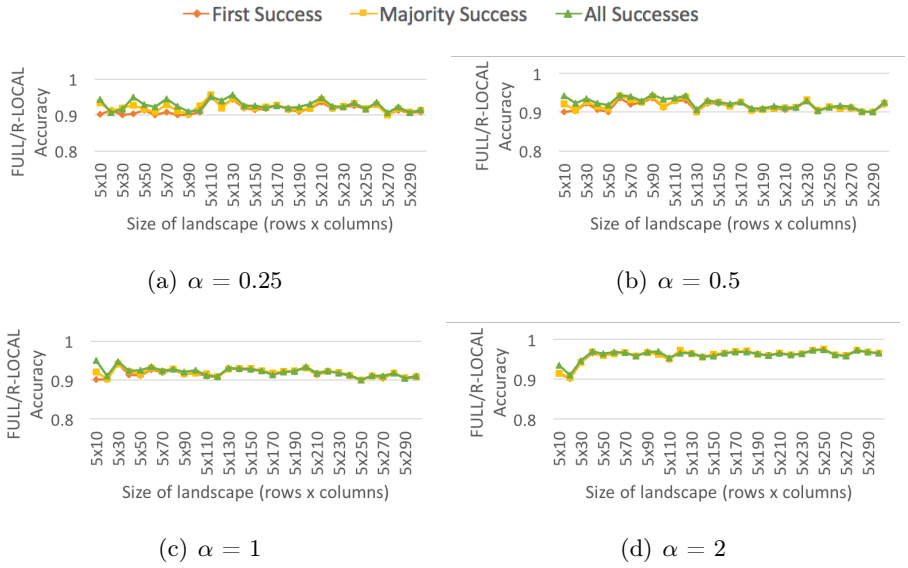


(g) $\alpha = 2$, *full* method



(h) $\alpha = 2$, *R-local* method

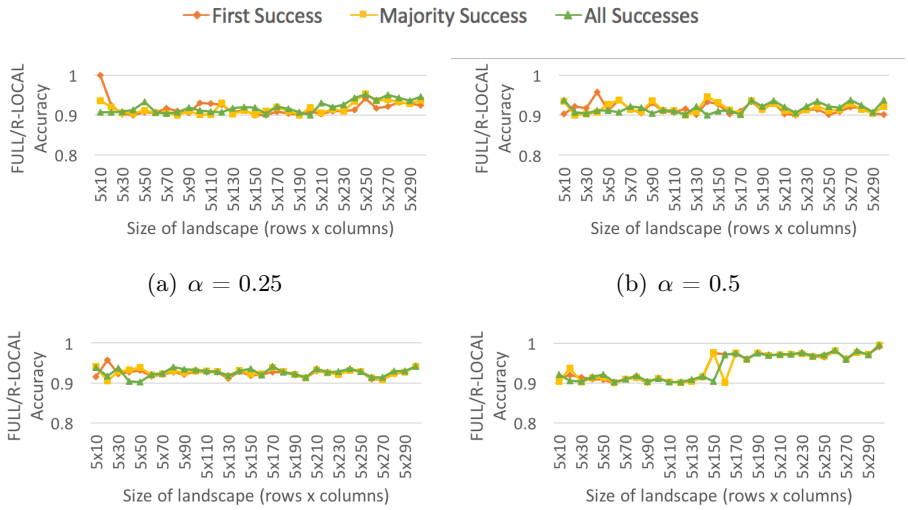
Figure 3.4: The average number of rounds needed for *first*, *majority* and *all* successes for each prefix in landscape of size 5×300 and of **high** quality when the dispersal coefficient $\alpha = 0.25, 0.5, 1, 2$ using *full* and *R-local* methods.



(a) $\alpha = 0.25$ (b) $\alpha = 0.5$

(c) $\alpha = 1$ (d) $\alpha = 2$

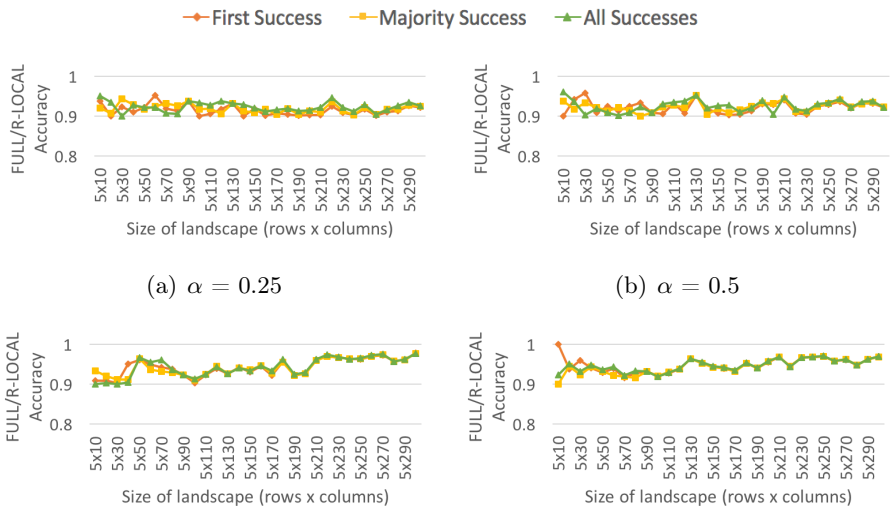
Figure 3.5: The FULL/R-LOCAL accuracy for each prefix in landscape of size 5×300 and of **low** quality when the dispersal coefficient α takes values of 0.25, 0.5, 1 and 2.



(a) $\alpha = 0.25$ (b) $\alpha = 0.5$

(c) $\alpha = 1$ (d) $\alpha = 2$

Figure 3.6: The FULL/R-LOCAL accuracy for each prefix in landscape of size 5×300 and of **medium** quality when the dispersal coefficient α takes values of 0.25, 0.5, 1 and 2.



(a) $\alpha = 0.25$ (b) $\alpha = 0.5$

(c) $\alpha = 1$ (d) $\alpha = 2$

Figure 3.7: The FULL/R-LOCAL accuracy for each prefix in landscape of size 5×300 and of **high** quality when the dispersal coefficient α takes values of 0.25, 0.5, 1 and 2.

Here we focus on the local distance R and the constant c. On the studied landscapes, values of opt R used in simulations and guaranteeing the FULL/R-LOCAL accuracy are provided in Figure 3.8(a), while Figure 3.8(b) shows the computed values of approx R using Equation (3.4). From the shown values of opt R in Figure 3.8(a), we observe that the dispersal coefficient α is the most important parameter in both simulation methods. In all qualities, it has been investigated that a larger local distance R is required when the dispersal coefficient α equals to 0.25, while for 0.5, 1 and 2 a smaller local distance R is sufficient. Therefore, the local distance R that ensures the FULL/R-LOCAL accuracy depends on the dispersal coefficient α, and hence with decreasing mean dispersal distance: R decreases with the increase in α. Furthermore, a logarithmic growth has been observed in the local distance R with the growth of landscape size. On the other hand, the difference in the landscape quality has not caused a significant difference in the local distance R. Comparison of the obtained results based on the proposed equation with simulation results demonstrates that the proposed equation (Equation (3.4)) gives a good estimate of the local distance R (see Figure 3.8).

Additionally, we define the error rate between approx R (when constant $c = 1$, i.e.,

Table 3.1: Error rate between approx R (when constant $c = 1$) and opt R for each 5×300 landscape of low, medium and high quality when the dispersal coefficient $\alpha = 0.25$, 0.5, 1, 2.

Landscape quality	α=0.25	α=0.5	α=1	α=2
Low quality	0.81	0.74	0.73	0.57
Medium quality	0.83	0.86	0.90	1.01
High quality	0.89	0.75	0.56	0.63

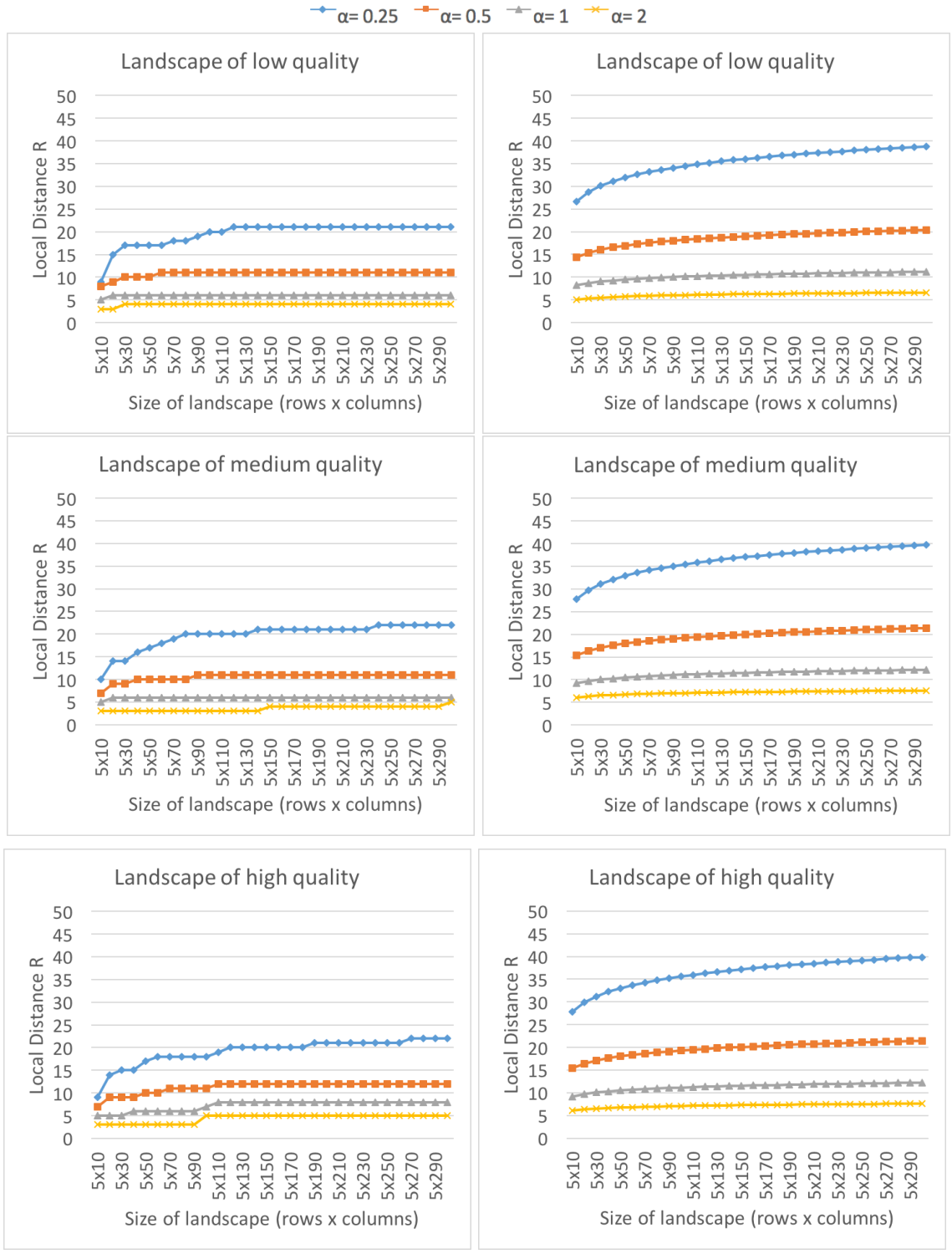
before interpolating c) and opt R as: $\frac{\sum_{j=1}^{z}(approxR_j - optR_j)}{\sum_{j=1}^{z} optR_j}$. Table 3.1 gives the error rates between approx R and opt R for each 5×300 landscape of low, medium, and high quality. All error rates in Table 3.1 are high, which means that we need to find the best constant c such that it minimises the error for various dispersal coefficients α and different qualities. For each studied landscape, Table 3.2 presents the interpolated constants c based on the defined objective functions ED, AB, and MM. These constants are affected by opt R for all prefixes in each landscape of different quality.

Table 3.2: The computed constant c by the objective functions ED, AB, and MM for each 5×300 landscape of low, medium and high quality when the dispersal coefficient $\alpha = 0.25$, 0.5, 1, 2.

Landscape quality	Parameters	Constant c	α=0.25	α=0.5	α=1	α=2
Low quality	$q_{min} = 0.01$ $q_{max} = 0.64$ $d_{min} = 2$	c_{ED}	0.55	0.55	0.6	0.65
		c_{AB}	0.55	0.55	0.55	0.65
		c_{MM}	0.6	0.65	0.7	0.75
Medium quality	$q_{min} = 0.01$ $q_{max} = 0.96$ $d_{min} = 3$	c_{ED}	0.55	0.55	0.55	0.5
		c_{AB}	0.55	0.55	0.5	0.5
		c_{MM}	0.6	0.6	0.65	0.7
High quality	$q_{min} = 0.01$ $q_{max} = 0.99$ $d_{min} = 3$	c_{ED}	0.55	0.55	0.65	0.6
		c_{AB}	0.55	0.6	0.65	0.65
		c_{MM}	0.6	0.65	0.75	0.75

3.6.3 Evaluating constants c_{ED}, c_{AB}, c_{MM} and selecting the most suitable one

In order to evaluate constants c_{ED}, c_{AB}, and c_{MM} in Table 3.2, we define the following three error rates:



(a) Opt R computed by simulations

(b) Approx R computed by Equation (3.4)

Figure 3.8: The local distance R for each prefix in landscape of size 5×300 and of low, medium, and high quality when the dispersal coefficient α takes values of 0.25, 0.5, 1 and 2. (a) Opt R that allows the FULL/R-LOCAL accuracy and computed by simulations. (b) Approx R computed by the proposed formula (Equation (3.4), $c = 1$).

1. error rate$(c_{ED}) = \frac{\sqrt{\sum_{j=1}^{z} (approxR_j \cdot c_{ED} - optR_j)^2}}{\sum_{j=1}^{z} optR_j}$,

2. error rate$(c_{AB}) = \frac{\sum_{j=1}^{z} |approxR_j \cdot c_{AB} - optR_j|}{\sum_{j=1}^{z} optR_j}$, and

3. error rate$(c_{MM}) = \frac{\sum_{j=1}^{z} (approxR_j \cdot c_{MM} - optR_j)}{\sum_{j=1}^{z} optR_j}$.

Observe that the error rate gives a measure of how well the constant c interpolated by the corresponding objective function minimises the error rate between approx R and opt R, for a given landscape. When approx R is very close or equal to opt R, the error will be small or zero. Table 3.3 provides the computed error rates between approx R and opt R for each 5×300 landscape of low, medium and high quality, when the constant c is equal to the interpolated c_{ED}, c_{AB}, and c_{MM} (constants in Table 3.2). As can be seen in Table 3.3, all error rates are between 0.01 and 0.4. Although c_{ED} and c_{AB} produce error rates smaller than c_{MM}, constant c_{MM} is the best among the three with respect to the following criteria. The best constant is the one that reduces the value of approx R to be greater than or equal to the value of opt R, in all studied landscapes. Based on this criteria, we found that constant c_{MM} is the one that satisfies that. On the other hand, c_{ED} and c_{AB} in some cases decrease approx R to be less than opt R, and that means they give approx R which does not allow the sought FULL/R-LOCAL accuracy.

3.7 Validation

3.7.1 Validation methodology for the *R-local* simulation method

One important characteristics to consider when computing the total time of invasion is also the number of repetitions needed for the (average) invasion time to stabilise on the outputted duration of invasion. Accordingly, we define the *Stabilisation Time (ST)* for a given landscape as the time t such that the change in the Average number of rounds For *All* Successes (AFAS) between t and $2t$ is less than or equal to 2%: $\forall \tau \in (t, 2t]$, $|AFAS(\tau) - AFAS(t)| \leq 0.02 \cdot AFAS(t)$, where $AFAS(\tau) = \frac{\sum_{j=1}^{\tau} AS(j)}{\tau}$ and $AS(j)$ is the number of rounds needed for the *All* Successes (AS) at iteration j.

Table 3.3: The error rate between approx R and opt R, when $c = c_{ED}, c_{AB}, c_{MM}$, for each 5×300 landscape of low, medium and high quality when $\alpha = 0.25, 0.5, 1, 2$.

Landscape quality	Parameters	Error rate	α=0.25	α=0.5	α=1	α=2
Low quality	$q_{min} = 0.01$ $q_{max} = 0.64$ $d_{min} = 2$	Error rate (c_{ED})	0.01	0.01	0.01	0.01
		Error rate (c_{AB})	0.04	0.05	0.06	0.05
		Error rate (c_{MM})	0.08	0.12	0.21	0.17
Medium quality	$q_{min} = 0.01$ $q_{max} = 0.96$ $d_{min} = 3$	Error rate (c_{ED})	0.01	0.01	0.01	0.02
		Error rate (c_{AB})	0.04	0.04	0.05	0.11
		Error rate (c_{MM})	0.1	0.11	0.23	0.4
High quality	$q_{min} = 0.01$ $q_{max} = 0.99$ $d_{min} = 3$	Error rate (c_{ED})	0.01	0.01	0.02	0.03
		Error rate (c_{AB})	0.05	0.05	0.08	0.12
		Error rate (c_{MM})	0.13	0.13	0.17	0.22

In order to test the robustness of our proposed method, we performed validation in a large number of different landscapes, 36 (cf., Figure 3.9). The 36 landscapes are divided into groups of nine landscapes and the four groups each has size: 5×50, 10×50, 15×50, and 20×50, respectively. To ensure robustness, we use different sizes from the sizes used in deriving R in Section 3.6.2. For each landscape size, the nine landscapes are further divided into subgroups of three and each group has associated low, medium and high quality, respectively. All landscapes are extracted randomly from different LCM2007 GB (aggregate classes) maps (i.e., aggregate classes in Table 2.2). The procedure to select a landscape randomly is described in the following steps.

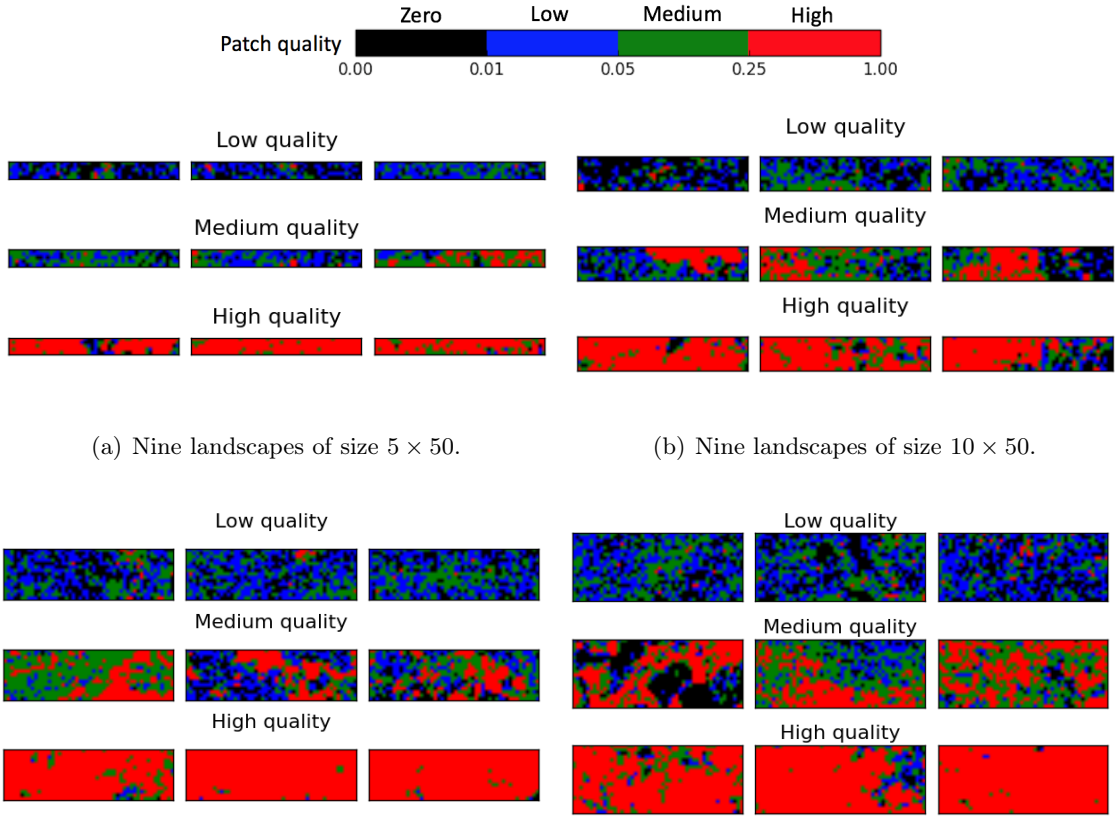
1. Specify a map from the ten maps presented in Table 2.2.

2. Specify the height H (number of rows), width W (number of columns) and the quality (low or medium or high) of the landscape that we will extract.

3. Select a random number, called r, in the range between one and the total number of rows in the specified map. Select another random number, called c, in the range between one and the total number of columns in the specified map. These two numbers indicate to the left upper corner of the landscape.

4. Check if r plus the specified number of rows for the landscape and c plus the specified number of columns for the landscape do not exceed the border of the selected map, then extract this landscape, otherwise discard it and choose another two random numbers.

5. Check the quality, by summing the patch quality over all patches and divided them by the total number of patches, of the selected landscape whether it is as the specified quality in point 2 or not.



(a) Nine landscapes of size 5×50.

(b) Nine landscapes of size 10×50.

(c) Nine landscapes of size 15×50.

(d) Nine landscapes of size 20×50.

Figure 3.9: The landscapes used for validation of *R-local* method. Nine landscapes of size (a) 5×50, (b) 10×50, (c) 15×50, and (d) 20×50, respectively. In each subfigure, all three landscapes in the top row are of low quality, middle row of medium quality, bottom row of high quality.

On the 36 extracted landscapes (landscapes in Figure 3.9) and for each considered value of the dispersal coefficient α, we run *full* and *R-local* simulations independently as described in the following steps, in order to get the estimated time of invasion for *first*, *majority* and *all* successes as well as the execution time of simulations.

1. Run *full* simulations many times independently, stop running simulations at the *Stabilisation Time (ST)* and record the results (i.e., the total number of rounds for *first*, *majority* and *all* successes as well as the execution time of *full* simulations) for each repetition. Following that, compute the average number of rounds for *first*, *majority* and *all* successes, and the average of the execution time of *full* simulations.

2. Compute the minimum quality q_{min}, the maximum quality q_{max} and the minimum distance d_{min}.

3. Compute three local distances R using Equation (3.4) with the constants (c_{ED}, c_{AB}, and c_{MM}) in Table 3.2, which are calculated as in Section 3.6.1.

4. Run *R-local* simulations with each of the three computed local distances R many times independently, stop running simulations at the *Stabilisation Time (ST)* and record the results (i.e., the total number of rounds for *first*, *majority* and *all* successes as well as the execution time of *R-local* simulations) for each repetition. Following that, compute the average number of rounds for *first*, *majority* and *all* successes, and the average of the execution time of *R-local* simulations.

5. For all three types of successful invasions and all local distances R computed in point 3, compute the FULL/R-LOCAL ratio and check whether it is within the desired 90% accuracy.

6. Specify which of the computed local distances R in point 3 is the opt, where the opt one is the smallest distance that gives the desired 90% accuracy.

7. Compute the ratio between the Average of the Execution Time of Simulations (AETS) for *full* and *R-local* methods.

3.7.2 Validation results

This section reports some results of implementing the above seven steps.

Figure 3.10 gives the average of the local distances R over the three landscapes in each subgroup with the same size and same landscape quality (results of computing R using Equation (3.4) with constant c_{MM}, as mentioned in point 3). Recall that these distances are computed using Equation (3.4) with constants c_{MM} given in Table 3.2. One could observe

that the distance R scales well with the growing size of the landscape (i.e., it is quite stable) and decreases sub-linearly with the growth of the dispersal coefficient α.

It has been illustrated by the validation experiments that running *full* and *R-local* simulations (until stabilisation point) on the 36 landscapes gives a good result as the FULL/R-LOCAL accuracy has been achieved as presented in Figure 3.11 (some results of implementing point 5). Detailed results of simulations (some results of implementing point 1 and 4), that give the desired accuracy in Figure 3.11, are presented in Figure 3.12. Furthermore, the MM objective function is the best function to be used among the three objective functions because it gives constant c_{MM} such that it reduces the approx R to be greater than or equal to the opt for all 36 landscapes (result of implementing point 6), thus avoiding underestimation of R and its consequence in uncontrolled growth of the invasion time. Finally, the computed parameter R is well scalable, in the sense that it does not depend much on the growing size of the landscape.

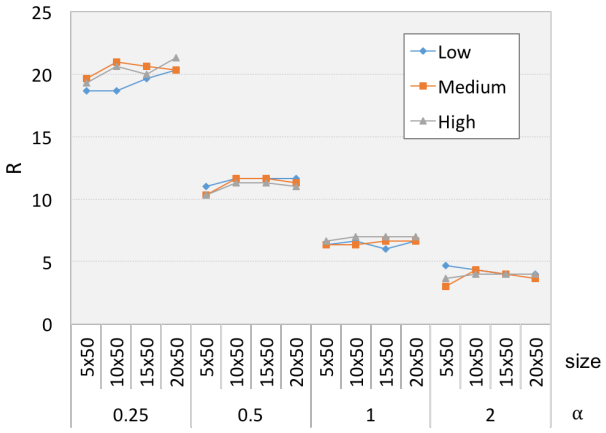


Figure 3.10: The average of the local distances R over the three landscapes in each subgroup with the same size and same landscape quality; R is computed using Equation (3.4) with constants c_{MM} in Table 3.2. This is done for different values of the dispersal coefficients α. These values of R allow the FULL/R-LOCAL accuracy presented in Figure 3.11.

3.8 The improvement in memory and time

This section shows how much memory and computational time we save when using the proposed *R-local* method for properly selected distance R to compute the invasion time.

(a) Landscapes of size 5×50

(b) Landscapes of size 10×50

(c) Landscapes of size 15×50

(d) Landscapes of size 20×50

Figure 3.11: The average of FULL/R-LOCAL accuracy over the three landscapes in each subgroup with the same size and same landscape quality; the R-LOCAL is with $R = approxR$ for $c = c_{MM}$. This is done for different values of the dispersal coefficient α.

3.8.1 Saved memory

In order to investigate the saved memory, we compute the *sparsification rate*. It is defined as the ratio of the total number of edges between patches in the landscape over the total number of edges between patches of distance at most R; in both numbers we count only edges with both ends of non-zeros quality. Figure 3.13 illustrates the sparsification rate versus the landscape size.

Generally, the relation between the sparsification rate and the landscape size can be seen as a linear tendency (except some peaks referring to the increase in R), where the linear coefficient grows linearly with α. As can be seen when comparing Figures 3.10 and 3.13, the sparsification rate depends on the local distance R: the rate increases with the decrease in R. For instance, the decline in the orange line of the rate in Figure 3.13 (medium quality, size=10×50, and α= 2) refers to an increase of R from 3 (for size 5×50) to 4 (for size 10×50), see the tendency of R in Figure 3.10. The results of sparsification rate in 30 prefixes of each studied landscape are given in Figure 3.14. It has been observed that there are three peaks in the sparsification rate in Figure 3.14. These peaks appeared in landscapes

of medium and high quality and for α equals 2 (yellow line), and caused by a change on the value of R. The first drop is between 5×140 and 5×150 medium prefixes and that because of an increase in the value of R from 3 to 4. The second drop is between 5×290 and 5×300 medium prefixes and that caused by an increase in R from 4 to 5. The third drop is between 5×90 and 5×100 high prefixes and that caused by an increase in R from 3 to 5.

3.8.2 Saved time

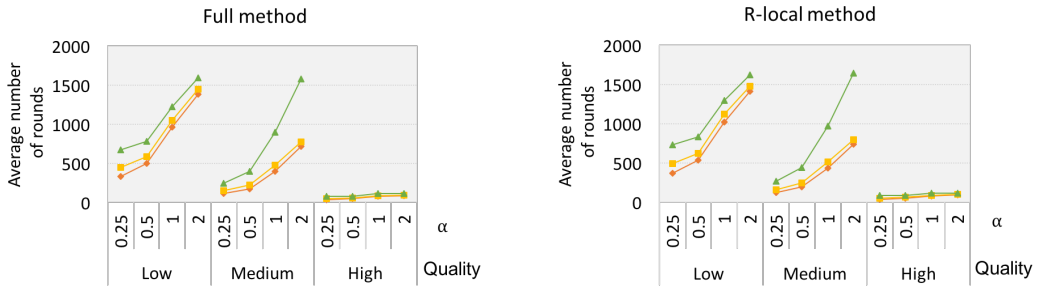
The validation experiments in this part of our work demonstrate that the Total Time (TT) of simulation execution, where TT = *Stabilisation Time (ST)* · Average of the Execution Time of Simulations (AETS), needed to compute the estimated duration of the invasion process is substantially reduced by the *R-local* simulation method for all landscapes of different qualities. Figure 3.16 illustrates how much the *R-local* method is faster than the *full* method for all three qualities. For many cases, the *full* method takes 5-10 times longer to compute, and this ratio can become as high as 75 for low quality landscapes. We note that in general, for a given landscape size, the speedup of the *R-local* method increases as the dispersal coefficient α increases. On the other hand, in most cases, when the dispersal coefficient α is fixed, the speedup increases as the size of landscape increases.

In more details, in the landscape of size 20×50 and of low quality, the average execution time of *full* simulation is 176.9 seconds while only 2.5 seconds in the *R-local* simulation. That means the *full* method takes around 70 times longer to compute. We could envisage that when we are running *full* simulation in very large landscapes e.g., landscape of size 500×500, the computation time will be substantially reduced from possibly weeks/days to hours.

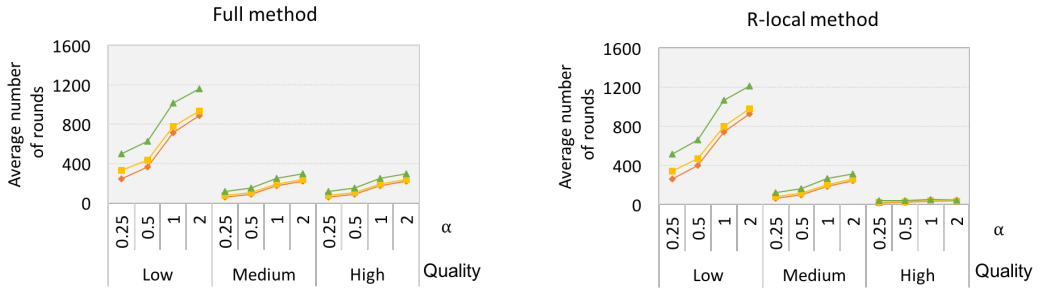
The ratio between the execution time of *full* and *R-local* simulation methods for some prefixes in each of the studied landscapes is presented in Figure 3.15.

To summarise, the validation experiments have confirmed the following: the desired 90% accuracy has been achieved using the developed formula for the local distance R (Equation (3.4)) and constants c_{MM} in Table 3.2. Moreover, saving in memory and computational time is rapidly growing with the dispersal coefficient α and the landscape size.

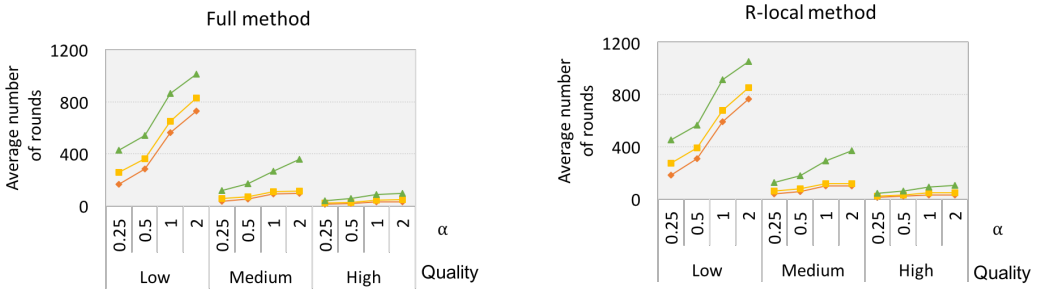
(a) Landscapes of size 5×50



(b) Landscapes of size 10×50



(c) Landscapes of size 15×50



(d) Landscapes of size 20×50

Figure 3.12: The average of the estimated time of invasion (i.e., the average number of rounds) for *first*, *majority* and *all* successes over the three landscapes in each subgroup with same size and same landscape quality using *full* and *R-local* methods; the R-LOCAL is with $R = approxR$ for $c = c_{MM}$. This is done for different values of dispersal coefficient α.

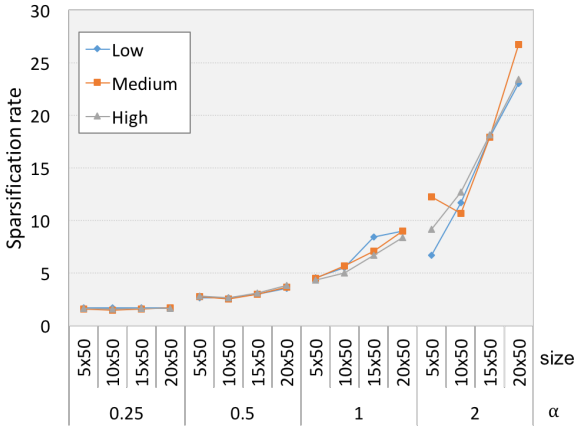


Figure 3.13: The sparsification rate versus the landscape size. That is done for different values of the dispersal coefficient α: 0.25, 0.5, 1 and 2, and for each landscape quality (low, medium, high).

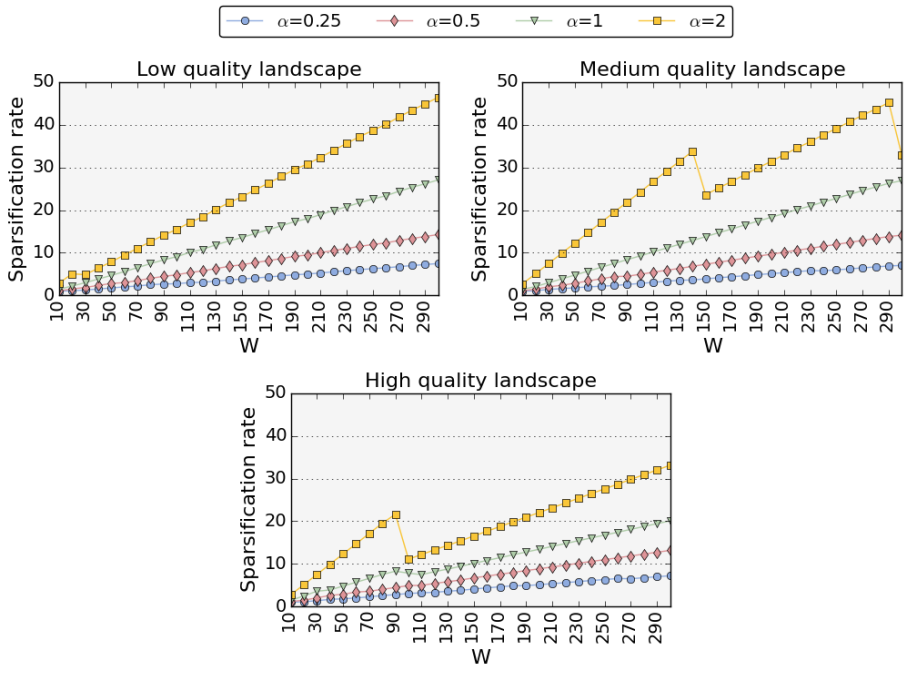


Figure 3.14: In each landscape quality (low, medium, high), the sparsification rate versus the landscape size. This is done for different values of the dispersal coefficient α: 0.25, 0.5, 1 and 2.

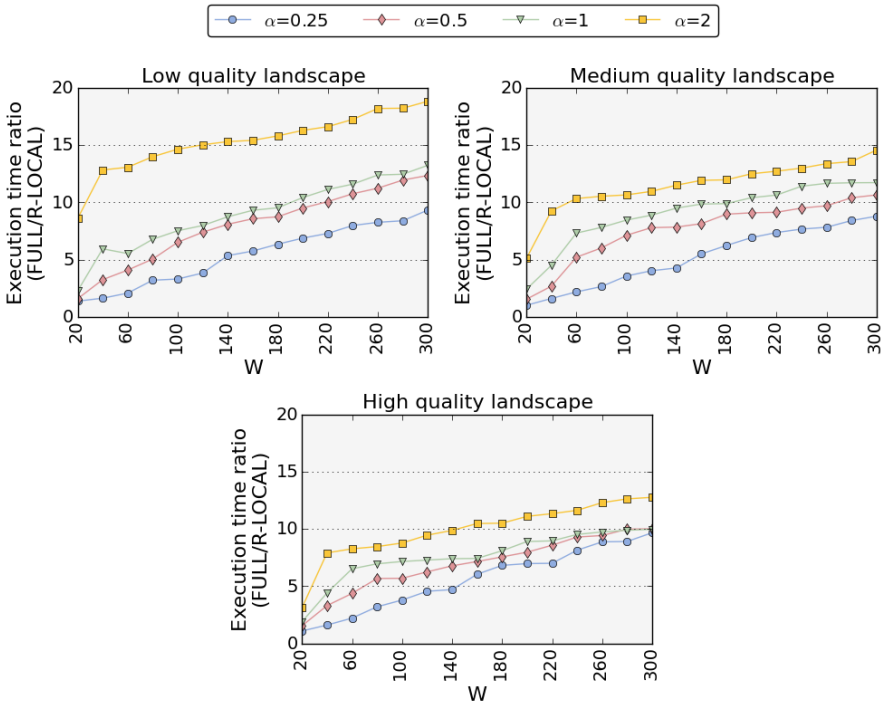
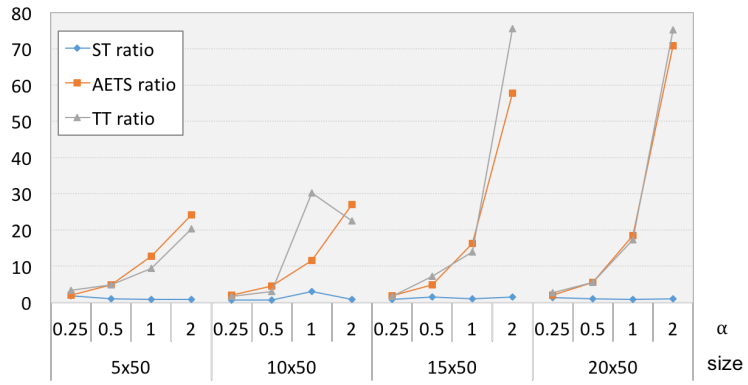
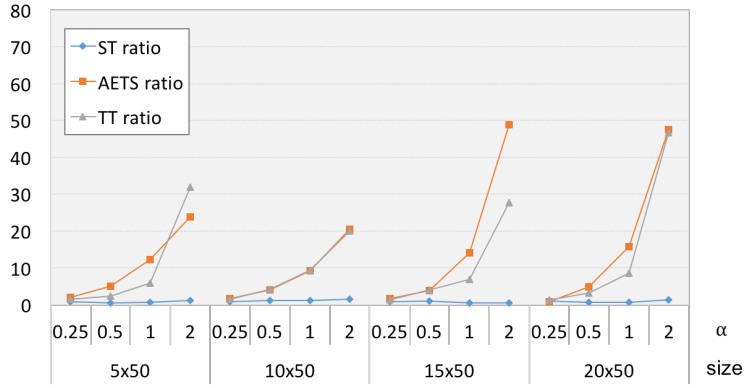


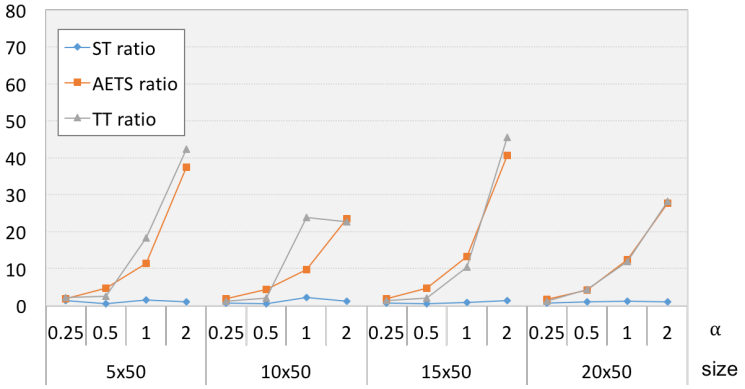
Figure 3.15: The ratio between the execution time of *full* and *R-local* simulations for some prefixes in each of the studied landscapes (low, medium, high). These prefixes are of sizes 5×20, 5×40, 5×60, ..., and 5×300. This is done for different values of the dispersal coefficient α: 0.25, 0.5, 1 and 2.



(a) Landscapes of low quality



(b) Landscapes of medium quality



(c) Landscapes of high quality

Figure 3.16: In each landscape quality (low, medium, high), the ratio between the *Stabilisation Time (ST)* of *full* and *R-local* simulations; the ratio between the Average of the Execution Time of Simulations (AETS) for *full* and *R-local* methods; and the ratio between the Total Time (TT) of *full* and *R-local* simulations; the R-LOCAL is with $R = approxR$ for $c = c_{MM}$. This is done in four different sizes of landscapes 5×50, 10×50, 15×50 and 20×50, and for different values of the dispersal coefficient α.

3.9 Further research

In this chapter, we introduced a new model that estimates the invasion time efficiently using graph sparsification approach. The new model has been compared with the state-of-the-art model to estimate the invasion time in heterogeneous real landscapes in Great Britain. We have shown that the actual computation time needed to compute the invasion time on large landscapes is substantially reduced by the new model while maintaining a comparable duration of the invasion.

The current approach applies sparsification to a specific problem with a single species and no other special circumstances. Introducing more species and other issues, for example the risk of population extinction, and the population density within a patch, into the model and problem would require more complex usage of sparsification, including formulas and algorithms, this is an open issue could be consider in the future.

Chapter 4

Modelling Invasion Process using Network Flow Theory

We present here a new way to estimate the time of invasion process using a powerful computational approach based on conductance and network flow theory. More specifically, we give a new formula for estimating the Invasion Time (IT) using a combination of network flow methodologies, and prove asymptotic bounds on the quality of the obtained approximation. We then analyse the proposed approach mathematically and apply it to real heterogeneous landscapes of Great Britain to estimate the duration of the invasion process; the theoretical bounds obtained are compared with simulation results. We finally evaluate the proposed approach and show its accuracy and efficiency in approximating the invasion time.

4.1 Overview

4.1.1 The problem and our model

In this part of our work, we propose a new formula for estimating time of the invasion process using conductance and network flow theory. Network flow [1] is one of the fundamental problems in many areas of computer science, including networks, optimisation, distributed systems, and distributed databases. The input of the problem is a directed, weighted landscape graph $G = (V, E)$, where each vertex represents a patch of habitat (henceforth patch) in the landscape and each edge weight represents the probability of spreading the

species in one step from the beginning to the end patch of the edge. We distinguish two sets of patches: the source patches represent initially populated patches in which species are located, and the target patches represent the target locations for the invasion process. The network is almost completed, in the sense that each vertex is connected to (almost) all other vertices. The invasion process is to populate any of target patches when starting from the source patches, and we aim to approximate the time performance of this process. The vertex is populated by means of an invasion protocol. The protocol keeps checking edges, one after another (see definitions of synchronous and asynchronous executions later in this section), by generating a random number from 0 to 1; if the number is smaller than the weight of the edge and the beginning of the edge is populated, then the end of the edge becomes populated as well (unless it has been populated earlier). We restrict our attention to the invasion protocol introduced in Chapter 2. In each round of the invasion process, each populated vertex tries to hit/populate (independently) all other unpopulated vertices in the landscape graph.

Throughout this part of our work, we consider discrete time steps, and we assume that one time step is sufficiently large so that the flow of the species between two vertices (i.e., population) completes. We analyse the invasion protocol in two models: *asynchronous* and *synchronous*. In the execution of the invasion protocol in the asynchronous model, simply called an asynchronous execution (of invasion protocol), at each step only one edge is selected, uniformly at random, among the m edges of the landscape graph. The selected edge is realised with a defined probability (weight) of populating unpopulated vertex u from populated vertex v, where vertices v and u are connected by the selected directed edge. In a synchronous execution of the invasion protocol, at each *synchronous round* all directed edges in the landscape graph are realised, independently with defined probabilities (weights); in this sense, each (synchronous) round consists of exactly m steps, each done with respect to a different edge. Sometimes, for simplicity, we will be calling asynchronous and synchronous executions of the invasion protocol by asynchronous and synchronous invasions, respectively.

In order to measure the *time performance*, or *runtime*, of invasion process, we measure the number of steps required so that every vertex in the target set is populated, regardless of which vertices are initially populated (worst case analysis). Since in asynchronous invasion only one edge is selected per step, while in synchronous invasion all edges are realised independently with corresponding probabilities (weights) per step, in order to compare them fairly we consider the concept of a *round*, that is a single synchronous round in case of synchronous executions and m consecutive steps in case of asynchronous executions.

In this part of our work, we define the Invasion Time (IT) as the estimated time (total number of rounds) to populate any of the target patches in a given landscape. We will define an invasion landscape network for a given landscape and find a correspondence between conductance and network flows in this network and the invasion time.

4.1.2 Conductance measure, related work and challenges

Conductance

Conductance is an important notion of expansion of a graph. There are different ways to define the conductance of a graph in the literature. Sinclair [105] gives a traditional definition of conductance, while there are also other, slightly different, definitions related to estimation of runtime of rumor spreading, c.f., Mosk-Aoyama et al. [92] and Censor-Hillel et al. [23, 24]. On the other hand, a definition of conductance that measures how *well-connected* the graph is, have been used by Giakkoupis [53] and Sauerwald and Stauffer [103].

In the next sections, we use a generalised version of this definition of conductance, due to its *graph theory* nature. The *conductance* $\Phi(G)$ of a graph $G = (V, E)$ is defined as follows:

$$\Phi(G) = \min_{X \subseteq V : 0 \leq \mathrm{vol}(X) \leq \mathrm{vol}(V)/2} \frac{\sum_{(v,u) \in |E(X, X^c)|} p(v,u)}{\mathrm{vol}(X)} \ ,$$

where $\mathrm{vol}(X)$ is the *volume* of set X and it is defined as $\sum_{v \in X} \sum_{u \in V, (v,u) \in E} p(v,u)$ and $p(v,u)$ is the weight of edge (v,u).

Related work

Species are being extinct faster than natural extinction due to changes in land use, climate change and pollution among others [68, 73, 98, 113]. Ecologists observed that species that are effectively able to respond to these threats do so by shifting/invading their geographical areas [25, 112], however the availability of suitable habitats may limit the ability of species to shift [68, 72, 98, 107]. If species are shifting very slowly or failing to shift, they become more vulnerable to extinct [108]. Therefore, in order to protect ecosystem functioning and services in different environmental changes, it becomes an important need to improve species performance and species interactions especially by facilitating their shifts to new regions with more suitable climate and landscape environment [69]. Conservationists are facing challenges to find out whether and how they can facilitate shifting of species [69]. A number of ecological studies have been shown the importance of spatial arrangement of habitats

on the speed of range shifting [67, 69, 113]. Hodgson et al. [69] found the evidence of the benefits of creating new habitats as "corridors" or "stepping stones" to allow species to shift through unsuitable landscapes and to help them colonise new regions. However, this notion of habitat creation is essentially difficult to test in large landscapes and it becomes even more expensive computation problem when different scenarios need to be tested.

The closest area to the estimation of invasion time is the analysis of rumor spreading protocols. The most popular rumor spreading protocols are: PUSH, in which an informed vertex selects randomly its (out-)neighbour to whom it sends a message, and PULL, in which an uninformed vertex selects its (in-)neighbour from whom it gets the message (provided that the chosen vertex has it). Their modifications and combinations were also considered. These processes differ from the invasion process in many ways. First, they were studied in undirected unweighted graphs; in other words, all weights were set to 1. Second, they run at vertices rather than at edges, which is the case of invasion. Nevertheless, in what follows we give an overview of related results in the rumor spreading area.

Runtime of PUSH-PULL has also been bounded on special graph classes. For complete graphs, it is known that $\Theta(\log n)$ rounds are sufficient [34, 42, 47, 74] while for social networks represented as Preferential Attachment graph, Chierichetti et al. [28] showed that PUSH-PULL protocol informs all vertices within $O(\log^2 n)$ rounds, whp. The latter has been further improved to $\Theta(\log n)$ by Doerr et al. [43]. Doerr et al. [43] also showed how to obtain optimal runtime for these types of graphs by modifying PUSH-PULL protocol.

In relating runtime of rumor spreading with conductance in general graphs, Chierichetti et al. [27] first showed that PUSH-PULL protocol broadcasts the message within $O(\Phi^{-6}(G)\log^4 n)$ rounds whp, and later the same authors improved the bound to $O(\frac{\log^2 1/\Phi}{\Phi}\log n)$ in [26]. Giakkoupis [53] closed the gap by providing a tight bound of $\Theta((1/\Phi)\log n)$.

Censor-Hillel and Shachnai [24] slightly modified the random protocol by adding some determinism in it. They analysed their protocol based on a different notion they called "weak conductance". Recently, Censor-Hillel [22] modified PUSH-PULL protocol in order to solve the information dissemination problem with no dependence on conductance and showed that this new protocol solves the rumor spreading problem in at most $O(D + \text{polylog}(n))$ rounds in a graph of diameter D. Nevertheless, their protocol is not the classical PUSH-PULL protocol.

The results mentioned above apply to the synchronous model. In the asynchronous model, Sauerwald [102] showed that for PUSH protocol, the expected number of rounds for rumor spreading is asymptotically equivalent to the expected number of rounds in the

synchronous model for PUSH protocol. He introduced a new measure that considers for each $1 \leq k \leq n-1$ the subset X of k vertices that minimises $\sum_{v \in X} deg_{X^c}(v)/deg(v)$ (denoted as Φ_k). The expected runtime of PUSH protocol in the asynchronous model has then been shown to be at most $\sum_{k=1}^{n-1} 1/\Phi_k$. More recently, Kowalski and Thraves Caro [77] defined another new measure that is relatively tight estimate of runtime of rumor spreading by PUSH-PULL protocol.

Challenges

Whilst it has been shown in [68, 69] how the threat of species' extinction could be limited by creating new habitats as "corridors" or "stepping stones" in order to allow species to shift through unsuitable landscapes and to help them colonise new suitable regions, however, it is still difficult for conservationists to make decisions that can facilitate range shifts in large landscapes and may even determine survivability of the species. From computational perspective, estimating the invasion time by running simulations is very time consuming, as the existing full model is based on a Markov Chain of exponential number of states with respect to the landscape size; therefore, in practice, this method is not suitable especially in case of frequent environmental changes or for environmental planning.

The main challenge in this part of our work is:

> *how to estimate the duration of the invasion process especially in large landscapes in a fast way with high accuracy.*

We first adapt the new measurement that Kowalski and Thraves Caro have recently developed [77]. This measurement gives a tighter estimate of the time of rumor spreading and therefore we analyse and relate this time to the expected time of the invasion process. Therefore, the challenge here is to extend the work from undirected unweighted network; in other words, all weights were set to 1 to directed weighted network, which is the case of invasion. Furthermore, the second related challenge is to give a more precise prediction formula for the invasion time such that we get a good trade-off between quality of estimate and actual computation time.

4.1.3 Contributions and organisation of the chapter

Contributions

Our contribution is three-fold. We firstly introduce a new theoretical measure γ that estimates the expected runtime of invasion process for a given landscape. In particular:

- We show that the proposed measure estimates from above the expected number of rounds in asynchronous and synchronous execution, in the latter with additional logarithmic (additive) component (Theorem 2).
- We prove that the new measure is inversely proportional to the conductance of the proposed flow network with a logarithmic factor (Theorem 3).

Secondly, based on our theoretical investigations, we introduce and justify more precise prediction formula for the Invasion Time (IT) (Equation (4.1)). We compare how good the obtained prediction formula is with respect to the previously used *full* simulation method, by applying these methods (i.e., IT formula and *full* simulation) to three real heterogeneous landscapes in Great Britain. These landscapes are of low, medium, and high quality. We then interpolate, by simulation, constant c in the prediction IT formula (Equation (4.1)) in order to make it even more accurate.

Finally, we verify our new prediction formula IT by performing the formula and simulation on a random selected landscape of mixed quality and of height that is different than the height in the studied landscapes (i.e., the three landscapes mentioned above) and comparing the obtained invasion times from these methods. Following that, we show that by the prediction IT formula we obtain quite accurate estimates of the invasion runtime in much lower computational cost (computation time and memory).

We believe that our theoretical and simulational results form a convincing background for further work in this direction, to obtain even better accuracy in prediction of invasion time for real landscapes.

Organisation of the chapter

Section 4.2 introduces a new theoretical measure γ that estimates the expected runtime of invasion process for a given landscape. It is also introduces a new prediction formula IT for the Invasion Time. Section 4.3 shows the theoretical analysis of the new theoretical measure γ to approximate the number of rounds (round complexity) in asynchronous and

synchronous executions of the invasion protocol. Section 4.4 focuses on adjusting the proposed prediction formula (i.e., IT formula) using simulation and presents the results for the considered landscapes of low, medium, and high quality. Section 4.5 focuses on verifying the prediction formula in a random selected landscape of mixed quality and presents the results of verification. Section 4.6 illustrates the improvement in memory and time using the proposed IT formula. Finally, Section 4.7 discusses some possible future work.

4.2 The new measure

Let us define a new measure to estimate invasion time — the γ measure. This measure can be seen as the weighted and edge-oriented version of the measure introduced by Kowalski and Thraves Caro [77] in the context of rumor spreading.

For a given network $N = (s, t, V, E)$, with source s, target t, set of intermediate vertices V and set of directed weighted edges E, we define the following new parameter:

$$\beta_k(N) = \min_{X: X \subseteq V, s \in X, t \notin X, |X|=k} \sum_{v \in X} \sum_{u \in X^c} p(v, u) \ ,$$

where $p(v, u)$ is the weight of edge (v, u). This parameter captures the worst-case bi-directional expansion through a cut of one border of size k. Our new measure, used later for estimating invasion time, is defined as follows:

$$\gamma(N) = \sum_{k=1}^{n-1} \frac{1}{\beta_k(N)} \ .$$

We may skip parameter N from the above parameters $\gamma(N)$, $\beta_k(N)$ if network N is clearly understood from the context.

4.2.1 Estimating parameters via network flow

We propose to use a network flow approach to model and estimate the invasion process. For this, we build a network to represent a given landscape graph G of size $H \times W$ in the case of invasion (recall the definition of invasion time in Section 4.1). We adapt our proposed sparsification method in Chapter 3 to compute the distance R using Equation (3.4) for the given landscape graph G, therefore the invasion time could be well estimated while restricting to links between patches of distance at most R. We number columns

starting from 0 and we assume that the area (patches) between column 0 and 9 inclusive is populated. We also assume that column number 19, 29, 39, ..., and $W-11$ as the first column of the target area, where the target area is of 10 columns length. For each prefix of size $H \times 29$, $H \times 39$, ..., $H \times W-2$, we define a sub-landscape graph G' of size $H \times (\max\{0, 10i-2R-10\} : 10i-1+9)$, where $i = [2, \frac{W-10}{10}]$. Then, we build an invasion network N to represent G' graph as follows.

Invasion network N and estimating Invasion Time (IT). The invasion network N for the sub-landscape graph G' of size $H \times (\max\{0, 10i-2R-10\} : 10i-1+9)$ is defined as follows. We distinguish two sets of vertices (each of length 10 columns): the initial populated set S which contains vertices between column $\max\{0, 10i-2R-10\}$ and column $\max\{0, 10i-2R-1\}$, including these two columns, and the target set T which involves vertices in the area between column $10i-1$ and column $10i-1+9$, including these two columns. We add a virtual source vertex s and connect it to each vertex in the initial populated set S by a directed edge with a weight λ, where λ is the maximum, over all vertices, of the sum of the weight of adjacent edges, $\lambda = \max\{\lambda_v : v \in V\}$, where $\lambda_v = \sum_{u \in V} p(v,u)$. We also add a virtual target vertex t and connect each vertex in the target set T to the additional target vertex t by a directed edge with weight λ. Each intermediate vertex (except the source vertex s and the target vertex t) is connecting to all other vertices by edges and given weights that equal to the *transition probabilities* $p(v,u)$ between the patches, using Equation (2.1). A constructed invasion network N for a given landscape is given in Figure 4.1. We compute the network flow of this constructed network with the weight as the capacity.

For a given landscape graph G of size $H \times W$, we consider column number $19, 29, 39, \ldots, W-11$ as the first column of the target area and we compute invasion time for each target area. The estimated Invasion Time (IT) for the target area of 10 columns length and started at column $10i-1$ is defined as:

$$IT(10i-1) = \begin{cases} \frac{\ln(\hat{\#}[0:10i-1+9])}{\text{max-flow}([0:10i-1+9])} \cdot c \\ \qquad\qquad \text{if } 1 < i \leq \frac{W-10}{10} \text{ and } 10i-1 < 2R+9 \\ IT(10i-11) + \frac{\ln(\hat{\#}[10i-2R-10:10i-1+9])}{\text{max-flow}([10i-2R-10:10i-1+9])} \cdot c \\ \qquad\qquad \text{if } 1 < i \leq \frac{W-10}{10} \text{ and } 10i-1 \geq 2R+9 \end{cases} \tag{4.1}$$

where c is a small constant to be interpolated by simulation in Section 4.4 and $\hat{\#}$ is the

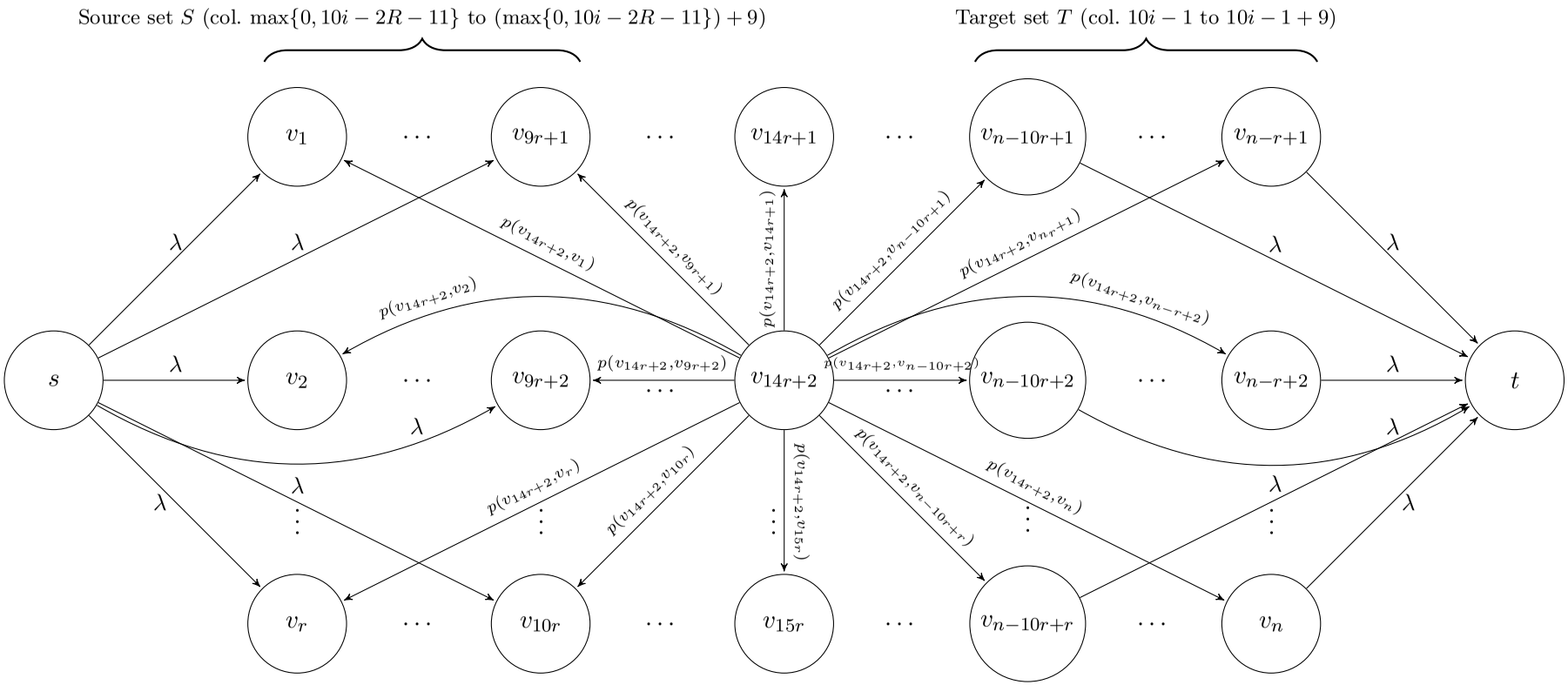


Figure 4.1: The constructed invasion network N for the sub-landscape graph G' of a given graph G of size $H \times W$. Each vertex (except s and t) is connecting to all other vertices by edges and given weights which is equal to the *transition probabilities* p between the patches, as shown by vertex v_{14r+2}.

total number of non-zero patches between two specified columns [start-column:end-column], including these two columns.

In the following Section 4.3 we develop a theory justifying that the formula defined by Equation (4.1) is a good asymptotic estimate of the invasion time.

4.3 Theoretical part — estimates of invasion runtime in real landscapes

Recall that for a given landscape graph $G = (V, E)$, distinguished source and target vertices (patches), the invasion process is to populate any of target vertices when starting from the source vertices, and we aim to approximate the time performance of this process. The vertex is populated by means of an invasion protocol. The invasion protocol keeps checking edges, one after another (recall definitions of synchronous and asynchronous executions in section 4.1.1), by generating a random number from 0 to 1; if the number is smaller than the weight of the edge and the beginning of the edge is populated, then the end of the edge becomes populated as well (unless it has been populated earlier).

Using the new measure γ, in this section we first give the upper bound of the expected number of rounds (round complexity) in asynchronous and synchronous execution of the invasion protocol on any given landscape graph. Then, we present the upper bound of the invasion time of the invasion protocol (asymptotically) on any given landscape graph.

Theorem 1. *For any landscape graph G, the expected* asynchronous *round complexity of invasion protocol is $O(\gamma(N))$, where N is the invasion landscape network of landscape graph G.*

Proof. We partition an asynchronous execution of invasion protocol into consecutive phases as follows: phase k contains steps in which exactly k vertices are populated. Note that some phases may be empty, and the partition into phases is related with the partition into asynchronous rounds, though the former depends on the random choices made in asynchronous steps while the latter is fixed (i.e., each round contains exactly m asynchronous steps).

Observe that the expected number of steps in phase k is at most $1/\beta_k(N)$. Indeed, let W be the set of k populated vertices during phase k. The probability that a single population operation populates some unpopulated vertex in W^c from some populated vertex in W (i.e., the probability of selecting an edge between W and W^c) is $\frac{1}{m}\sum_{v\in W}\sum_{u\in W^c} p(v,u)$.

Therefore, the probability of populating some unpopulated vertex, when the phase will terminate, is at least

$$\frac{1}{m}\sum_{v\in W}\sum_{u\in W^c} p(v,u) \geq \frac{\beta_k(N)}{m}\ .$$

Because population operation is applied independently for edges in consecutive steps, the expected time for termination of the phase is at most $\frac{m}{\beta_k(N)}$. Hence, the expected time of invasion, in terms of steps, is the sum of expected numbers of steps over $1 \leq k \leq n-1$, which gives at most $\sum_{k=1}^{n-1} \frac{m}{\beta_k(N)}$ steps, which in turn is equal to $\gamma(N)$ asynchronous rounds. □

The proof of the following theorem is analogous to the one in [77] (the main difference is that we consider random sequence of edges instead of vertices) and we give the proof for completeness.

Theorem 2. *For any landscape graph G, the number of rounds in an asynchronous execution of invasion protocol is asymptotically not bigger than the number of rounds in a synchronous execution of invasion protocol plus $\log n$, with high probability. On the other hand, the expected number of rounds in a synchronous execution of invasion protocol on landscape graph G is $O(\gamma(N)+\log n)$, where N is the invasion landscape network of landscape graph G.*

Proof. Consider an asynchronous execution of invasion protocol. Partition asynchronous steps into consecutive rounds. Consider a modified invasion protocol, executed in asynchronous model, in which each vertex remembers whether it has already populated some unpopulated vertex in the current round or not; if it has, it skips any other possible action in this round even if it is selected by the asynchronous mechanism (i.e., by the model feature that randomly schedules edges to execute their populating operation in the execution). This protocol is clearly not faster than the original invasion protocol in asynchronous model.

In order to compare the round complexity of an asynchronous execution of the modified protocol with a synchronous execution of invasion protocol, let us fix a sequence of random bits used by edges in a synchronous execution of invasion protocol, i.e., bits for performing populating operation. Let G be a given n-vertex landscape graph. Consider a vertex v and the path from the source of the invasion to vertex v through which populating v is performed in the considered execution of invasion with the fixed bits (i.e., to each vertex on this path, the first populated vertex comes through its predecessor on this path). Now consider an asynchronous execution of the modified protocol for the same set of random bits. Note that we fix only bits used for populating operation, while bits used to generate an asynchronous order of active edges are still random.

The probability that the number of rounds needed for populating vertex v through the same path in asynchronous execution of the modified protocol is asymptotically different than the number of rounds used in the corresponding synchronous execution (i.e., for the same fixed sequence of random bits for invasion) plus/minus $\log n$, is polynomially small. Here the probability distribution is over random bits used for selecting an asynchronous order of edges. This result holds by applying Chernoff bound and the fact that the probability of not skipping a round by a vertex on the path is at least $1 - (1 - 1/n)^n = \Omega(1)$. Taking a union bound of the above events, over all sequences of bits for invasion operation and over all edges m, we obtain the result claimed in the first part of the theorem.

In order to prove the second part, we observe that the modified protocol in asynchronous environment completes invading targets on landscape graph G in $O(\gamma(N))$ rounds, in expectation. Indeed, we enhanced the proof of Theorem 1 in a way to capture the skipped steps made by the modified protocol: an invasion step is skipped with a constant probability, as the expected number of repetitions of vertices within a round is a constant. This probability introduces an additional constant in front of all the formulas in the original proof of Theorem 1, which however does not change the asymptotic result of $O(\gamma(N))$ rounds, in expectation. Next we apply the relation between the round complexity of a synchronous execution of invasion and an asynchronous execution of the modified protocol, prove in the previous paragraph. This immediately implies the expected $O(\gamma(N) + \log n)$ round complexity of a synchronous execution of invasion protocol. □

It follows that the asymptotic bounds on synchronous invasion protocol apply also to asynchronous invasion protocol, modulo an additive logarithm of n.

The following theorem is the main theoretical finding, which will be later used for comparison with simulation results. Denote $\max_{v \in V} \mathrm{vol}(v)$ by $\mathrm{vol}_{\max}$ and $\min_{v \in V} \mathrm{vol}(v)$ by $\mathrm{vol}_{\min}$, and assume that they are constants (depending on the landscape).

Theorem 3. *For every landscape with n patches and its invasion landscape network N, it holds that the invasion time of the invasion protocol is $O\left(\lceil \frac{\log n}{\Phi(N)} \rceil\right)$, where $\Phi(N)$ is the conductance of N.*

Proof. By Theorem 2, it is enough to show that $\gamma(N) = O\left(\frac{\log n}{\Phi(N)}\right)$. The following holds:

$$\begin{aligned}
\beta_k(G) &= \min_{X:X\subseteq V,|X|=k} \sum_{v\in X} \sum_{u\in X^c,(v,u)\in E} p(v,u) \\
&\geq \min_{X:X\subseteq V,|X|=k} \left(k \cdot \mathrm{vol}_{\min} \cdot \frac{\sum_{v\in X} \sum_{u\in X^c,(v,u)\in E} p(v,u)}{\sum_{v\in X} \mathrm{vol}(v)} \right) \\
&\geq k \cdot \mathrm{vol}_{\min} \cdot \Phi(N) \ .
\end{aligned}$$

Finally, we get

$$\gamma(G) = \sum_{k=1}^{n-1} \frac{1}{\beta_k(G)} \leq \sum_{k=1}^{n-1} \frac{1}{k \cdot \mathrm{vol}_{\min}\Phi(N)} = \frac{H_{n-1}}{\mathrm{vol}_{\min}\Phi(N)} = O\left(\frac{\log n}{\Phi(N)}\right) \ .$$

□

Observe that, although the exact computation of conductance (or γ) is hard, for small landscapes N of width $2R$ with added source and target patches, where R is the sparsification distance from Equation (3.4), $\Phi(N)$ could be well estimated by minimum cut in the graph, which is equal to maximum flow in the graph (by the well-known min-max theorem). This combined with Theorem 3 justifies the (asymptotic) prediction formula IT introduced in Equation (4.1): the formula is a sum of estimates of the invasion times over a sequence of small and partly overlapping landscapes (of width $2R$). In Section 4.4 we will interpolate constants in the formula, and in Section 4.5 we verify it for a randomly selected landscape.

4.4 Adjusting the prediction formula by simulations

In this section we presents the studied landscapes in this chapter that used to examine the proposed prediction formula (Equation (4.1)). We also describe the simulation environment and adjust the prediction formula using simulations.

4.4.1 The studied landscapes

As mentioned in Chapter 2, the dataset of the 1km resolution raster version of the Land Cover Map 2007 (LCM2007) for Great Britain [91] is used for evaluation purposes. To examine and evaluate the proposed prediction formula, three landscapes from different maps

of the aggregate classes are extracted. Recall that in such an extracted landscape, if the average of all patches' qualities is between 0% and 5%, 5% and 25%, 25% and 100%, then the landscape is of low, medium, and high quality, respectively. For each quality type, we extract a rectangular landscape that consists of 5 rows/height and 299 columns/width (see Figure 4.2). Landscapes of low and medium qualities were extracted from the semi-natural grassland GB map (aggregate class), while the one of high quality from the improved grassland GB map (aggregate class).

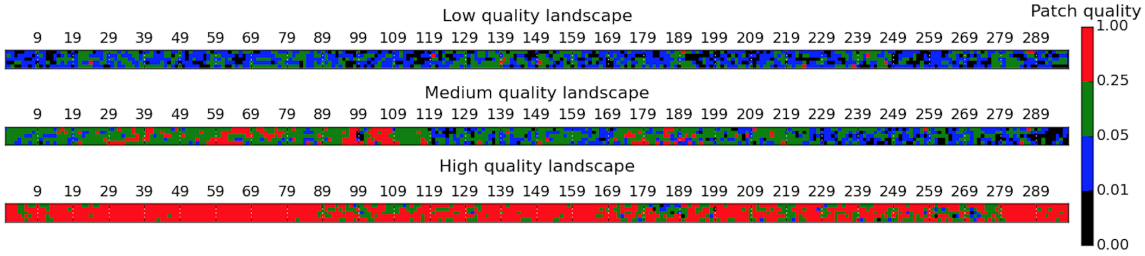


Figure 4.2: The studied landscapes. Three rectangular landscapes of size 5×299 and of low, medium and high quality extracted from LCM2007 GB maps (aggregate classes). In each landscape, the colour corresponds to the quality of each patch; black, blue, green, and red corresponds to zero, low (0.01-0.05), medium (0.05-0.25), and high quality (0.25-1), respectively.

On these extracted rectangular landscapes, it has been assumed that all patches at the first 10 columns of each landscape are occupied by species and the goal is estimate the time needed to populate any of patches at an target area. It has been assumed that the length of the target area is 10 columns. Columns number 19, 29, 39, ..., W-11 have been considered as the first column of the target areas for the occupied patches at the first 10 columns. Therefore, each landscape has been extracted according to the following criteria:

1. All source patches at the first 10 columns are non-zeros in quality.
2. At least one of the patches at each target area (i.e., 19-28, 29-38, 39-48 etc.) is non-zero in quality.

4.4.2 Simulation setting and adjusting prediction

The simulation is directed at three goals. The first is to compute the invasion time using the *full* simulation method and investigate how the dispersal coefficient α and landscape quality

affect on the invasion duration. The second is to compute the invasion time based on the proposed prediction method in Section 4.2.1 (Equation (4.1)), which is based on computing maximum flow for the constructed invasion landscape network N using one of the known algorithms[1]. The third goal is to adjust the prediction formula IT (Equation (4.1)) using simulations. Finally, we aim to combine results from simulation and the proposed prediction formula for invasion and compare them.

One important characteristics to consider when computing the total time of invasion is also the number of iterations needed for the (average) invasion time to stabilise on the outputted duration of invasion. Accordingly, we define the *Stabilisation Time (ST)* for a given landscape as the time t such that the change in average number of rounds for the Invasion Time (IT) between t and $2t$ is less than or equal to 2%: $\forall \tau \in (t, 2t]$, $|IT(\tau) - IT(t)| \leq 0.02 \cdot IT(t)$, where $IT(\tau) = \frac{\sum_{j=1}^{\tau} Rounds(j)}{\tau}$ and $Rounds(j)$ is the number of rounds needed for invasion at iteration j.

For each prefix $5 \times 30, 5 \times 40, 5 \times 50, \ldots, 5 \times 299$ in each extracted rectangular landscape in Figure 4.2 we run the following procedure. We run *full* simulation until stabilisation point and compute the average number of rounds over 2ST independent iterations (approximated invasion time) and the execution time of simulation. Then, we compute the invasion time based on computing maximum flow for the constructed invasion landscape network N (as described in Section 4.2.1) and the execution time of computing flows (as presented in Algorithm 3 with input parameters and output variables presented in Table 4.1). Finally, we compare the invasion time produced by these two methods by computing the ratio between them (invasion time by simulation over invasion time by prediction formula). That has been done for different values of the dispersal coefficient α: 0.25, 0.5, 1 and 2.

Interpolating constant c in IT formula (Equation (4.1)) based on simulations. In order to interpolate constant c in the IT prediction formula (Equation (4.1)), we do the following. We first compute the ratio of invasion time (simulation results over prediction results assuming $c = 1$) for each prefix in each of the studied rectangular landscapes (low, medium, high). Then, we divide the ratio of the medium and high qualities by the ratio of low quality. Finally, we compute constant c that gives the smallest distance between simulation and prediction curves (the best c that makes the invasion time ratio closer to 1). Table 4.2 shows the interpolated constant c (using simulation) for each landscape quality and each dispersal coefficient α.

[1]We use NetworkX package in Python programming language to calculate maximum flow.

Table 4.1: Input parameters and output variables for Algorithm 3.

Input parameters:
1. G: 2-dimensional array stores qualities of patches in a given real landscape
2. S: vector containing indices (i, j) of initial populated patches (source patches)
3. T: vector containing indices (i, j) of unpopulated target patches
4. α: given number > 0
Output variables:
1. Maximum flow for the invasion network
2. Execution time of computing max-flow

Algorithm 3 Modelling invasion process using network flow method (G, S, T, α)

function COMPUTE MAX-FLOW(G, S, T, α)
  start ← record start time of computing max-flow
  rows ← number of rows in G
  columns ← number of columns in G
  patches ← rows · columns
  $s \leftarrow 0$
  $t \leftarrow$ patches+1
  Create array $C[patches + 2, patches + 2] \leftarrow 0$
  for each non-zero patch in S **do**
    $C(s, patch) \leftarrow \lambda$
  for each non-zero patch in T **do**
    $C(patch, t) \leftarrow \lambda$
  for $i \leftarrow 1$ to patches **do**
    for $j \leftarrow 1$ to patches **do**
      $C(i, j) \leftarrow$ *Transition probability* between patches i and j
  Max-flow← compute maximum flow (C, s, t)
  end ← Record end time of computing max-flow
  ExecutionTime ← end - start
  return Max-flow, ExecutionTime

4.4.3 Simulation results vs. prediction results

For each prefix $5 \times 30, 5 \times 40, 5 \times 50, \ldots, 5 \times 299$ in each extracted rectangular landscape of low, medium, and high quality, we compute the approximated invasion time (i.e., average

Table 4.2: The interpolated constant c for each landscape quality and dispersal coefficient α.

Landscape quality	α=0.25	α=0.5	α=1	α=2
Low	1	1	1	1
Medium	1.3	1.45	1.65	1.8
High	1.35	1.65	1.95	2.35

number of rounds over 2ST independent repetitions) using *full* simulation method and the proposed method (i.e., Equation (4.1) with constants in Table 4.2) which based on computing maximum flow in the constructed invasion landscape network. Both simulation and prediction results show that the invasion time grows with the growth of the prefix width. Prefixes of low quality take longer time (larger number of rounds) than medium quality to be invaded, while high quality prefixes need shorter time. The increase in the dispersal coefficient α from 0.25 to 2 shows an increase in the invasion time in all qualities (for detail results, see Figures 4.3-4.5).

Importantly, based on theoretical analysis of the new γ measure to approximate the number of rounds (invasion time) in Section 4.3 we compute the ratio between the computed invasion times by simulation and prediction for each prefix. Comparison of the obtained results based on the *full* and flow methods shows that our proposed flow-based estimation (Equation (4.1) with constants in Table 4.2) gives a good approximation for the invasion time, as shown in Figure 4.6. The invasion time ratio is around one in all types of quality (low, medium, and high) and for different values of the dispersal coefficient α.

4.5 Verification of the prediction formula

In order to test the robustness of our proposed prediction method (i.e., Equation (4.1) with constants in Table 4.2), we performed verification in a landscape of mixed quality which is selected randomly from semi-natural grassland GB map [91], see Figure 4.7. To ensure robustness, we verify our new method on a landscape of 10 rows and 299 columns, where the height is different than the height in landscapes used in Section 4.4 (we double the height).

For each prefix $10 \times 30, 10 \times 40, 10 \times 50, \ldots, 10 \times 299$ on this extracted landscape and for each considered value of the dispersal coefficient α, we run *full* simulation method and our proposed prediction method (i.e., Equation (4.1) with constants in Table 4.2) in order to get the estimated time of invasion as well as the execution time of simulation and prediction. Then, we compute the ratio of the estimated time of invasion (i.e., invasion time

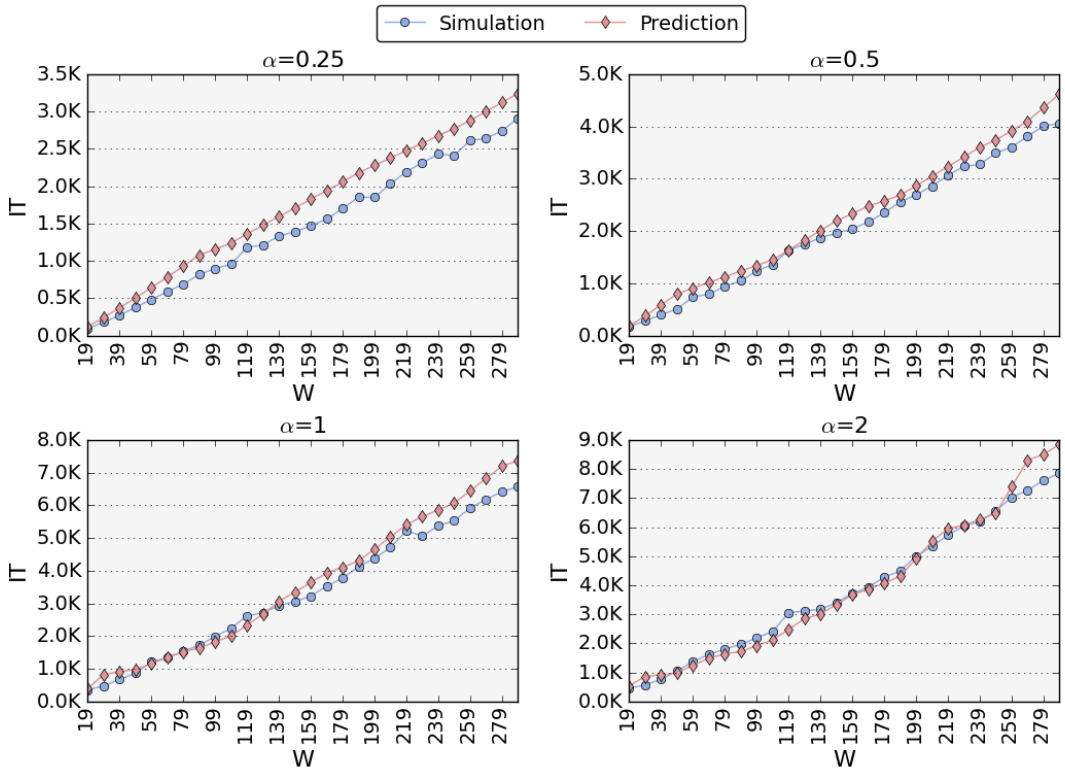


Figure 4.3: The Invasion Time (IT) using *full* simulation method and the proposed prediction method for each prefix in the landscape of size 5×299 and of **low** quality. This is done for different values of the dispersal coefficient α: 0.25, 0.5, 1 and 2.

by simulation over invasion time by prediction).

Figure 4.9 gives the ratio of invasion time. As presented in the figure, the ratio is between 0.5 and 1 for α equal to 0.25 and 0.5 and is much better and much closer to one for α equal to 1 and 2 (for detail results, see Figure 4.8).

Additionally, we compute the error rate between the presented invasion time ratios in Figures 4.6 and 4.9 to see the precision of our prediction when the height of the landscape is doubled. The error rate for a fixed α is defined as: $\frac{\sum_{i=1}^{j}(a-b)^2}{j}$, where a is the average of the invasion time ratio over all three types of quality, b is the invasion time ratio in the mixed qualities landscape, and j is the total number of prefixes. It has been illustrated by the verification experiment that the accuracy of our new prediction method is very good as when the height of the landscape is increased to the double and the chosen landscape is of

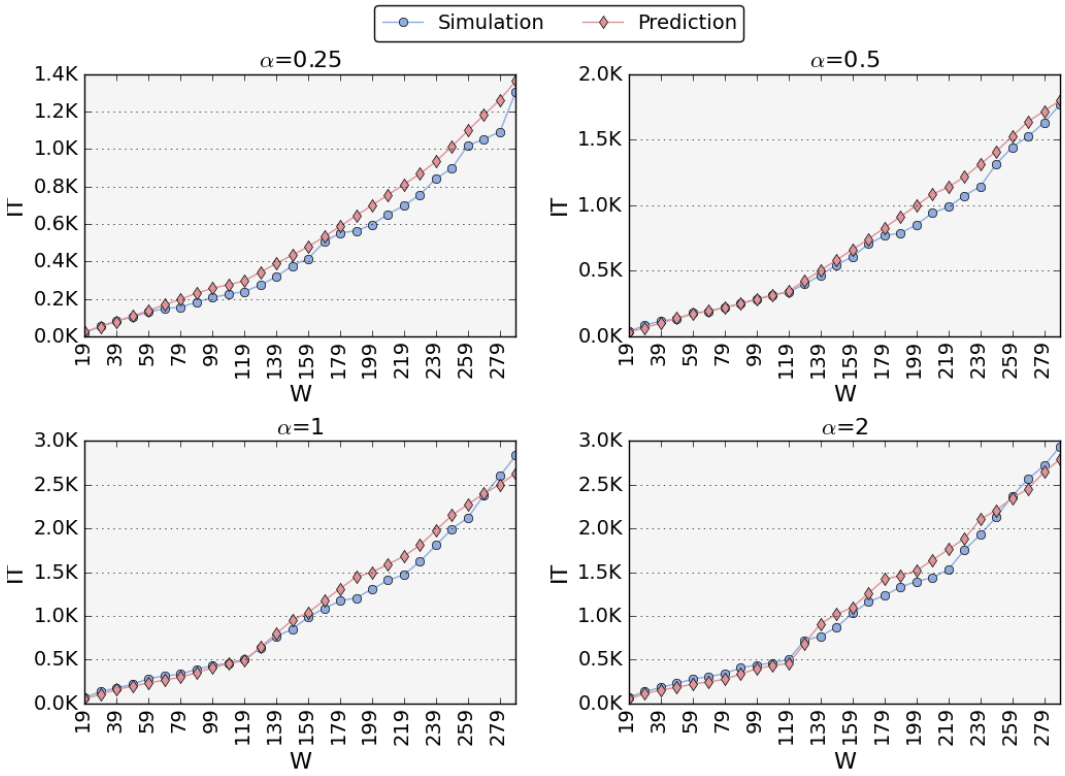


Figure 4.4: The Invasion Time (IT) using *full* simulation method and the proposed prediction method for each prefix of the **medium** quality landscape of size 5×299, for $\alpha = 0.25, 0.5, 1, 2$.

mixed qualities, the error rate is small between 2% and 4%. The error rates are 0.04, 0.02, 0.04, and 0.04 when α are 0.25, 0.5, 1, and 2, respectively.

To summarise, the verification experiment has confirmed that the desired accuracy (simulation over prediction $\simeq$1) has been achieved using the developed formula for the invasion time. Moreover, the error rate between the invasion time ratios (Figures 4.6 and 4.9) is small (0.02 & 0.04) for all values of the dispersal coefficient α. On the other hand, it seems that the precision of our prediction drops with the increase in the height of the landscape.

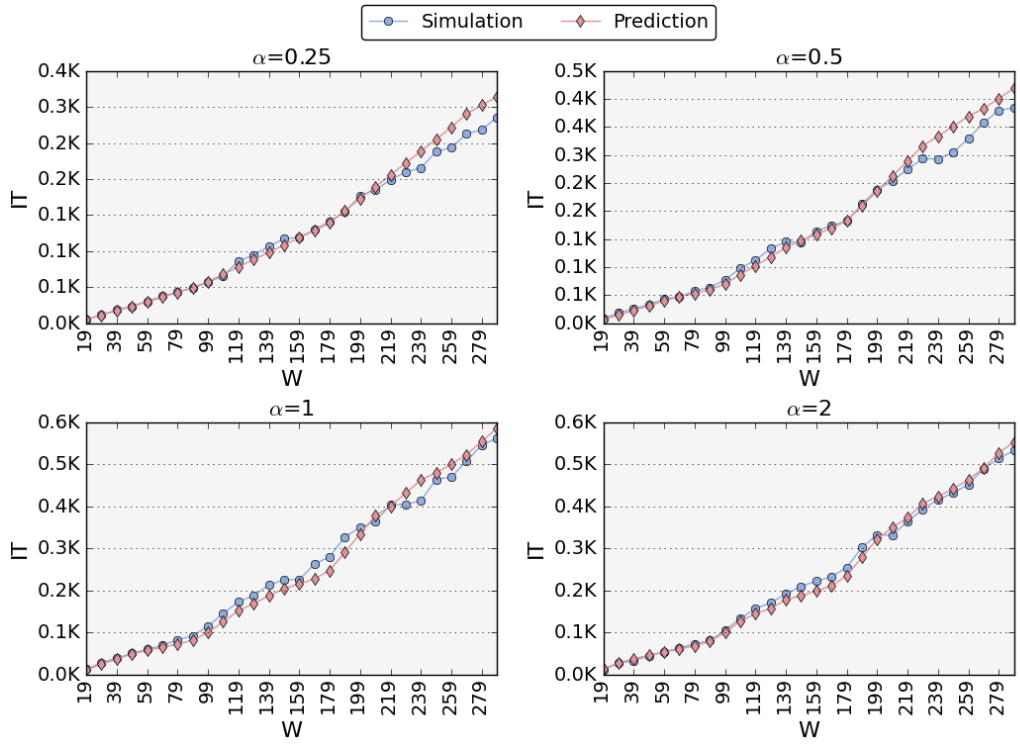


Figure 4.5: TheInvasion Time (IT) using *full* simulation method and the proposed prediction method for each prefix of the **high** quality landscape of size 5×299, for $\alpha = 0.25, 0.5, 1, 2$.

4.6 The improvement in time and memory

This section shows how much computational time and memory we save when using the proposed prediction method (Equation (4.1)) for properly selected distance R (using Equation (3.4)) to compute the invasion time.

4.6.1 Saved time

The carried experiments in this part of our work demonstrate that the actual execution time needed to compute the estimated duration of the invasion process is substantially reduced by our new prediction method (Equation (4.1)) for all landscapes of different qualities. Figure 4.10 illustrates how much the prediction method is faster than the *full* simulation method for all four considered landscapes. In many cases, the *full* simulation method takes 10-10000 times longer to compute invasion time and this ratio can become as high as

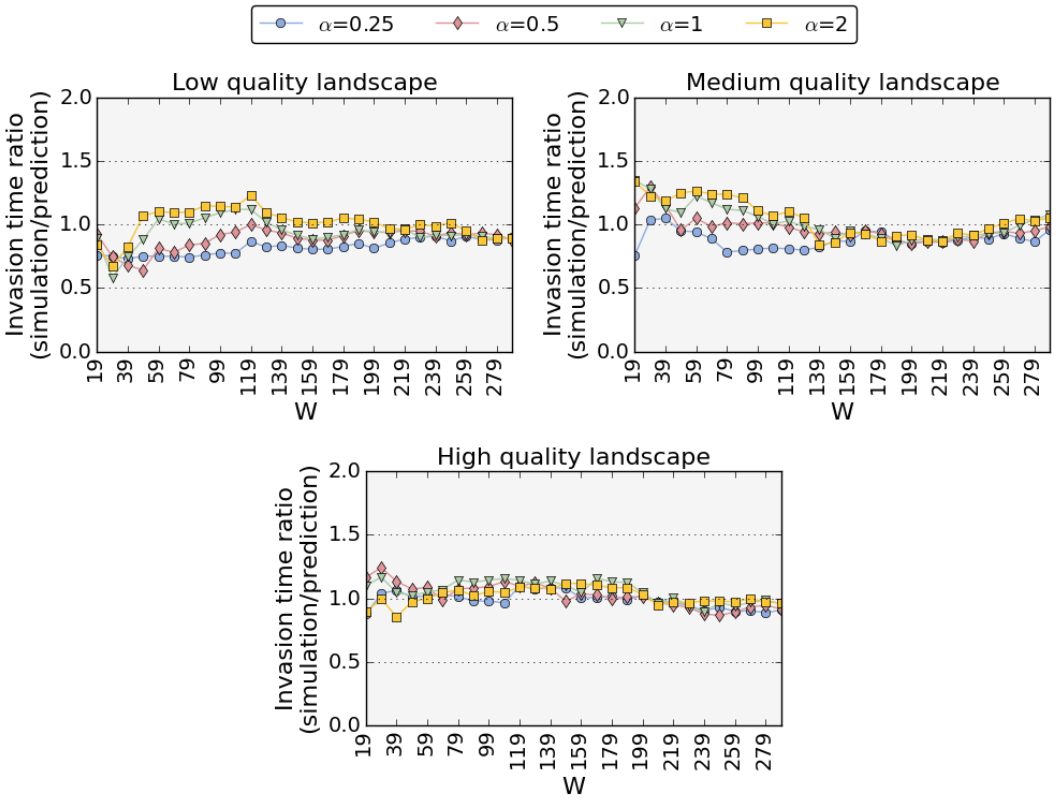


Figure 4.6: The ratio of invasion time (simulation over prediction) for each prefix in each of the studied landscapes (low, medium, high) when the dispersal coefficient α takes four values: 0.25, 0.5, 1 and 2.

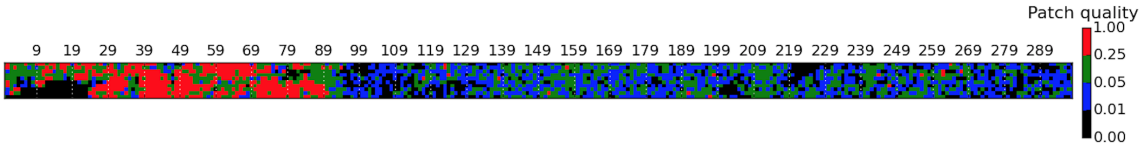


Figure 4.7: Mixed quality landscape of size 10×299 used to verify the new proposed prediction method. The colour corresponds to the quality of each patch; black, blue, green, and red corresponds to zero, low (0.01-0.05), medium (0.05-0.25), and high quality (0.25-1), respectively.

94000 for low quality landscapes. We note that in general, for a given landscape width, the speedup of the prediction method increases as the dispersal coefficient α increases. On the

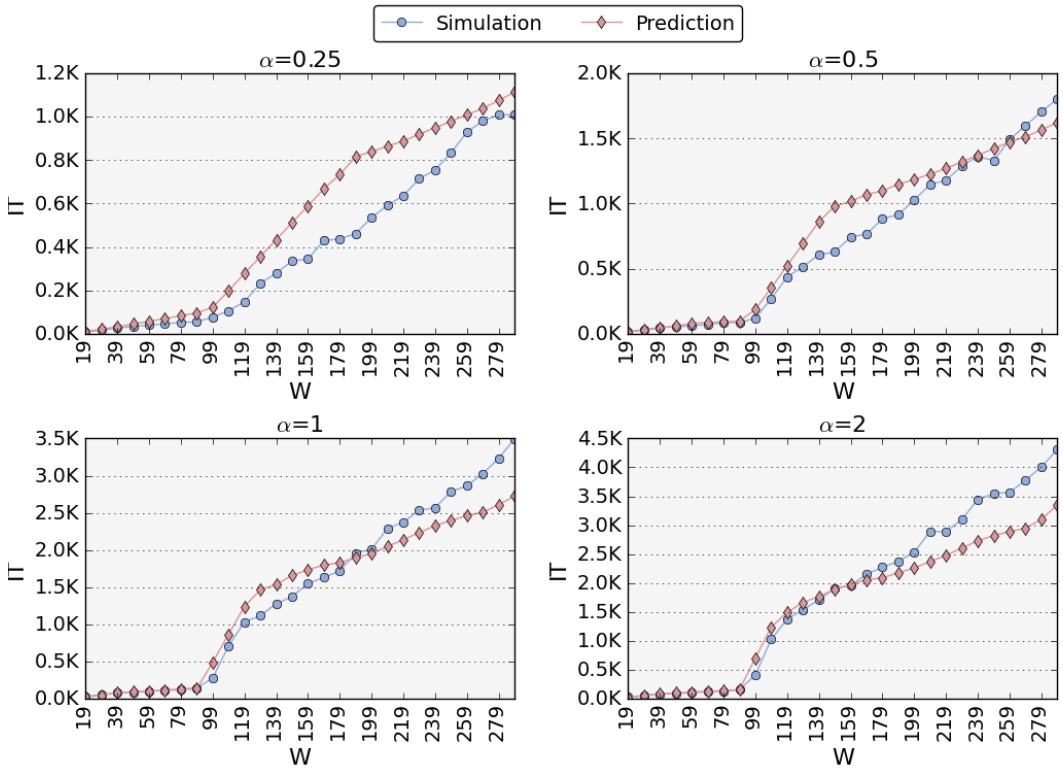


Figure 4.8: The Invasion Time (IT) using *full* simulation method and the proposed prediction method for each prefix in 10×299 landscape of **mixed** qualities, for $\alpha = 0.25, 0.5, 1, 2$.

other hand, in most cases, when α is fixed, the speedup increases as the width of landscape increases.

In more details, in the landscape of size 5×299 and of low quality, the execution time of *full* simulation is 9086.29 seconds while it takes only 0.09 seconds to compute invasion time using the new prediction method. That means the *full* simulation method takes around 94000 times longer to compute. We could envisage that when we are running the *full* simulation in a very large landscape e.g., a landscape of size 500×500, the computation time will be substantially reduced from maybe days/weeks to hours.

4.6.2 Saved memory

In order to investigate the saved memory, we compute the sparsification rate. It is defined as the ratio of the total number of edges between patches in the landscape over the total

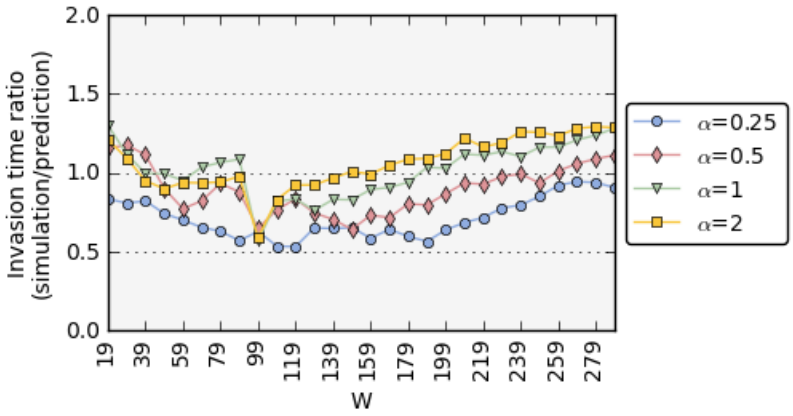


Figure 4.9: The ratio of invasion time (simulation over prediction) for each prefix in 10×299 landscape of mixed qualities when the dispersal coefficient α takes values: 0.25, 0.5, 1 and 2.

number of edges between patches in the constructed invasion landscape network N; in both numbers we count only edges with both ends of non-zeros quality. Figure 4.11 illustrates the sparsification rate versus the landscape width.

Generally, the relation between the sparsification rate and the landscape width can be seen as a linear tendency, where the linear coefficient grows linearly with α. As can be seen in Figure 4.11, the sparsification rate depends on the computed local distance R for each α: the rate increases with the decrease of R. For instance, the rate increase in all qualities for $\alpha = 2$ refers to a decline in R; where R depends on α: R declines with the increase in α from 0.25 to 2 (see R values in Table 4.3).

Table 4.3: The computed distance R (using Equation (3.4)) for each landscape quality and dispersal coefficient α.

Landscape quality	α=0.25	α=0.5	α=1	α=2
Low	38	21	12	8
Medium	40	22	13	8
High	42	22	12	7
Mixed	41	22	12	10

To summarise, the experimental part in this part of our work has confirmed the following: the desired accuracy (simulation over prediction $\simeq$1) has been achieved using the developed formula for the invasion time (Equation (4.1)) and constants c in Table 4.2. Moreover,

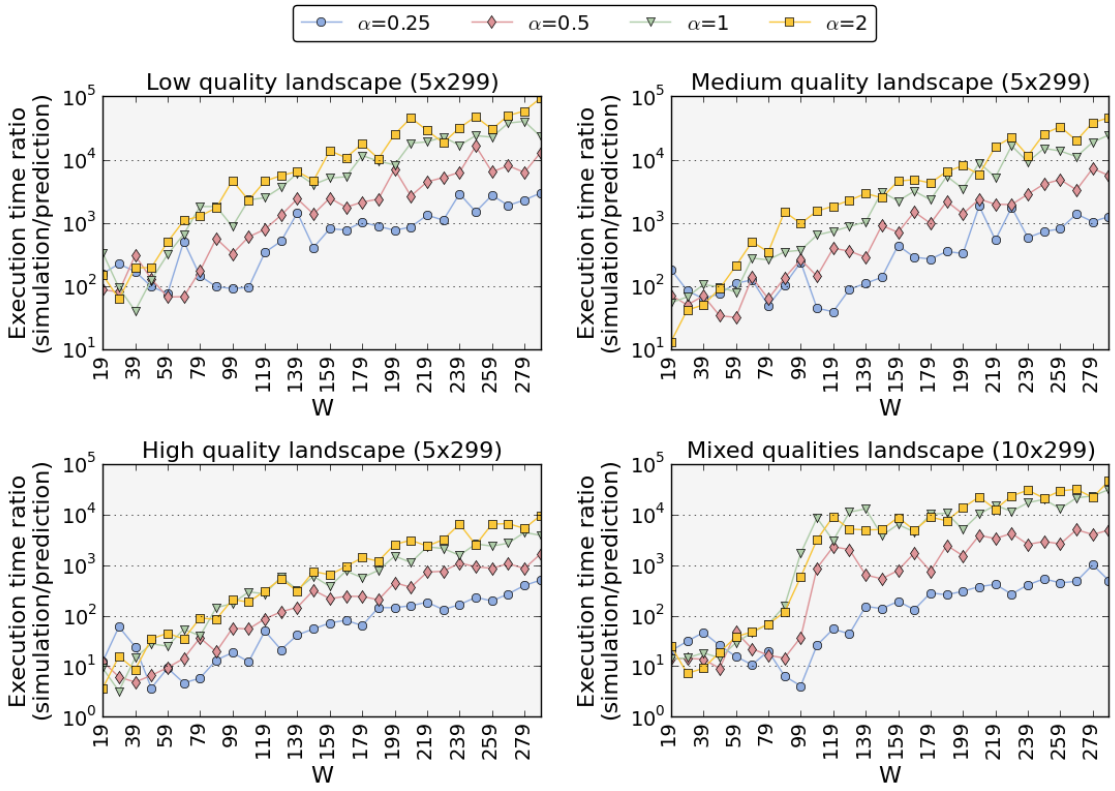


Figure 4.10: The ratio of the execution time (simulation over prediction) with logarithmic scale versus the landscape width in each of the considered landscape (low, medium, high, and mixed) when the dispersal coefficient α equals: 0.25, 0.5, 1, and 2.

saving in computational time and memory is rapidly growing with the dispersal coefficient α and the landscape width.

4.7 Further research

In this chapter, we introduced a new way to estimate the time of invasion process using a powerful computational approach based on conductance and network flow theory. A further goal of our research is to extend the work to consider estimation of invasion time in two-dimensional squared landscapes as it seems that the precision of our prediction formula drops with the increase in the height of the landscape. We will first consider

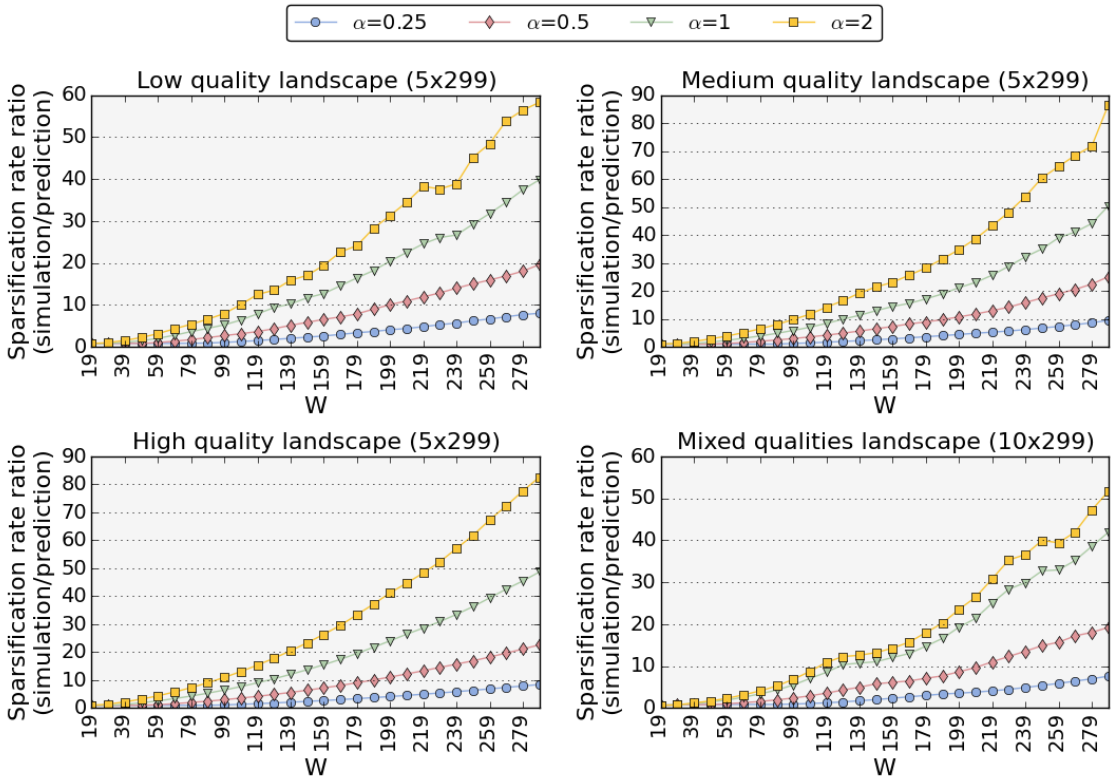


Figure 4.11: The ratio of the sparsification rate (simulation over prediction) versus the landscape width. That done for different values of the dispersal coefficient α: 0.25, 0.5, 1 and 2, and for each landscape quality (low, medium, high, and mixed).

applying the prediction formula (i.e., the method as it is without any modification) directly to two-dimensional square landscapes, then investigate how to modify it to obtain a good accuracy in the prediction of invasion time, if needed.

Chapter 5

Minimising Duration of Invasion Process

In this chapter we show how to manipulate landscapes within some pre-defined budget in order to minimise the duration of the invasion process. We first use the method proposed in Chapter 4, which is based on the network flow theory, to approximate the time taken by species to invade landscapes and reach the targets. Based on this, we propose and test a new method that can help to compute the best locations in real landscapes in order to restore habitat which leads to minimising the expected time taken by species to invade and reach targets. Following that, we compare the new optimisation method with other two baseline methods by running simulations on the new modified landscapes, which are produced by the three methods (i.e., our new optimisation method and the two baseline methods), and study the effect of different landscapes' modifications on the invasion process. We finally show by evaluation that the proposed optimisation method outperforms the competitive baseline methods in terms of proposing landscape modifications that minimise the expected time of the invasion process.

5.1 Min-invasion-time problem

Climate change is a growing threat to species throughout the world and interacts with the major threats to biodiversity, which are habitat loss and fragmentation [20, 96]. It has been observed that species can respond to these threats by (shifting) invading their geographic ranges [25, 112], however in order to do so, they need sufficient habitat in the range where

they exist, in the range where they are going to be as well as any intermediate areas in between to enable species population survival and colonisation [62, 66, 70]. This means that the availability of suitable habitat is very important to protect and conserve species [73, 98]. One of the best ways to protect species against extinction is to prevent their deterioration by restoring habitat. To protect and restore habitat on landscapes, computational decision support tools are needed to find the best locations in landscapes for habitat restoration, given that land and funding for conservation are limited. In a changing climate, the best locations are arguably the places that are most likely to lead to a large decline in the expected time for species to invade landscapes and speed up reaching the target locations.

The field of 'systematic conservation planning' already uses some optimisation techniques, mostly based on integer or linear programming [89], however these have not been applied to the goal of minimising invasion time, partly because this is a more complex, likely non-linear problem. Hodgson et al. in [71] made some progress in developing heuristics to improve landscapes in a stepwise manner, by showing that some network properties (based on circuit theory) can predict the marginal change in invasion time upon adding or removing a network node.

In this chapter, we aim to study how to improve the invasion process, in particular, how to increase the speed of invasion, by modifying landscapes. We consider the scenario where we are given a certain budget that we can use to modify the landscape, e.g., but not restricted to, to enlarge patches that are populated or add new patches to form stepping stones or corridors. We restrict our attention here to increase quality of some selected patches within the given budget. This problem can be seen as an optimisation problem in which we want to determine the best locations to do modifications (i.e., to spend budget) on landscapes, which will lead to minimising the duration of the invasion process.

Based on the developed method in [4] (our new method presented in Chapter 4), *we first propose and test a new method that determines the best locations in the network, hence the landscape, to spend a bounded given budget.* We model and solve this optimisation problem, called min-invasion-time problem, as a convex program. Then, *we compare our optimisation method with two baseline methods by implementing them on real landscapes to produce improved landscapes.* On these improved landscapes, we run simulations to find out how the duration of the invasion process is influenced by the landscapes' modifications.

5.2 The solutions to the min-invasion-time problem

In the min-invasion-time problem, the goal is to find the best locations (i.e., vertices or patches) to increase their qualities by bounded weights, where the overall of the added weights is restricted to a limited budget, such that the invasion time is minimised. We first introduce two baseline optimisation approaches, called *random allocation approach* and *heuristic approach*, that we use to evaluate the quality of our new optimisation approach, called *Convex Programming for the Inverse of Maximum Flow (CP-IMF)*, to the problem.

5.2.1 Random allocation approach

The random allocation approach is a very simple and natural approach in which for a given landscape graph, distinguished source and target patches, we choose a patch v randomly that does not belong to the source and target patches and its quality value $q(v)$ is less than one. Then, we increase the quality value of the chosen patch v by a weight $w(v)$, which is a generated random number between zero and $1 - q(v)$. This scenario is repeated many times until there is no more budget to be spent.

We repeat the whole process four times to produce four different random solutions. For each random solution, we run simulations 100 times, independently, to estimate the expected invasion time in each of these four landscapes. Then we compute the average expected invasion time over the four expected invasion times computed in simulations.

5.2.2 Heuristic approach

The heuristic approach is an optimisation method developed by Hodgson et al. in [71] to manipulate landscapes to improve the expected time of invasion. In their work, they used the electrical circuit theory to approximate the duration of the invasion process. In particular, they approximated the time for the species to reach the target by the overall resistance of a circuit with patches as vertices (equivalent to the vertices of our graph G), colonisation times as links (symmetric and equivalent to our edge *transition probabilities* (Equation (2.1) in Chapter 2)), and fixed potentials at the source (1) and target (-1) nodes. Based on this, their optimisation method is to compute the electrical power of every link (graph edge) as current flow $\cdot$ potential difference, find the link with the highest power and add habitat to the cell located halfway along that link (see [71] for more details). In our implementation, we increase the quality $q(v)$ of the patch v, which is located halfway

along the link with the highest power between two patches by a weight $w(v)$, which is equal to $1 - q(v)$. We repeat this scenario until we spend the whole budget. Both the heuristic approach and the random approach are applied to the entire graph G, rather than to the sub-networks N. Addition of habitat to the source and target sets S and T is not allowed.

5.2.3 Convex Programming for the Inverse of Maximum Flow (CP-IMF)

The CP-IMF approach is the main contribution of this chapter – a new optimisation method based on convex programming. This approach is basically use our new method introduced in Chapter 4, i.e., the invasion time formula (Equation (4.1)). Recall that for a given landscape graph G, we consider $\frac{W}{10} - 2$ sub-landscapes covering the whole landscape such that each one of these sub-landscapes has width of $2R$ with added source and target patches, where R is a sparsification parameter corresponding to the landscape G (recall the computation of parameter R for a given landscape in Equation (3.4), Chapter 3). For each of these sub-landscapes, we define a constructed network N (recall the construction of network N in Chapter 4, Section 4.2.1); the family of all these sub-landscapes and their corresponding networks is denoted by $\mathcal{N}$. Consider a directed network $N \in \mathcal{N}$ with a set of vertices V_N, which consists of all vertices of the sub-landscape, and set of edges E_N. The populated source patches and the unpopulated target patches in network N are denoted by S_N and T_N, respectively. Recall that each vertex $v \in V_N \setminus \{s, t\} \subseteq V$ is associated with a quality $q(v)$. In the min-invasion-time problem, each edge $e = (v, u) \in E_N$ has a nonnegative capacity $c(v, u)$ which represents the maximum flow that can pass through the edge and equal to the *transition probability* $p(v, u)$ (Equation (2.1) in Chapter 2) between the vertices plus an additional weight $w(v)$. Let B denote the budget to be distributed and added to the landscape graph, where the budget is a number given as input such that, $0 < B \leq n - Q$. There are three decision variables. The first decision variable is $w(v)$ per vertex $v \in V$. Each $w(v)$ represents a weight of vertex v to be added to the quality $q(v)$ of vertex v and satisfies the following allocation restrictions:

1. Weight constraint:

$$0 \leq w(v) \leq 1 - q(v), \text{ for each vertex } v \in V.$$

2. Budget and weight constraint:

$$\sum_{v \in V} w(v) \leq B.$$

The second decision variable is a function f_N assigning each edge $e = (v, u) \in E_N$ a value $f_N(e)$ in $[0, 1]$. Each $f_N(e)$ represents a flow from vertex v to vertex u in a network N. Function f_N satisfies the following constraints:

1. Capacity constraints:

$$0 \leq f_N(e) \leq \lambda, \text{ for each edge } e \text{ from } s \text{ to } v \in S_N$$

$$0 \leq f_N(e) \leq \lambda, \text{ for each edge } e \text{ from } v \in T_N \text{ to } t$$

$$0 \leq f_N(e) \leq \big[q(v) + w(v)\big] \cdot \frac{\exp\left(-\alpha d(v,u)\right)}{\left(\frac{2\pi}{\alpha^2}\right) - 1},$$

for each edge e from v to u, where v and $u \in V_N \setminus \{s, t\}$.

Recall that λ is an associated capacity to each directed edge from the virtual source vertex s to each vertex in the initial populated set S_N, and to each directed edge from each vertex in the target set T_N to the additional target vertex t; where λ is the maximum, over all vertices V, of the sum of the weight of adjacent edges, $\lambda = \max\{\lambda_v : v \in V\}$, where $\lambda_v = \sum_{u \in V} p(v, u)$.

2. Flow conservation constraint:

$$\sum_{e \to v} f_N(e) = \sum_{e \leftarrow v} f_N(e), \text{ for each vertex } v \in V_N \setminus \{s, t\}.$$

The third decision variable is M_N per network $N \in \mathcal{N}$. Each M_N represents the value of a maximum flow in a network N and satisfies the following constraint:

$$\sum_{e \leftarrow s} f_N(e) \geq M_N, \text{ for each network } N \in \mathcal{N}.$$

For each network $N \in \mathcal{N}$, C_N denotes a constant, which is the average of the total number of vertices in network N that have non-zero quality and the total number of all vertices in network N.

The min-invasion-time problem is to find the best allocation of a given budget B in a given landscape graph that minimises the total cost (i.e., the expected time of invasion) subject to the budget allocation restrictions as well as the capacity and flow conservation

constraints. It can be written as a convex program:

$$\begin{aligned}
&\text{minimise} \quad \sum_{N\in\mathcal{N}} \frac{C_N}{M_N} \\
&\text{subject to:} \\
&0 \le w(v) \le 1 - q(v), \quad \forall v \in V \\
&\sum_{v\in V} w(v) \le B \\
&0 \le f_N(e) \le \lambda, \quad \forall N \in \mathcal{N} \quad \forall e = (s,v) \in E_N \quad v \in S_N \\
&0 \le f_N(e) \le \lambda, \quad \forall N \in \mathcal{N} \quad \forall e = (v,t) \in E_N \quad v \in T_N \\
&0 \le f_N(e) \le \big[q(v) + w(v)\big] \cdot \frac{\exp\left(-\alpha d(v,u)\right)}{\left(\frac{2\pi}{\alpha^2}\right) - 1}, \quad \forall N \in \mathcal{N} \quad \forall e = (v,u) \in E_N \quad v,u \in V_N \setminus \{s,t\} \\
&\sum_{e\rightarrow v} f_N(e) = \sum_{e\leftarrow v} f_N(e), \quad \forall N \in \mathcal{N} \quad \forall v \in V_N \setminus \{s,t\} \\
&\sum_{e\leftarrow s} f_N(e) \ge M_N, \quad \forall N \in \mathcal{N}.
\end{aligned}$$

We formulate and solve this convex program using IPOPT (Interior Point OPTimizer) package in Python programming language, which is a software package designed to find (local) solutions of nonlinear optimisation problems.

5.3 Evaluating the quality of the CP-IMF method

In this section we describe how we evaluate the quality of our new optimisation method (CP-IMF), landscapes that we used for evaluation, and present the results of evaluation.

5.3.1 Evaluation methodology

In order to evaluate the quality of the CP-IMF method, we first extract two landscapes (see the landscapes later in Section 5.3.2) from the dataset mentioned in Chapter 2. Then, we do the following steps for each extracted landscape (see Figure 5.1):

1. Implement the three optimisation methods (i.e., *random allocation method*, *heuristic method*, and *CP-IMF method*) on the selected landscapes within several values of the budget which are equal to 1-5%$(n - Q)$. The results from implementing each method are new improved/modified landscapes.

2. For each modified landscape, enhanced by the budget and returned by each method/solver,

we do the following. Assume that the source area is the first ten columns and the target area is the last ten columns. Then, run the *full* simulations, which is presented in Chapter 2, 100 times independently to compute the average number of rounds for the invasion process (invasion time) over the 100 independent repetitions.

3. Following that, we compare the computed invasion times in point 2 to evaluate the quality of the CP-IMF method with respect to the two baseline methods.

The above three steps are done for different values of the dispersal coefficient α: 0.25, 0.5, 1 and 2.

5.3.2 The landscapes

In this section, we give details of the landscapes used in this chapter. As mentioned in Chapter 2, the dataset of the 1km resolution raster version of the Land Cover Map 2007 (LCM2007) for Great Britain [91] is used for evaluation purposes. To examine and evaluate the proposed optimisation method, two landscapes from semi-natural grassland map are extracted. Recall that in such an extracted landscape, if the average of all patches' qualities is between 0% and 5%, 5% and 25%, 25% and 100%, then the landscape is of low, medium, and high quality, respectively. We extract a rectangular landscape that consists of 5 rows/height (pixels) and 49 columns/width (pixels) for low and medium quality (see Figure 5.1).

In this part of our work, we aim to minimise the invasion time in low and medium quality landscapes only and we leave high quality landscapes out without improvement because the invasion times are already short in these landscapes so there is not much need for improvement.

It has been assumed that all patches at the first 10 columns of each landscape are occupied by species and the goal is to compute the average number of rounds (i.e., the expected time of invasion) for populating any of patches at the target area, which is the last 10 columns. Since our new optimisation method (CP-IMF) to minimise the invasion time is based on the developed method in [4] (presented in Chapter 4), which defines the invasion time as the sum of estimates of the invasion times over a sequence of small and partly overlapping sub-landscapes, we do the following. We number columns starting from 0, therefore the area between column 0 and 9 is populated (source patches). We consider columns number 19, 29, and 39 as the first column of the target areas, which are of 10

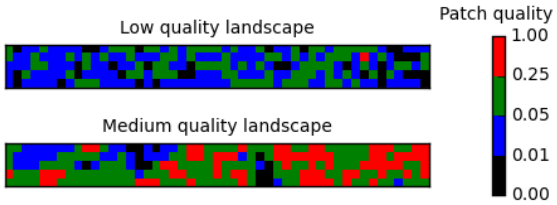


Figure 5.1: Two landscapes of size 5×49 and of low and medium quality extracted from semi-natural grassland map (LCM2007 GB maps). The colour in each landscape corresponds to the quality of each patch; black, blue, green, and red corresponds to zero, low (0.01-0.05), medium (0.05-0.25), and high quality (0.25-1), respectively.

columns length, for the occupied patches at the first 10 columns. Therefore, each landscape in Figure 5.1 has been extracted according to the following criteria. At least one of the patches at each target area (i.e., 19-28, 29-38, 39-48) has non-zero quality.

5.3.3 Evaluation results

Minimising invasion time using three methods

The average number of rounds over 100 independent repetitions (i.e., the estimated time of invasion) decreases as more budget (i.e., additional quality) is added to a landscape (see Figures 5.2 and 5.3). For different amounts of the budget, the CP-IMF method always chooses/finds the best locations (patches) to add budget. Therefore, the estimated time of the invasion in the improved landscapes and produced by the CP-IMF method is the minimum among the three methods. If the budget is added at random patches, the invasion time tends to be reduced gradually. If the budget is added to the patch that is located in the halfway across the highest power link between two patches, the invasion time tends to be reduced gradually as well. On the other hand, the invasion time tends to be reduced significantly to be the minimum over the three methods, if the budget is added by the CP-IMF method.

Figures 5.4 and 5.5 show the improvement in minimising the invasion time by the CP-IMF method, for α equal to 0.25, 0.5, 1 and 2, and in 5×49 low and medium quality landscapes, respectively. The improvement is shown by computing the ratio of the average number of rounds on improved landscapes by the random allocation method to the average number of rounds on improved landscapes by the CP-IMF method. The same ratio is computed

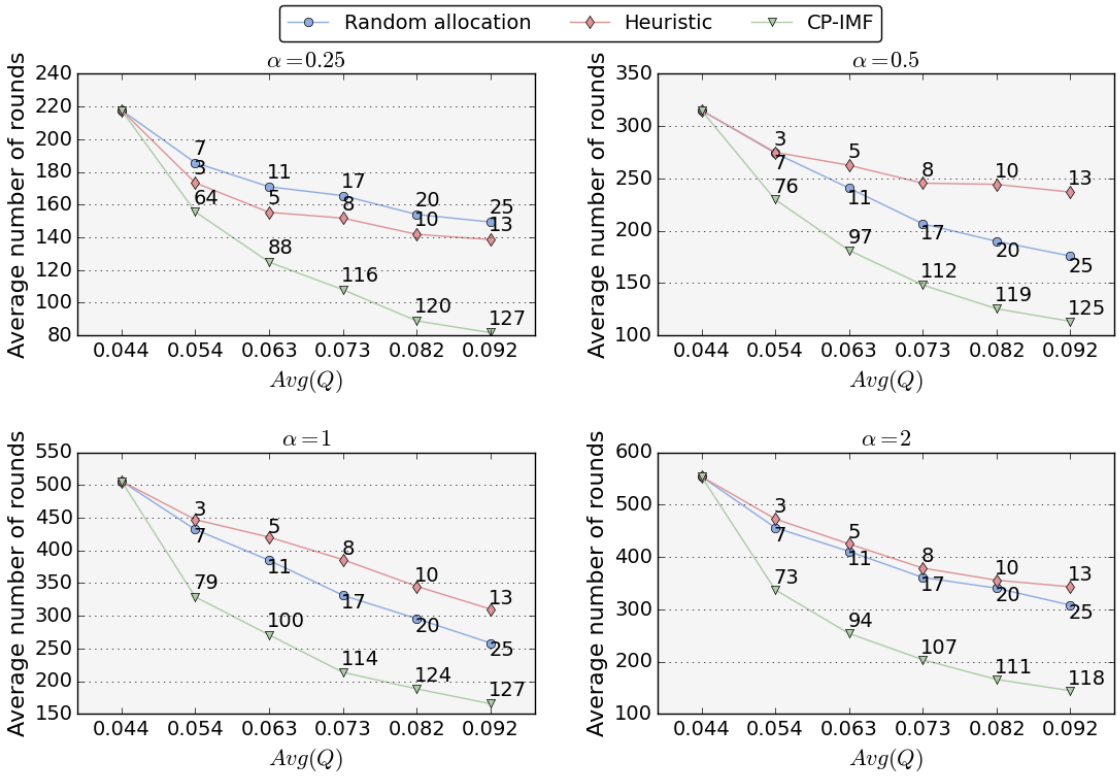


Figure 5.2: The average number of rounds computed by *full* simulations for 5×49 landscapes of **low** quality enhanced by the budget (additional quality) and returned by random allocation, heuristic, and CP-IMF optimisation methods. In the random case, four different random arrangements are used. The x-axis in all figures gives the exact value of the average qualities in the landscape after adding budget. The first mark in each figure represents the average number of round in the original landscape (without modifications). The total number of modified patches is associated with each point for each method. That is done for different values of the dispersal coefficient α: 0.25, 0.5, 1 and 2.

with respect to the heuristic and CP-IMF methods. The obtained improvement using the CP-IMF method is between 1-2.5 and 1-1.5 in low and medium landscapes, respectively. The CP-IMF method outperforms the others especially in the low quality landscape, and for less dispersive species, and these are the situations where conservation intervention is most needed (i.e., where species would be less likely to keep up with climate change under the status quo).

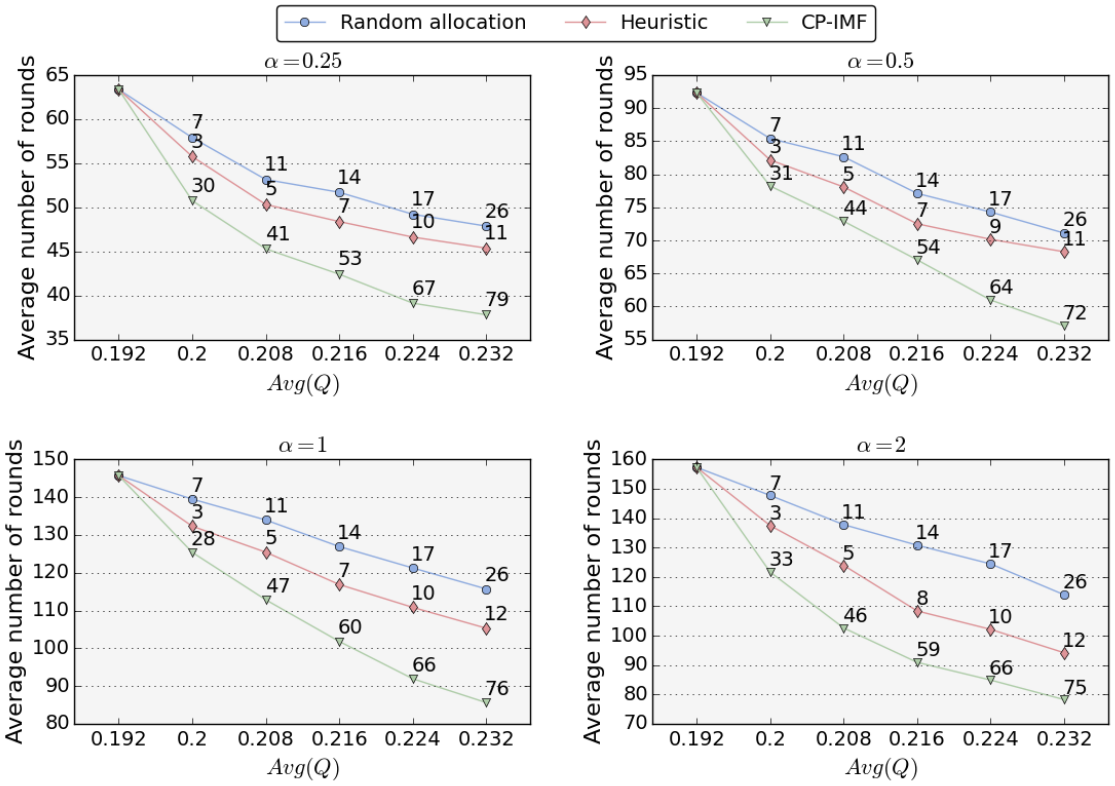


Figure 5.3: The average number of rounds computed by *full* simulations for 5×49 landscapes of **medium** quality enhanced by the budget (additional quality) and returned by random allocation, heuristic, and CP-IMF optimisation methods. In the random case, four different random arrangements are used. The x-axis in all figures gives the exact value of the average qualities in the landscape after adding budget. The first mark in each figure represents the average number of round in the original landscape (without modifications). The total number of modified patches is associated with each point for each method. That is done for different values of the dispersal coefficient α: 0.25, 0.5, 1 and 2.

The spatial allocation of budgets

Allocating the budget for low quality landscape, using CP-IMF and heuristic methods and for α equal to 0.25 and 2, is shown in Figures 5.6 and 5.7 (see the remaining figures, Figures A.1 and A.10, in Appendix A for other values of α and for low and medium quality landscapes). For small values of α (i.e., $\alpha = 0.25, 0.5$), the heuristic method tends to

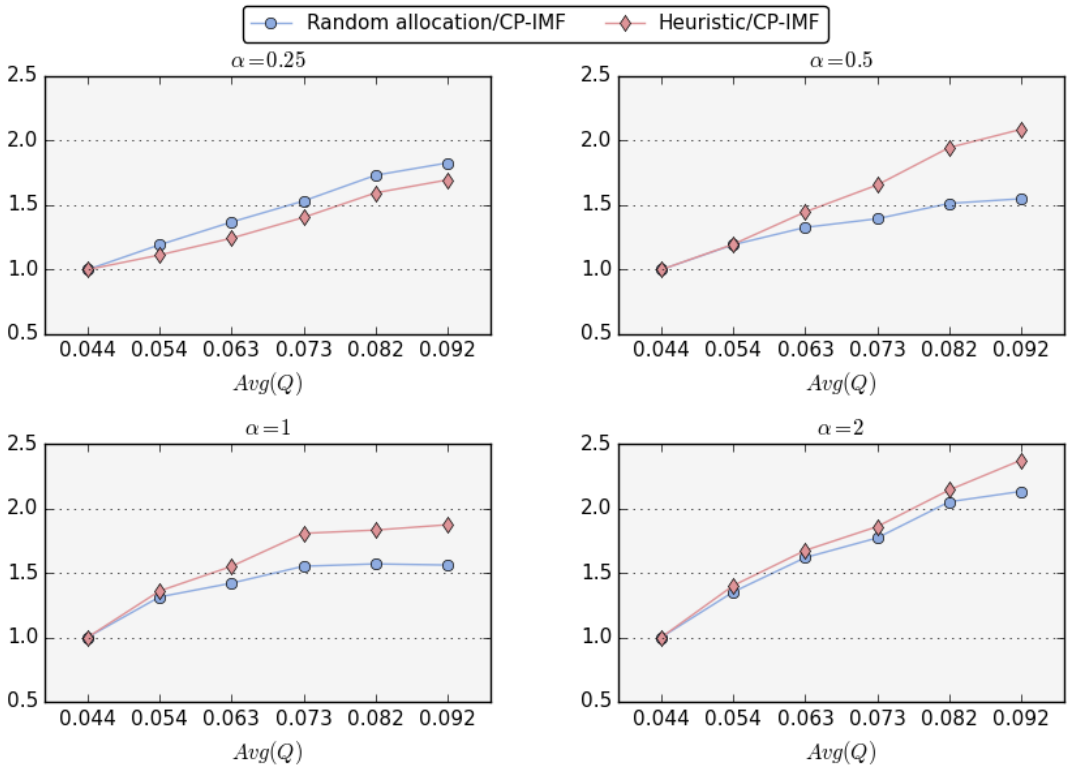


Figure 5.4: Ratio of the average number of rounds on **low** quality landscapes produced by random allocation and heuristic methods to the average number of rounds on **low** quality landscapes produced by the CP-IMF method, for α: 0.25, 0.5, 1 and 2.

allocate the budget close to the source and target patches. While for the large values of α (i.e., $\alpha = 1, 2$), the heuristic method tends to allocate the budget in a way to form corridors or paths from the source patches (the first ten columns) to the target patches (the last ten columns). This could be because low α implies a species that can disperse a long distance and hence 'jump' over unsuitable areas. When α is high, the range expansion will be more dependent on many short-distance colonisation events, and these become more likely in a 'corridor' of uniform quality.

It is very striking that the CP-IMF always allocates its budget in small quotients over many of the landscapes' cells. By contrast, the heuristic method is restricted to improve each chosen cell to $q = 1$ before selecting another cell. We can speculate that the heuristic

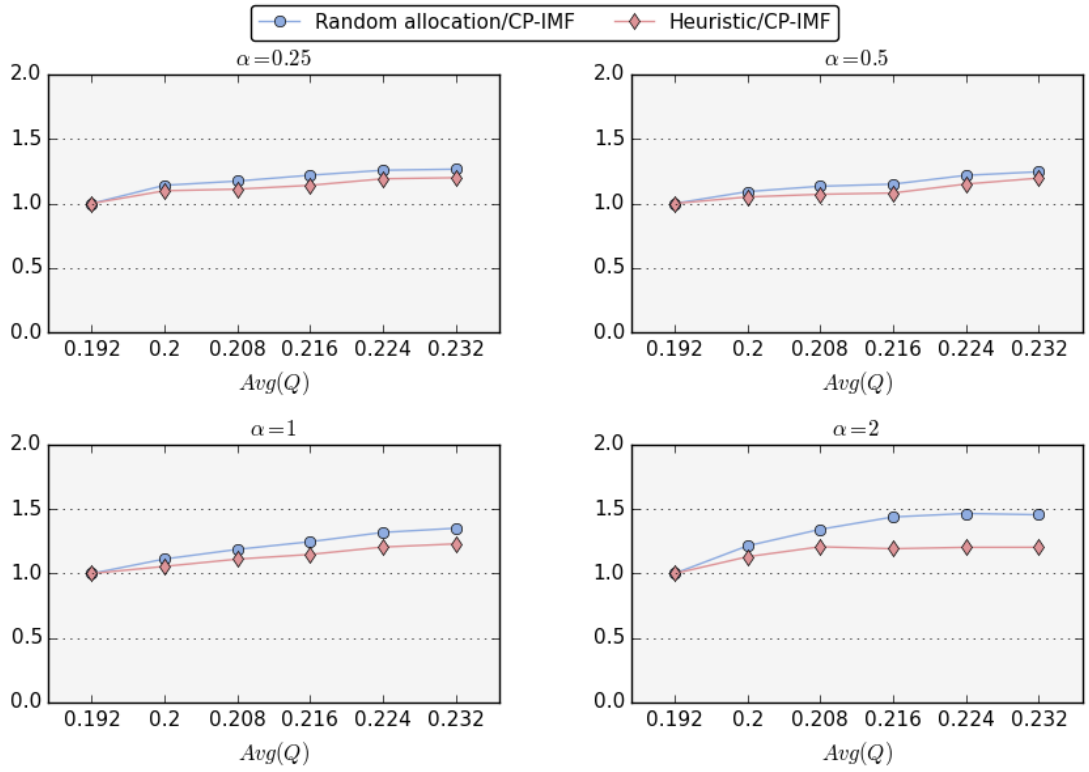
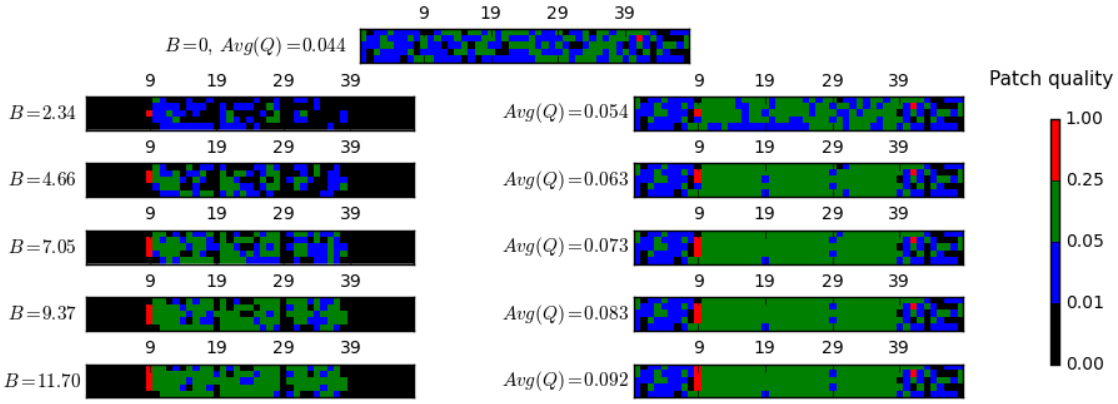


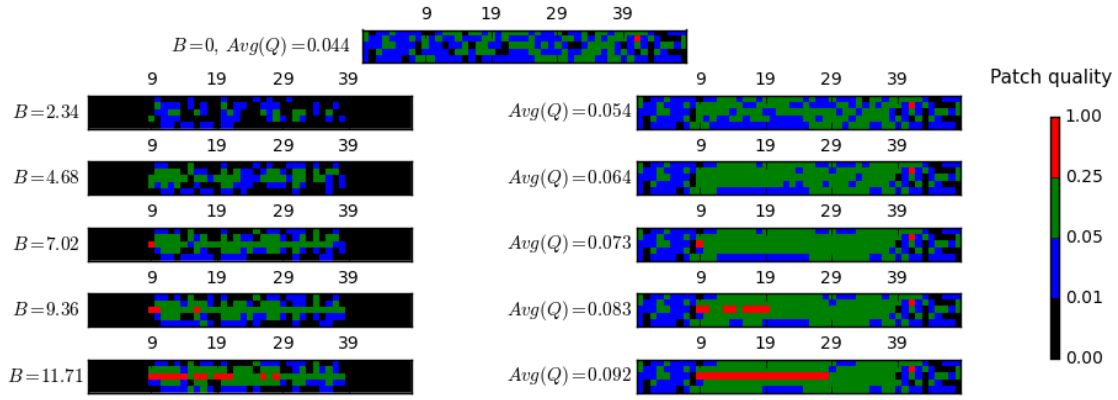
Figure 5.5: Ratio of the average number of rounds on **medium** quality landscapes produced by random allocation and heuristic methods to the average number of rounds on **medium** quality landscapes produced by the CP-IMF method, for α: 0.25, 0.5, 1 and 2.

method might produce better results if this restriction were relaxed. The resultant landscapes produced by the CP-IMF method are often quite homogeneous in quality (Figure 5.6), especially for landscapes of low quality, yet one could still notice some 'islands' or (more or less regular) 'corridors' of higher quality created automatically by the CP-IMF despite of its general tendency of spreading the budget across the whole area. If uniform quality was generally a pattern to maximise invasion speed, this would be a very simple message for conservation in practice, however we suspect that this is not a general rule: for example, Hodgson et al. in [69] found that concentrated corridors of habitat led to faster invasion than the same amount of habitat spread evenly across a square landscape. Since the landscapes used here are relatively narrow strips, it is difficult to tell whether there is some optimal

'width' for a corridor to promote invasion. It will be valuable to test the CP-IMF method in larger landscapes, and because it works by analysing smaller sub-networks, we hope that this can be achieved without too much increase in computation time.

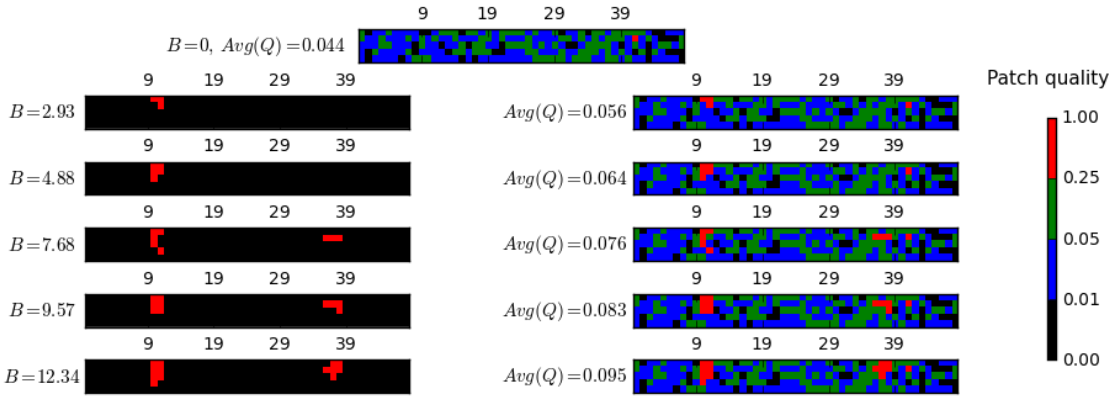


(a) $\alpha = 0.25$

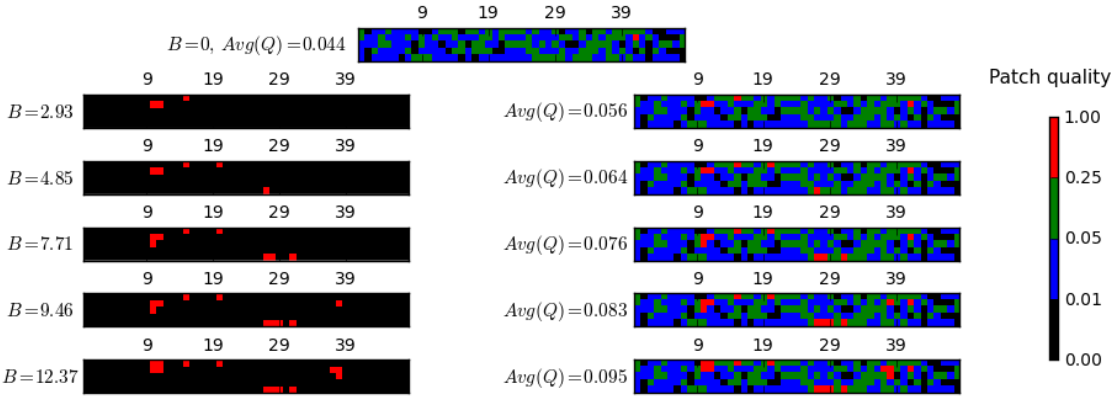


(b) $\alpha = 2$

Figure 5.6: The results of improving 5×49 landscape of **low** quality for $\boldsymbol{\alpha}$ equal to **0.25** (top part) and **2** (bottom part). The map at the top of each part shows the landscape before improvement. The maps in the left column show only the allocation of the budget B by the **CP-IMF** method. The maps in the right column show the landscape after improvement.



(a) $\alpha = 0.25$



(b) $\alpha = 2$

Figure 5.7: The results of improving 5×49 landscape of **low** quality for $\boldsymbol{\alpha}$ equal to **0.25** (top part) and **2** (bottom part). The map at the top of each part shows the landscape before improvement. The maps in the left column show only the allocation of the budget B by the **heuristic** method. The maps in the right column show the landscape after improvement.

5.4 Further research

In this chapter we introduced a new optimisation method (CP-IMF) which chooses the best locations in real landscapes to spend a limited given budget, to minimise the invasion time and therefore to assist decision-making. We believe that it has great potential to be used in practical landscape restoration planning.

As for the future work, we will promote results for larger landscape and presented them in a full version of the paper. In addition, we aim to study corresponding methods that modify landscapes in order to *slow-down* the invasion process.

Chapter 6

Optimal Genetic Code Graph Partition

We describe in this chapter the structure of the standard genetic code (SGC) by a graph, in which codons are vertices and edges correspond to single nucleotide mutations occurring between the codons. We also introduce weights (W) for mutation types to distinguish transversions from transitions (see the definition of mutation types in the next section, Section 6.1). Following that, we develop a clustering algorithm, which searches for the codes characterised by the minimum average conductance calculated from the set W-conductance of codon groups. Recall that the conductance $\Phi \in (0, 1]$ is a measure that is widely used in the graph theory and is defined roughly as the minimum ratio of the edges leaving a set of vertices over the volume of that set, i.e., the total number of edges incident to the set (see Section 6.2 for the precise definitions of W-conductance and average W-conductance). From the biological point of view, the average code W-conductance measure describes the code robustness against amino acid and stop translation signal replacements resulting from single nucleotide substitutions between codons. By applying our developed clustering algorithm, we find three genetic codes that are best with respect to the average code W-conductance for various ranges of the applied weights. We finally compare the structure of the three obtained codes with the structure of the standard genetic code.

6.1 Previous work and our contribution

The questions about the origin and the structure of the standard genetic code (SGC) have puzzled biologists since the first codons assignments were discovered [75, 94]. This nearly universal, with some rare exceptions, set of coding rules is responsible for transmitting genetic information stored in DNA molecules into the protein world. The code uses all possible 64 nucleotide triplets, i.e., codons, to encode 20 canonical amino acids and also the signal for stopping the protein synthesis, i.e., the translation. Since the total number of codons is greater than the number of encoded labels, the SGC must be degenerate, i.e., there must exist an amino acid that is encoded by more than one codon. These redundant codons, called synonymous, are organised in specific groups. In most cases, the codons in such groups differ at the third position, which can be called a degenerate position. This fact suggested to Francis Crick that only the first two codon positions were important in a primordial code [33].

The redundancy of the SGC causes other interesting consequences related to the process of single nucleotide mutations. If these changes occur in the degenerate codon position, then the originally encoded amino acid will not be changed. These mutations are called synonymous or silent, whereas those that change the encoded amino acid or stop translation signal are named nonsynonymous. It should be noted that there are two types of nucleotide changes, transitions and transversions. In the case of transition, a purine nucleotide, i.e., adenine or guanine, mutates to another purine (A↔G), or a pyrimidine nucleotide, i.e., cytosine or thymine, changes into another pyrimidine (C↔T). Transversion are changes in which a purine mutates to a pyrimidine or *vice versa* (A↔C, A↔T, G↔C, G↔T). There are four possible transitions and eight possible transversions. Transitions are more often observed in sequences than transversions [44, 57, 79, 84, 85, 97, 99, 111]. It may result from a higher mutation rate of transitions than transversions in nucleic acids due to physicochemical similarity of the nucleotides. Moreover, transitions are accepted with a greater probability because they rarely lead to amino acid substitutions in encoded proteins due to the specific codon degeneracy. The transitions are also more frequent during protein synthesis [49].

It should be noted, that the synonymous substitutions do not have to be completely neutral mutations, even though they do not change a coded amino acid. The specific codon usage can be associated with co-translational modifications of amino acids, efficiency and accuracy of translation as well as co-translational folding of synthesised proteins [21, 65, 119].

The synonymous codon usage can be also modified as a consequence of selection at the amino acid level [12, 90].

The tendency to minimise the number of nonsynonymous substitutions were noticed in the SGC and this property suggested that the code could have evolved to minimise harmful consequences of mutations and translational errors [7, 8, 35, 41, 48, 49, 50, 51, 52, 56, 58, 59, 61, 115]. The robustness of the code was usually measured as a difference between the polarity values of amino acids encoded by codons before and after a single-point mutation.

Since the genetic code is a set of codons which are related, e.g., by nucleotide mutations, the general structure of this code can be well described by the methodology taken from graph theory [82, 9]. Similarly to [109, 10], we assume that the code encodes 21 items, i.e., 20 amino acids and stop translation signal, and all 64 codons create the set of vertices of the graph, in which the set of edges corresponds to all possible single nucleotide mutations occurring between the codons. This graph is undirected, unweighted and regular. Moreover, according to this representation, each genetic code is a partition of the set of vertices into 21 disjoint subsets. Therefore, the question about the potential genetic code optimality in regard to the mutations can be reformulated into the optimal graph clustering problem.

In the present study, we investigate the properties of the SGC using a more general model including in the graph representation information about transition to transversion ratio, which was not considered by [10, 109]. From a mathematical point of view, we consider a weighted graph, in which all weights are dependent on the type of nucleotide substitutions. We also modify the set conductance measure, which is widely used in the graph theory [82] and has many practical interpretations, for example in the theory of random walks [83] and social networks [19]. In the problem considered here, the conductance of a codon group is the ratio of the weights of nonsynonymous mutations to the weights of all possible single nucleotide mutations, in which the codons in this group are involved. Therefore, this parameter can be used as a measure of robustness against the potential changes in protein-coding sequences generated by the single nucleotide mutations. Basing on the methodology described in [10], we found some solutions, i.e., the genetic code structures, of the optimal graph clustering problem.

6.2 Preliminaries

6.2.1 Model description

To study the general structure of the genetic code we developed its graph representation. Let $G(V, E)$ be a graph in which V is the set of vertices representing all possible 64 codons, whereas E is the set of edges connecting these vertices. All connections fulfil the property that the vertices, i.e., codons $u, v \in V$ are connected by the edge $e(u, v) \in E$ ($u \sim v$) if and only if the codon u differs from the codon v in exactly one position. Moreover, we claim that all transitions are given a weight which equals always to one, while the transversions are given a weight W, where $W \in [0, \infty)$. The larger weight indicates that the transversions are more important than transitions, respectively. The weight can be interpreted as transversion to transition ratio. Hence, the graph G is undirected, weighted and regular with the vertices degree equal to 9. Moreover, from a biological perspective, the set of edges represents all possible single nucleotide substitutions, which occur between codons in a DNA sequence. What is more, this model includes two important types of mutations.

Following the methodology presented in [10], each potential genetic code C, which encodes 20 amino acids and stop translation signal is a partition of the set V into 21 disjoint subsets, i.e., groups of codons, S. Thus, we obtain the following representation of the genetic code C:

$$C = \{S_1, S_2, \cdots, S_{20}, S_{21} : S_i \cap S_j = \emptyset, S_1 \cup S_2 \cup \cdots \cup S_{21} = V\} .$$

In Figure 6.1 we showed an example of the partition of the graph G, which corresponds to the standard genetic code. From a biological point of view, it is interesting to study the code structure according to the types and also the number of connections between and within the codon groups because these connections correspond to nonsynonymous and synonymous substitutions, respectively. It should be noted that each potential genetic code that minimises the number of the nonsynonymous substitutions is regarded the best in terms of decreasing the biological consequences of mutations. Therefore, the conditions under which the partitions of the graph vertices describe the best genetic code, are worth finding.

There are many methods of the optimal graph partitioning, which are based on different approaches. In this work, to investigate the theoretical features of genetic codes in terms of the connections between the codon groups, we decided to use the set conductance measure, which plays a central role in the spectral graph clustering method. The definition of the set

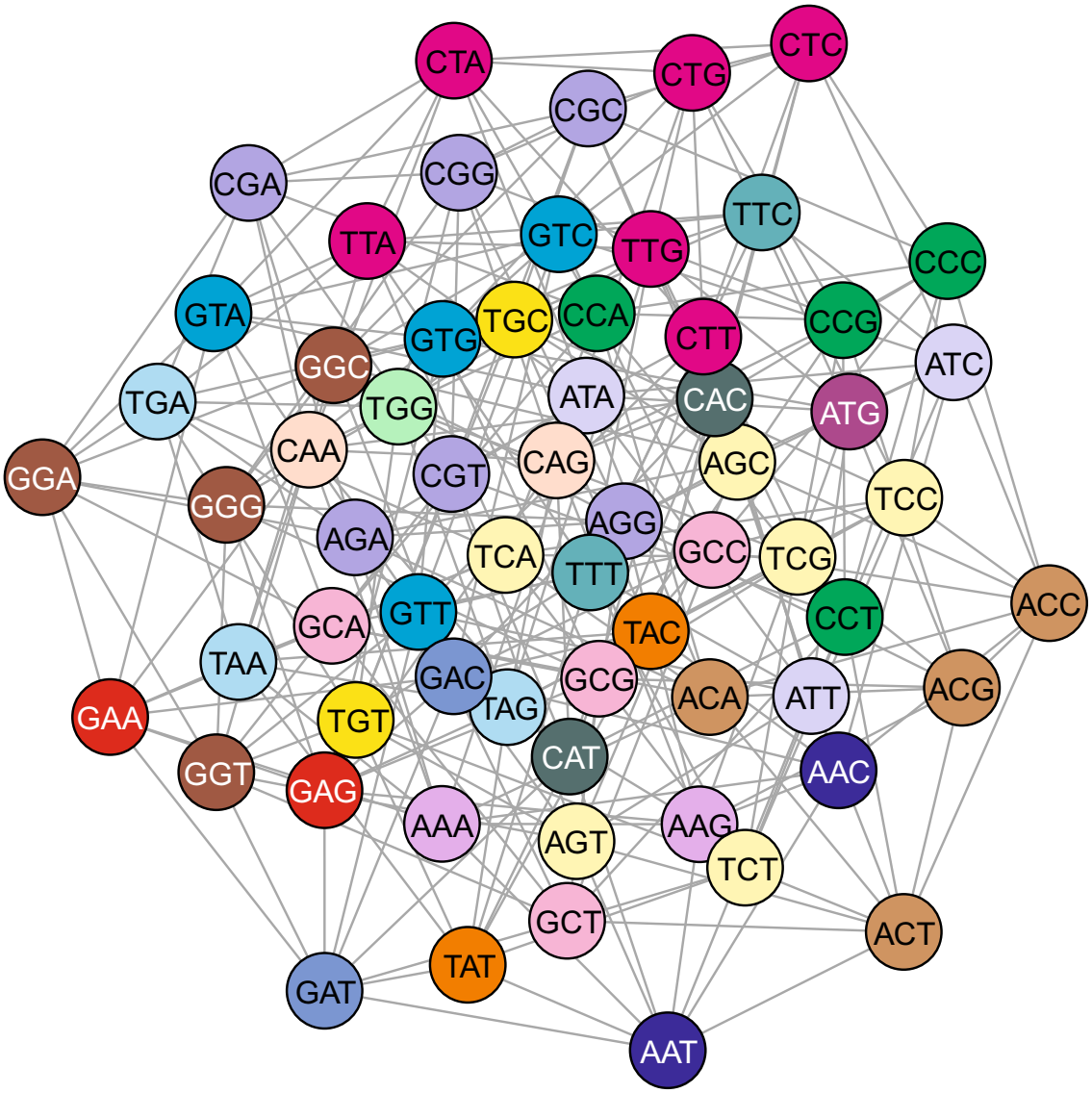


Figure 6.1: The standard genetic code as an example of the partition of the graph $G(V, E)$. Every group of vertices with the same colour corresponds to the respective set of codons, which code for the same amino acid or stop translation signal. The edges represent all possible single nucleotide substitutions. According to [10], modified.

W-conductance measure including weights for edges is as follows:

Definition 1. *For a given weighted graph G let W be a weight of transversion connections in G and S be a subset of V. The W-conductance of S is defined as:*

$$\phi_S(W) = \frac{E_{tr}(S, \overline{S}) + E_{trv}(S, \overline{S}) \cdot W}{|S| \cdot (3 + 6W)} \ ,$$

where $E_{tr}(S, \overline{S})$ is the total number of transition edges crossing from S to its complement $\overline{S}$ whereas $E_{trv}(S, \overline{S})$ is the total number of transversion edges crossing from S to its complement $\overline{S}$, and $|S|$ is the number of vertices belonging to S.

The definition of the set W-conductance is a good staring point to describe a quality measure of a given codon group. Large values of this measure mean that a substantial fraction of substitutions in which these codons are involved are nonsynonymous, i.e., they change one amino acid to another. From the robustness point of view, small values are desirable because in this case many substitutions are neutral (synonymous) and do not change coded amino acids.

What is more, this approach allows us to characterise the properties of the whole genetic code because following the definition of the set W-conductance we define the average W-conductance of a genetic code:

Definition 2. *The average W-conductance of a given genetic code C and a given weight W is defined as:*

$$\overline{\Phi}_C(W) = \frac{1}{21} \sum_{S \in C} \phi_S(W) \ .$$

Using the definition presented above, we are able to describe the best code in terms of the average W-conductance, which is defined as follows:

$$\overline{\Phi}_{min}(W) = \min_C \overline{\Phi}_C(W) \ .$$

$\overline{\Phi}_{min}(W)$ gives us the lower bound of the genetic code robustness measured in terms of the average code W-conductance.

6.2.2 The clustering algorithm

In this work we propose a new randomised clustering algorithm to find the optimal genetic code with respect to the minimum average W-conductance. More formal description of Algorithm 4 provides the structure of the clustering algorithm. The generic structure of the

clustering algorithm contains inputs, outputs (cf. input parameters and output variables in Table 6.1), and three functions, namely: AverageConductance, PickFirstNode, and PickSecondNode. The main function is the AverageConductance function, which aims to find the optimal genetic code with the minimum average W-conductance. The function includes nested loops of two levels. The main loop (lines 5-14) counts the average conductance for each iteration. The second level loop (lines 7-14) is for picking and merging nodes from the graph until we have 21 clusters (super nodes). The AverageConductance terminates when the graph is clustered to 21 clusters for each iteration and returns the best genetic code with the minimum average conductance over all independent iterations. The PickFirstNode (lines 16-30) and the PickSecondNode (lines 31-47) associate a probability for each node in the graph and each function pick a node randomly.

Table 6.1: Input parameters and output variables for Algorithm 4.

Input parameters:
1. Adjacent matrix of 64 codons, called A, where $A[i,j]$ can take: $$A[i,j] = \begin{cases} 1 & \text{if } i \neq j \text{ and if and only if } i \text{ differs from } j \text{ in exactly one position} \\ 0 & \text{otherwise} \end{cases}$$ 2. Adjacent transition matrix of codons, called B, where $B[i,j]$ can take 1 only with transition connections, i.e., A↔G, C↔T, otherwise 0 3. $W \in [0, \infty)$ 4. #iterations $\leftarrow 20,000$
Output variables:
1. The minimum average conductance 2. The structure of the best genetic code that gives the minimum average conductance

Algorithm 4 The clustering algorithm.

function AVERAGECONDUCTANCE($A, B, W, iterations$)
  $D = A - B$ ▷ Create transversion matrix
  $M = B + (W \cdot D)$ ▷ Create matrix M
  min-ave-cond $\leftarrow 2$
  for each iteration **do**
    g = [(node, edges) for each node in M] ▷ List g stores each node i in M and its edges
    while ($len(g) > 21$ nodes) **do** ▷ Pick & merge nodes until we have 21 clusters
      $u \leftarrow$ PICKFIRSTNODE(g)
      $v \leftarrow$ PICKSECONDNODE(g)
      Merge nodes u and v
    conductance = compute conductance for each cluster in g ▷ List conductance stores conductance of 21 clusters using $\phi_S(W)$ formula in Definition 1
    if min-ave-cond>sum(conductance)/len(conductance) **then**
      min-ave-cond = sum(conductance)/len(conductance)
      clusterings-min-ave-cond = g ▷ Stores the structure of the genetic code
  return min-ave-cond, clusterings-min-ave-cond
function PICKFIRSTNODE(g)
  $cond \leftarrow [(i, \phi_i(W))$ for each node i in $g]$ ▷ List to store conductance for each node i in g
  for for each node i in $cond$ **do**
    $weight[i] \leftarrow (i, cond[i]^{20})$ ▷ List to store weight for each node i in $cond$ list
  for for each node i in $weight$ **do**
    $prob[i] \leftarrow (i, \frac{weight[i]}{sum(weight)})$ ▷ List to store prob. of selecting each node i in $weight$ list
  $R \leftarrow$ Generate a random number between 0 and 1
  $j \leftarrow 0$
  $a \leftarrow prob[0]$
  while ($R > a$) **do**
    $j \leftarrow j + 1$
    $a \leftarrow a + prob[j]$
  return j
  $u \leftarrow cond[j]$ ▷ Select the j^{th} node in the $cond$ list
  return u

```
function PickSecondNode(g)
    cond1 ← cond − u                          ▷ Copy cond list without the selected node u
    for for each node i in cond1 do
        edges[i] ← (i, #edges between i and u)
    for for each node i in cond1 do
        weight[i] ← (i, (edges[i] + 1)^10 · cond1[i]^20)   ▷ Compute weight for each node i in cond1 list
    for for each node i in weight do
        prob[i] ← (i, weight[i] / sum(weight))   ▷ List to store prob. of selecting each node i in weight list
    R ← Generate a random number between 0 and 1
    j ← 0
    a ← prob[0]
    while (R > a) do
        j ← j + 1
        a ← a + prob[j]
    return j
    v ← cond1[j]                              ▷ Select the j^th node in the cond1 list
    return v
```

6.3 Our results — towards the optimal genetic code with respect to average conductance

The main goal of our work is to find the optimal genetic codes in terms of the average W-conductance $\overline{\Phi}_{min}(W)$. Furthermore, we compare the properties of the obtained codes with the standard genetic code, which is interesting from the biological point of view.

We run the clustering algorithm (Algorithm 4) 20,000 times independently to find the minimum of the W-average conductance and the structure of the genetic code that gives the minimum average W-conductance. We carried out the calculations for the transversion weight $W \in [0, 10]$. The weights can be interpreted as a relative ratio between transversions and transitions. Smaller weights mean that transitions are more frequent than transversions, while larger weights indicate that the transversions dominate among point mutations.

We found three genetic codes that are best for different ranges of $W \in [0, \infty]$. The genetic code $C1$ was best for every $W \in [0, \frac{37}{70}]$, the code $C2$ for every $W \in [\frac{37}{70}, 1]$ and the code $C3$ for every $W \in [1, \infty]$. The average W-conductance for these codes in the function of weight W is $\overline{\Phi}_{C1}(W) = \frac{1}{21} \cdot \frac{126W+31}{6W+3}$, $\overline{\Phi}_{C2}(W) = \frac{1}{21} \cdot \frac{308W+130}{9(2W+1)}$, and $\overline{\Phi}_{C3}(W) = \frac{1}{21} \cdot \frac{94W+52}{6W+3}$ for every $W \in [0, \frac{37}{70}], [\frac{37}{70}, 1]$, and $[1, \infty]$, respectively. We conjecture that these codes are

optimal in the range of weights corresponding to the observed transversion/transition ratio in natural mutational pressures; though, as the algorithm is randomised (but repeated a large number of times to reduce the probability of finding sub-optimal solutions), the formal proof of their optimality is still an open question. Figure 6.2 shows the average W-conductance for the best codes and the SGC depending on the transversion weight.

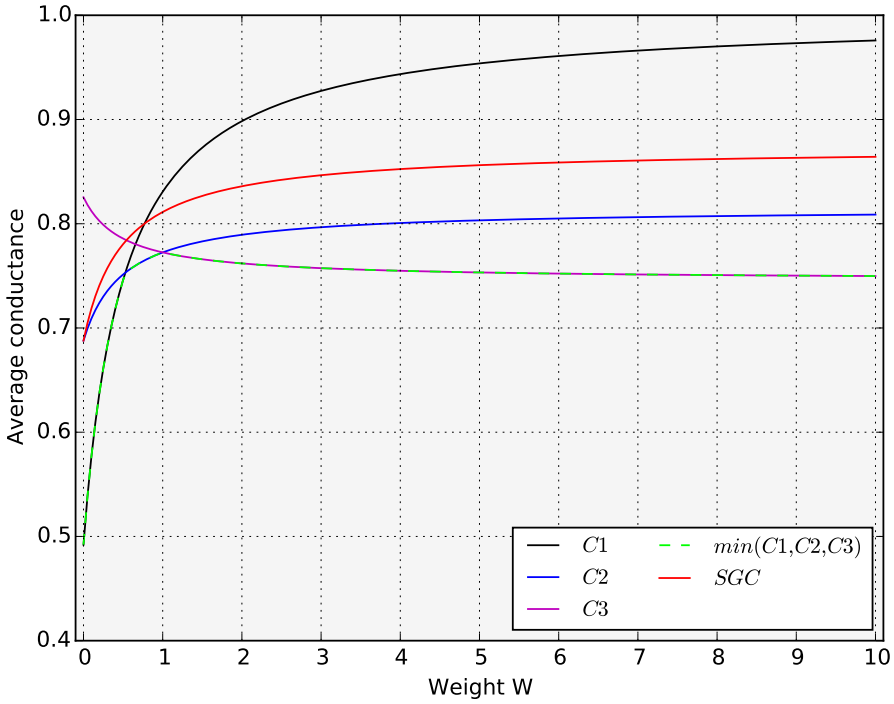


Figure 6.2: The average conductance for the best codes $C1, C2, C3$, and $\min(C1, C2, C3)$ as well as the standard genetic code (SGC) for the weights of transversions $W \in [0, 10]$.

The average conductance for the codes that are best for $W < 1$ increases rapidly with W and then stabilises for large values. In the case of the code $C3$, its conductance decreases at first and then also approaches a certain value. For small W values, the average conductance is the smallest for the code $C1$ and the largest for code $C3$. In turn, the opposite is true for large W values. The code $C1$ is characterised by the biggest difference between its average conductance values. The code is very well optimised for the excess of transitions over transversions but it is very bad in the opposite case. The average W-conductance of the SGC shows the general course similar to that of the $C1$ and $C2$ codes.

To compare the properties of the best codes with the standard genetic code, we computed the function of the average W-conductance for the SGC, $\overline{\Phi}_{SGC}(W) = \frac{1}{21} \cdot \frac{10(33W+13)}{9(2W+1)}$. Then, we subtracted $\overline{\Phi}_{SGC}(W)$ from each of the average W-conductance function of each best codes $(C1, C2, C3)$ and calculated the derivative for each produced function as follows:

1. Define $f1(W) = \overline{\Phi}_{SGC}(W) - \overline{\Phi}_{C1}(W)$, then

$$f1(W) = \frac{1}{21} \cdot \frac{10(33W+13)}{9(2W+1)} - \frac{1}{21} \cdot \frac{126W+31}{6W+3} \ .$$

   The derivative of $f1(W)$ is

$$f1'(W) = -\frac{122}{189(2W+1)^2} \ .$$

   At $W = 0$, the value of $f1$ is equal to 0.2 and at $W = \frac{37}{70}$ $f1$ is equal to 0.03. The values of the conductance function for both codes are the same at $W = \frac{37}{48}$. Below this weight the $C1$ code shows a smaller $\overline{\Phi}$ than the SGC and above this weight, the opposite is true.

2. Define $f2(W) = \overline{\Phi}_{SGC}(W) - \overline{\Phi}_{C2}(W)$, then

$$f2(W) = \frac{1}{21} \cdot \frac{10(33W+13)}{9(2W+1)} - \frac{1}{21} \cdot \frac{308W+130}{9(2W+1)} \ .$$

   The derivative of $f2(W)$ is

$$f2'(W) = \frac{22}{189(2W+1)^2} \ .$$

   At $W = 0$, the $C2$ and the SGC codes have the same values of $\overline{\Phi}$, i.e., $f2 = 0$. Next, with the growth of W, $f2$ increases, which means that the average W-conductance of the SGC becomes larger than that of the $C2$ code. At $W = \frac{37}{70}$, the value of $f2$ is equal to 0.03 and at $W = 1$ $f2$ is equal to 0.04.

3. Define $f3(W) = \overline{\Phi}_{SGC}(W) - \overline{\Phi}_{C3}(W)$, then

$$f3(W) = \frac{1}{21} \cdot \frac{10(33W+13)}{9(2W+1)} - \frac{1}{21} \cdot \frac{94W+52}{6W+3} \ .$$

The derivative of $f3(W)$ is

$$f3'(W) = \frac{100}{189(2W+1)^2} \ .$$

For $W < \frac{13}{24}$, $\overline{\Phi}_{SGC}$ is smaller than $\overline{\Phi}_{C3}$. Above this weight, the opposite is true. At $W = 1$, the value of $f3$ is equal to 0.04 and at $W = 3$ $f3$ is equal to 0.09.

The functions $f1(W)$, $f2(W)$ and $f3(W)$ are differences in the average W-conductance between the SGC and three best codes depending on the transition weight W. It is evident that the standard genetic code is optimised for more frequent transitions than transversions. The SGC can obtain the same average W-conductance as each of the optimised codes but for different transversion weights. Nevertheless, the W is always smaller than 1 in these cases. The minimum distance between the SGC and the best genetic codes in terms of the average conductance is 0.03 for $W = \frac{37}{70} = 0.53$ (Figure 6.3). Since there are twice as many possible transversions as transitions, the expected ratio should be 2, if all nucleotide substitutions happen with the same probability. Interestingly, the weights for which $\overline{\Phi}_{SGC}$ is close to $\overline{\Phi}$ of the best codes is in the range of the transversion/transition ratio observed in genomic mutational pressures, i.e., from 1.44 to 0.10 [14, 76]. However, for each transversion weight, it is possible to find a code better optimised than the SGC in terms of the average W-conductance, so this code is not perfectly optimised. Our preliminary analyses of the alternative genetic codes in this respect showed that the relationship between their average conductance depending on the transversion weight has the course very similar to that of the SGC.

Examples of the structure of these best genetic codes are presented in Table 6.2. Although the code $C1$ and $C3$ are best for different and extreme W values, they have the same number of two- and four-codon groups, 10 and 11, respectively. The code $C2$ has in addition 4 groups consisting of three codons as well as 8 two-codon groups and 9 four-codon groups. The SGC is more diversified in this respect because it has 2 one-codon groups, 9 two-codon groups, 2 three-codon groups, 5 four-codon groups and 3 six-codon groups. Thereby, it is more similar to the code $C2$.

The code $C1$ is best for smaller weights of transversions. Therefore, such mutations are preferably involved in changes between codon groups of this code in order to minimise these changes. Consequently, all synonymous substitutions in this code are transitions. In the case of the code $C3$, which is best for larger W, transversions were eliminated from

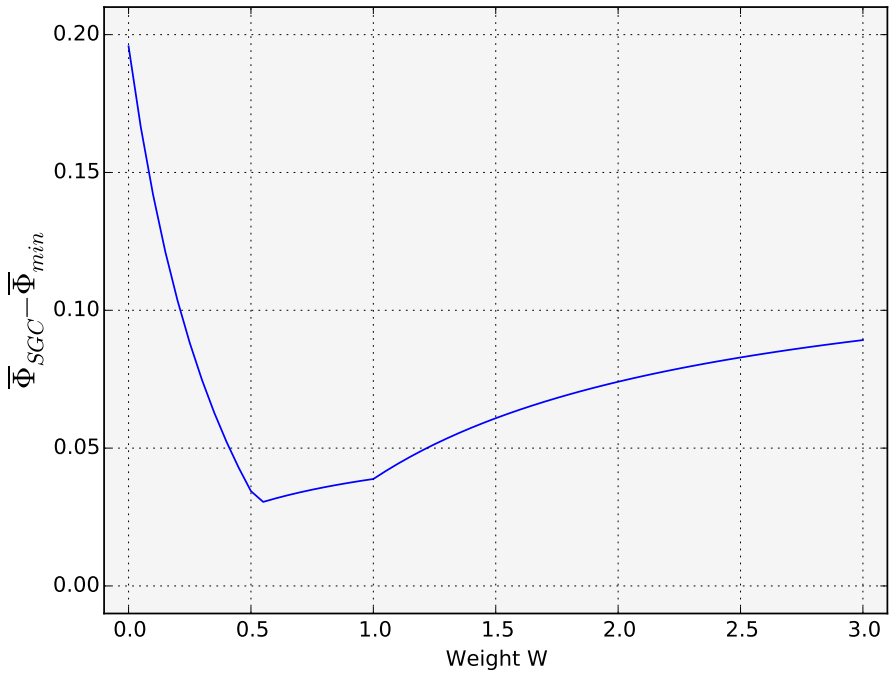


Figure 6.3: The difference between the average conductance for the standard genetic code (SGC) ($\overline{\Phi}_{SGC}$) and the best codes ($\overline{\Phi}_{min}$) for the weights of transversions $W \in [0, 3]$.

changes between codon groups as much as possible to increase the number of transitions. In consequence, all changes within two-codon groups of this code are transversions. Since there are only two purines and two pyrimidines, it is not possible to create four-codon groups that can change to each other by only transversions. Therefore, changes within such groups are both transitions and transversions. The code $C2$ is a mixture in this respect because the codons in its two-codon groups can change to each other only by transitions, while in the other groups by the two types of mutations. Considering only one point mutations in the SGC, all changes within two-codon groups are also transitions and within other groups both transitions and transversions with exception to the stop codon group, which also involves only transitions. Then the SGC is again more similar to the code $C2$ in this respect.

The changes between codons in one group of the code $C3$ can occur only in one fixed codon position, the first or the second one. The third codon position can also be mutated in the code $C2$. However, the code $C1$ contains also the groups in which any two codon positions can be changed. The SGC contains many codon groups with synonymous mutations

Table 6.2: The structure of the best genetic codes $C1, C2$, and $C3$ for $W \in [0, \frac{37}{70}]$, $W \in [\frac{37}{70}, 1]$, and $W \in [1, \infty]$, respectively. Each row describes the codon group for a cluster.

	$C1$	$C2$	$C3$
1	$\{AAA, AAG, AGG, AGA\}$	$\{AAG, ACG, ATG\}$	$\{ATA, AAA, AGA, ACA\}$
2	$\{ATT, ATC\}$	$\{AGA, AGG\}$	$\{AGC, ACC\}$
3	$\{TAG, TAA\}$	$\{AGC, ATC, AAC, ACC\}$	$\{ACT, TCT\}$
4	$\{TAC, TAT, TGT, TGC\}$	$\{ACA, AAA, ATA\}$	$\{ACG, AAG, ATG, AGG\}$
5	$\{TTT, CTT\}$	$\{TAA, TAG\}$	$\{TTA, TAA\}$
6	$\{TGA, TGG\}$	$\{TAC, CAC\}$	$\{TTC, TAC, TCC, TGC\}$
7	$\{TCT, CCT\}$	$\{TTA, CTA, GTA\}$	$\{TGG, TAG, TCG, TTG\}$
8	$\{TCG, TTG, TTA, TCA\}$	$\{TTG, CTG\}$	$\{TCA, TGA\}$
9	$\{TCC, TTC, CTC, CCC\}$	$\{TGA, TGG, TGC, TGT\}$	$\{GAT, AAT, TAT, CAT\}$
10	$\{GTA, ATA, GTG, ATG\}$	$\{TCA, TCT, TCC, TCG\}$	$\{GTT, ATT, TTT, CTT\}$
11	$\{GTC, GTT\}$	$\{GAT, AAT, TAT, CAT\}$	$\{GGA, GCA, GAA, GTA\}$
12	$\{GGA, GAA, GAG, GGG\}$	$\{GAC, GGC, GCC, GTC\}$	$\{GGT, CGT, AGT, TGT\}$
13	$\{GGT, GAT, AGT, AAT\}$	$\{GTG, GAG, GGG, GCG\}$	$\{GCG, GAG, GGG, GTG\}$
14	$\{GGC, AGC, AAC, GAC\}$	$\{GGT, AGT\}$	$\{GCC, GTC, GAC, GGC\}$
15	$\{GCG, GCA, ACA, ACG\}$	$\{GCA, GGA, GAA\}$	$\{CAA, CTA\}$
16	$\{GCC, ACC, GCT, ACT\}$	$\{GCT, ACT\}$	$\{CAC, AAC\}$
17	$\{CAT, CGT, CGC, CAC\}$	$\{CAG, CAA\}$	$\{CTG, CGG, CAG, CCG\}$
18	$\{CTA, CTG\}$	$\{CTT, GTT, ATT, TTT\}$	$\{CTC, ATC\}$
19	$\{CGA, CAA\}$	$\{CTC, TTC\}$	$\{CGC, CCC\}$
20	$\{CGG, CAG\}$	$\{CGC, CGG, CGT, CGA\}$	$\{CCA, CGA\}$
21	$\{CCG, CCA\}$	$\{CCA, CCC, CCT, CCG\}$	$\{CCT, GCT\}$

in the third codon position but there are also three codon groups involving single changes in two codon positions.

The comparison of structures of the genetic codes show that the assignments of amino acids to codons is not ideally optimised in the SGC. Some similarity of the SGC to the code $C2$ suggests that the standard genetic code could evolve under the transversion/transition for which the code $C2$ is best.

6.4 Further research

In this Chapter, we introduce a new clustering algorithm which found some solutions, i.e., the genetic code structures, of the optimal graph clustering problem. A further goal of our research is to formally prove the optimality of the found codes. Further study may involve extending the code by new purines (i.e., extending the alphabet by new letters and the semantic of the code by new amino acids).

Chapter 7

Conclusions

This thesis has presented a variety of solutions to several problems that can all be categorised in the area of Computer Science — Computational Biology and Modelling, and more specifically in the field of Algorithmic Graph Theory and its application to bioinformatics.

The results presented in the preceding Chapters are the result of published work carried out by the author, their supervisors as well as other collaborators from different institutions, with the exception of Chapter 5 which is under review at the time of the write-up of this thesis. Below is a more detailed look at the conclusions that can be obtained from the main results presented in this thesis.

7.1 Modelling and estimating invasion time using graph sparsification approach

The study in Chapter 3 was prompted by a desire to construct the *R-local* model that visualises the invasion process based on the landscape network sparsification tool to efficiently estimate a duration of the invasion process. The capability of our new model is to reduce the time needed to compute the estimated duration of the invasion process on large landscapes while maintaining a comparable duration of the invasion.

We have shown by simulations how the local distance R depends on two factors: the dispersal coefficient α and the landscape quality. A small dispersal coefficient α requires a large local distance R in all types of quality, while a small R is sufficient for a large α. The difference in landscape quality does not cause a significant difference in the needed R. Indeed, the tool of landscape network sparsification illustrates its efficiency in computing the

invasion time especially for large landscapes. Even when the size of landscape is increased, the local distance R does not grow significantly (see Figure 3.8). This implies a relatively sparser landscape networks for larger landscapes, and therefore the time needed to compute the invasion duration decreases substantially.

7.2 Modelling and estimating invasion time using network flow theory

Chapter 4 introduces a new theoretical measure γ that estimates the expected time performance for asynchronous and synchronous executions of the invasion protocol. The γ measure can be upper bounded by $\frac{\ln n}{\Phi(N)}$, which is a parameter that can be fastly approximated for small landscapes by using network flow algorithm. We extend this theoretical result into prediction formula IT, in which we interpolate constants by simulations. The capability of our model is to approximate accurately the number of rounds needed to invade large homogeneous landscapes in a short computational time; this has been clearly achieved as demonstrated in Figures 4.6 and 4.9 (accuracy) and in Figures 4.10 and 4.11 (saved computation time and memory).

We show, by simulations, that the dispersal coefficient α and the landscape quality affect the invasion time and the computation time needed for it. A large α requires a long time to compute the invasion time, in all types of quality, while a short time is sufficient for a small α. Landscapes of low quality require the longest time to compute the invasion time among the three formed qualities (low, medium, high).

7.3 Manipulating/Improving landscapes to minimise invasion time

The study in Chapter 5 was prompted by a desire to propose and test a new method that chooses the best locations in real landscapes to spend a limited given budget, to increase the speed of invasion and therefore to assist decision-making. Our new method is based on the developed method by Aloqalaa et al. in [4] (introduced in Chapter 4), which approximates the invasion time using the network flow approach. We compare our new optimisation method with two baseline methods by implementing them on real heterogeneous landscapes of Great Britain to produce improved landscapes. Then, we show, by running simulations

on the obtained and improved landscapes, the capability of our new method to propose landscapes' modifications, which lead to maximise invasion speed. We believe that it has great potential to be used in practical landscape restoration planning.

7.4 The impact of the transversion/transition ratio on the optimal genetic code graph partition

Our results in Chapter 6 show that the general structure of the genetic code and the problem of the genetic code optimality can be successfully reformulated using a methodology adapted from graph theory in the context of optimal clustering of a specific graph. To evaluate the quality of the genetic code, we calculated the average code W-conductance including weights for mutation types. Thereby, we distinguished transitions and transversions. From the biological point of view, this measure describes the code robustness against amino acid and stop translation signal replacements resulting from single nucleotide substitutions between codons.

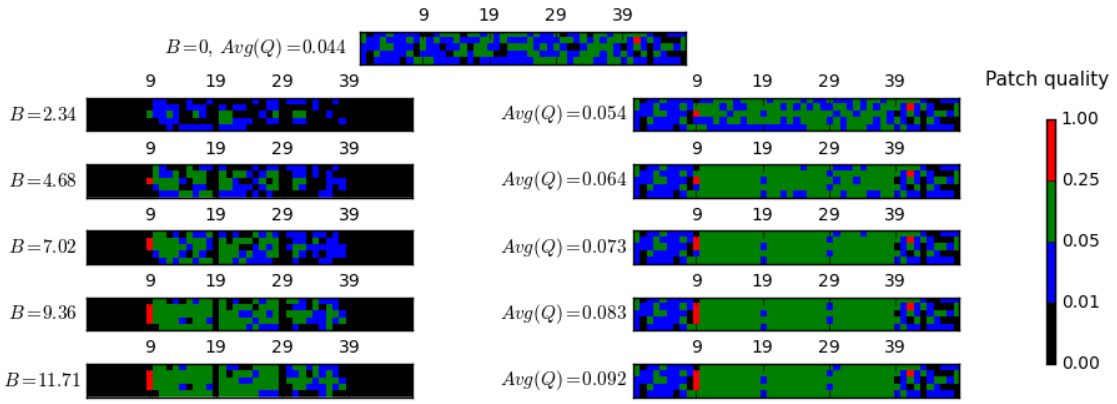
We found three best codes with respect to the average code conductance for various ranges of the applied weight. The structure of the codes was different in comparison to the standard genetic code. The W-conductance of the SGC was the most similar to that of the best codes in the range of weights corresponding to the observed small transversion/transition ratio in the mutational pressure. Other researches also showed that the SGC performs better for the excess of transitions over transversions [49, 50]. It indicates that the SGC is optimised to some extent in terms of the minimisation of amino acid and stop translation replacements. However, the optimisation was not ideal and for each weight better theoretical codes could be found. In agreement with that, other investigations also showed that the SGC is not perfectly robust against point mutations or mistranslations [15, 16, 87, 95, 100, 101, 114].

Most likely, the robustness against mutations was not the main force that drove the evolution of the genetic code and amino acids were assigned to codons according to expansion of biosynthetic pathways synthesising amino acids [36, 37, 38, 39, 40, 116, 118, 117]. In this case, the potential minimisation of mutation errors could have occurred by the direct optimisation of the mutational pressure around the established genetic code [11, 13, 14, 45, 86].

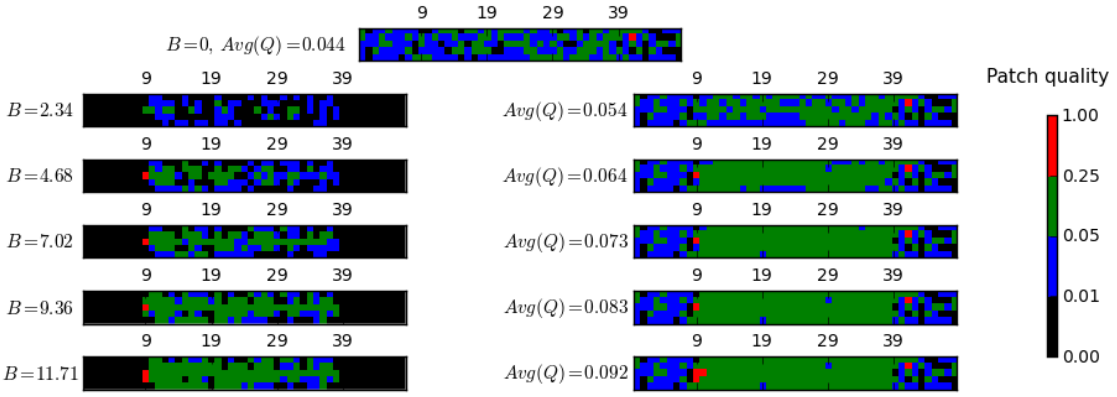
Appendix A

Appendix – Additional Results of the Spatial Allocation of Budgets on Landscapes

This Chapter contains results of the spatial allocation of budgets on 5×49 landscapes of low and medium qualities. Allocating the budget, using three allocation optimisation methods, CP-IMF, heuristic and random allocation, and for α equal to 0.25, 0.5, 1 and 2, is shown in Figures S1-S4 and Figures S5-S10 for low and medium landscapes, respectively.

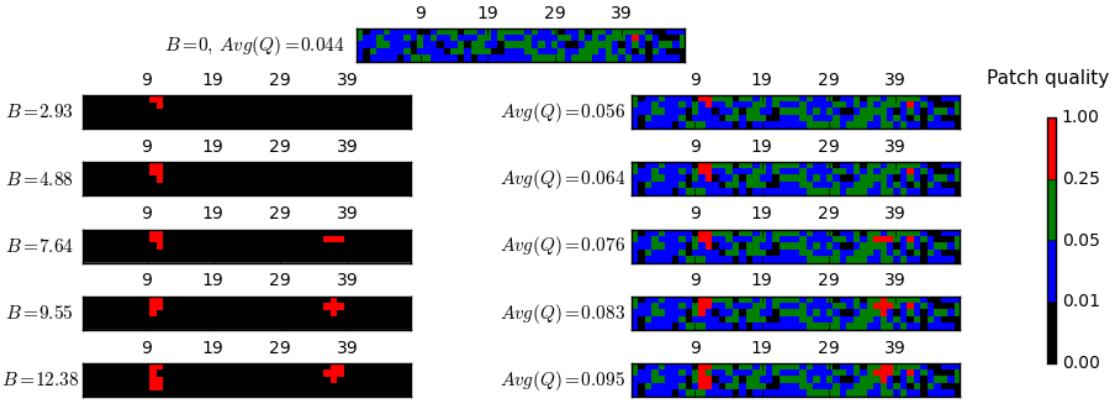


(a) $\alpha = 0.5$

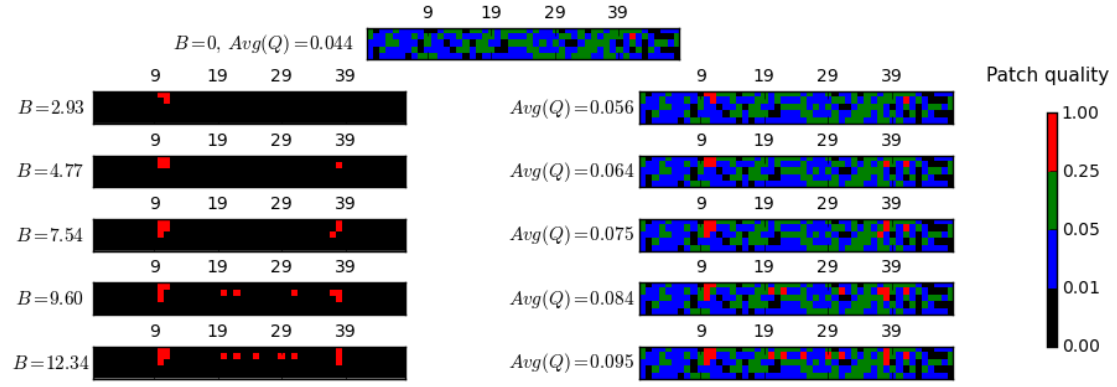


(b) $\alpha = 1$

Figure A.1: The results of improving 5×49 landscape of **low** quality for $\boldsymbol{\alpha}$ equal to **0.5** (top part) and **1** (bottom part). The map at the top of each part shows the landscape before improvement. The maps in the left column show only the allocation of the budget B by the **CP-IMF** method. The maps in the right column show the landscape after improvement.

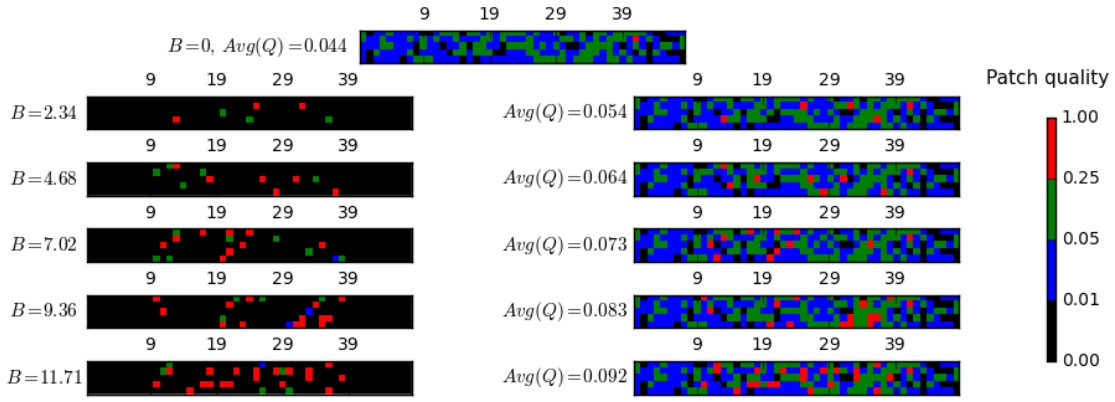


(a) $\alpha = 0.5$

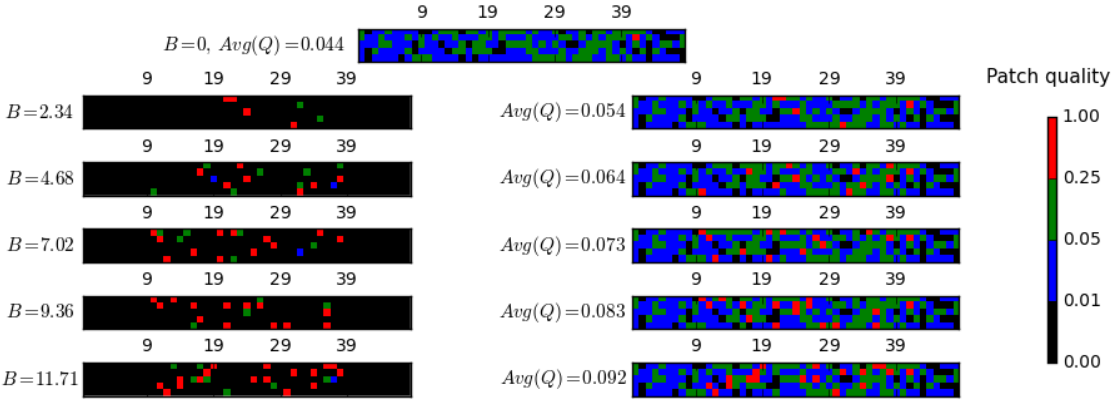


(b) $\alpha = 1$

Figure A.2: The results of improving 5×49 landscape of **low** quality for $\boldsymbol{\alpha}$ equal to **0.5** (top part) and **1** (bottom part). The map at the top of each part shows the landscape before improvement. The maps in the left column show only the allocation of the budget B by the **heuristic** method. The maps in the right column show the landscape after improvement.

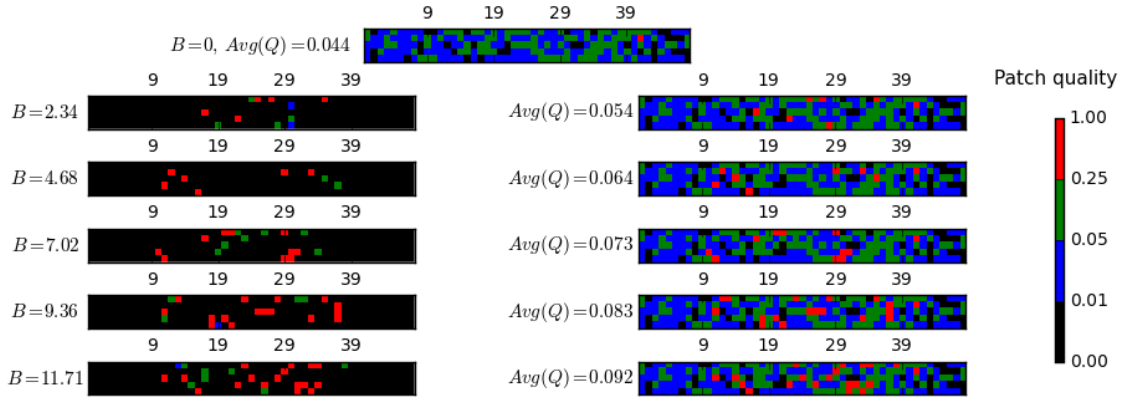


(a) $\alpha = 0.25$

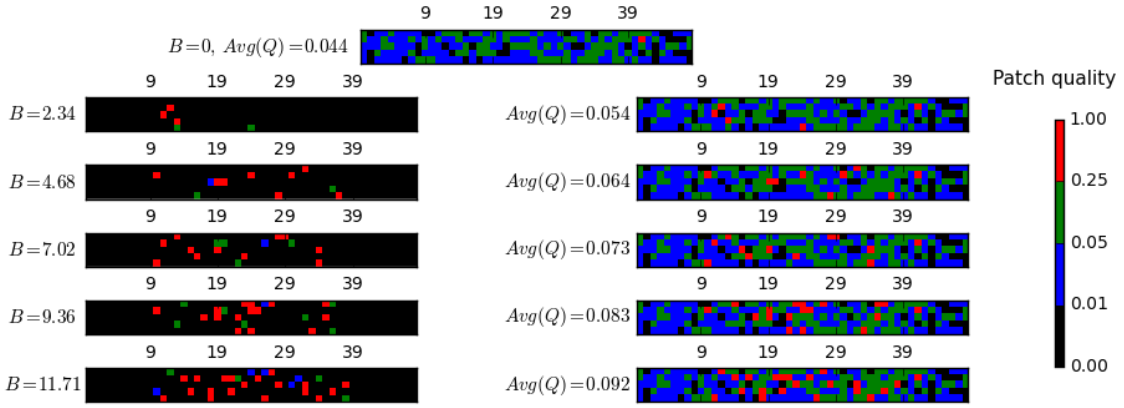


(b) $\alpha = 0.5$

Figure A.3: The results of improving 5×49 landscape of **low** quality for $\boldsymbol{\alpha}$ equal to **0.25** (top part) and **0.5** (bottom part). The map at the top of each part shows the landscape before improvement. The maps in the left column show only the allocation of the budget B by the **random** method. The maps in the right column show the landscape after improvement.

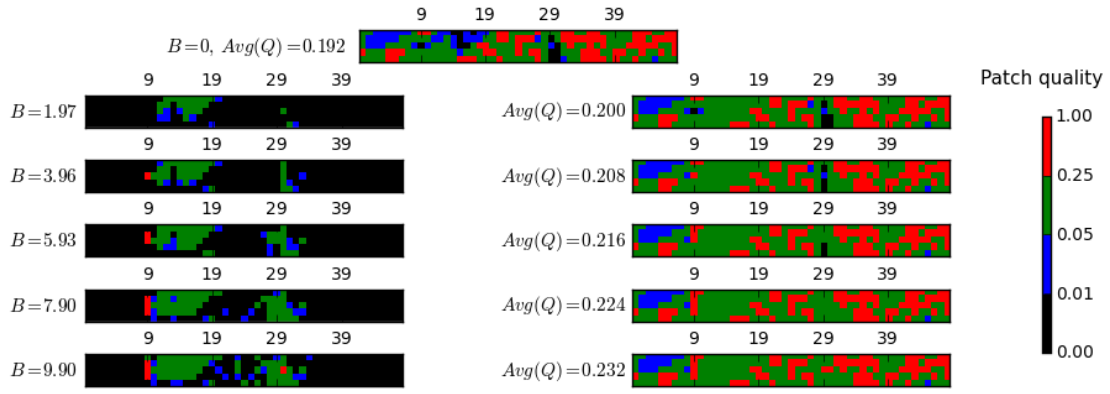


(a) $\alpha = 1$

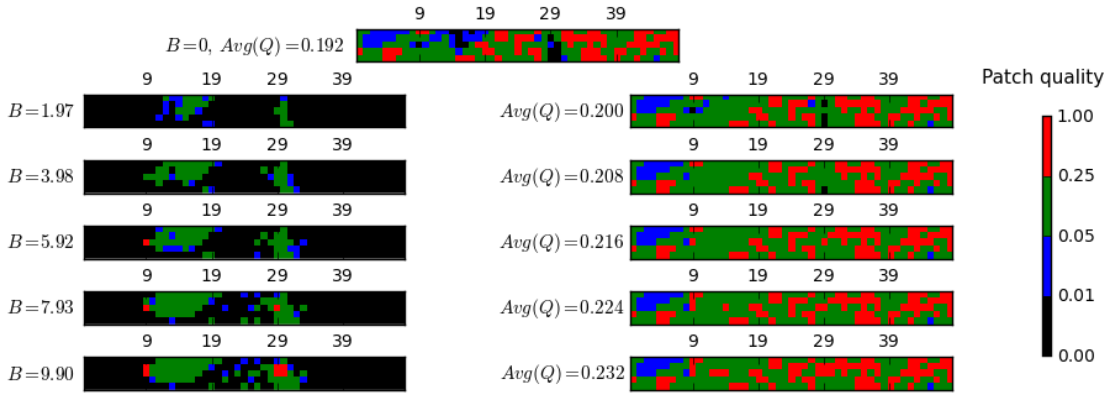


(b) $\alpha = 2$

Figure A.4: The results of improving 5×49 landscape of **low** quality for $\boldsymbol{\alpha}$ equal to **1** (top part) and **2** (bottom part). The map at the top of each part shows the landscape before improvement. The maps in the left column show only the allocation of the budget B by the **random** method. The maps in the right column show the landscape after improvement.

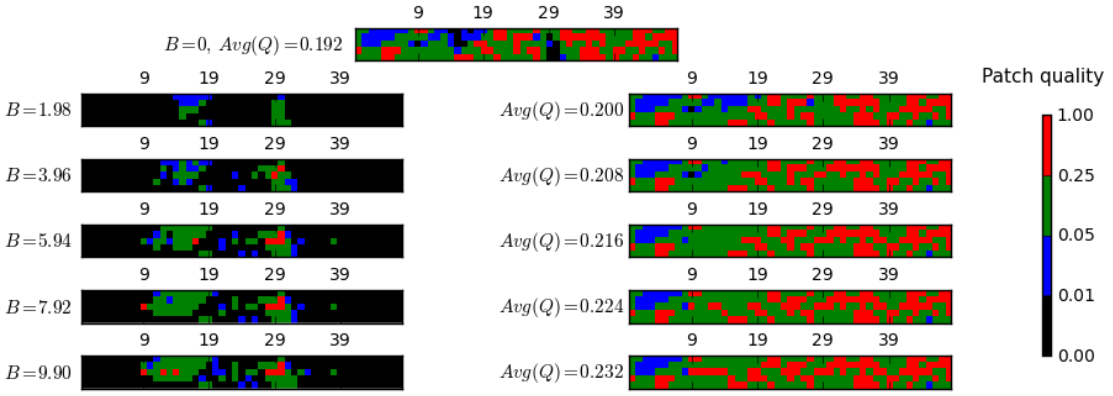


(a) $\alpha = 0.25$

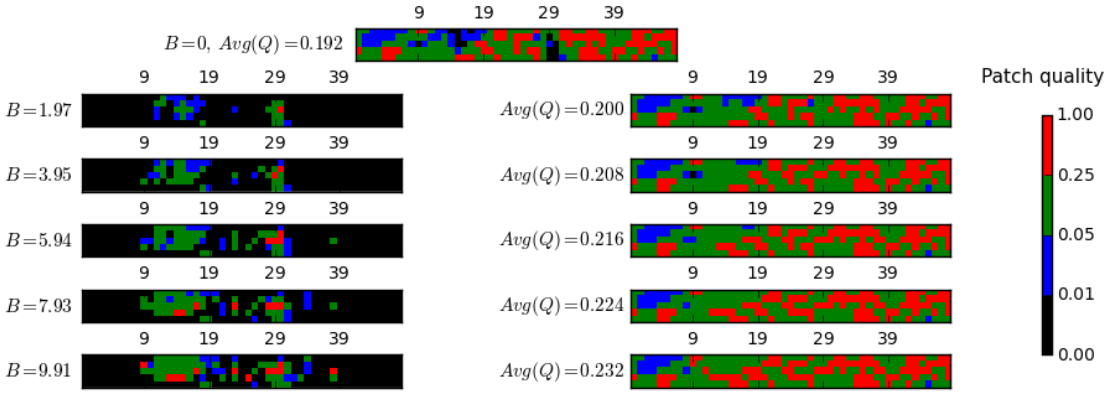


(b) $\alpha = 0.5$

Figure A.5: The results of improving 5×49 landscape of **medium** quality for $\boldsymbol{\alpha}$ equal to **0.25** (top part) and **0.5** (bottom part). The map at the top of each part shows the landscape before improvement. The maps in the left column show only the allocation of the budget B by the **CP-IMF** method. The maps in the right column show the landscape after improvement.

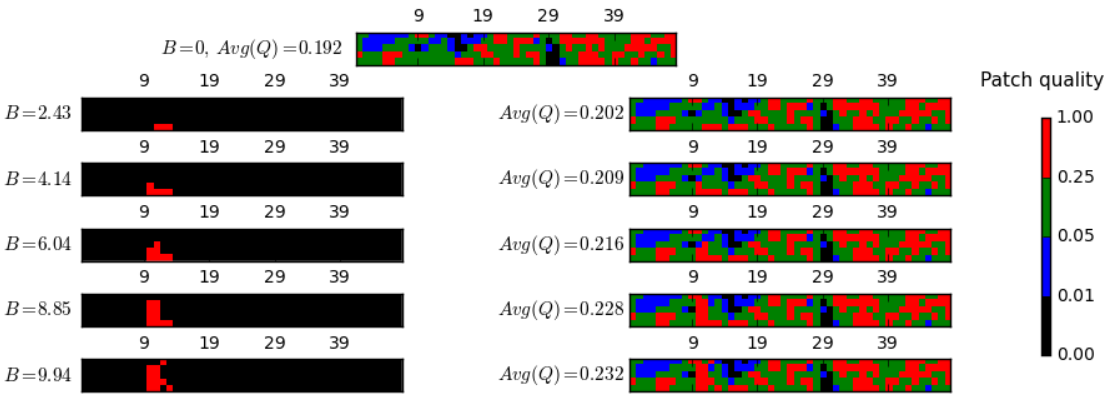


(a) $\alpha = 1$

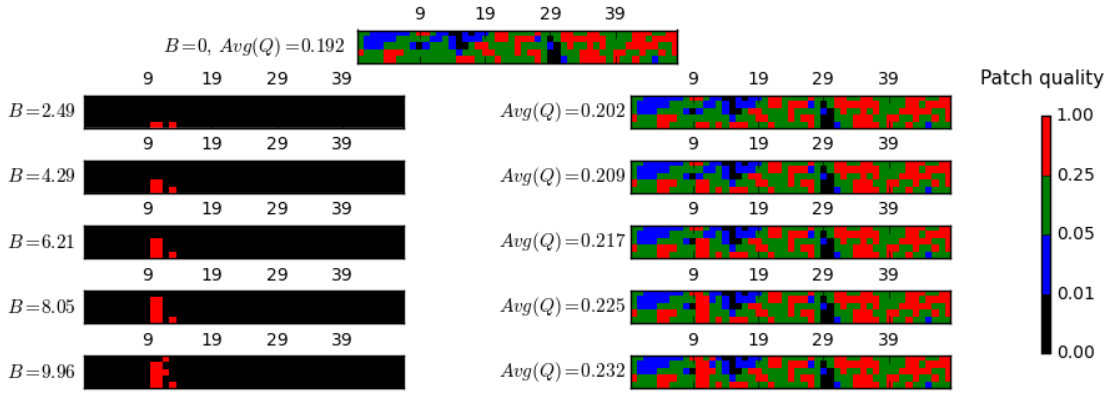


(b) $\alpha = 2$

Figure A.6: The results of improving 5×49 landscape of **medium** quality for $\boldsymbol{\alpha}$ equal to **1** (top part) and **2** (bottom part). The map at the top of each part shows the landscape before improvement. The maps in the left column show only the allocation of the budget B by the **CP-IMF** method. The maps in the right column show the landscape after improvement.

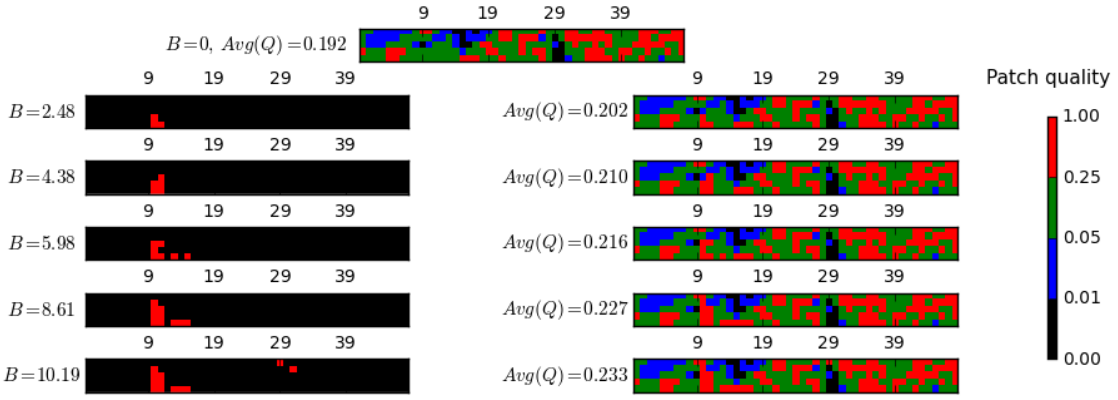


(a) $\alpha = 0.25$

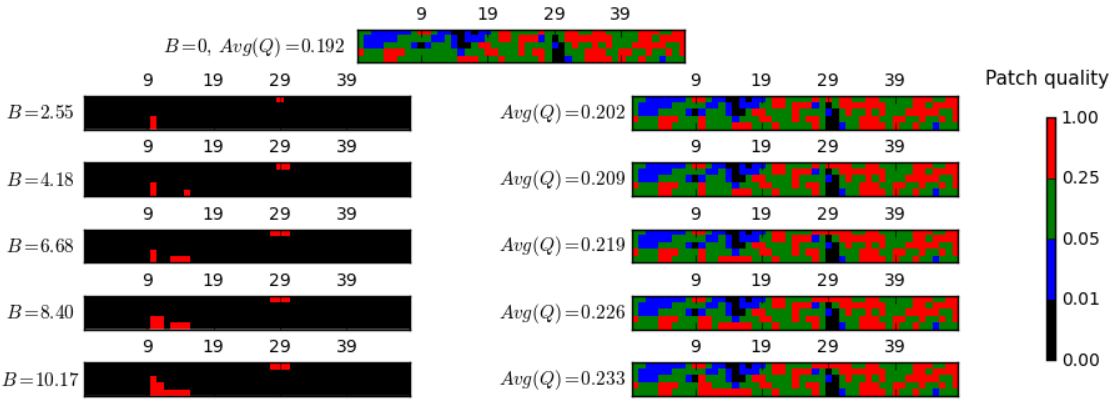


(b) $\alpha = 0.5$

Figure A.7: The results of improving 5×49 landscape of **medium** quality for $\boldsymbol{\alpha}$ equal to **0.25** (top part) and **0.5** (bottom part). The map at the top of each part shows the landscape before improvement. The maps in the left column show only the allocation of the budget B by the **heuristic** method. The maps in the right column show the landscape after improvement.

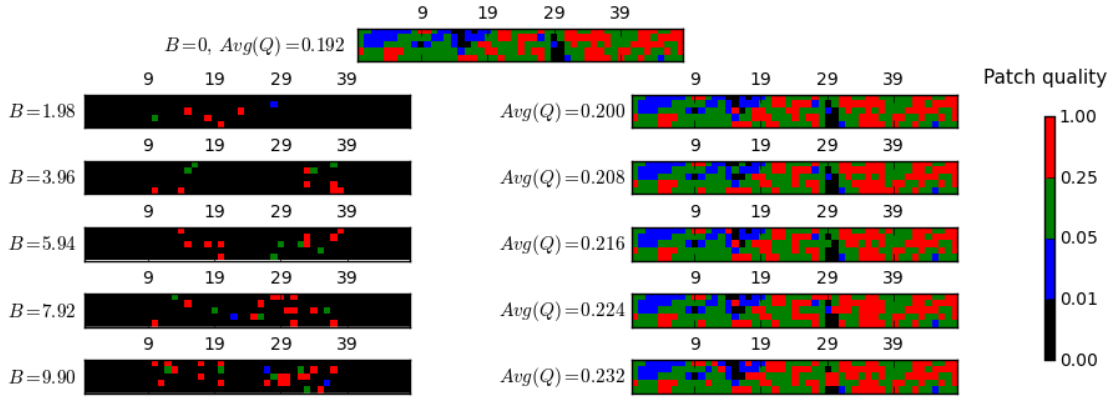


(a) $\alpha = 1$

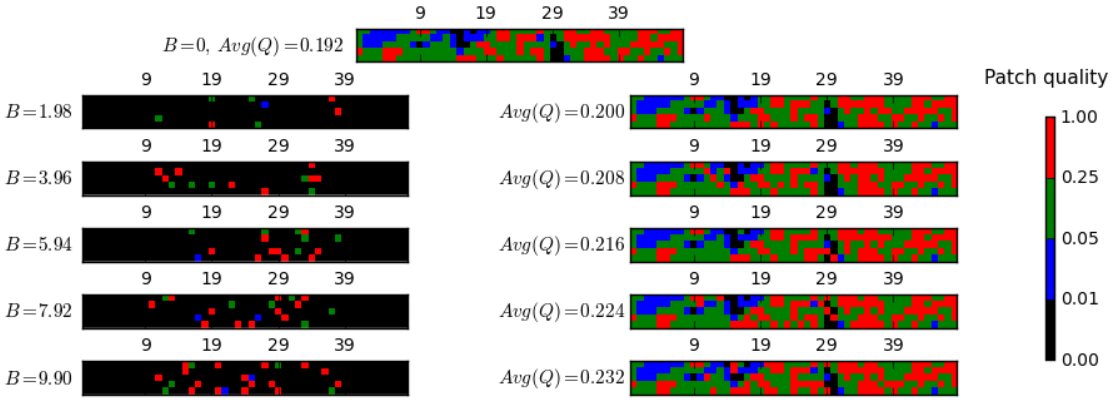


(b) $\alpha = 2$

Figure A.8: The results of improving 5×49 landscape of **medium** quality for $\boldsymbol{\alpha}$ equal to **1** (top part) and **2** (bottom part). The map at the top of each part shows the landscape before improvement. The maps in the left column show only the allocation of the budget B by the **heuristic** method. The maps in the right column show the landscape after improvement.

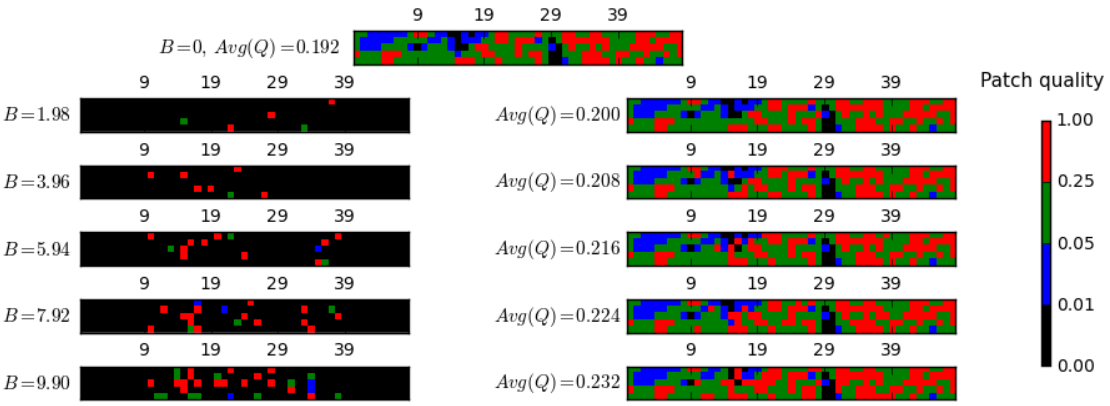


(a) $\alpha = 0.25$

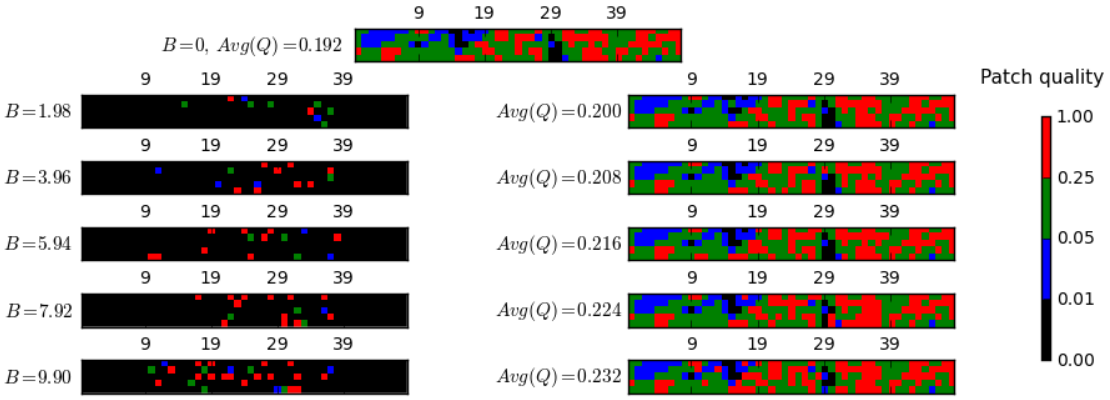


(b) $\alpha = 0.5$

Figure A.9: The results of improving 5×49 landscape of **medium** quality for $\boldsymbol{\alpha}$ equal to **0.25** (top part) and **0.5** (bottom part). The map at the top of each part shows the landscape before improvement. The maps in the left column show only the allocation of the budget B by the **random** method. The maps in the right column show the landscape after improvement.



(a) $\alpha = 1$



(b) $\alpha = 2$

Figure A.10: The results of improving 5×49 landscape of **medium** quality for $\boldsymbol{\alpha}$ equal to **1** (top part) and **2** (bottom part). The map at the top of each part shows the landscape before improvement. The maps in the left column show only the allocation of the budget B by the **random** method. The maps in the right column show the landscape after improvement.

Bibliography

[1] Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1993.

[2] Daniyah Aloqalaa, Dariusz Kowalski, Pawel Blazej, Malgorzata Wnetrzak, Dorota Mackiewicz, and Pawel Mackiewicz. The impact of the transversion/transition ratio on the optimal genetic code graph partition. In *10th International Conference on Bioinformatics Models, Methods and Algorithms (BIOINFORMATICS 2019)*, 2019. To appear.

[3] Daniyah A. Aloqalaa, Jenny A. Hodgson, Dariusz R. Kowalski, and Prudence W. H. Wong. The impact of landscape sparsification on modelling and analysis of the invasion process. 2018. Submitted (journal version).

[4] Daniyah A. Aloqalaa, Jenny A. Hodgson, Dariusz R. Kowalski, and Prudence W.H. Wong. Estimating invasion time in real landscapes. In *2nd International Conference on Computational Biology and Bioinformatics (ICCBB 2018)*, 2018. To appear.

[5] Daniyah A. Aloqalaa, Jenny A. Hodgson, Dariusz R. Kowalski, and Prudence W.H. Wong. Testing methods to minimise range-shifting time with conservation actions. 2019. Submitted.

[6] Daniyah A. Aloqalaa, Jenny A. Hodgson, and Prudence W. H. Wong. The Impact of Landscape Sparsification on Modelling and Analysis of the Invasion Process. In Costas S. Iliopoulos, Solon P. Pissis, Simon J. Puglisi, and Rajeev Raman, editors, *16th International Symposium on Experimental Algorithms (SEA 2017)*, volume 75 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 32:1–32:16, Dagstuhl, Germany, 2017. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[7] David H Ardell. On error minimization in a sequential origin of the standard genetic code. *Journal of Molecular Evolution*, 47(1):1–13, 1998.

[8] David H Ardell and Guy Sella. On the evolution of redundancy in genetic codes. *Journal of Molecular Evolution*, 53(4-5):269–281, 2001.

[9] Lowell W Beineke and Robin J Wilson. *Topics in algebraic graph theory*. Cambridge University Press, Cambridge, UK; New York, 2005.

[10] Paweł Błażej, Dariusz Kowalski, Dorota Mackiewicz, Małgorzata Wnetrzak, Daniyah Aloqalaa, and Paweł Mackiewicz. The structure of the genetic code as an optimal graph clustering problem. *bioRxiv*, 2018.

[11] Paweł Błażej, Dorota Mackiewicz, Małgorzata Grabinska, Małgorzata Wnetrzak, and Paweł Mackiewicz. Optimization of amino acid replacement costs by mutational pressure in bacterial genomes. *Scientific Reports*, 7:1061, April 2017.

[12] Paweł Błażej, Dorota Mackiewicz, Małgorzata Wnetrzak, and Paweł Mackiewicz. The impact of selection at the amino acid level on the usage of synonymous codons. *G3-Genes Genomes Genetics*, 7(3):967–981, 2017.

[13] Paweł Błażej, Paweł Mackiewicz, Stanisław Cebrat, and Małgorzata Wańczyk. Using evolutionary algorithms in finding of optimized nucleotide substitution matrices. In *Genetic and Evolutionary Computation Conference, GECCO'13*, pages 41–42. Companion ACM, 2013.

[14] Paweł Błażej, Błażej Miasojedow, Małgorzata Grabinska, and Paweł Mackiewicz. Optimization of mutation pressure in relation to properties of protein-coding sequences in bacterial genomes. *PLoS One*, 10:e0130411, 2015.

[15] Paweł Błażej, Małgorzata Wnetrzak, Dorota Mackiewicz, and Paweł Mackiewicz. Optimization of the standard genetic code according to three codon positions using an evolutionary algorithm. *PLoS One*, 13(8):e0201715, 2018.

[16] Paweł Błażej, Małgorzata Wnętrzak, and Paweł Mackiewicz. The role of crossover operator in evolutionary-based approach to the problem of genetic code optimization. *Biosystems*, 150:61–72, 2016.

[17] Greta Bocedi, Damaris Zurell, Björn Reineking, and Justin M. J. Travis. Mechanistic modelling of animal dispersal offers new insights into range expansion dynamics across fragmented landscapes. *Ecography*, 37:1240–1253, 2014.

[18] Greta Bocedi1, Stephen C.F. Palmer, Guy Pe'er, Risto K. Heikkinen, Yiannis G. Matsinos, Kevin Watts, and Justin M.J. Travis. Rangeshifter: a platform for modelling spatial eco-evolutionary dynamics and species' responses to environmental changes. *Methods Ecol. Evol.*, 5:388–396, 2014.

[19] Béla Bollobás. Modern graph theory, graduate texts in mathematics vol. 184, 1998.

[20] Barry W Brook, Navjot S Sodhi, and Corey JA Bradshaw. Synergies among extinction drivers under global change. *Trends in ecology & evolution*, 23(8):453–460, 2008.

[21] Michael Bulmer. The selection-mutation-drift theory of synonymous codon usage. *Genetics*, 129(3):897–907, 1991.

[22] Keren Censor-Hillel, Bernhard Haeupler, Jonathan A. Kelner, and Petar Maymounkov. Global computation in a poorly connected world: fast rumor spreading with no dependence on conductance. In *Proceedings of the 44rd annual ACM symposium on Theory of computing*, STOC '12, pages 961–970. ACM, 2012.

[23] Keren Censor-Hillel and Hadas Shachnai. Partial information spreading with application to distributed maximum coverage. In *PODC*, pages 161–170, 2010.

[24] Keren Censor-Hillel and Hadas Shachnai. Fast information spreading in graphs with large weak conductance. In *ACM Sym. Discrete Algorithms (SODA)*, pages 440–448, 2011.

[25] I-Ching Chen, Jane K Hill, Ralf Ohlemüller, David B Roy, and Chris D Thomas. Rapid range shifts of species associated with high levels of climate warming. *Science*, 333(6045):1024–1026, 2011.

[26] Flavio Chierichetti, Silvio Lattanzi, and Alessandro Panconesi. Almost tight bounds for rumour spreading with conductance. In *ACM Sym. Theory Computing (STOC)*, pages 399–408, 2010.

[27] Flavio Chierichetti, Silvio Lattanzi, and Alessandro Panconesi. Rumour spreading and graph conductance. In *SODA*, pages 1657–1663, 2010.

[28] Flavio Chierichetti, Silvio Lattanzi, and Alessandro Panconesi. Rumor spreading in social networks. *Theor. Comput. Sci.*, 412(24):2602–2610, 2011.

[29] Jean Clobert, Michel Baguette, Tim G Benton, and James M Bullock. *Dispersal ecology and evolution*. Oxford University Press, 2012.

[30] Jean Clobert, Jean-François Le Galliard, Julien Cote, Sandrine Meylan, and Manuel Massot. Informed dispersal, heterogeneity in animal dispersal syndromes and the dynamics of spatially structured populations. *Ecology letters*, 12(3):197–209, 2009.

[31] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. MIT Press, 2009.

[32] Daniel J Cowley, Oliver Johnson, and Michael JO Pocock. Using electric network theory to model the spread of oak processionary moth, thaumetopoea processionea, in urban woodland patches. *Landscape Ecology*, 30(5):905–918, 2015.

[33] Francis HC Crick. The origin of the genetic code. *Journal of Molecular Biology*, 38(3):367–379, 1968.

[34] Alan Demers, Dan Greene, Carl Houser, Wes Irish, John Larson, Scott Shenker, Howard Sturgis, Dan Swinehart, and Doug Terry. Epidemic algorithms for replicated database maintenance. *SIGOPS Oper. Syst. Rev.*, 22:8–32, January 1988.

[35] Massimo Di Giulio. The extension reached by the minimization of the polarity distances during the evolution of the genetic code. *Journal of Molecular Evolution*, 29(4):288–293, 1989.

[36] Massimo Di Giulio. The coevolution theory of the origin of the genetic code. *Journal of Molecular Evolution*, 48(3):253–254, 1999.

[37] Massimo Di Giulio. An extension of the coevolution theory of the origin of the genetic code. *Biology Direct*, 3, 2008.

[38] Massimo Di Giulio. The lack of foundation in the mechanism on which are based the physico-chemical theories for the origin of the genetic code is counterposed to the credible and natural mechanism suggested by the coevolution theory. *Journal of Theoretical Biology*, 399:134–140, 2016.

[39] Massimo Di Giulio. Some pungent arguments against the physico-chemical theories of the origin of the genetic code and corroborating the coevolution theory. *Journal of Theoretical Biology*, 414:1–4, 2017.

[40] Massimo Di Giulio. A discriminative test among the different theories proposed to explain the origin of the genetic code: The coevolution theory finds additional support. *Biosystems*, 169:1–4, 2018.

[41] Massimo Di Giulio and Mario Medugno. Physicochemical optimization in the genetic code origin as the number of codified amino acids increases. *Journal of Molecular Evolution*, 49(1):1–10, 1999.

[42] Benjamin Doerr and Mahmoud Fouz. Asymptotically optimal randomized rumor spreading. In *ICALP*, pages 502–513, 2011.

[43] Benjamin Doerr, Mahmoud Fouz, and Tobias Friedrich. Social networks spread rumors in sublogarithmic time. In *Proceedings of the 43rd annual ACM symposium on Theory of computing*, STOC '11, pages 21–30. ACM, 2011.

[44] Sebastián Duchêne, Simon YW Ho, and Edward C Holmes. Declining transition/transversion ratios through time reveal limitations to the accuracy of nucleotide substitution models. *BMC Evolutionary Biology*, 15(1):36, 2015.

[45] Malgorzata Dudkiewicz, Pawel Mackiewicz, Aleksandra Nowicka, Maria Kowalezuk, Dorota Mackiewicz, Natalia Polak, Kamila Smolarczyk, Joanna Banaszak, Miroslaw R. Dudek, and Stanislaw Cebrat. Correspondence between mutation and selection pressure and the genetic code degeneracy in the gene evolution. *Future Generation Computer Systems*, 21(7):1033–1039, 2005.

[46] Calvin Dytham, Justin M. J. Travis, Karen Mustin, and Tim G. Bento. Changes in species' distributions during and after environmental change: which eco-evolutionary processes matter more? *Ecography*, 37:1210–1217, 2014.

[47] Robert Elsässer. On the communication complexity of randomized broadcasting in random-like graphs. In *Proceedings of the eighteenth annual ACM symposium on Parallelism in algorithms and architectures*, SPAA '06, pages 148–157, 2006.

[48] Charles J Epstein. Role of the amino-acid "code" and of selection for conformation in the evolution of proteins. *Nature*, 210(5031):25–28, 1966.

[49] Stephen J Freeland and Laurence D Hurst. The genetic code is one in a million. *Journal of Molecular Evolution*, 47(3):238–248, 1998.

[50] Stephen J Freeland and Laurence D Hurst. Load minimization of the genetic code: history does not explain the pattern. *Proceedings of the Royal Society of London B: Biological Sciences*, 265(1410):2111–2119, 1998.

[51] Stephen J Freeland, Robin D Knight, Laura F Landweber, and Laurence D Hurst. Early fixation of an optimal genetic code. *Molecular Biology and Evolution*, 17(4):511–518, 2000.

[52] Stephen J Freeland, Tao Wu, and Nick Keulmann. The case for an error minimizing standard genetic code. *Origins of Life and Evolution of the Biosphere*, 33(4-5):457–477, 2003.

[53] George Giakkoupis. Tight bounds for rumor spreading in graphs of a given conductance. In *STACS*, pages 57–68, 2011.

[54] George Giakkoupis and Thomas Sauerwald. Rumor spreading and vertex expansion. In *ACM Sym. Discrete Algorithms (SODA)*, pages 1623–1641, 2012.

[55] Mark A Gilbert, Steven M White, James M Bullock, and Eamonn A Gaffney. Spreading speeds for stage structured plant populations in fragmented landscapes. *Journal of Theoretical Biology*, 349:135–149, 2014.

[56] Dimitri Gilis, Serge Massar, Nicolas J Cerf, and Marianne Rooman. Optimality of the genetic code with respect to protein stability and amino-acid frequencies. *Genome Biology*, 2(11):research0049–1, 2001.

[57] Takashi Gojobori, Wen-Hsiung Li, and Dan Graur. Patterns of nucleotide substitution in pseudogenes and functional genes. *Journal of Molecular Evolution*, 18(5):360–369, 1982.

[58] Alfred L Goldberg and Robert E Wittes. Genetic code: aspects of organization. *Science*, 153(3734):420–424, 1966.

[59] Hani Goodarzi, Hamed Shateri Najafabadi, and Noorossadat Torabi. Designing a neural network for the constraint optimization of the fitness functions devised based on the load minimization of the genetic code. *Biosystems*, 81(2):91–100, 2005.

[60] Geoffrey Grimmett and David Stirzaker. *Probability and random processes*. Oxford university press, 2001.

[61] David Haig and Laurence D Hurst. A quantitative measure of error minimization in the genetic code. *Journal of Molecular Evolution*, 33(5):412–417, 1991.

[62] Lee Hannah, Guy Midgley, Sandy Andelman, Miguel Araújo, Greg Hughes, Enrique Martinez-Meyer, Richard Pearson, and Paul Williams. Protected area needs in a changing climate. *Frontiers in Ecology and the Environment*, 5(3):131–138, 2007.

[63] Ilkka Hanski. A practical model of metapopulation dynamics. *Journal of animal ecology*, pages 151–162, 1994.

[64] Nicole E Heller and Erika S Zavaleta. Biodiversity management in the face of climate change: a review of 22 years of recommendations. *Biological Conservation*, 142(1):14–32, 2009.

[65] Ruth Hershberg and Dmitri A Petrov. Selection on codon bias. *Annual Review of Genetics*, 42:287–299, 2008.

[66] Jane K Hill, Chris D Thomas, and Brian Huntley. Climate and habitat availability determine 20th century changes in a butterfly's range margin. *Proceedings of the Royal Society of London B: Biological Sciences*, 266(1425):1197–1206, 1999.

[67] Jenny A Hodgson, Atte Moilanen, Brendan A Wintle, and Chris D Thomas. Habitat area, quality and connectivity: striking the balance for efficient conservation. *Journal of Applied Ecology*, 48(1):148–152, 2011.

[68] Jenny A Hodgson, Chris D Thomas, Steve Cinderby, Howard Cambridge, Paul Evans, and Jane K Hill. Habitat recreation strategies for promoting adaptation of species to climate change. *Conservation Letters*, 4:289–297, 2011.

[69] Jenny A Hodgson, Chris D Thomas, Calvin Dytham, Justin MJ Travis, and Stephen J Cornell. The speed of range shifts in fragmented landscapes. *PLoS ONE*, 7:e47141, 2012.

[70] Jenny A Hodgson, Chris D Thomas, Brendan A Wintle, and Atte Moilanen. Climate change, connectivity and conservation decision making: back to basics. *Journal of Applied Ecology*, 46(5):964–969, 2009.

[71] Jenny A Hodgson, David W Wallis, Ritesh Krishna, and Stephen J Cornell. How to manipulate landscapes to improve the potential for range expansion. *Methods in Ecology and Evolution*, 7(12):1558–1566, 2016.

[72] Olivier Honnay, Kris Verheyen, Jan Butaye, Hans Jacquemyn, Beatrijs Bossuyt, and Martin Hermy. Possible effects of habitat fragmentation and climate change on the range of forest plant species. *Ecol Lett*, 5:525–530, 2002.

[73] Brian Huntley, Yvonne C Collingham, Stephen G Willis, and Rhys E Green. Potential impacts of climatic change on European breeding birds. *PLoS ONE*, 3(1):e1439, 2008.

[74] Richard M. Karp, Christian Schindelhauer, Scott Shenker, and Berthold Vöcking. Randomized rumor spreading. In *Proceedings of the 41st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 565–574, 2000.

[75] Har Gobind Khorana, H Büuchi, H Ghosh, N Gupta, TM Jacob, H Kössel, R Morgan, SA Narang, E Ohtsuka, and RD Wells. Polynucleotide synthesis and the genetic code. In *Cold Spring Harbor Symposia on Quantitative Biology*, volume 31, pages 39–49. Cold Spring Harbor Laboratory Press, 1966.

[76] Maria Kowalczuk, Pawel Mackiewicz, Dorota Mackiewicz, Aleksandra Nowicka, Malgorzata Dudkiewicz, Miroslaw R Dudek, and Stanislaw Cebrat. High correlation between the turnover of nucleotides under mutational pressure and the dna composition. *BMC Evolutionary Biology*, 1(1):13, 2001.

[77] Dariusz R. Kowalski and Christopher Thraves Caro. Estimating time complexity of rumor spreading in ad-hoc networks. In *Ad-hoc, Mobile, and Wireless Network - 12th International Conference, ADHOC-NOW 2013, Wrocław, Poland, July 8-10, 2013. Proceedings*, pages 245–256, 2013.

[78] Vidyadhar G Kulkarni. *Modeling and analysis of stochastic systems*. CRC Press, 2016.

[79] Sudhir Kumar. Patterns of nucleotide substitution in mitochondrial protein coding genes of vertebrates. *Genetics*, 143(1):537–548, 1996.

[80] Joshua J Lawler. Climate change adaptation strategies for resource management and conservation planning. *Annals of the New York Academy of Sciences*, 1162(1):79–98, 2009.

[81] John Lawton, Peter Brotherton, Valerie Brown, Clive Elphick, Alastair Fitter, Jane Forshaw, Ross Haddow, Stephanie Hilbourne, Richard Leafe, Georgina Mace, Mark Southgate, William Sutherland, Tom Tew, John Varley, and Graham Wynne. Making space for nature: a review of england's wildlife sites and ecological network. *Report to DEFRA*, 107, 2010.

[82] James R Lee, Shayan Oveis Gharan, and Luca Trevisan. Multiway spectral partitioning and higher-order cheeger inequalities. *Journal of the ACM (JACM)*, 61(6):37, 2014.

[83] David A Levin, Yuval Peres, and Elizabeth L Wilmer. *Markov chains and mixing times*. American Mathematical Society, Providence, Rhode Island, 2009.

[84] Michael Lynch. Rate, molecular spectrum, and consequences of human mutation. *Proceedings of the National Academy of Sciences*, 107(3):961–968, 2010.

[85] Daniel M Lyons and Adam S Lauring. Evidence for the selective basis of transition-to-transversion substitution bias in two rna viruses. *Molecular Biology and Evolution*, 34(12):3205–3215, 2017.

[86] Paweł Mackiewicz, Przemysław Biecek, Dorota Mackiewicz, Joanna Kiraga, Krystian Baczkowski, Maciej Sobczynski, and Stanisław Cebrat. Optimisation of asymmetric mutational pressure and selection pressure around the universal genetic code. *Computational Science - ICCS 2008, Proceedings, Lecture Notes in Computer Science*, 5103:100–109, 2008.

[87] Steven E Massey. A neutral origin for error minimization in the genetic code. *Journal of Molecular Evolution*, 67(5):510–516, 2008.

[88] Michael Mitzenmacher and Eli Upfal. *Probability and computing: Randomized algorithms and probabilistic analysis*. Cambridge university press, 2005.

[89] Atte Moilanen, Kerrie A Wilson, and Hugh Possingham. *Spatial conservation prioritization: quantitative methods and computational tools*. Oxford University Press, Oxford, 2009.

[90] Brian R Morton. Selection at the amino acid level can influence synonymous codon usage: Implications for the study of codon adaptation in plastid genes. *Genetics*, 159(1):347–358, 2001.

[91] Daniel Morton, Clare Rowland, Claire Wood, Lorna Meek, Christopher Marston, and Geoff Smith. Land Cover Map 2007 (1km percentage aggregate class, GB) v1.2, 2014.

[92] Damon Mosk-Aoyama and Devavrat Shah. Fast distributed algorithms for computing separable functions. *IEEE Transactions on Information Theory*, 54(7):2997–3007, 2008.

[93] Mark Newman. *Networks: An Introduction*. Oxford University Press, 2010.

[94] Marshall Nirenberg, T Caskey, R Marshall, R Brimacombe, D Kellogg, B Doctor, D Hatfield, J Levin, F Rottman, S Pestka, et al. The rna code and protein synthesis. In *Cold Spring Harbor symposia on quantitative biology*, volume 31, pages 11–24. Cold Spring Harbor Laboratory Press, 1966.

[95] Artem S. Novozhilov, Yuri I. Wolf, and Eugene V. Koonin. Evolution of the genetic code: partial optimization of a random code for robustness to translation error in a rugged fitness landscape. *Biology Direct*, 2(1):24, 2007.

[96] Martin L. Parry, Osvaldo F. Canziani, Jean P. Palutikof, Paul J. Van der Linden, and Clair E. Hanson. Climate Change 2007-Impacts, Adaptation and Vulnerability: Contribution of Working Group II to the Fourth Assessment Report of the IPCC. *Cambridge University Press, Cambridge, UK*, 4, 2007.

[97] Dmitri A Petrov and Daniel L Hartl. Patterns of nucleotide substitution in drosophila and mammalian genomes. *Proceedings of the National Academy of Sciences of the United States of America*, 96(4):1475–1479, 1999.

[98] Steven J Phillips, Paul Williams, Guy Midgley, and Aaron Archer. Optimizing dispersal corridors for the cape proteaceae using network flow. *Ecological Applications*, 18(5):1200–1211, 2008.

[99] Michael S Rosenberg, Sankar Subramanian, and Sudhir Kumar. Patterns of transitional mutation biases within and among mammalian genomes. *Molecular biology and evolution*, 20(6):988–993, 2003.

[100] José Santos and Ángel Monteagudo. Simulated evolution applied to study the genetic code optimality using a model of codon reassignments. *BMC Bioinformatics*, 12, 2011.

[101] José Santos and Ángel Monteagudo. Inclusion of the fitness sharing technique in an evolutionary algorithm to analyze the fitness landscape of the genetic code adaptability. *BMC Bioinformatics*, 18(1):195, Mar 2017.

[102] Thomas Sauerwald. On mixing and edge expansion properties in randomized broadcasting. *Algorithmica*, 56(1):51–88, 2010.

[103] Thomas Sauerwald and Alexandre Stauffer. Rumor spreading and vertex expansion on regular graphs. In *ACM Sym. Discrete Algorithms (SODA)*, pages 462–475, 2011.

[104] Frank M Schurr, Jörn Pagel, Juliano Sarmento Cabral, Jürgen Groeneveld, Olga Bykova, Robert B O'Hara, Florian Hartig, W Daniel Kissling, H Peter Linder, and Guy F Midgley. How to understand species' niches and range dynamics: a demographic research agenda for biogeography. *Journal of Biogeography*, 39(12):2146–2162, 2012.

[105] Alistair Sinclair. *Algorithms for random generation and counting - a Markov chain approach*. Progress in theoretical computer science. Birkhäuser, 1993.

[106] John Gordon Skellam. Random dispersal in theoretical populations. *Biometrika*, 38(1/2):196–218, 1951.

[107] Flemming Skov and Jens-Christian Svenning. Potential impact of climatic change on the distribution of forest herbs in europe. *Ecography*, 27:366–380, 2004.

[108] Chris D Thomas, Alison Cameron, Rhys E Green, Michel Bakkenes, Linda J Beaumont, Yvonne C Collingham, Barend FN Erasmus, Marinez Ferreira De Siqueira, Alan Grainger, and Lee Hannah. Extinction risk from climate change. *Nature*, 427(6970):145–148, 2004.

[109] Tsvi Tlusty. A colorful origin for the genetic code: Information theory, statistical mechanics and the emergence of molecular codes. *Physics of Life Reviews*, 7(3):362–376, 2010.

[110] Patrick C Tobin, Christelle Robinet, Derek M Johnson, Stefanie L Whitmire, Ottar N Bjørnstad, and Andrew M Liebhold. The role of allee effects in gypsy moth, *Lymantria dispar* (l.), invasions. *Population Ecology*, 51:373–384, 2009.

[111] John Wakeley. The excess of transitions among nucleotide substitutions: new methods of estimating transition bias underscore its significance. *Trends in Ecology & Evolution*, 11(4):158–162, 1996.

[112] Gian-Reto Walther, Eric Post, Peter Convey, Annette Menzel, Camille Parmesan, Trevor JC Beebee, Jean-Marc Fromentin, Ove Hoegh-Guldberg, and Franz Bairlein. Ecological responses to recent climate change. *Nature*, 416(6879):389–395, 2002.

[113] M. S. Warren, J. K. Hill, J. A. Thomas, J. Asher, R. Fox, B. Huntley, D. B. Roy, M. G. Telfer, S. Jeffcoate, P. Harding, G. Jeffcoate, S. G. Willis, J. N. Greatorex-Davies, D. Moss, and C. D. Thomas. Rapid responses of british butterflies to opposing forces of climate and habitat change. *Nature*, 414:65–69, 2001.

[114] Małgorzata Wnetrzak, Paweł Błażej, Dorota Mackiewicz, and Paweł Mackiewicz. The optimality of the standard genetic code assessed by an eight-objective evolutionary algorithm. *BMC Evolutionary Biology*, in press, DOI: 10.1186/s12862-018-1304-0, 2018.

[115] Carl R Woese. On the evolution of the genetic code. *Proceedings of the National Academy of Sciences of the United States of America*, 54(6):1546–1552, 1965.

[116] Jeffrey Tze-Fei Wong. A co-evolution theory of the genetic code. *Proceedings of the National Academy of Sciences of the United States of America*, 72(5):1909–12, 1975.

[117] Jeffrey Tze-Fei Wong. Coevolution theory of the genetic code: A proven theory. *Origins of Life and Evolution of Biospheres*, 37(4-5):403–408, 2007.

[118] Jeffrey Tze-Fei Wong, Siu-Kin Ng, Wai-Kin Mat, Taobo Hu, and Hong Xue. Coevolution theory of the genetic code at age forty: Pathway to translation and synthetic life. *Life (Basel)*, 6(1):E12, 2016.

[119] Tong Zhou, Mason Weems, and Claus O Wilke. Translationally optimal codons associate with structurally sensitive sites in proteins. *Molecular Biology and Evolution*, 26(7):1571–1580, 2009.