Aix*Marseille université
Socialement engagée

NNT/NL : 2023AIXM0107/003ED184

THÈSE DE DOCTORAT

Soutenue à Aix-Marseille Université
le 10 mars 2023, par

François HAMONIC

Algorithmes pour la conservation et la restauration des habitats et paysages écologiques

Discipline
Informatique

École doctorale
ED 184 MATHÉMATIQUES ET INFORMATIQUE

Laboratoire/Partenaires de recherche
Laboratoire d'Informatique & Systèmes, LIS

Institut Méditerranéen de Biodiversité et d'Ecologie marine et continentale, IMBE

Composition du jury

François MUNOZ LIPhy	Rapporteur
Gautier STAUFFER HEC Lausanne	Rapporteur
Laurent VIENNOT INRIA, IRIF	Président du jury
Stéphanie MANEL CEFE	Examinatrice
Benoit GESLIN IMBE	Examinateur
Cécile ALBERT IMBE	Co-encadrante
Basile COUËTOUX LIS	Co-encadrant
Yann VAXÈS LIS	Directeur de thèse

LIS Laboratoire d'Informatique & Systèmes
imbe institut méditerranéen de biodiversité et d'écologie marine et continentale
RÉGION SUD PROVENCE ALPES CÔTE D'AZUR
natural solutions

Affidavit

I, undersigned, François Hamonic, hereby declare that the work presented in this manuscript is my own work, carried out under the scientific direction of Cécile Albert, Basile Couëtoux and Yann Vaxès, in accordance with the principles of honesty, integrity and responsibility inherent to the research mission. The research work and the writing of this manuscript have been carried out in compliance with both the french national charter for Research Integrity and the Aix-Marseille University charter on the fight against plagiarism.

This work has not been submitted previously either in this country or in another country in the same or in a similar version to any other examination body.

Marseille, April 8, 2023

Publication list and participation in conferences

List of publications realised as part of the PhD thesis:

1. Hamonic, François, Albert, Cécile, Couëtoux, Basile, and Vaxès, Yann. "Optimizing the ecological connectivity of landscapes". In: *Networks* (2022). DOI: 10.1002/net.22131 [59]

2. François Hamonic, Cécile Albert, Basile Couëtoux, Yann Vaxès : Cumulative effects on habitat networks : How greedy should we be? Biological Conservation 2022 (Submitted)

3. Open source code repositories :
   - https://gitlab.lis-lab.fr/francois.hamonic/landscape_opt_networks_submission.git
   - https://github.com/fhamonic/melon
   - https://github.com/fhamonic/mippp
   - https://github.com/fhamonic/landscape_opt

Participation to conferences and summer schools throughout the duration of the thesis:

1. Congrès national de la ROADEF : Association Française de Recherche Opérationnelle et d'Aide à la Décision, Montpelier, 2020 (participation + presentation)

2. Ecole Jeunes Chercheurs en Informatique et Mathématiques, En Ligne, 2020 (participation)

3. Metric Graph Theory and Related Topics, Marseille, 2021 (participation)

4. Journées Algorithmiques et Graphes, Online, 2021 (participation + presentation)

5. Ecole Jeunes Chercheurs du GRD RO : Ordonnancement, Planification & Applications, Paris, 2021 (participation)

6. Congrès national de la ROADEF : Association Française de Recherche Opérationnelle et d'Aide à la Décision, Lyon, 2022 (participation + presentation)

Résumé

La *connectivité* est une caractéristique importante des paysages écologiques qui est devenue un outil essentiel pour la conservation et la restauration de la biodiversité au cours des deux dernières décennies. Définie comme le degré selon lequel un paysage facilite le mouvement des organismes entre les zones d'habitat, la connectivité des paysages joue un rôle crucial dans la survie à long terme des espèces en facilitant l'accès aux ressources vitales, le flux génétique entre les populations et même l'adaptabilité au changement climatique. Un paysage écologique peut être considéré comme un graphe dirigé dont les n sommets représentent les zones d'habitat du paysage et les m arcs représentent les connexions entre ces zones. Chaque sommet est associé à un poids correspondant à la qualité écologique de la zone qu'il représente et chaque arc est associé à une longueur qui représente la difficulté pour un individu d'effectuer le déplacement correspondant. La *Probabilité de Connectivité* du graphe est alors calculée à partir des distances de plus court chemin dans ce graphe pondéré et est souvent utilisée par les écologues pour évaluer la connectivité du paysage et identifier les zones à prioriser pour la conservation ou la restauration.

Dans cette thèse, nous nous intéressons au problème de la maximisation de la Probabilité de Connectivité d'un paysage sous contrainte budgétaire. Ce problème consiste à choisir parmi un ensemble d'améliorations du paysage qui modifient les poids du graphe, un sous-ensemble dont le coût ne dépasse pas le budget et qui augmente autant que possible la Probabilité de Connectivité. Nous donnons une formalisation pour ce problème et montrons qu'elle peut exprimer de nombreuses problématiques de conservation et de restauration. Nous proposons une formalisation en programmation linéaire en nombres entiers basée sur la notion de flot avec multiplicateur ainsi qu'une technique de prétraitement qui permet de réduire de manière significative la taille des programmes linéaires à résoudre. Pour mettre en œuvre ce prétraitement de manière efficace, nous donnons un algorithme en temps $O(m + n \log n)$ pour résoudre le problème suivant : étant donné un ensemble de scénarios caractérisés par le choix des longueurs des arcs et un arc (u, v) , calculer l'ensemble des sommets t tel que (u, v) est sur un plus court chemin de u à t pour tout scénario. Nous appliquons ensuite notre formalisation à divers cas d'étude afin de comparer la solution optimale obtenue avec notre méthode aux solutions sous-optimales obtenues avec les algorithmes plus simples utilisés en pratique par les écologues.

Mots clés : connectivité écologique ; conservation ; restauration ; biodiversité ; optimisation combinatoire ; programmation linéaire ; graphe ; algorithme ; plus court chemin ; intervales de longueurs

Abstract

Landscape connectivity is an important feature of ecological landscapes that has become an essential tool for biodiversity conservation and restoration over the past two decades. Defined as the degree to which a landscape facilitates the movement of organisms between habitat areas, landscape connectivity plays a crucial role in the long-term survival of species by facilitating access to vital resources, gene flow between populations and even adaptability to climate change. An ecological landscape can be viewed as a directed graph with n vertices representing the habitat areas of the landscape and m arcs representing the connections between these areas. Each vertex is associated with a weight corresponding to the ecological quality of the area it represents, and each arc is associated with a length that represents the difficulty for an individual to make the corresponding travel. The *Probability of Connectivity* is then calculated from the shortest path distances in this weighted graph and is often used by ecologists to assess landscape connectivity and identify areas to prioritize for conservation or restoration.

In this thesis, we are interested in the problem of maximizing the Probability of Connectivity of a landscape under a budget constraint. This problem consists in choosing among a set of landscape improvements that modify the weights of the graph, a subset whose cost does not exceed the budget and which increases as much as possible the Probability of Connectivity. We give a formalization for this problem and show that it can express many conservation and restoration problems. We propose a formalization in mixed integer linear programming based on the notion of flow with multipliers as well as a preprocessing technique that allows to significantly reduce the size of the linear programs to be solved. To implement this preprocessing efficiently, we give a $O(m + n \log n)$ time algorithm to solve the following problem: given a set of scenarios characterized by the choice of arc lengths and an arc (u, v), compute the set of vertices t such that (u, v) is on a shortest path from u to t for any scenario. We then apply our formalization to various case studies in order to compare the optimal solution obtained with our method to the suboptimal solutions obtained with the simpler algorithms used in practice by ecologists.

Keywords: landscape connectivity ; biological conservation; probability of connectivity ; combinatorial optimization ; linear programming ; graph ; algorithm ; shortest path ; interval data

Remerciements

Arrivé au bout de ce travail à la saveur d'accomplissement, le temps est enfin venu d'utiliser trop peu de mots pour exprimer tant de gratitude aux nombreuses personnes qui ont été amenées à influencer le cours de cette aventure doctorale.

Mes premiers mots vont évidemment à Cécile Albert, Basile Couëtoux et Yann Vaxès pour leur encadrement sans faille ainsi que pour l'amitié et la bienveillance qui ont su nous lier pendant ces années de thèse et auront définitivement coloré cette expérience en me permettant de progresser tant sur le plan scientifique que d'un point de vue personnel et humain.

Je souhaite également faire part de ma gratitude envers les membres du jury pour l'attention qu'ils ont pu porter à mon travail, pour leurs retours constructifs, leurs questions stimulantes et leur approbation.

Merci également à la Région SUD et à Natural Solutions pour avoir financé ce projet.

Merci à tout le personnel du LIS, de l'IMBE, du LIEU et autres laboratoires, que j'ai eu la chance de côtoyer et m'ont permis d'avancer. Merci également à la sacrosainte machine à café qui, même en n'en consommant que rarement le fruit, possède un indéniable pouvoir sociabilisant.

De manière générale, merci à tous ceux que j'ai la chance de pouvoir appeler mes amis, qui par leur présence, leur soutien, leurs conseils et parfois leur humour ont quelque part tous contribué à une fraction non nulle de ce travail. Merci aux doctorants de Luminy, passés et présents, auxquels je n'envisage toujours pas de refuser la moindre pause café/partie de cartes. Merci particulier à Alexandre D'Ambra pour son amitié et, entre autres, son accompagnement dans mes projets de bricolage loufoques.

Et enfin, Merci à ma maman, Maryvonne Hamonic, pour son soutien, sa patience et son amour sans limites.

Contents

List of Definition Boxes

List of Figures

List of Tables

Glossary

BC-PC-Opt	Budget Constrained PC Optimization, see Important Box 4.1	40–44, 46, 48, 51, 56–58, 64, 74, 81, 86, 104

CSPP	Constrained Shortest Path Problem, see Definition Box 15	74, 75
ECA	Equivalent Connected Area, see [108]	27–29, 54, 82
IIC	Integral Index of Connectivity, see [96]	14, 15, 26, 27
MILP	Mixed Integer Linear Programming, see Definition Box 13	10, 14, 55–57, 63, 64, 67, 68, 74, 84, 91, 94–97, 100, 104
MSS-SPP	Minimax regret Single-Source Shortest Path Problem	68
NP-**Hard**	Non-determinstic Polynomial Hard, see Definition Box 7	13, 45–47, 49, 52, 55, 75, 76, 104
PC	Probability of Connectivity, see Important Box 3.2.1	10, 14, 15, 26–34, 36, 38, 39, 41–44, 49–51, 54, 55, 57, 58, 61, 62, 82, 84–86, 89, 94, 101–104
$PTAS$	Polynomial Time Approximation Scheme, see Definition Box 8	14, 45–47
RDSPP	Robust Deviation Shortest Path Problem	67, 68
SAO-BC-PC-Opt	Single-Arc Only Budget Constrained PC Optimization, see Section 4.3.2	9, 43, 44, 49–51, 55, 56, 66, 89
SPP	Shortest Path Problem, see Definition Box 2	67, 74
SVO-BC-PC-Opt	Single-Vertex Only Budget Constrained PC Optimization, see Section 4.3.1	9, 46–48, 51

1 Introduction

Table of contents

1.1 Nature crises and landscape connectivity

Climate change is only one of the many consequences of our modern civilization on planet Earth. Although it is certainly the most noticeable one for the uninitiated public, it is surely not the most striking and perhaps not (yet) the most worrying. As a matter of facts, land artificialisation, overexploitation, pollution and the introduction of invasive species are as many causes that, like climate change, put pressure on biodiversity and provoke its decline. The IPBES report [17], which is the equivalent of the IPCC (GIEC in French) report [25] for biodiversity, is clear on this point: Nature is declining globally at rates unprecedented in human history. The WWF's Living Planet Report of 2022 [7] states, among other alarming news, that the populations of monitored vertebrate species declined on average by 69% between 1970 and 2018, accounting for 31,821 populations of 5,230 species of mammals, birds, fishes, reptiles and amphibians around the world. Furthermore, insect populations are also estimated to have declined by at least 75% in the last 30 years [58]. Overall, 1 million species of plants and animals are now at risk of extinction [17]. While global warming has its part in these disasters, particularly in the ruthless and accelerating decline of coral populations [17, 29], it is far from being the only explanation for such a collapse of biodiversity. The IPBES report [17] identifies economic growth as a key driver of nature loss by establishing that the causes of biodiversity losses are, in decreasing order of severity, land and sea use change, overexploitation, climate change, pollution and invasive species. Moreover, just like CO_2 emissions [82, 56], an absolute decoupling between economics growth and biodiversity degradation seems more and more unlikely to occur in the near future without major transformations of the economic systems [94].

Therefore, in the meantime, measures are being taken to mitigate the deleterious effects on biodiversity. For two decades now, connectivity conservation has been identified as an essential lever for biodiversity conservation in the face of natural

habitat loss and fragmentation [31, 63, 77]. *Landscape connectivity* is defined by [117] as the degree to which a landscape facilitates the movement of individuals between habitat areas. This feature of landscapes is not only important for the capacity of individual to move around and access vital resources but also to increase gene flow among populations and improve their adaptability to climate change [31, 77].

1.2 Connectivity optimization

Implementing connectivity conservation requires that practitioners identify areas where habitat conservation, restoration, or recreation will be the most effective and cost-efficient for landscape connectivity [11]. For this purpose, graph-theoretical approaches are very useful to model and study habitat connectivity [121]. Indeed, a landscape can be viewed as a directed graph in which vertices represent the habitat areas and each arc indicates a way for individuals to travel from one area to another. Each vertex is associated with a weight corresponding to the ecological quality of the area it represents, and each arc is associated with a length that represents the difficulty for an individual to make the corresponding travel. This difficulty is often assessed with a function of the euclidean distance between habitat areas. Interestingly, this approach can be used for a variety of ecological systems like terrestrial (patches of forests in an agricultural area, networks of lakes or wetlands), riverine (segments of river than can be separated by human constructions like dams that prevent fishes' movement) or marine (reefs that are connected by flows of larvae transported by currents). With this formalism, ecologists have developed many connectivity indicators [96, 109, 88] that aim to quantify the quality of a landscape with respect to the connections between its habitat areas. Therefore, a multitude of frameworks have been developed to address the prioritization of areas for conservation or restoration, under budget constraint, for a variety of contexts and biological systems [81, 5, 116]. The problem facing decision makers is to select, from a given set of possible conservation or restoration options, a subset whose cost does not exceed the available budget and whose impact on improving the quality of the ecological landscape is as great as possible. For most of the connectivity indicators used in practice, the problem is very difficult to solve in terms of computational complexity (In chapter 4 we will see that they are NP-Hard). Indeed, the number of possible solutions grows exponentially with the number of available options. The proposed frameworks thus do not seek optimal resolution and largely struggle to account for the cumulative effects that can occur when multiple impacts or small-scale decisions accumulate over large habitat networks [45, 127, 54]. Indeed, until now, ecologists have mostly tackled these problems by ranking each conservation or restoration option by its independent contribution to the connectivity, i.e., the amount by which the chosen connectivity indicator varies if the option is purchased alone. Such an approach overlooks the cumulative effects of the decisions made like unnecessary redundancies or potential synergistic effects. This could lead to solutions that are more expensive or less beneficial to the landscape connectivity than an optimal solution. Some studies have tried to overcome this limitation by

considering tuples of options [104, 98]. In [104], the authors show that the brute force approach, that consists in testing all possible combinations of options, rapidly becomes impractical for landscapes with more than 20 options. Few studies have explored the search for optimal solutions or guaranteed approximations. In [129], the authors propose a polynomial approximation scheme (*PTAS*) for optimizing the PC indicator when the underlying graph is a tree. The authors of [131] have introduced a XOR sampling method based on a mixed integer formulation for optimizing the PC indicator. To our knowledge, this is the only linear programming formulation already proposed for optimizing the PC. However, their mixed integer formulation does not scale to landscapes with few hundreds of improvement options. More recently, the PhD thesis [70] introduced a constraint programming method for optimizing the IIC indicator. Nowadays, the PC and the IIC are the two most commonly used indicators for connectivity conservation [72].

1.3 Shortest path problems with interval data

Given a landscape graph where each arc is associated to a length, computing the value of the PC indicator requires finding the shortest path distance for each pair of vertices. As we will see in Chapter 4, some improvement of the landscape can be represented by modifying the lengths associated to certain arcs of the graph. Thus, optimizing PC is closely related to solving the shortest path problem on a graph with different length functions, namely one length function for each combination of improvement options. In the sequel, we call such a combination a *scenario*. A very powerful approach to solve this kind of hard combinatorial optimization problems is to express them as Mixed Integer Linear Program (MILP), i.e. as the problem of maximizing a linear objective function subject to linear and integrality constraints. Indeed, state-of-the-art linear programming solvers are able to solve MILPs of increasing size more and more efficiently. However, the size of the resulting MILP models often remains a limit, even to these advanced solvers. An approach has been proposed in [21] for the robust shortest path problem to address this issue. It consists in a preprocessing step that identify a subset of arcs that can either be removed or contracted to reduce the size of the graph considered when computing the shortest path to a given vertex t. In [21], given an arc (u,v) and a vertex t, the authors give a sufficient (but not necessary) condition for (u,v) to be on a shortest path from u to t in every possible scenario and use it to design a $O(m+n\log n)$ time recognition algorithm, where m is the number of arcs and n the number of vertices of the graph. However, since the condition is only sufficient, their algorithm may fail to identify some pairs of arc (u,v) and vertex t for which the property holds. The authors also proposed a $O(m+n\log n)$ time algorithm to test whether a given arc (u,v) is never on a shortest path from u to t. As explained bellow, in this thesis, we have extended this line of research with the objective of improving the algorithmic implementation of this preprocessing and study its efficiency in accelerating the resolution of landscape optimization problems.

1.4 Contributions and organization of the thesis

The contribution of this thesis is twofold. On the one hand, we formalize a versatile problem for the optimization of the PC indicator under a budget constraint and give a mixed integer formulation for solving it efficiently. This new optimization method allows us to compute optimal solutions for restoration and conservation case studies of unprecedented sizes and compare these solutions to those obtained with simpler but suboptimal algorithms used in practice by practitioners. On the other hand, we improve and introduce general purpose processing algorithms for optimization problems involving shortest path computations with changing arc lengths. These algorithms are not only useful for reducing the size of our mixed integer formulations, but have been studied as preprocessing steps for other optimization problems such as robust shortest paths problems [71, 21, 79] and may be of similar interest for optimizing current and future connectivity indicators based on shortest paths, like PC and IIC.

In Chapter 2, we define the essential mathematical notations used throughout the manuscript. In Chapter 3, we analyze the state of the art on the methods used for quantifying landscape connectivity, motivate our choice of the PC indicator and describe modelization techniques to represent landscapes with graphs. Then, in Chapter 4, we introduce the *Budget-Constrained PC Optimization* problem (BC-PC-Opt), show that it can capture many conservation and restoration problematics, prove its NP-Hardness and inapproximability with a constant factor greater than $(1-\frac{1}{e})$, and give our mixed integer formulation for solving it efficiently. Our new formulation is based on a generalized flow formulation instead of a standard network flow formulation. This leads to two improvements. Firstly, our formulation has a linear objective function as opposed to the model of [131] that was using a piecewise constant approximation and additional binary variables to handle non-linearity. Secondly, the new formulation aggregates into a single generalized flow the contribution to the connectivity of several source/sink pairs having the same source whereas the previous model treated every pair separately. More precisely, our model improves on the formulation given in [131] by requiring n times less continuous variables and constraints and n^2 times less integer variables, where n is the number of vertices of the graph. In Chapter 5, we design and analyze an $O(m+n\log n)$ time algorithm that, given an arc (u,v), computes the set of all vertices t such that (u,v) is always on a shortest path from u to t or the set of all vertices t for which (u,v) is never on a shortest path from u to t. Our algorithm improves the one of [21] in two ways. First, we replace the sufficient condition defined in [21] by a necessary and sufficient condition that allows us to identify all the vertices t such that (u,v) is always on a shortest path from u to t instead of a subset of them. We also improve the time complexity for computing this set by a factor n since we only need to run our algorithm once instead of running it for each vertex t. Chapter 6 is dedicated to the greedy algorithms used in practice to solve problems like BC-PC-Opt. We describe in detail these algorithms and provide some instances where they compute solutions that are far from optimal. We also show how to modify these algorithms to improve their execution time in practice, while keeping the same time

complexity. Finally, in Chapter 7, we briefly present the software packages developed during the thesis and compare our optimization approach to the greedy algorithms mentioned above in terms of running times and quality of the solutions found on a set of experimental cases. These experiments show that the preprocessing is very effective and that the greedy approach performs quite well on these instances.

2 Prerequisites

Table of contents

Since this thesis is interdisciplinary, we define many notions of computer science. These definitions are intended to be either read by the neophyte and interested reader or skipped by the more expert one. We have chosen to provide only the essential prerequisites for understanding the mathematical statements in this section, and to introduce other prerequisites and definitions when necessary throughout the manuscript in Definition Boxes.

DEFINITION BOX 1: DEFINITION BOXES

This gray box is a Definition Box. Definition Boxes are used to provide definitions and additional information that are needed at a more local scale.

This is an "Important Box". This kind of boxes is used to highlight the core definitions of the thesis and allow the reader to recover them easily.

2.1 Set theory basics

Set theory is a very powerful tool for representing mathematical objects that is commonly employed as a foundational system for the whole of mathematics. A *set* is simply a mathematical model for representing a collection of different objects. We can describe a set by listing its elements separated by commas within braces $\{\}$. The set containing no elements, called the empty set, is denoted by the symbol $\emptyset$. Given a set S containing exactly the elements 1, 2 and 4, which we can denote as $S = \{1,2,4\}$, Table 2.1 shows true formal statements on S and their transcription in plain words.

Statement	Transcription
$S = \{1,2,4\}$	The set S contains exactly the numbers 1, 2 and 4.
$\|S\| = 3$	The size of the set S is 3.
$1 \in S$	1 is in the set S.
$3 \notin S$	3 is not in the set S.
$\{1,4\} \subseteq S$	$\{1,4\}$ is a subset of S.
$S \cup \{2,3\} = \{1,2,3,4\}$	The set of elements either in S or $\{2,3\}$ is $\{1,2,3,4\}$.
$S \cap \{1,2,3\} = \{1,2\}$	The set of elements both in S and $\{1,2,3\}$ is $\{1,2\}$.
$S \cap \{7,8\} = \emptyset$	The sets S and $\{7,8\}$ share no elements in common.
$S \setminus \{2,6\} = \{1,4\}$	The set S minus the elements of $\{2,6\}$ is $\{1,4\}$
$\sum_{k \in S}(k) = 7$	The sum of the elements of S is 7.
$\prod_{k \in S}(k) = 8$	The product of the elements of S is 8.
$\forall_{k \in S}(k < 5)$	For all element k of S, k is smaller than 5.
$\min_{k \in S}(3-k) = -1$	The minimum value of $(3-k)$ for all $k \in S$ is -1.

Table 2.1: Set theory notations and their transcription in plain words.

Another way of describing a set is the set-builder notation. This notation consists in characterizing the elements of a set by a predicate, i.e., a logical formula, such that the described set contains all the objects for which this predicate is true. For example, the intersection of two sets S and T can be expressed as $S \cap T = \{e : e \in S, e \in T\}$. Figure 2.1 illustrates the interpretation of this set-builder notation of $S \cap T$.

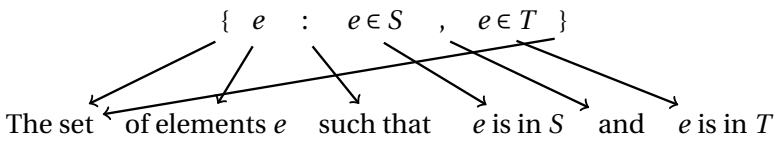


Figure 2.1: Interpretation of the set-builder notation of $S \cap T$.

A set can also contain others sets as its elements, in which case it is often called a *family*. We can describe the intersection of all the sets of a family $\mathcal{F}$ by $\bigcap_{S \in \mathcal{F}} S$. Finally, given two sets A and B, we use the notation A^B to describe the set of all the functions $f : B \to A$ that associate to each element of B an element of A. We often describe the function $f : B \to A$ as a vector of $|B|$ elements of A that is indexed by the elements of B. For example, $x \in \{0,1\}^A$ is a vector such that, for each $a \in A$, either $x_a = 0$ or $x_a = 1$. We refer to [38] for a more in-depth introduction to set theory.

2.2 Graph theory

Graphs are very useful mathematical structures for describing networks of all kinds. Usually, the term graph refers to undirected graphs. An undirected graph G is characterized by a couple of sets $G = (V, E)$ where V is the set of the vertices (or nodes) of the graph and E is the set of the edges (or links) that connects pairs of vertices together.

Formally, the elements of V can be anything from integers to train stations, and E must be a subset of all the pairs that can be formed between all the vertices, i.e.,

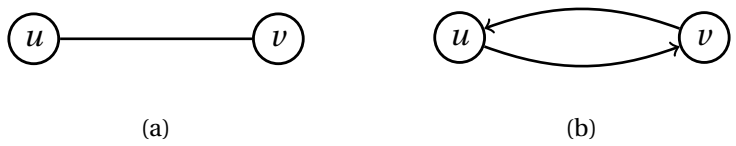
$$E \subseteq \{\{u, v\} : u \in V, v \in V, u \neq v\}. \tag{2.1}$$

On the other hand, a directed graph is formed by a couple (V, A) where V is still the set of vertices but A is a set of arcs (or directed edges) that can only be followed in one direction, i.e.,

$$A \subseteq \{(u, v) : u \in V, v \in V, u \neq v\}. \tag{2.2}$$

Given an arc $a \in A$, let source(a) and target(a) respectively denote its source and target vertices, i.e., if $a = (u, v)$ then source$(a) = u$ and target$(a) = v$.

Most of the time, graphs are assumed to be simple and without loop, that is, they do not contain multiple edge (resp. arc) between to vertices u and v and do not contain any edge (resp. arc) connecting a vertex to itself, hence $u \neq v$. In this thesis we rarely use the notion of undirected graphs since the notion of directed graph is more expressive in our context. Indeed, an edge linking two vertices u and v can be seen as the equivalent of two reciprocal arcs (u, v) and (v, u), see Figure 2.2. Therefore, we use the term graph to designate simple directed graphs without loop.



(a) (b)

Figure 2.2: Topologically equivalent undirected (a) and directed (b) graphs.

A subgraph H of $G = (V, A)$ is a couple $H = (U, F)$ with $U \subseteq V$ and $F \subseteq A$. Given a subset of vertices $W \subseteq V$, we denote by $G[W]$ the subgraph induced by the vertices of W, that is, the graph on the vertices of W that contains all the arcs of A that have both endpoints in W, i.e.,

$$G[W] = \big(W, \{a \in A : \text{source}(a) \in W, \text{target}(a) \in W\}\big). \tag{2.3}$$

Given a graph $G = (V, A)$ and a vertex $w \in V$, we denote δ^{in}_w and δ^{out}_w the sets that correspond respectively to the incoming arcs of w and outgoing arcs of w such that $\delta^{\text{in}}_w = \{a \in A : \text{target}(a) = w\}$ and $\delta^{\text{out}}_w = \{a \in A : \text{source}(a) = w\}$.

The notion of *path* is very important for graph theory applications in general and particularly in the context of landscape connectivity optimization. An st-path in the graph G is a set of arcs $P \subseteq A$ that can be ordered in a sequence $P = \{a_1, \ldots, a_n\}$ of arcs that can be consecutively followed to go from the vertex s to the vertex t, i.e., that fulfills the following properties:

$$\begin{cases} \text{source}(a_1) = s & (2.4) \\ \text{target}(a_n) = t & (2.5) \\ \text{target}(a_i) = \text{source}(a_{(i+1)}) \quad \text{for each } i \in \{1, \ldots, n-1\} & (2.6) \end{cases}$$

If each arc $a \in A$ is associated with a length $l_a \in \mathbb{R}^+$, the length of a path P is defined as the sum of the length of its arcs, i.e., $\sum_{a \in P} l_a$, and the length of a shortest st-path is called the distance from s to t and denoted $d(s,t)$ such that

$$d(s,t) = \min_{P:st\text{-path}} \left(\sum_{a \in P} l_a \right). \tag{2.7}$$

2.3 Complexity of algorithms

In computer science, the computational complexity of an algorithm is a measure of the amount of resources needed to run it. These resources are time and space. Or, in practical terms, "How long should I wait for the result?" and "Does my computer have enough RAM?". The amount of time (and memory) required by an algorithm depends on the size of the input data. Thus, if n describes the size of the input data, the time complexity of an algorithm is a function $f(n)$ that associates each value of n to the number of elementary operations needed to execute the algorithm on the input data. Similarly, the space complexity is usually a function $g(n)$ that associates each value of n to the number of memory bits required to run the algorithm on the input. We are not interested in the exact values of $f(n)$ and $g(n)$ but rather in how they grow with n, i.e. their asymptotic behavior when n tends to infinity. Therefore, the complexity is expressed using big O notation. The big O notation is a standard mathematical notation used for comparing the rate of growth of functions. Given a function $g(n)$, $O(g(n))$ denote the set of all the functions $f(n)$ that are dominated by $g(n)$ when n tends to infinity. Formally, it is the set of functions $f(n)$ for which it exists two constants c and m such that $f(n) < c \cdot g(n)$ for any $n > m$.

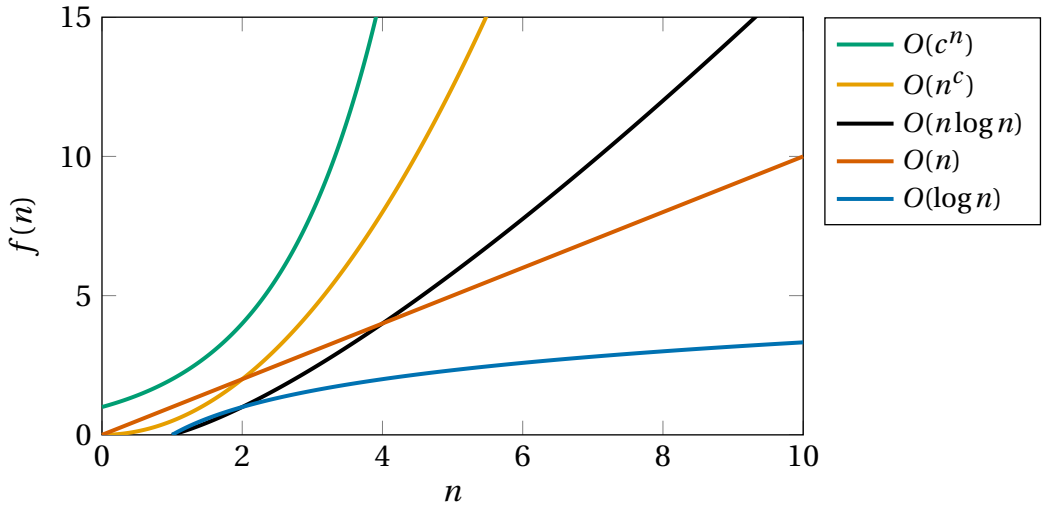


Figure 2.3: Common functions used for big O notations.

One of the goal of computer science is to study mathematical problems, and particularly graph related problems, and find the algorithms with the smaller time complexity possible for solving them. One typical algorithmic problem that has been studied very intensively by computer scientists is the single-source shortest path problem. In the following box, we define this problem and illustrate how we will present an algorithm and its time complexity.

DEFINITION BOX 2: SINGLE-SOURCE SHORTEST PATH PROBLEM

The *Single-Source Shortest Path Problem* (SSSPP) is a fundamental and very well studied combinatorial optimization problem [111]. Given a graph $G = (V, A)$ where each arc $a \in A$ is associated with a length $l_a \in \mathbb{R}^+$ and a vertex $s \in V$, the SSSPP consists in finding the lengths of the shortest paths from s to every other vertices of G, i.e. finding $d(s,t)$ for each $t \in V$.
The SSSPP arises in numerous applications that are not only limited to the fields of telecommunications and transportation with, for example, landscape ecology. The Dijkstra algorithm [35] is well known for solving this problem efficiently. Indeed, it can be implemented to run in $O(|A| + |V| \log |V|)$ time. This algorithm can be described as the following pseudocode:

Algorithm 1: Dijkstra

Input : $G = (V, A)$, $(l)_{a \in A}$, $s \in V$
Output: $dist : V \to \mathbb{R}^+$ such that $dist(t) = d(s, t)$

$dist(s) \leftarrow 0$
foreach $(s, w) \in \delta_s^{out}$ **do**
  $dist(w) \leftarrow l_{sw}$
$S \leftarrow \{s\}$
while $S \neq V$ **do**
  Pick $t \in V - S$ with smallest $dist(t)$
  foreach $(t, w) \in \delta_t^{out}$ **such that** $dist(t) + l_{tw} \leq dist(w)$ **do**
    $dist(w) \leftarrow dist(t) + l_{tw}$
  $S \leftarrow S \cup \{t\}$
return $dist$

Correctness and complexity analysis of Algorithm 1 are provided in standard reference manuals on algorithms such as [28].

Generally, we consider that an algorithm is efficient when its time complexity is polynomial in the size of the input data. We define more advanced notions of computational complexity, like NP-Hardness and Approximation algorithms in the Definition Boxes 7 and 8.

3 Quantify landscape connectivity

Table of contents

In the last two decades, landscape connectivity has become an essential tool for biodiversity conservation and restoration. The first challenge in implementing landscape connectivity is to be able to identify what makes a landscape "well-connected". And, more specifically, how to determine whether a local change in the landscape is beneficial or detrimental to the ability of individuals to move around. In this chapter we go from the state of the art concerning the methods for quantifying landscape connectivity to the description of the method we chose to be our optimization criterion.

3.1 State of the art

Since the 1980s, human activities related to land use have become an increasing concern for biodiversity. During this period, urban and agricultural expansion has destroyed more and more natural habitats, to the point of stimulating environmentalists' interest in the landscape processes affected by these destructions [120]. The studies that followed highlighted the fact that ecological processes are influenced by the landscape at a much larger scale than the local scale that was traditionally considered [128, 37]. An example of local changes that induced global problems in a landscape can easily be found in the consequences of the land consolidation policies that took place in European countries [124]. Land consolidation consists in merging adjacent agricultural parcels for more efficient use. However, it also involves removing the hedgerows that separated these fields. While locally these hedgerows provided little habitat and resources, their removal has had a much greater impact on the landscape

as a whole, c.f. Fig. 3.1. Indeed, due to their rectilinear shape, hedgerows serve as movement and dispersal corridors for many forest species [47, 19]. In addition to that, they are considered key factors for regulating soil erosion and wind speed.

(a) 1950-1965 (b) 2000-2005

Figure 3.1: Aerial photographs of a region of Normandy, France, with, on the left, a photography taken between 1950 and 1965 before the implementation of land consolidation policies and, on the right, a photography taken between 2000 and 2005, IGN [66].

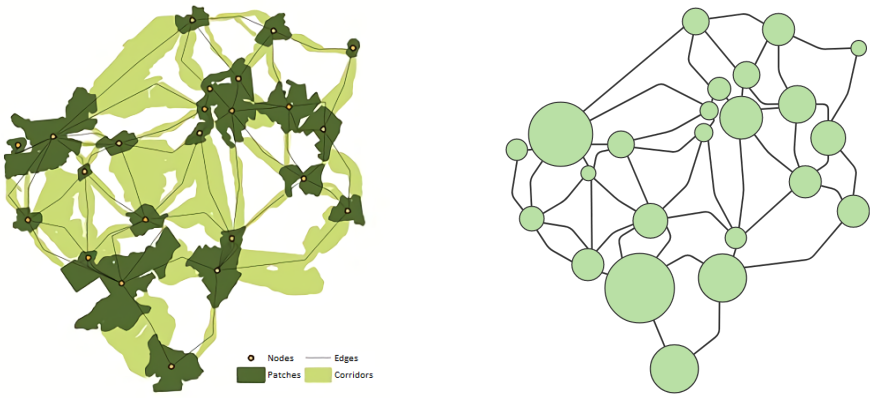
DEFINITION BOX 3: HABITAT PATCH

In landscape ecology, patches are the basic elements of a landscape. A patch is often defined as a contiguous parcel of the same land cover, e.g., forest. However, this definition is highly dependent on the scale adopted because landscapes are fundamentally heterogeneous.

Despite the lack of colors on the first photography of Figure 3.1, we can see that the amount of forest is approximately the same between the two photos whereas two major changes occurred between 1965 and 2000. On the one hand, according to the land consolidation policy, agricultural parcels have been merged, which resulted in the disappearance of most of the landscape's hedgerows and thus undermined the connections between the forest patches. On the other hand, even if some parcels densified and became forests, some parts of the forest patch on the right were destroyed and became fields, breaking the contiguity of this forest patch. In landscape ecology these two effects are recognized under the term *habitat fragmentation* which is the process by which large and contiguous patches of habitats get divided into smaller and isolated ones [40]. Habitat loss and fragmentation are known to be the primary causes of declines in global biodiversity [57]. Fragmented ecological landscapes are

commonly represented as mosaics of patches that provide habitat and resources for species, e.g., forest, above a more dominant and possibly inhospitable environment called the *matrix*, e.g., agricultural fields, roads. Landscape connectivity is then defined as "the degree to which the landscape facilitates or impedes movement among resource patches" [117]. The relative importance of habitat loss versus habitat fragmentation is not clear and depends on many factors such as the degree of landscape fragmentation and the species being studied [8, 40]. Usually, habitat connectivity become more and more important as the proportion of habitat within the landscape decreases. It is therefore necessary to be able to quantify landscape connectivity in order to arbitrate the trade-off between quantity and connectivity of the habitat in conservation and restoration planning. For this purpose a lot of connectivity metrics have been developed in the literature. The first ones are based on simple statistics and information theory such as the percentage of habitat within the landscape, the average size of the habitat patches or the Shannon diversity index. In [86], dated from 1995, the authors present the software FRAGSTATS version 2 together with an exhaustive list of the connectivity indices that were used at the time. In 2000, the authors of [68] introduced three new measures of fragmentation that outperform previous metrics on nine suitability criterions. However, today, most of these connectivity indices would be classified as structural connectivity indicators. As opposed to functional connectivity, which is related to the behavioral response of organisms to the landscape structure, structural connectivity considers only the geometrical properties of the landscape [119, 118].

At that time, landscapes were mainly represented by rasters, i.e., grids of cells, or vector maps, i.e., polygons, that delineate land use types. These two types of representation are ubiquitous in the GIS programs that were becoming more and more popular, thus facilitating spatial analyses. The articles [18, 121] are recognized for popularizing a third tool for representing landscapes, namely graph theory. Indeed, a landscape can be viewed as a graph $G = (V, E)$ whose vertices are the habitat patches and edges represent the connections that individuals can use to travel from one patch to another.



(a) ecological landscape [105]

(b) a possible graph representation

Figure 3.2: Modelisation of an ecological landscape by a graph.

Usually, each patch is associated with a weight representing its ecological value and each edge is associated with a weight representing the distance between the two patches it connects. One strength of this formalism is to allow including species specific variables. For example, each patch can be associated to the estimated number of individual living in it and each edge can be associated to the probability for an individual of the considered species to successfully travel from one patch to the other. This proved to be a good compromise between data requirements and model relevance [20]. Based on this new formalism many connectivity metrics have been created to assess landscape connectivity. In fact graph-based indicators of connectivity are applicable to raster maps since a raster can be represented by a graph where cells are the vertices of the graph and adjacent cells are connected by an edge. In [96] (2006), the authors introduced three graph-based connectivity metrics, namely, the *Class Coincidence Probability* (CCP), the *Landscape Coincidence Probability* (LCP) and the *Integral Index of Connectivity* (IIC). LCP is a generalization of the *degree of coherence* index of [68] to arbitrary graphs. The authors then compared these three indices with previous ones on seven examples to test for the sensibility of the metrics to minor changes in landscape. Each example depicts two versions of the same landscape with a minor change where an ideal connectivity metric should consistently prefer one version over the other. They found that, among the tested metrics, IIC is the only one to consistently prioritize the desired case and that LCP is the second best metric by these criteria. However, these metrics apply to graphs with weighted vertices but unweighted edges. One year later, the authors published another article presenting the *Probability of Connectivity* indicator [109]. Combining their approach on the IIC with the earlier works of [18] and [121] on the Flux metric, they showed that the PC indicator outperforms all previous metrics while allowing to modulate the strength of the connection between pair of patches. Since then, most of the graph-based connectivity indicators that have been proposed, including the PC, have been based on the least-cost path approach, i.e., the connectivity between two patches is assessed by a function of the path of least resistance between these patches. This approach has

been recognized to be a great tool for modeling and calculating inter-patch distances modified with landscape structure and movement behavior [1] but can be criticized for not accounting for the number of different ecological routes that exist between two patches. In [88] (2008), the authors proposed an approach based on electrical circuit theory to assess the connectivity of a landscape when it is represented as a raster grid. The notion of effective resistance that they use in their method has the advantages of being closely related to random walks and thus to account for all the paths from one patch to another.

Afterwards, many reformulations of the PC indicator have been proposed such as the *Probability of Functional Connectivity* (PFC) [126], the *Equivalent Connected Area* (ECA) [108], the *Protected Connected* (ProtCon) [107] and the *Connectivity through Intact lands* (ConnIntact) [125]. There is also the *Density Weighted Connectivity* (DWC) indicator [90] whose formula shares a lot of similarities with the PC. In [72] (2021), the authors provide a review of the existing connectivity metrics used for conservation planning. They conclude that planners should use more than one connectivity metric to prioritize natural lands for inclusion in a protected area network such as in the framework proposed in [5]. In [62], the authors conduct a review of the graph-based connectivity indicators used in studies between 2014 and 2021. Among the 114 studies they reviewed, 58 indicators are mentioned with PC, IIC, LCP and ECA being the four most used indicators with respectively 30%, 18.75%, 8.12% and 2.5% of studies using them. Overall, PC is the most widely used connectivity indicator and has many similarities with other indicators, making it a prime candidate for studying connectivity optimization problems.

3.2 The Probability of Connectivity (PC) indicator

3.2.1 Definition

In general terms, a landscape can be modelled for the PC indicator as a directed graph $G = (V, A, w, \pi)$ with weights on the vertices and the arcs that depends on the studied species that we call a *landscape graph*. The vertices correspond to non-overlapping regions of the landscape and the arcs represent the ability for an individual to travel from one region to another. Each region, i.e., vertex, $u \in V$ is associated with a weight $w_u \in \mathbb{R}^+$ that represents its ecological value for the considered species, that is usually the area of available habitat within the region, and each arc (u, v) is associated with a probability π_{uv} representing the probability for an individual to succeed in his move from the region u to the region v. In Section 3.3, we present more details about the methods used for creating such graphs.

The PC indicator is computed as:

$$PC\,(G=(V,A,w,\pi)) = \frac{\sum_{s\in V}\sum_{t\in V}\left(w_s \cdot w_t \cdot \Pi(s,t)\right)}{\mathcal{W}^2}, \tag{3.1}$$

where $\Pi(s,t)$ is the probability of the most reliable path from s to t, that we often call the *probability of connectivity* from s to t, and $\mathcal{W}$ is a constant greater than $\sum_{u\in V} w_u$ used to normalize the value between 0 and 1. Usually, W is the area of a rectangle containing the landscape under study.

DEFINITION BOX 4: MOST RELIABLE PATH

Given a graph $G=(V,A)$ where each arc $(u,v)\in A$ is associated with a probability π_{uv}, the most reliable path from a vertex $s\in V$ to a vertex $t\in V$ is the path maximizing the product of the probabilities of its arcs:

$$\Pi(s,t) = \max_{P:st\text{-path}} \prod_{a\in P} \pi_a. \tag{3.2}$$

The problem of identifying the most reliable path is closely related to that of identifying the shortest path. Indeed, the computation of $\Pi(s,t)$ can be reduced to the computation of a shortest path from s to t:

$$\begin{aligned}
\Pi(s,t) &= \exp\left(\ln\left(\max_{P:st\text{-path}} \prod_{a\in P} \pi_a\right)\right) && (3.3)\\
&= \exp\left(\max_{P:st\text{-path}} \sum_{a\in P} \ln(\pi_a)\right) && (3.4)\\
&= \exp\left(-\min_{P:st\text{-path}} \sum_{a\in P} -\ln(\pi_a)\right) && (3.5)
\end{aligned}$$

where $\left(\min_{P:st\text{-path}} \sum_{a\in P} -\ln(\pi_a)\right)$ is the length of the shortest path from s to t using $-\ln(\pi_a)$ as the length of the arc a. Since $0\le\pi_a\le 1$ for each $a\in A$, the weights $-\ln(\pi_a)$ are positive. Thus, given a source vertex $s\in V$, the Dijkstra algorithm [35] allows to compute the probabilities of connectivity $\Pi(s,t)$ for all $t\in V$ in one run of time complexity $O(|E|+|V|\log|V|)$.

The ECA indicator [108] is a reformulation of the PC indicator such that:

$$ECA(G) = \sqrt{PC_{num}} \tag{3.6}$$

where PC_{num} is the numerator of the PC formula (3.1),

$$PC_{num}(G) = \sum_{s \in V} \sum_{t \in V} \left(w_s \cdot w_t \cdot \Pi(s,t) \right). \tag{3.7}$$

Since the area containing the landscape under study, $\mathscr{W}$, is a constant term and the square root is a monotonically increasing function, PC and ECA are equivalent in the sense that, given two landscape graphs G and H, $PC(G) > PC(H)$ if and only if $ECA(G) > ECA(H)$. The ECA indicator is often used to make the resulting value more meaningful since $ECA(G)$ is an area value that actually corresponds to the area of a single habitat patch that would have the same PC value as $PC(G)$.

In this thesis we focus on the PC indicator, but since it is the basis for many other connectivity indicators, all our results can be applied, with minor modifications, to its derived indicators such as ECA [108], PFC [126], ProtConn [107], ConnIntact [125] and DWC [90].

3.2.2 Properties

Given a landscape graph $G = (V, A, w, \pi)$ the following properties hold :

Lemma 3.1. *Increasing the quality w_u of a vertex $u \in V$ strictly increases $PC(G)$.*

Proof. The PC formula is the sum, for every pair of vertices s and t, of the products $w_s \cdot w_t \cdot \Pi(s,t)$ where all terms are positive, the increase of w_u can therefore not make $PC(G)$ decrease. However, w_u appears in the product $w_u \cdot w_u \cdot \Pi(u,u)$ where, by definition, $\Pi(u,u) = 1$. Supposing w_u is about to be increased by $\delta > 0$, $PC(G)$ increases then at least by $\delta^2 + 2w_u\delta$. □

Corollary 3.2. *Decreasing the quality w_u of a vertex $u \in V$ strictly decreases $PC(G)$.*

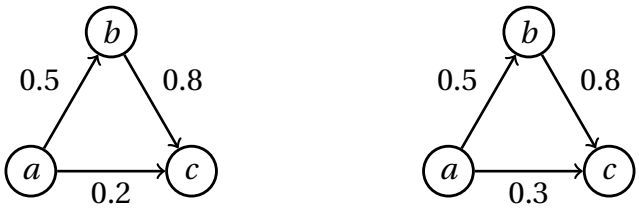
Lemma 3.3. *Increasing the probability π_a of an arc $a \in A$ can only increase $PC(G)$.*

Proof. Let $a = (u, v)$ and suppose π_{uv} is about to be increased to $\pi^+_{uv} > \pi_{uv}$. For every pair of vertices s and t whose original probability of connectivity is $\Pi(s,t)$, we can compute their new *probability of connectivity as $\Pi^+(s,t) = \max\{\Pi(s,t), \Pi(s,u) \cdot \pi^+_{uv} \cdot \Pi(v,t)\}$ thus $\Pi^+(s,t) \geq \Pi(s,t)$. Since the PC formula is the sum, for every pair of vertices s and t, of the products $w_s \cdot w_t \cdot \Pi(s,t)$ where all terms are positive, the increase of π_a can only make $PC(G)$ increase. □

Corollary 3.4. *Decreasing the probability π_a of an arc $a \in A$ can only decrease $PC(G)$.*

Remark. Increasing the probability π_a of an arc a may not increase $PC(G)$.

The two graphs G and H, described by Figure 3.3 have the same value of PC, i.e., assuming $w_a = w_b = w_c = 1$, $PC(G) = PC(H) = 1.7$, despite the fact that the arc (a,c) has a higher probability in H than in G. Indeed, in both cases the most probable path from a to c passes by b and has a probability of $\Pi(a,c) = 0.5 \cdot 0.8 = 0.4$.



(a) G (b) H

Figure 3.3: Two graphs, G (a) and H (b) with H being obtained from G by increasing the probability of the arc (a,c) from 0.2 to 0.3.

Lemma 3.5. *If two vertices s and t of G are connected with a probability of connectivity $\Pi(s,t) = \Pi(t,s) = 1$ then $PC(G) = PC(H)$ where $H = (V', A', w', \pi')$ is the graph obtained from G by contracting the vertices s and t, i.e., replacing them by a vertex u such that $V' = (V \setminus \{s,t\}) \cup \{u\}$ with $w'_u = w_s + w_t$ and $w'_v = w_v$ for any $v \in V' \setminus \{u\}$, and replacing the occurrences of s and t in the arcs of A by u.*

Proof. Since, in G, s and t are connected with probability 1, their probability of connectivity to all the other vertices of the graph are the same, i.e., $\Pi_G(s,v) = \Pi_G(t,v)$ for any $v \in V$, furthermore contracting them does not affect the probability of connectivity between any of the remaining vertices. Let $S = V \setminus \{s,t\}$ be the set of vertices that G and H have in common, we have $\Pi_G(x,y) = \Pi_H(x,y)$ for any pair of vertices $x, y \in S$, and $\Pi_H(u,v) = \Pi_G(s,v) = \Pi_G(t,v)$ for any vertex $v \in S$. Let $F_v^G = \sum_{x \in S} w_x \cdot (\Pi_G(x,v) + \Pi_G(v,x))$ such that $w_v \cdot F_v^G$ is the contribution to $PC(G)$ of all the pairs formed by v and a vertex of S, clearly $F_u^H = F_s^G = F_t^G$. We can then reformulate $PC(G)$ as follows:

$$
\begin{aligned}
PC(G) &= \sum_{v \in S} \left(w_v \cdot F_v^G\right) + w_s \cdot F_s^G + w_t \cdot F_t^G + (w_s)^2 + (w_t)^2 + 2 \cdot w_s \cdot w_t && (3.8)\\
&= \sum_{v \in S} (w_v \cdot F_v) + w'_u \cdot F_u^H + (w'_u)^2 && (3.9)\\
&= PC(H) && (3.10)
\end{aligned}
$$

□

3.2.3 Interpretation

In the article introducing the *PC* indicator [109] as well as in other articles [14, 110], the authors give the idea of a probabilistic interpretation of this indicator as "the probability that two points randomly placed within the landscape fall into habitat areas that are reachable from each other". In this section, we develop this interpretation further, as it seems to explain many of the good properties of the *PC* indicator.

Let $\mathscr{W}$ be the area of a rectangle containing the landscape under study. We consider a stochastic process that consists in choosing two points p and q uniformly at random in the rectangle. The indicator PC is the expected value of a random variable equal to 0 if either p or q does not belong to habitat areas and $\Pi(s,t)$ if p belongs to the habitat of s and q belongs to the habitat of t (recall that $\Pi(s,t) = 1$ if $s = t$). Let w_u denote the quantity of available habitat in the region u. Since the probability that p belongs to the habitat of u is $\frac{w_u}{\mathscr{W}}$ and the events $p \in s$ and $q \in t$ are independent, by linearity of expectation, PC can be expressed as follows:

$$PC(G) = \sum_{s \in V} \sum_{t \in V} \left(\frac{w_s}{\mathscr{W}} \cdot \frac{w_t}{\mathscr{W}} \cdot \Pi(s,t) \right). \tag{3.11}$$

One important assumption of this interpretation is that two points belonging to the same region, i.e., vertex, are assumed to be connected with probability 1 . Although the travel costs of an individual can easily be considered lower in a habitat patch than in the matrix, this can be a rather extreme hypothesis when the regions described by the vertices of the graph are very large with respect to the dispersal capability of the species considered. On the other hand, it seems a reasonable approximation in highly fragmented landscape where habitat patches are small.

3.2.4 Empirical support and limitations

While being widely used in practice [72], metrics based on least-cost paths, such as PC, are sometimes criticized for their ecological interpretation and relevance. Indeed, in most of the least cost path models, the connectivity between two habitat patches is assessed only with the path that connect these patches and whose total resistance to movement is minimum. This leads to two drawbacks: (1) individuals are assumed to have knowledge of the paths presenting the least movement costs among the landscape and (2) PC does not promote the presence of multiple disjoint paths between habitat patches.

To remedy these problems, other measures have been proposed. For example, the resistance distance [88] between two vertices u and v of a graph takes account for all the possible paths between these two vertices. The effective resistance between u and v is linked to the expected number of steps needed in a random walk to go from u to v and back to u by selecting randomly an incident edge to cross at each step. This can be seen as the opposite hypothesis of (1) where individuals are assumed to have absolutely no knowledge of the structure of the landscape. However, some studies have shown that depending on the studied species the least cost path approach product better results, i.e., that sticks more to the reality, than the effective resistance model. For example, least-cost path approaches perform slightly better than circuit theory for predicting the migration routes of elks [84] and, more generally, to predict immigration rates at the patch scale [103]. In [65, 87], the authors suggest that least-cost path methods and circuit theory ones should be used conjointly to find a compromise between the two extreme hypothesis they carry.

Figure 3.3 provides an example of the drawback pointed by (2). While the arc (a, c) has a higher probability in the graph of 3.3b than the one in 3.3b, the PC values of the two graphs are equal since the arc (a, c) does not belong to any least cost path. We can nevertheless argue that an ideal connectivity indicator should be prioritizing the case 3.3b according to the well accepted definition of landscape connectivity: "the degree to which the landscape facilitates or impedes movement among resource patches" [117]. In [102], the authors propose two approaches that evaluate the connectivity between two patches based on many of the shortest paths connecting them.

As the saying goes, "all models are wrong, but some are useful" [16], and despite these criticisms, the PC indicator has not been formally refuted and has received encouraging empirical support [99, 10] while being one of the most used connectivity indicator [62].

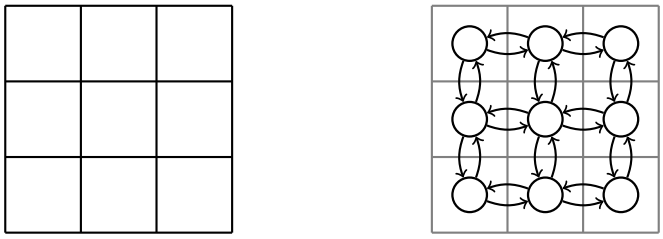
3.3 Modelling landscapes for the PC indicator

As with any quantitative model, assessing the connectivity of a habitat network based on its graph representation critically relies on the input data and the modeling process. This includes characterizing the regions of the landscape that would be represented by the vertices of the graph, weighting these regions according to their ecological quality and modelling the connections between these regions, i.e., determining which pairs of vertices should be linked by arcs and the probabilities to be associated with these arcs. In this section, we present the two most commonly used techniques for representing ecological landscapes with graphs, namely raster-based and patch-based modeling. While the PC indicator can be applied to both [109], these two models share many differences that need to be addressed. We argue that any patch-based model can be viewed as a simplification of a raster-based one and that raster-based models allow for a more accurate modeling of the response of individuals to the landscape structure at the expense of higher computation time. The goal here is not to provide an exhaustive guide for modeling ecological landscapes – this is far beyond the scope of this thesis – but rather to give more precise ideas of what can be meant by graph representation of an ecological landscape when considering the PC indicator.

3.3.1 Raster-based models

Raster data is ubiquitous in all domains where geospatial information are needed. Its basic principle is to subdivide the continuous 2D terrain in cells that are usually squares. The parallel can be made with a camera sensor where the incident light represents the 2D terrain and is divided in pixels. Each cell is then associated with a value, in our analogy, a color, that is the average value of the infinitely many 2D points that lie within the cell. We define a *raster* as a way of subdividing the 2D plane in regular cells. Thus, there are many kinds of rasters, with different cells shapes. For the moment, we consider a standard raster whose cells are square and its associated *raster graph* where each vertex represents a cell and each cell is linked to its four adjacent

neighbors with reciprocal arcs, as depicted in Figure 3.4. From this point, we have



(a) (b)

Figure 3.4: Square raster grid (a) and its associated graph (b)

to determine for each cell the weights of its corresponding vertex and of its incident arcs. Obtaining accurate and reliable data on the biological response of individuals to the landscape structure is complicated, time-consuming and expensive. In general, the weight w_u of a cell u is defined as the area of potential habitat within it. Indeed, habitat quality and habitat area seems to be the most important properties explaining population size [64]. The probability π_{uv} of an arc (u,v) is assessed by a function of the center to center distance between the cells u and v. A basic method for estimating this probability is given in the paper introducing the PC indicator [109] as:

$$\pi_{uv} = \exp(-\alpha \cdot l_{uv}), \tag{3.12}$$

where $l_{uv} = \|u - v\|$ is usually the center-to-center euclidean distance between the cells u and v and α is a constant used to fit the function to the relationship between the travel distance and the dispersal probability of this species. The constant α can be calculated:

$$\alpha = \frac{-\ln(0.5)}{\theta}, \tag{3.13}$$

with θ being the median dispersal distance of the target species. Two cells s and t whose center-to-center distance is θ would then be connected with an arc (s,t) of probability:

$$\pi_{st} = \exp(-\alpha \cdot \theta) = 0.5, \tag{3.14}$$

that is, starting from s, half of the individuals would not make it to t while the remaining half would succeed and continue to disperse.

With additional data, such as the predominant soil type of each cell, it is possible to refine the calculation of both w_u for each vertex u and π_{uv} for each arc (u,v). For example, the habitat quality of a cell u can be assessed by taking into account the land cover and the spatial arrangement of the habitat [5]. Indeed, the edge and the interior of the forest do not present the same attractiveness for some species. Similarly, the estimation of the probability π_{uv} can be refined by taking into account not only the distance between the cells u and v but the movement cost that an individual would pay to get from u to v. In this case, l_{uv} is called the effective distance between

u and v and is computed from the center-to-center euclidean distance between the cells multiplied by a constant r that depends on the soil types of u and v, and the way in which the species under study responds to them, i.e., $l_{uv} = r \cdot \|u - v\|$. Since $\pi_{uv} = \exp(-\alpha \cdot l_{uv})$, this is equivalent to defining a value of α for each type of soil. However, this approach require knowing these values of α. While these data can be accessible in other published works, some frameworks go further and probe the environment to obtain it. For example, in the article introducing the Density Weighted Connectivity (DWC) metric [90], which is very close to the PC one, the authors use spatial capture-recapture models to estimate both the quality of the cells and the probabilities of the arcs based on the fluxes of individuals across the landscape.

The choice of a raster and of its associated graph can have a strong impact on the computed value of the PC indicator. Indeed, the distances in the square raster with 4-neighborhood of Figure 3.4 does not correspond exactly to the euclidean distance.

DEFINITION BOX 5: SPANNER GRAPH

Given an undirected graph $G = (V, E)$ where each edge $e \in E$ has a weight l_e and a real number $r \geq 1$, a r-spanner of G is a graph $H = (V, F)$ with $F \subseteq E$ such that, for any pair of vertices $s, t \in V$, the distance between s and t in H is at most r times the distance between s and t in G:

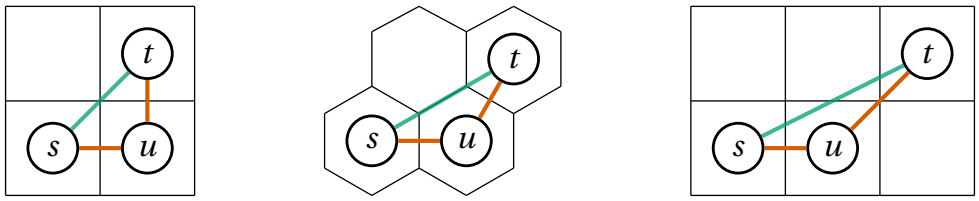
$$\forall_{s,t \in V} d_H(s,t) \leq r \cdot d_G(s,t), \tag{3.15}$$

where $d_G(s,t)$ is the length of the shortest path between s and t in G using l_e as the length of the edge e. The smallest r such that H is a r-spanner of G is called the *stretch factor* of H. This concept extends well to the case where each edge is associated with a probability π_e, and we are interested to the most reliable paths between pairs of vertices. In this case, H is a r-spanner of G if and only if:

$$\forall_{s,t \in V} \Pi_H(s,t) \geq \left(\Pi_G(s,t)\right)^r, \tag{3.16}$$

where $\Pi_G(s,t)$ is the probability of the most reliable path from s to t in the graph G. This definition follows the equivalence between shortest paths and most reliable paths that we addressed in the Definition Box 4 since $\exp(r \cdot \ln(\pi)) = (\pi)^r$.

Thus, in practice, most of the studies employing raster-based modelization uses the square raster with 8-neighbohood, in which each square cell is connected to its eight surrounding neighbors, since its stretch factor for the euclidean distance is closer to 1, see Figure 3.5. We found only one study [115] that uses the hexagonal raster to model landscapes and assess the connectivity of landscapes with the PC indicator. The hexagonal raster has the advantage of having 6 times more arcs than vertices which places it right between the square rasters with 4 and 8 neighborhoods that have, adequately, 4 and 8 times more arcs than vertices, while having a stretch factor



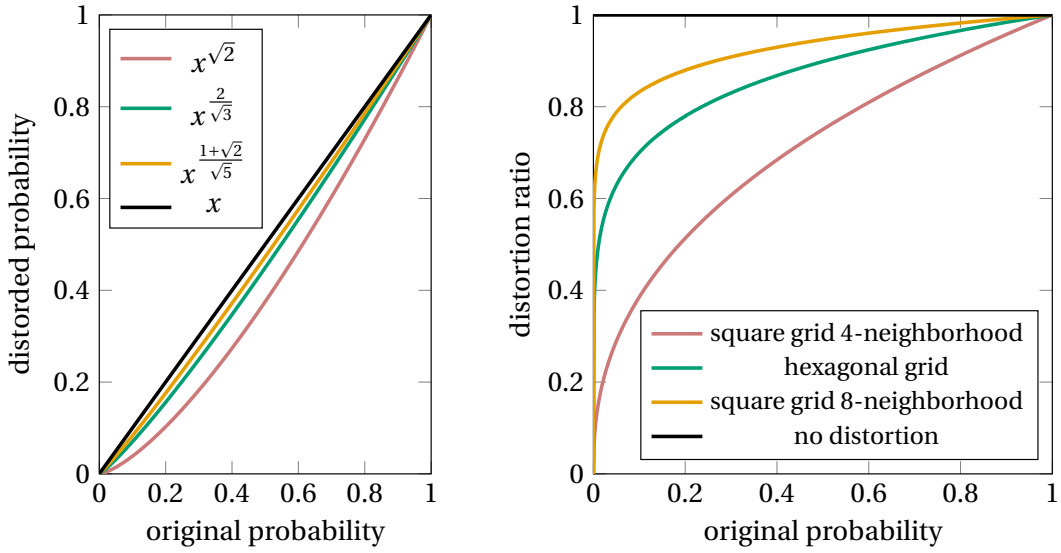
(a) square raster with 4-neighborhood

(b) hexagonal raster

(c) square raster with 8-neighborhood

Figure 3.5: Worst cases illustrating the stretch factors of different raster graphs for the euclidean distance. The shortest st-paths in these graphs are represented in red while the euclidean distance is represented in green by the segment st. The square raster with 4-neighborhood (a) has a stretch factor of $\sqrt{(2)} \simeq 1.41$, the hexagonal raster (b) stretch factor is $\frac{2}{\sqrt{3}} \simeq 1.15$ and the square raster with 8-neighborhood (c) is $\frac{1+\sqrt{(2)}}{\sqrt{5}} \simeq 1.08$.

smaller than the average of the two square rasters. The stretch factor for the euclidean distance of a raster graph can be interpreted as a distortion of the probabilities of connection. Indeed, assuming that the probability of each arc (u, v) is assessed by the function $\exp(-\|u-v\|)$, the probability of connection between two cells s and t in a raster of stretch factor r, that is normally defined as $\exp(-\|t-s\|)$, can be as low as $\left(\exp(-\|t-s\|)\right)^r$. Figure 3.6 shows the worst cases of probability distortion that can happen in those raster graphs.



(a)

(b)

Figure 3.6: Worst cases of probability distortion in raster graphs.

We believe that the hexagonal raster is an excellent compromise between the square rasters with 4 and 8 neighborhoods in terms of number of arcs and stretching factor but that square rasters are preferred for ease of implementation. Indeed, a square raster can easily be represented by two-dimensional arrays whereas it is less trivial for a hexagonal raster.

3.3.2 Patch-based models

In the patch-based model, the vertices of the graph correspond exactly to the patches of habitat of the landscape, i.e., the contiguous area of habitat. Each arc (u, v) then represents the ability for an individual to travel directly from a patch u to a patch v with a success probability of π_{uv}. Like raster-based models, in the absence of more data, the weight w_u of a vertex u is often set to the area of the corresponding habitat patch. However, these weights can also be assessed with additional data, like spatial capture-recapture models [90], or by taking into account the land cover and the spatial arrangement of the habitat [5]. Again, the probability π_{uv} of an arc (u, v) can be assessed with the formula

$$\pi_{uv} = \exp(-\alpha \cdot l_{uv}), \tag{3.17}$$

where, this time, l_{uv} denotes the border-to-border distance between the patches u and v [109]. Thus, in this model, the inner patch movements are assumed to be "free", i.e. they are not accounted in the movement costs of the individuals, only the costs incurred by the matrix are accounted for. Arc probabilities can also be refined by accounting for the heterogeneity of the matrix. For example, a raster model can be used to compute the probabilities of the arcs [1, 41]. This is equivalent to creating a raster model where each cell has a positive weight if it belongs to habitat and a null weight if it belongs to the matrix, and every two adjacent cells corresponding to a habitat area are linked with reciprocal arcs of probability 1. Indeed, by Lemma 3.5, we can then contract all the cells corresponding to a contiguous habitat into a single vertex representing the habitat patch and whose weight is the sum of those of the cells, i.e., the total habitat area. The probability of an arc linking two patches in the patch-based model is then defined by the most reliable path between the corresponding vertices of this raster-based model.

Remark. The PC value between these two models would be essentially the same since only the vertices with positive weights are accounted in the sum of the PC formula.

There is still to decide which vertices must be linked by arcs in a patch-based model. Given a set of patches of the landscape, the simplest graph representation is the complete graph where every pair of patches u and v are connected with reciprocal arcs (u, v) and (v, u). However, for n patches, the complete graph contains $O(n^2)$ arcs which can quickly become intractable for graphs with few thousand of vertices, even for simple graph algorithms. Moreover, since inner patch movements are considered "free", many of the arcs of the complete graph would not be used in most reliable paths and probably do not correspond to actual movements of individuals. To overcome this problem, it is interesting to try to remove a maximum of arcs from the complete

graph while keeping the distances between pairs of vertices as close as possible to the original ones. In fact, finding methods to construct a graph on a set of arbitrary points in 2D whose shortest path distances approximate those of the euclidean distance is a well studied topic of research in graph theory. Such graphs are called *geometric spanners*.

DEFINITION BOX 6: GEOMETRIC SPANNERS

A geometric t-spanners is a graph $G = (V, E)$ whose vertices correspond to points in the 2D plane, i.e., $V \subseteq \mathbb{R}^2$, and where each edge has a length l_e such that the shortest path distances in G approximate the euclidean distance as follows:

$$\forall_{u,v \in V} d_G(u, v) \leq t \cdot ||v - u||, \tag{3.18}$$

where $d_G(u, v)$ is the length of the shortest path between u and v in G using l_e as the length of the edge e and $||v - u||$ is the euclidean distance between u and v. Geometric spanner graphs are well studied [91] with the objective of obtaining good trade-offs between graph size and stretch factor. Indeed, for points in general position, i.e., no 3 of them are aligned, the complete graph is the only geometric spanner with a stretch factor of $t = 1$.
In [15], the authors presents a review of the types of planar graphs that are geometric spanners. While a complete graph possesses $O(|V|^2)$ edges, planar graphs are guarantied to have no more than $(3 \cdot |V| - 6)$ edges which is $O(|V|)$. This result is a corollary of the famous Euler polyhedron formula [80]. The most common plane geometric spanner in the literature are types of Delaunay triangulation. For example, the *triangle distance* (TD) Delaunay triangulation is long known for approximating the euclidean distance in two dimensions with a stretch factor of at most 2 [97]. Afterwards, it was proved that the standard Delaunay triangulation approximates the euclidean distance with a stretch factor of at most 2.42 [73] and, more recently, this upper bound has been improved to 1.998 [130].

Two graph structures are typically used to represent landscape with the patch-based model [50]. The *Complete Thresholded Graph* (CTG) is obtained from the complete graph by removing the arcs whose probability is below a certain threshold γ that corresponds to the maximum distance that organisms can travel. The choice of γ must be checked by expert opinion since a value of γ that is too low will remove too few arcs from the graph while a value that is too high would disconnect the graph, artificially isolating some patches from the others. Therefore, the CTG of an arbitrary graph G is not guarantied to be a spanner. The *Minimum Planar Graph* (MPG), introduced in [41], is the other main structure used for patch-based models. The authors present the MPG as a generalization of the Delaunay triangulation. Indeed, since the distance between two patches is often defined as a function of the border-to-border distance instead of the center-to-center one, patches does not correspond to points in 2D.

Thus, the Delaunay triangulation can not be directly be computed on a set of patches. To our knowledge, there are no known upper bounds on the stretch factor of the MPG with respect to the chosen distances between the vertices. But, since the MPG is a generalization of the standard Delaunay triangulation, its stretch factor (which is defined by the worst case) is not better than those of the Delaunay triangulation. However, in most practical cases, it seems to provide a good approximation of the complete graph [41].

3.3.3 Discussion

The first major difference between raster and patch-based models is the number of vertices of the resulting graph. In a raster-based model, the number of vertices is determined by the type of raster adopted and its resolution, i.e. the chosen cell size, whereas, in the patch-based model there are as many vertices as habitat patches. Consequently, raster based models are generally many order of magnitude bigger than patch-based models in terms of number of vertices. For this reason, most studies that run simulations on the graph use patch-based models for their computational efficiency. However, this larger size of the raster-based models comes with a greater expressiveness. Indeed, since the matrix is entirely represented in raster-based graphs, it is much easier to simulate changes in the matrix that may affect the probability of connection between many pairs of habitats areas and to fully account for landscape heterogeneity, which proved insufficient in the early patch-based models [118]. Raster-based models have also the advantage of sticking more to the interpretation of the PC indicator given in Section 3.2.3. Indeed, in patch-based models, inner patches movements are assumed to have absolutely no cost for individuals which can lead to overestimating the landscape connectivity in the case of very large patches of habitat. Patched-based models can be seen as simplifications of raster-based models in order to make them more computationally tractable at the expense of losing expressiveness about the matrix and making some assumptions about the graph structure (CTG vs MPG) and the inner patch movements.

Overall, the trade-off between raster-based and patch-based models is driven by the need to represent the heterogeneity of the landscape, particularly its matrix, and the constraints in computational resources. Our advice would be to prefer raster-based models as soon as the computational requirements are reasonable. Otherwise, two modifications can be done. First, increasing the size of the cells can significantly reduce the number of vertices in the model, but the incurred loss of resolution can make the model underestimate the impacts of thin barriers like roads. Then, if the inner patch movements can be neglected, contracting the contiguous cells of habitat into a single vertex, as described previously, is an option to greatly reduce the number of vertices pairs contributing in the PC formula while keeping a representation of the whole matrix. We adopted this technique for the Aix case study in our experiments of Chapter 7.

While the goal of this thesis is to improve the solving methods for optimizing the PC indicator in the general case, our contribution in Chapter 5 is even more interesting

for improving the computational tractability of raster-based models since it allows to automatically simplify the parts of the landscape whose contribution to the PC are always the same, regardless of the landscape improvements selected.

4 A Mixed Integer Linear Programming approach for optimizing PC

Table of contents

One of the key issues for which landscape connectivity indicators have been used in the last years is to help conservation practitioners take the most effective and cost-efficient actions to conserve, restore or recreate the habitat patches or ecological corridors that preserve or enhance landscape connectivity [5]. This often translates into identifying the set of vertices or arcs that optimally maintains a good level of connectivity. In this chapter we first introduce the *Budget-Constrained PC Optimization* problem (BC-PC-Opt), that we are concerned with, in Section 4.1, see how it fits in the state of the art in Section 4.2, give difficulty and inapproximability results in Section 4.3 and finally present a mixed integer formalization for solving it efficiently in Section 4.4.

4.1 Problem Statement

For the BC-PC-Opt problem, we give a formulation that could deal with both patch-based and raster-based models as well as address restoration or conservation problematics. The idea is to have a landscape graph G, a set Φ of options and a budget β.

Each option corresponds to either a conservation or restoration action whose effects can be modeled by increasing the weights of some vertices and arcs of the graph and for which we have an estimation of the cost. The BC-PC-Opt problem then consists in finding the best combination of options fitting the budget in order to maximize the PC of the resulting landscape:

BC-PC-Opt

Input: A landscape graph $G = (V, A, w, \pi)$, a budget $\beta \in \mathbb{R}^+$ and a set options Φ such that each option $i \in \Phi$ is associated with a cost $c_i \in \mathbb{R}^+$, a weight $w_u^i \in \mathbb{R}^+$ for each vertex $u \in V$ and a probability $\pi_a^i \in [\pi_a, 1]$ for each arc $a \in A$.

Output: A subset of options $S \subseteq \Phi$ of total cost at most β that maximizes $PC(G' = (V, A, w', \pi'))$, where $w'_u = w_u + \sum_{i \in S} w_u^i$ and $\pi'_{uv} = \max\{\pi_{uv}, \max_{i \in S} \pi_{uv}^i\}$.

Interestingly, this problem is relevant both when the goal is to restore existing landscape elements or when the goal is to identify landscape elements that are at risk of being degraded and need to be protected, while ensuring the highest possible PC value.

In the case of restoration, G is the landscape graph representing the current landscape, Φ is the set of restoration options and the realization of an option $i \in \Phi$ is modeled by increasing the weight of each vertex $u \in V$ by w_u^i and increasing the probability of each arc $(u, v) \in A$ to π_{uv}^i. In practice, the impacts of restoration options are local and very unlikely to affect all the vertices or all the arcs of the graph. For each vertex $u \in V$ (resp. arc $a \in A$) that is not affected by an option $i \in \Phi$, its corresponding weight is simply set to $w_u^i = 0$ (resp. $\pi_a^i = \pi_a$). However, we allow many options to impact the same vertex or arc, in which case we consider that the increases in weight of a vertex adds up whereas the increases in probability of an arc are concurrent and that only the maximum value counts. To model the cumulative improvement of an arc (u, v) by two options i and j, we can replace (u, v) with two arcs (u, x) and (x, v) where the option i improves the probability of (u, x) and the option j improves those of (x, v). This requires adding an intermediary vertex x to the graph such that $w_x = 0$ and $\pi_{uv} = \pi_{ux} \cdot \pi_{xv}$.

In the case of conservation, $G = (V, A, w, \pi)$ represents the landscape as it would be if no conservation action is taken, i.e., if all elements subject to degradation are effectively degraded and Φ is the set of conservation options. The realization of an option $i \in \Phi$ corresponds to preventing the weight of each vertex $u \in V$ from decreasing by w_u^i and the probability of each arc $a \in A$ from dropping below π_{uv}^i. Let $\overline{w}_u$ and $\overline{\pi}_a$ denote the current weights, i.e., before degradation, of each vertex $u \in V$ and each arc $a \in A$, such that $\overline{w}_u \geq w_u$ and $\overline{\pi}_a \geq \pi_a$. If all potential degradation can be prevented by the conservation option, we have:

$$\overline{w}_u = w_u + \sum_{i \in \Phi} w_u^i \tag{4.1}$$

and

$$\overline{\pi}_a = \max\left\{\pi_{uv}, \max_{i \in \Phi}\left(\pi_{uv}^i\right)\right\}. \tag{4.2}$$

Solving the BC-PC-Opt problem in this case amounts to finding the set of conservation options $S \subseteq \Phi$ that guarantees the highest PC value if all the landscape elements that risk degradation and are not protected, i.e. those concerned by the options $\Phi \setminus S$, are indeed degraded. The restoration and conservation cases are thus equivalent, given that they do not start with the same state of the graph.

Concrete instances are described in Chapter 7.

4.2 State of the art

Although, to our knowledge, the BC-PC-Opt problem has never been formalized like this, its scheme can be applied to many studies dealing with conservation or restoration problematics. One of the most typical workflows, widely applied by conservation practitioners, is to represent the landscape with a patch-based model, assess its connectivity using metrics such as the PC, and perform simulation experiments on the graph to associate with each patch, i.e., vertex, the amount by which its removal, i.e., vertex deletion, affects the PC, namely its relative contribution to the PC [14, 115]. This quantity, often denoted dPC_k for a given patch k, is then used to rank the patches of the landscape by order of importance and establish priorities for conservation or restoration. The method of ranking patches based on the effect of their removal on a connectivity indicator has been used since [121], but is known to overlook cumulative effects such as potential synergistic effects or redundancies. For example, in Figure 4.1a, removing the patches b or c makes $PC(G)$ decrease by disconnecting the patches a and d such that $dPC_b = dPC_c = 2$, but removing both patches does not make $PC(G)$ decrease more than that. In this case the two patches act in synergy and are contributing to the PC only if both are selected in the conservation plan. On the other hand, in Figure 4.1b the patches g and h are redundant since they both provide a path from e to f of probability $\Pi(e, f) = 1$. Thus, the deletion of only g or h does not decrease $PC(H)$, i.e., $dPC_g = dPC_h = 0$ although the removing both patches does decrease $PC(H)$. Therefore, dPC_k can either overestimate or underestimate the contribution of a patch k to the PC of the final solution and relying on these values for decision-making can lead to solutions that are more expensive or less beneficial to the PC than the optimal solution. Some studies tried to overcome this limitation by computing the dPC for tuples of n patches instead of single patches [104, 98]. This method, called multi-node approach, as opposed to the single-node approach, is however more computationally demanding since it requires computing the PC for $\frac{|V|!}{(|V|-n)!}$ subsets of patches rather than for each patch, which rapidly becomes impractical for landscape with $|V| \geq 20$ and $n = 5$ [104]. In [51] and [46], the authors show that the

dPC method can not only be used to identify the most important habitat patches in the current landscape, but also to assess the potential gains that can be achieved by reforesting certain regions of the landscape, i.e., adding habitat. In this case, if Φ is the set of regions of the landscape that can be reforested, dPC_i denotes the increase in PC achieved by reforesting the region $i \in \Phi$ which can either be modeled by adding a new vertex to the graph or by increasing the weight of an existing vertex. The authors of [46] then propose a simple greedy algorithm for selecting N regions of Φ in order to maximize the PC. This algorithm consists in N steps where, at each step, it computes dPC_i for each remaining region $i \in \Phi$, adds the region with the highest dPC to the solution and update the landscape graph accordingly. Such greedy algorithms also struggle to take into account all the cumulative effects of the decisions and can provide arbitrarily bad solutions. We go into more details about this in Chapter 6.

(a) G (b) H

Figure 4.1: Examples of cumulative effects: (a) is a landscape graph G on four vertices a, b, c and d with $w_a = w_d = 1$ and $w_b = w_c = 0$. (b) is a landscape graph H on four vertices e, f, g and h with $w_e = w_f = 1$ and $w_g = w_h = 0$. In both cases, all the arcs have a probability of 1.

However, the problem addressed by [51], i.e., selecting N regions of the landscape for reforestation in order to maximize the PC, is actually a special case of BC-PC-Opt. In this case, all the vertices and arcs that would be added to the graph to model the reforestation of some regions are already part of the base graph but with null weights, i.e., 0 for the weights of the vertices and the probabilities of the arcs. Each region that can be reforested then corresponds to a restoration option that consists in increasing the weight of the corresponding vertex and the probabilities of its adjacent arcs to their nominal values. Solving the BC-PC-Opt problem consists in finding the best combination of reforestation options to maximize the PC, all cumulative effects considered.

Only few studies have explored the search for an optimal or a guaranteed approximate solution to maximize landscape connectivity. In most cases these studies focus on dendritic networks such as rivers where the landscape is represented by a tree graph such as in [75, 113]. We only find two articles that address problems directly applicable to the optimization of the PC indicator. In [129], the authors propose a polynomial time approximation scheme for the bidirectional barrier removal problem in river network. When applied to the optimization of the PC, this problem is a special case of BC-PC-Opt, that we later define as SAO-BC-PC-Opt, when the landscape graph

is restricted to be a tree. For any real $\epsilon > 0$, their dynamical programming algorithm with rounding allows to compute a $(1-\epsilon)$-approximated solution in $O(|V|^8/\epsilon)$ time. More recently, [131] introduced a sampling method based on a mixed integer formulation that is applicable to the same problem without requiring the graph to be a tree. To our knowledge, this is the only linear programming formulation of SAO-BC-PC-Opt for general graphs described in the literature. However, on its own, their mixed integer formulation only approximates the PC because they used a standard shortest path linear formulation to identify the most reliable path between each pair of vertices s and t and a piecewise constant approximation to compute $\Pi(s,t)$ from this path. This formulation does not scale to landscape graphs with few hundreds of vertices, as it requires at least $|V|^2 \cdot |A|$ flow variables, $|V|^3$ constraints and $k * |V|^2 + |\Phi|$ integer variables, where k is the number of possible values for each $\Pi(s,t)$.

4.3 Complexity results

In order to be as general as possible concerning the improvements of a landscape graph, the BC-PC-Opt problem is rather complicated, involving a lot of possibly changing weights. In this section, we prove its NP-hardness and inapproximability through the intermediary of two simpler derived problems, namely, the *Single-Vertex-Only* (SVO) and the *Single-Arc-Only* (SAO) variants.

DEFINITION BOX 7: NP-HARDNESS

Two of the most fundamental questions underlying computer science are "What problems can be solved by a computer?" and "How much time and memory does it require?". Soon enough in this area of research, many complexity classes were created in order to arrange problems by order of difficulty. The complexity class P regroups the decision problems, i.e., whose answers are "yes" or "no", that can be solved efficiently [23]. Here, *efficiently* means that, given a problem $A \in P$, the time required to compute the solution of any instance I of A is at most proportional to a polynomial of the number of symbols needed to describe I, i.e., $O(poly(n))$ where n is the size of the instance I. Another important class of complexity is the class NP. A problem B is in NP if, for any instance J, we can provide the solution with a proof of correctness that can be verified efficiently. Clearly, $P \subseteq NP$, since any problem of P can be solved efficiently there is no need for a proof, but, interestingly, the question "$P = NP$?" remains open and is considered the most important challenge of theoretical computer science. The biggest step towards answering this question is the Cook-Levin theorem [27, 78] proving that the Boolean Satisfiability (SAT) problem is NP-Complete. A decision problem $B \in NP$ is said to be NP-Complete if for any problem $A \in NP$, we can efficiently transform an instance I of A into an instance J of B such that solving J gives the solution for I. This transformation, called a

polynomial reduction, is required to be time polynomial, which allows to chain the reductions since $poly(poly(n))$ is also a polynomial. In this way, if SAT is polynomially reducible to another problem of NP, this other problem is also NP-complete. It is then only necessary to find a polynomial algorithm for any of the NP-Complete problems to be able to claim that $P = NP$. Unfortunately (or not [26]) such an algorithm has not yet been found and many consider its existence very unlikely. NP-Hard [52] is the class of problems that are at least as difficult as NP-Complete ones. An optimization problem is said to be NP-Hard if its decision variant is NP-Complete. For example, given a maximization problem A, the associated decision problem is "Given an instance I of A and a value k, does I admit a solution of value at least k?".

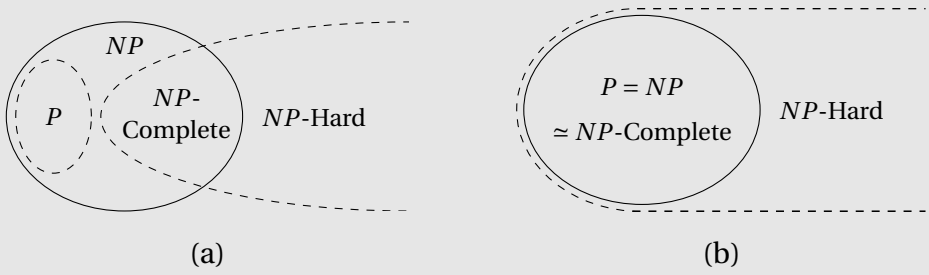


Figure 4.2: Euler diagram for P, NP, NP-Complete and NP-Hard under the hypothesis $P \neq NP$ (a) and under the hypothesis $P = NP$ (b).

The bottom line is that NP-Hard problems are very unlikely to be solved efficiently. In practice, some instances of any NP-Hard optimization problem may require a time that is an exponential function of their size, which corresponds to the need for testing all possible combinations to find the optimal solution.

DEFINITION BOX 8: POLYNOMIAL TIME APPROXIMATION ALGORITHMS

Since NP-Hard optimization problems are expensive to solve, research in this field has moved to concede optimality in order to obtain algorithms that are efficient but still guarantee the quality of their solutions to a certain extent. Unfortunately, for some optimization problems, even approximation has turned out to be difficult to achieve in polynomial time. Given a maximization problem A, an α-approximation algorithm for A, with $0 \leq \alpha < 1$, is a time polynomial algorithm that can associate to any instance I of A a solution y that has at least α times the value of an optimal solution for I [122]. Symmetrically, for a minimization problem $\alpha > 1$ and an α-approximated solution is guaranteed to be at most α times the optimal solution value. A problem admitting such an approximation algorithm is said to be in the APX complexity class. Similarly to the NP class, APX contains a class of problems that are "easy", namely $PTAS$

which stands for Polynomial Time Approximation Scheme. An optimization problem A is in *PTAS* if it admits an α-approximation algorithm for any α.

4.3.1 NP-hardness of SVO-BC-PC-Opt

The *Single-Vertex-Only Budget-Constrained PC Optimization* problem (SVO-BC-PC-Opt) is a specialization of the BC-PC-Opt where each improvement option concerns only the weight of a single vertex of the graph and each vertex corresponds to exactly one improvement option.

SVO-BC-PC-Opt
Input: A landscape representation $G = (V, A, w, \pi)$, a budget $\beta \in \mathbb{R}^+$ and, for each vertex $u \in V$, a cost $c_u \in \mathbb{R}^+$ and a quality gain $w_u^+ \in \mathbb{R}^+$.
Output: A subset of vertices $S \subseteq V$ of total cost at most β that maximizes $PC(G' = (V, A, w', \pi))$, where $w'_u = w_u + w_u^+$ if $u \in S$ and $w'_u = w_u$ otherwise.

In this case, since the probabilities of the arcs are fixed, the probability of connectivity $\Pi(s,t)$ is known in advance for every pair (s,t). This problem can then be formulated as the following mixed integer quadratic program:

$$(SVO\text{-}BC\text{-}PC\text{-}Opt) \begin{cases} \text{maximize} & \sum_{s \in V} \sum_{t \in V} y_s \cdot y_t \cdot \Pi(s,t) & & (4.3) \\ \text{subject to} & y_u = w_u + w_u^+ \cdot x_u & u \in V & (4.4) \\ & \sum_{u \in V} c_u \cdot x_u \leq \beta & & (4.5) \\ & x_u \in \{0,1\} & u \in V & (4.6) \end{cases}$$

Proposition 4.1. *SVO-BC-PC-Opt is NP-Hard.*

Proof. We describe a polynomial reduction from Densest-k-Subgraph to SVO-BC-PC-Opt to show the NP-hardness of the latter.

DEFINITION BOX 9: DENSEST-k-SUBGRAPH

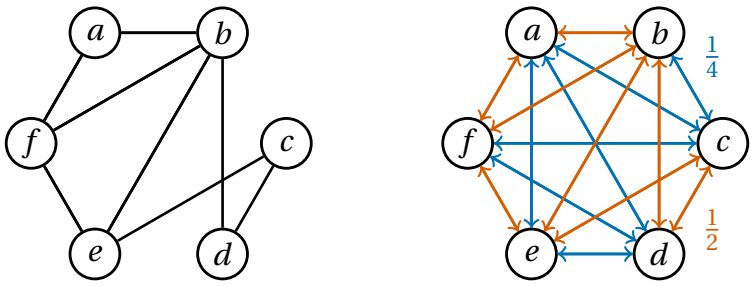
Given a graph $G = (V, E)$, the Densest-k-Subgraph problem (DkS) consists in finding a subgraph of G on exactly k vertices whose number of edges is maximal. This amounts to find a subset $S \subseteq V$ of size k that maximizes $|E(G[S])|$ which

can be formulated as the following mixed integer quadratic program:

$$\text{(Densest-}k\text{-Subgraph)} \begin{cases} \text{maximize} & \sum_{\{u,v\}\in E} x_u x_v & (4.7) \\ \text{subject to} & \sum_{u\in V} x_u = k & (4.8) \\ & x_u \in \{0,1\} \qquad u \in V & (4.9) \end{cases}$$

This problem is known under various names in the literature [114] such as the heaviest unweighted subgraph problem, the k-cluster problem or the k-cardinality subgraph problem. It is *NP*-Hard on general graph and remains *NP*-Hard on bipartite and chordal graphs. It is currently unknown if it exists a constant approximation algorithm for the general case, but it has been shown that, under the assumption $NP \nsubseteq \bigcap_{\epsilon>0} BPTIME(2^{n^\epsilon})$, there is no *PTAS* for this problem [74]. However, it exists *PTAS* for specific cases like dense graphs [9] and interval graphs [93]. Some articles also proposed algorithms with approximation ratio of $\frac{|V|}{2k}$ [43] and $|V|^{1/4}$ [13].

Let I be an instance of the Densest-k-Subgraph problem characterized by a graph $G = (V, E)$ and an integer k. We assign to it the instance J of SVO-BC-PC-Opt characterized by the complete directed graph $H = (V, A, w, \pi)$. Each vertex $u \in V$ has a base quality of $w_u = 0$ that can be increased by $w_u^+ = 1$ by paying the cost $c_u = 1$. Each arc $a \in A$ has a probability of $\pi_a = \frac{1}{2}$ if it corresponds to an edge of G and a probability of $\pi_a = \frac{1}{4}$ otherwise. The transformation from I to J can be done in $O(|V|^2)$ time for creating the complete graph and adding the weights of vertices and arcs.



(a) G (b) H

Figure 4.3: Example of the polynomial reduction from (a) the undirected graph G of an instance of Densest-k-Subgraph to (b) the landscape graph H of the corresponding instance of SVO-BC-PC-Opt where blue arcs have a probability of $\frac{1}{4}$ and red arcs a probability of $\frac{1}{2}$.

Let us adapt the mixed integer quadratic formulation of SVO-BC-PC-Opt (4.3) to the instance J. Since in the instance J $w_u = 0$ and $w_u^+ = 1$ for each $u \in V$, we can directly

use x_u in the objective function and remove the constraint (4.4), by Lemma-3.1, we can also assume that the budget inequality (4.5) is tight since all costs are unitary. This gives:

$$\begin{cases} \text{maximize} \quad \sum_{s\in V}\sum_{t\in V} x_s \cdot x_t \cdot \Pi(s,t) & (4.10) \\ \text{subject to} \quad \sum_{u\in V} x_u = k & (4.11) \\ \qquad\qquad x_u \in \{0,1\} \qquad\qquad u \in V & (4.12) \end{cases}$$

This reformulation shares the same variables and the same constraints with the Densest-k-Subgraph one. It remains to show that their objective functions are positive monotonic transformation of each other. Given that H is a complete graph and each arc has either a probability of $\frac{1}{2}$ or $\frac{1}{4}$, $\Pi(s,t) = \pi_{st}$ for every pair of distinct vertices s and t since any path of at least two arcs has a probability of at most $\frac{1}{2}\cdot\frac{1}{2}$. We can then reformulate (4.10) as:

$$\begin{aligned} \sum_{s\in V}\sum_{t\in V} x_s \cdot x_t \cdot \Pi(s,t) &= \sum_{u\in V} x_u^2 + \sum_{(s,t)\in A} x_s \cdot x_t \cdot \pi_{st} && (4.13) \\ &= k + 2\cdot\alpha\cdot\frac{1}{2} + 2\cdot\left(\binom{k}{2} - \alpha\right)\cdot\frac{1}{4} && (4.14) \\ &= k + \alpha + \frac{k(k-1)}{4} - \frac{1}{2}\cdot\alpha && (4.15) \\ &= \frac{1}{2}\left(\alpha + \frac{k(k+3)}{2}\right), && (4.16) \end{aligned}$$

where α is the number of edges in the subgraph induced by the purchased vertices $\alpha = |\{\{u,v\} \in E : x_u = x_v = 1\}|$. Since $\frac{k(k+3)}{2}$ is a constant term, we just reformulate the objective function of SVO-BC-PC-Opt on instance J as an increasing function of α, completing the proof that SVO-BC-PC-Opt is NP-Hard. □

Remark. Unfortunately, we were not able to derive inapproximability results for SVO-BC-PC-Opt with this polynomial reduction from Densest-k-Subgraph. Indeed, the constant term $\frac{k(k+3)}{2}$ can represent an arbitrary fraction of the optimal value since $0 \le \alpha \le \frac{k(k-1)}{2}$.

Corollary 4.2. *Since SVO-BC-PC-Opt is a special case of BC-PC-Opt, BC-PC-Opt is also NP-Hard.*

4.3.2 NP-hardness and inapproximability of SAO-BC-PC-Opt

The *Single-Arc-Only Budget-Constrained PC Optimization* problem (SAO-BC-PC-Opt) is a specialization of the BC-PC-Opt where each improvement option concerns only

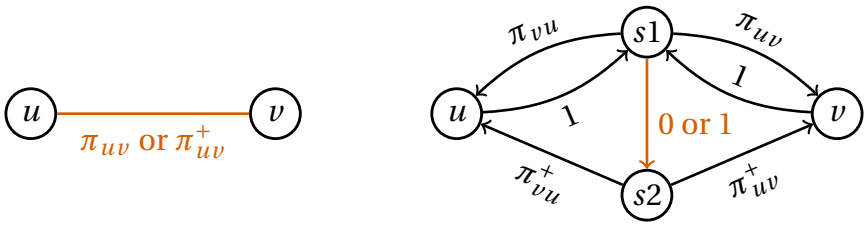
the probability of a single arc of the graph and each arc corresponds to exactly one improvement option.

SAO-BC-PC-Opt

Input: A landscape representation $G = (V, A, w, \pi)$, a budget $\beta \in \mathbb{R}^+$ and, for each arc $a \in A$, a cost $c_a \in \mathbb{R}^+$ and an improved probability $\pi_a^+ \geq \pi_a$.

Output: A subset of arcs $S \subseteq A$ of total cost at most β that maximizes $PC(G' = (V, A, w, \pi'))$, where $\pi'_a = \pi_a^+$ if $a \in S$ and $\pi'_a = \pi_a$ otherwise.

Remark. The SAO-BC-PC-Opt problem where each improvement option concerns only a single arc is more general than the analogous problem on undirected graphs where the improvement options would concern single edges. Indeed, Figure 4.4 shows a construction for passing from the case where an edge $\{u, v\}$ has a probability π_{uv} that can be improved to π^+_{uv} to the case where the improvement of a single arc has the same impact on the probabilities of connectivity between u and v, $\Pi(s, t)$ and $\Pi(t, s)$.



(a) (b)

Figure 4.4: Example of the construction for reducing the version of the SAO-BC-PC-Opt problem on undirected graphs to SAO-BC-PC-Opt.

We define the value of a solution to the SAO-BC-PC-Opt problem as the amount by which the solution increases the PC value of the original landscape.

Proposition 4.3. *SAO-BC-PC-Opt is NP-Hard and can not be approximated in polynomial time with a constant factor greater than* $1 - \frac{1}{e} \simeq 0.632$, *assuming* $P \neq NP$.

Proof. To prove the above proposition, we describe a polynomial reduction from the Max-Coverage problem that preserves approximation.

DEFINITION BOX 10: MAX-COVERAGE

Given a set of elements E, a family $\mathcal{F}$ of subsets of E and an integer k, the Maximum-Coverage problem consists in finding k sets of $\mathcal{F}$ that maximize the number of covered elements $|\bigcup_{S \in \mathcal{F}} S|$. It can be formulated as the following

integer program:

$$\text{(Max-Coverage)} \begin{cases} \text{maximize} \quad \sum_{e \in E} x_e & & (4.17) \\ \text{subject to} \quad x_e \leq \sum_{S \in \mathscr{F}_e} x_S & e \in E & (4.18) \\ \qquad \sum_{S \in \mathscr{F}} x_S \leq k & & (4.19) \\ \qquad x_e \in \{0,1\} & e \in E & (4.20) \\ \qquad y_S \in \{0,1\} & S \in \mathscr{F} & (4.21) \end{cases}$$

where $\mathscr{F}_e = \{S \in \mathscr{F} : e \in S\}$ is the set of sets containing e.
Max-Coverage is a NP-Hard problem which is widely taught in the context of approximation algorithms. Indeed, the greedy algorithm consisting in iteratively selecting a set covering the largest number of uncovered elements provides a $1 - \frac{1}{e} \simeq 0.632$ approximation guaranty, i.e., the returned solution is guaranty to cover at least $1 - \frac{1}{e}$ times the number of elements covered in the optimal solution. Furthermore, it has been shown that, unless $P = NP$, no polynomial algorithm can achieve a better approximation factor [42].

Let I be an instance of the Max-Coverage problem characterized by a set E, a family $\mathscr{F}$ of subsets of E and an integer k. We assign to it the instance J of SAO-BC-PC-Opt characterized by the landscape graph $G = (V, A, w, \pi)$, a budget k and, for each arc $a \in A$, an improved probability π_a^+ and a cost c_a. There is a vertex in G for each element $e \in E$, for each set $s \in \mathscr{F}$ and an extra vertex t, i.e., $V = E \cup \mathscr{F} \cup \{t\}$. Each vertex $u \in V$ has a weight $w_u = 1$ if it corresponds to an element of E or if $u = t$ and a weight $w_u = 0$ otherwise. For each set $s \in \mathscr{F}$, there is an arc $(s,t) \in A$ with a default probability of $\pi_{st} = 0$, an increased probability $\pi_{st}^+ = 1$ and a cost $c_{st} = 1$. Finally, there is an arc (e,s) for each couple of element $e \in E$ and set $s \in \mathscr{F}$ such that $e \in s$ with a probability of $\pi_{es} = 1$ which can't be improved. Figure 4.5 shows the graph construction for an arbitrary instance of Max-Coverage. This construction can be done in polynomial time since there is $O(|E| + |\mathscr{F}|)$ vertices and at most $O(|E| \cdot |\mathscr{F}|)$ arcs.

In this construction, there is no path between the vertices of E since they have no incoming arcs, and, without improvements, all the paths leading to t have a null probability. The default value of the PC is then only the contribution of the individual vertices, i.e., $PC(G) = |E| + 1$. Now, it is easy to see that the only couples of vertices that can contribute to the PC are the couples (e,t) where $e \in E$ and that $w_e \cdot w_t \cdot \Pi(e,t) = 1$ if an arc (s,t) corresponding to a set $s \in \mathscr{F}$ with $e \in s$ is selected in the solution and 0 otherwise. Finding the optimal solution for the instance J amounts then to finding the set of arcs S which connects the maximum number of vertices of E to t or, equivalently that maximizes $|\bigcup_{(s,t) \in S} s|$. Thus, an optimal solution for the instance J gives an optimal solution for the instance I of Max-Coverage, proving that

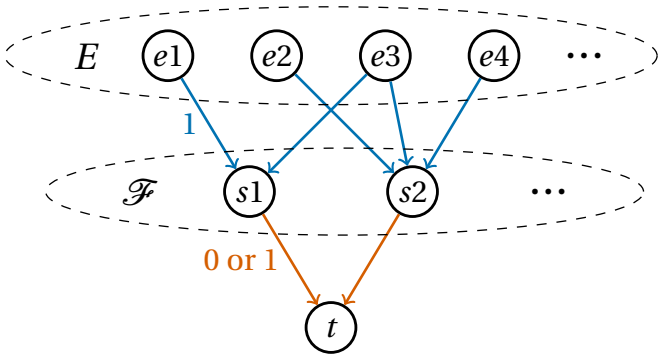


Figure 4.5: Example of the polynomial reduction from an instance of Max-Coverage to the graph of the corresponding instance of SVO-BC-PC-Opt where blue arcs have a probability of 1 and red arcs a default probability of 0 that can be increased to 1 if selected in the solution.

SAO-BC-PC-Opt is NP-Hard.

As in the instance J the value of a solution S is exactly the number of covered elements for the instance I, an approximation algorithm for SAO-BC-PC-Opt with a constant factor $r < 1$ would give an approximation algorithm with factor r for the Max-Coverage problem. Since it is well known that Max-Coverage can not be approximated with a constant factor greater than $1 - \frac{1}{e}$, under the assumption $P \neq NP$, SAO-BC-PC-Opt can also not be approximated with a constant factor greater than $1 - \frac{1}{e}$. □

Corollary 4.4. *Since SAO-BC-PC-Opt is a special case of BC-PC-Opt, BC-PC-Opt cannot be approximated with a constant factor greater than* $1 - \frac{1}{e} \simeq 0.632$, *assuming* $P \neq NP$.

4.4 A MILP formulation for solving BC-PC-Opt

In this section we gradually introduce our mixed integer linear formulation for the BC-PC-Opt problem. We first present a linear program allowing to compute the PC of a given landscape graph, then describe a mixed integer linear formulation for the SAO-BC-PC-Opt problem and show how to extend this formulation to the BC-PC-Opt problem. Finally, we provide ways to preprocess the formulation in order to significantly reduce its size and improve its resolution time in practice.

4.4.1 A Linear Program to compute PC

Since $PC(G) = \frac{PC_{\text{num}}(G)}{\mathcal{W}^2}$ and $\mathcal{W}^2$ is a constant, see Section 3.2 for details, we only need to compute $PC_{\text{num}}(G)$. We decompose it as $PC_{\text{num}}(G) = \sum_{t \in V} w_t \cdot f_t$, where $f_t = \sum_{s \in V} w_s \cdot \Pi(s,t)$, and describe how f_t can be expressed as the optimal value of a linear programming formulation of a generalized flow problem.

DEFINITION BOX 11: LINEAR PROGRAMMING

Linear Programming (LP), also called Linear Optimization, is a technique that can be used to solve some optimization problems. An instance of Linear Programming, called linear program, is characterized by a vector $x \in (\mathbb{R}^+)^n$ of n unknown variables, a linear function $c^\intercal x$ where the vector $c \in \mathbb{R}^n$ defines the coefficients of the variables of x, i.e.

$$c^\intercal x = \sum_{i=1}^{n} c_i \cdot x_i \,, \tag{4.22}$$

and a set of m linear constraints often described by a matrix $A \in \mathbb{R}^{m \times n}$ with m rows and n columns, and a vector $b \in \mathbb{R}^m$. Each linear constraint is then represented by a row $j \in \{1, ..., m\}$ of A and the entry b_j such that

$$\sum_{i=1}^{n} a_{ji} \cdot x_i \leq b_j \,, \tag{4.23}$$

where a_{ji} is the entry positioned at row j and column i of A. Solving a linear program therefore consists in finding values for the variables of x such that the constraints are satisfied and $c^\intercal x$ is maximum.

$$(\mathrm{P}) \begin{cases} \text{maximize} \quad c^\intercal x & (4.24) \\ \text{subject to} \quad Ax \leq b & (4.25) \\ \qquad\qquad\quad x \geq 0 & (4.26) \end{cases}$$

Linear programming was a major breakthrough in the history of combinatorial optimization. The first algorithm for solving it in most cases is attributed to George Dantzig in 1947 [32] and called the simplex method. Although there is no known variant of this algorithm whose time complexity is polynomial in the worst case, it is very efficient in practice and is still used as the basis of modern LP solvers. It is only in 1979 that Leonid Khachiyan showed that all linear programs can be solved in polynomial time [49]. One important feature of LP is that the set of feasible solution $S = \{x \in (\mathbb{R}^+)^n : Ax \leq b\}$ is a convex subset of $\mathbb{R}^n$, i.e., for any solutions $x, y \in S$ the solutions lying on the segment $[x, y]$ are also in S. Thus, LP belongs to the domain of convex optimization, which often concerns polynomial problems, whereas non-convex optimization problems are generally *NP*-Hard. A wide range of optimization problems can be modeled in PL, including minimum cost flow problems such as shortest path, maximum flow and the assignment problem [4]. We refer to [22, 83] for an in-depth introduction to linear programming.

DEFINITION BOX 12: GENERALIZED FLOW PROBLEMS

Network flow problems arise in many fields that usually involve transportation. Be it electricity, vehicles or nutriments, the network is modeled as a directed graph where x_{uv} usually represents the amount of flow moving from a vertex u to a vertex v along the arc (u,v). One elementary rule in such systems is *flow conservation*, i.e., no losses or gains are incurred in the transport of flow through vertices and arcs. Thus, the quantity of flow arriving at a vertex u from its incoming arcs must equal the quantity of flow leaving u using its outgoing arcs and if x_{uv} units of flow are sent into the arc (u,v) from u, exactly x_{uv} units of flow are received into v from the arc (u,v). Generalized flow problems are, appropriately, a generalization of flow problems, where the conservation of flow along the arcs is not assumed. Instead, each arc (u,v) is associated with a multiplier μ_{uv} such that if x_{uv} units of flow enters (u,v) from u, $\mu_{uv} \cdot x_{uv}$ units of flow exit the arc at the vertex v. We recommend [2] for an introduction to network flow problems.

The linear program $(\mathscr{P})$ having f_t as optimum value can be defined as follows. Its decision variables are flow variables $(\phi_a)_{a\in A}$ that represent the quantity of flow entering each arc $a \in A$ and a variable z that represents the total quantity of flow sent to t. The objective of $(\mathscr{P})$ is to maximize the value of the variable z. For each vertex $u \in V$, let δ_u^{out} be the set of arcs leaving u and δ_u^{in} the set of arcs entering u.

$$
(\mathscr{P}) \begin{cases}
\text{maximize } z & & & (4.27)\\
\text{subject to } \displaystyle\sum_{a\in\delta_u^{\text{out}}} \phi_a - \sum_{b\in\delta_u^{\text{in}}} \pi_b \cdot \phi_b \le w_u & u \in V\setminus\{t\} & (4.28)\\
\displaystyle\sum_{a\in\delta_t^{\text{out}}} \phi_a - \sum_{b\in\delta_t^{\text{in}}} \pi_b \cdot \phi_b = w_t - z & & (4.29)\\
\phi_a \ge 0 & a \in A & (4.30)
\end{cases}
$$

Constraints (4.28) require that the total quantity of flow leaving u is at most w_u plus the total quantify of flow entering u, i.e., the quantity of flow available in vertex u. Constraint (4.29) requires that z is equal to the total quantity of flow entering t plus w_t minus the total quantity of flow leaving t. Finally, constraints (4.30) state that each arc carries a non-negative quantity of flow.

Lemma 4.5. *Any optimal solution of $(\mathscr{P})$ is obtained by sending, for every vertex $s \in V\setminus\{t\}$, w_s units of flow from s to t along most reliable st-paths.*

Proof. First notice that the quantity of flow arriving at t when w_s units flow are sent from s to t along an st-path P is w_s times the probability of the path P. Let ϕ' be an optimal solution of $(\mathscr{P})$ maximizing the quantity of flow routed along a path which is

not a most reliable path. Suppose, by contradiction, that ϕ' sends $\epsilon > 0$ units of flow along an st-path P' whose probability is smaller than the probability of a most reliable st-path P. Let ϕ be the flow obtained from ϕ' by decreasing the flow sent on path P' by ϵ and increasing the flow sent on path P by ϵ. Since the probability of P is larger than the probability of P', ϕ sends more flow to t than ϕ', leading to a contradiction with the choice of ϕ'. □

Corollary 4.6. *The optimal value of* $(\mathscr{P})$ *is* $f_t = \sum_{s \in V} w_s \cdot \Pi(s,t)$.

Proof. Since the objective is to maximize z, no flow leaves t in any optimal solution of $(\mathscr{P})$, i.e., $\sum_{a \in \delta_t^{\text{out}}} \phi_a = 0$. Constraint (4.29) ensures that z is the quantity of flow received by t plus w_t. By Lemma 4.5, there exists an optimal solution ϕ such that every vertex s distinct from t send w_s units of flow on a most reliable st-path. Hence, for every vertex s distinct from t, the flow received by t from s is $w_s \cdot \Pi(s,t)$ and thus the value of z is $\sum_{s \in V} w_s \cdot \Pi(s,t)$. □

To compute $PC_{\text{num}}(G)$ we simply have to combine the linear programming formulations of f_t for each $t \in V$. To distinguish the variables from the different linear formulations, we annotate the variables coming from the formulation of f_t. For instance, ϕ_a^t correspond to the flow variable of the arc $a \in A$ in the linear program used to compute f_t. $PC_{\text{num}}(G)$ can then be computed with the following linear program.

$$
(PC_{\text{num}}) \begin{cases}
\text{maximize} & \displaystyle\sum_{t \in V} w_t \cdot f_t & & (4.31) \\
\text{subject to} & \displaystyle\sum_{a \in \delta_u^{\text{out}}} \phi_a^t - \sum_{b \in \delta_u^{\text{in}}} \pi_b \cdot \phi_b^t \leq w_u & t \in V,\ u \in V \setminus \{t\} & (4.32) \\
& \displaystyle\sum_{a \in \delta_t^{\text{out}}} \phi_a^t - \sum_{b \in \delta_t^{\text{in}}} \pi_b \cdot \phi_b^t = w_t - f_t & t \in V & (4.33) \\
& \phi_a^t \geq 0 & t \in V,\ a \in \Phi & (4.34)
\end{cases}
$$

Remark. To obtain the exact value of the PC with this formulation it suffices to multiply the objective function by $\frac{1}{W^2}$, whereas, to obtain the value of the ECA we need to take its square root.

4.4.2 MILP formulation for SAO-BC-PC-Opt

DEFINITION BOX 13: MIXED INTEGER LINEAR PROGRAMMING

Mixed Integer Linear Programming (MILP) is a generalization of Linear Programming (LP) where each variable x_i can also be constrained to lie in the integer domain $x_i \in \mathbb{N}$ instead of the continuous one $x \in \mathbb{R}^+$. While the problem of finding an optimal solution is polynomial for LP, this small modification renders the problem *NP*-Hard for MILP.

With this linear programming definition of PC_{num} we can now present our MILP formulation of SAO-BC-PC-Opt. As a reminder, an instance of SAO-BC-PC-Opt is composed of a landscape graph $G = (V, A, w, \pi)$ where each arc $a \in A$ is also associated with an improved probability π_a^+ and a cost c_a, and a budget β. The objective is then to find the set of arcs fitting into the budget that maximizes the PC. For each arc $a = (u, v) \in A$, we add another arc $a' = (u, v)$ of probability $\pi_{a'} = \pi_a^+$ that can be viewed as an improved copy of the arc a. This improved copy can be used only if the improvement of the arc a is purchased. For each arc $a \in A$, x_a is equal to one if the improvement of the arc a is purchased and zero otherwise. We denote by A' the set of improved copies of the arcs of A. In the following MILP program, δ_u^{out}, δ_u^{in} and the flow variables ϕ are defined with respect to the set of arcs $\Psi = A \cup A'$.

(*SAO-BC-PC-Opt*):

$$
\begin{cases}
\text{maximize} & \displaystyle\sum_{t \in V} w_t \cdot f_t & & (4.35) \\
\text{subject to} & \displaystyle\sum_{a \in \delta_u^{\text{out}}} \phi_a^t - \sum_{b \in \delta_u^{\text{in}}} \pi_b \cdot \phi_b^t \leq w_u & t \in V,\ u \in V \setminus \{t\} & (4.36) \\
& \displaystyle\sum_{a \in \delta_t^{\text{out}}} \phi_a^t - \sum_{b \in \delta_t^{\text{in}}} \pi_b \cdot \phi_b^t = w_t - f_t & t \in V & (4.37) \\
& \phi_{a'}^t \leq M_a \cdot x_a & t \in V,\ a \in A & (4.38) \\
& \displaystyle\sum_{a \in A} c_a \cdot x_a \leq \beta & & (4.39) \\
& x_a \in \{0, 1\} & a \in A & (4.40) \\
& \phi_a^t \geq 0 & t \in V,\ a \in \Psi & (4.41)
\end{cases}
$$

Constraints of (4.36) and (4.37) are simply the constraints of (4.32) and (4.33). For each arc $a \in A$, the big-M constraints (4.38) ensures that if the improvement of the arc a is not purchased, i.e., $x_a = 0$, then the flow on arc a' is null. The constant M_a is an upper bound of the flow that can enter the arc a such that, if the improvement is purchased, i.e., $x_a = 1$, the flow on a' is not restricted by the big-M constraint. We can

simply take $M_a = \sum_{u \in V} w_u$ for every $a \in A$, but we show in the following Section 4.5 how to compute a much better, i.e., smaller, upper bound since these bounds play a critical role in the resolution time of the MILP [12]. The constraint (4.39) ensures that the total cost of the improvements is at most β. This MILP program has $O(|V| \cdot |A|)$ flow variables, $|A|$ binary variables and $O(|V|^2 + |V| \cdot |A|)$ constraints.

Remark. In practical instances, not all arcs can be improved, i.e., for some arcs $a \in A$, $\pi_a^+ = \pi_a$. In this case, we can remove the improved copy a' from Ψ, its associated constraint of (4.38) and the binary variable x_a.

4.4.3 Extension to BC-PC-Opt

Now that we have a MILP formulation for the SAO-BC-PC-Opt problem, we explain how to extend it to the more general BC-PC-Opt problem. As opposed to SAO-BC-PC-Opt, an instance of BC-PC-Opt is characterized by a landscape graph $G = (V, A, w, \pi)$ and a set of options Φ where each option $i \in \Phi$ costs c_i and has the effect of increasing the weight of each vertex $u \in V$ by w_u^i and improving the probability of each arc $a \in A$ to π_a^i. We first focus on the arcs improvements. In the BC-PC-Opt problem, many arcs can be improved by the purchase of a single option while in the SAO-BC-PC-Opt problem each arc can be purchased independently. Since in the SAO-BC-PC-Opt formulation the purchase of an arc $a \in A$ is determined by the value of the binary variable x_a in the associated constraint of (4.38), to couple the arcs improvements associated with an option $i \in \Phi$, it suffices to replace x_a by the binary variable x_i in the corresponding constraints. It remains to address the improvements of the vertices. Let x_i be the binary variable equal to 1 if the option i is purchased and 0 otherwise. Recall that the purchase of i has the effect of increasing the weight of each vertex $u \in V$ by w_u^i. Then, for each $u \in V$, we process all the occurrences of w_u as follows. When w_u appears as an additive constant, we simply replace it by $w_u + w_u^i \cdot x_i$. Note that w_u only appears as a coefficient in the objective function in the form $w_u f_t$. In this case, to avoid a quadratic term, we use a standard McCormick linearization [85]. We replace the product $w_u f_t$ by $w_u f_t + w_u^i \cdot f_t^i$ where f_t^i is a new variable that is equal to f_t if $x_i = 1$ and 0 otherwise. To achieve this values of f_t^i, for all $u \in V$, we add the constraints $f_t^i \leq f_t$ and $f_t^i \leq M_t \cdot x_i$ where M_t is larger than any values of f_t. As it is a maximization program and f_t^i appears with a positive coefficient in the objective function, the first constraint guaranties that $f_t^i = f_t$ if $x_i = 1$ in any optimal solution and the second constraint guaranties that $f_t^i = 0$ if $x_i = 0$.

(BC-PC-Opt):

$$
\begin{cases}
\text{maximize} & \displaystyle\sum_{t\in V}\left(w_t\cdot f_t+\sum_{i\in\Phi}w_t^i\cdot f_t^i\right) & & (4.42)\\
\text{subject to} & \displaystyle\sum_{a\in\delta_u^{\text{out}}}\phi_a^t-\sum_{b\in\delta_u^{\text{in}}}\pi_b\cdot\phi_b^t\le w_u+\sum_{i\in\Phi}w_u^i\cdot x_i & t,u\in V,\ u\in V\setminus\{t\} & (4.43)\\
& \displaystyle\sum_{a\in\delta_t^{\text{out}}}\phi_a^t-\sum_{b\in\delta_t^{\text{in}}}\pi_b\cdot\phi_b^t=w_t+\sum_{i\in\Phi}w_t^i\cdot x_i-f_t & t\in V & (4.44)\\
& f_t^i\le f_t & t\in V,\ i\in\Phi & (4.45)\\
& f_t^i\le M_t\cdot x_i & t\in V,\ i\in\Phi & (4.46)\\
& \phi_{a^i}^t\le M_a^t\cdot x_i & t\in V,\ a\in A,\ i\in\Phi & (4.47)\\
& \displaystyle\sum_{i\in\Phi}c_i\cdot x_i\le\beta & & (4.48)\\
& x_i\in\{0,1\} & i\in\Phi & (4.49)\\
& \phi_a^t\ge 0 & t\in V,\ a\in\Psi & (4.50)
\end{cases}
$$

4.5 Preprocessing the MILP formulation

In this section we discuss how to preprocess the MILP formulation of an instance of the BC-PC-Opt problem in order to reduce the computing time required to solve it. In Section 4.5.1, we first provide methods to reduce the size of the formulation in terms of numbers of variables and constraints and, in Section 4.5.2, we show how to compute tighter upper bounds for the big-M constraints to improve the linear relaxation.

4.5.1 Reducing the size of the graph

The MILP formulation of the BC-PC-Opt problem, later designated as the MILP, can be viewed as a linear formulation that, given a vector $x\in\{0,1\}^\Phi$, where $x_i=1$ if the option $i\in\Phi$ is purchased and $x_i=0$ otherwise, computes the value of the PC of the graph in which the options designated by x are performed. We call $x\in\{0,1\}^\Phi$ a scenario and denote by $G^x=(V,A,w^x,\pi^x)$ the landscape graph obtained by realizing the options induced by the scenario x, where

$$w_u^x=w_u+\sum_{i\in\Phi}w_u^i\cdot x_i\,,\text{for each } u\in V\,, \tag{4.51}$$

and

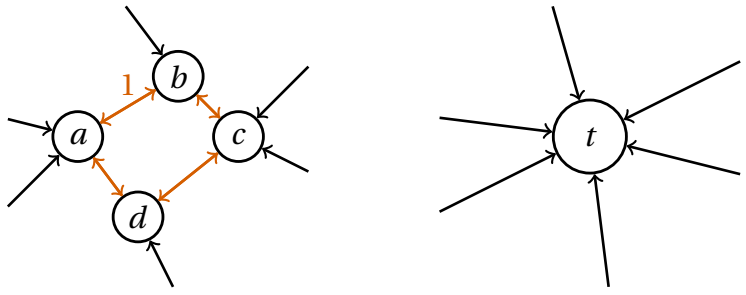
$$\pi_a=\max\left\{\pi_a,\max_{i\in\Phi}\left(\pi_a^i\cdot x_i\right)\right\},\text{for each } a\in A. \tag{4.52}$$

Solving the BC-PC-Opt problem amounts to finding a scenario $y \in \{0,1\}^{\Phi}$ with $\sum_{i\in\Phi} c_i \cdot x_i \leq \beta$ that maximizes $PC(G^y)$. Thus, the goal is to reduce the size of the MILP while keeping the same optimal scenario. One way of doing this is to maintain that, for every scenario x, the objective value of the reduced MILP is $PC(G^x)$. Similarly to the linear formulation of the PC, we decompose $PC(G^x)$ as $\sum_{t\in V} w_t^x \cdot f_t^x$ where $f_t^x = \sum_{s\in v} \Pi^x(s,t) \cdot w_s^x$ is the optimum value of the generalized flow problem to the vertex t, with $\Pi^x(s,t)$ being the probability of the most reliable path from s to t using π_a^x as the probability of each arc a. We respectively denote by $\overline{w}_u = w_u + \sum_{i\in\Phi} w_u^i$ and $\overline{\pi}_a = \max\{\pi_a, \max_{i\in\Phi}(\pi_a^i)\}$ the weight of the vertex u and the probability of the arc a in the scenario where all options are purchased. It is clear that, for every scenario x, $\overline{w}_u$ is an upper bound on the weight of any vertex u, i.e., $\overline{w}_u \geq w_u^x$, and $\overline{\pi}_a$ is an upper bound on the probability of any arc a, i.e., $\overline{\pi}_a \geq \pi_a^x$.

Contract the strongly connected components that are connected with probability 1. In Lemma 3.5 we proved that if two vertices $u, v \in V$ are connected with probability $\Pi(u,v) = \Pi(v,u) = 1$, we can contract them into a vertex t such that $w_t = w_u + w_v$ without changing the value of the PC. It is easy to show by induction that this is also true for any set of vertices T for which any vertices $s, w \in T$ are connected with probability $\Pi(s,w) = \Pi(w,s) = 1$. Indeed, since the flow can pass from one to the other of these vertices without losses, $f_s^x = f_w^x$ for any scenario x. We can then contract the vertices of T in a single vertex t, as illustrated in Figure 4.6, with $w_t = \sum_{u\in T} w_u$ and $w_t^i = \sum_{u\in T} w_u^i$ for each option $i \in \Phi$. In this way

$$\sum_{u\in T} w_u^x \cdot f_u^x = w_t^x \cdot f_T^x, \tag{4.53}$$

where f_T^x is the value of the generalized flow to any of the vertices of T. All the maximal sets of vertices that are connected with probability 1 can easily be identified as the strongly connected components of the graph $G' = (V, A')$ where $A' = \{a \in A : \pi_a = 1\}$ in $O(|V| + |A|)$ time with a depth-first-search.



(a) (b)

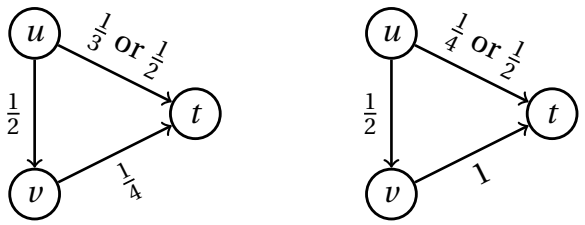
Figure 4.6: Illustration of the contraction of a strongly connected component $T = \{a, b, c, d\}$ into a new vertex t such that $w_t = w_a + w_b + w_c + w_d$.

Remove the computation of $\mathbf{f_t}$ if $\overline{\mathbf{w}}_\mathbf{t} = \mathbf{0}$. Given a vertex t, if its best possible weight is equal to 0, i.e., $\overline{w}_t = 0$, then there is no need to compute f_t since $w_t^x \cdot f_t = 0$ for every scenario x. We can then remove the formulation of the generalized flow problem to t from the MILP.

Remove the uncrossable arcs, i.e., arcs $\mathbf{a \in A}$ with $\overline{\pi}_\mathbf{a} = \mathbf{0}$. The same goes with an arc $a \in A$ whose probability is null in any scenario, i.e., $\overline{\pi}_\mathbf{a} = \mathbf{0}$. Such arc can be omitted in the formulation of the generalized flow problems since any flow passing through it is lost.

Remove the components that are not connected to $\mathbf{t}$ in the formulation of $\mathbf{f_t}$. After removing each arc $a \in A$ whose probability is $\overline{\pi}_\mathbf{a} = \mathbf{0}$, the only flow that can reach a vertex t can only come from vertices from which it exists a path to t. We can identify this set of vertices in $O(|V| + |A|)$ time with a depth-first-search starting at the vertex t in the reversed graph $G' = (V, A')$ where $A' = \{(v, u) : (u, v) \in A\}$.

To further reduce the size of the MILP, we are interested in the arcs of the generalized flow formulation to a particular vertex t. For any scenario x, the optimal solution of the generalized flow to t is obtained by routing w_u units of flow along most reliable ut-paths for each vertex u. However, it is possible that, for any scenario, some arcs (u, v) may never be on a most reliable path from u to t, like in Figure 4.7a, and then never carry flow in an optimal solution. We denote the set of such arcs $\mathbb{W}(t)$. At the opposite, some arcs (u, v) may always be on a most reliable path from u to t and funnel all the flow arriving in u to v, like in Figure 4.7b. We denote the set of such arcs $\mathbb{S}(t)$.



(a) (b)

Figure 4.7: Examples of useless and strong arcs. In (a), the arc (u, v) is t-useless since the best path from u to t passing by (u, v) has a lower probability than the arc (u, t) in every scenario. In (b), the arc (u, v) is t-strong since the path from $(u, v), (v, t)$ has a probability greater or equal to the arc (u, t) in every scenario.

More formally,

$$\mathbb{S}(t) = \{(u, v) \in A : \forall x \in \{0, 1\}^\Phi, \Pi^x(u, t) = \pi_{uv}^x \cdot \Pi^x(v, t)\}, \tag{4.54}$$

and

$$\mathbb{W}(T) = \{(u,v) \in A : \forall x \in \{0,1\}^{\Phi}, \Pi^x(u,t) < \pi^x_{uv} \cdot \Pi^x(v,t)\}. \tag{4.55}$$

One difficulty in the definition of $\mathbb{S}(t)$ and $\mathbb{W}(t)$ is that an option may affect the probability of more than one arc at the same time and that the probability of an arc may be increased differently by several options, in which case only the option with the greater increase matters. In the following part of the preprocessing we relax the notion of scenario and consider that the probability of each arc $a \in A$ can vary in the interval $[\pi_a, \overline{\pi}_a]$ independently of the probabilities of the other arcs. We call $x \in [0,1]^A$ a continuous scenario and define the probability of an arc $a \in A$ under the continuous scenario x as $\pi^x_a = \pi_a + x_a \cdot (\overline{\pi}_a - \pi_a)$. Clearly, the set of probability functions defined by the scenarios of $\{0,1\}^{\Phi}$ is included in the set of probability functions defined by the continuous scenarios $[0,1]^A$, i.e.

$$\{l^x : x \in \{0,1\}^{\Phi}\} \subseteq \{l^x : x \in [0,1]^A\}, \tag{4.56}$$

thus, any property that is verified under all the scenarios of $[0,1]^A$ is also verified for every scenario $x \in \{0,1\}^{\Phi}$. We can then define the following sets

$$S(t) = \{(u,v) \in A : \forall x \in [0,1]^A, \Pi^x(u,t) = \pi^x_{uv} \cdot \Pi^x(v,t)\} \tag{4.57}$$

and

$$W(T) = \{(u,v) \in A : \forall x \in [0,1]^A, \Pi^x(u,t) < \pi^x_{uv} \cdot \Pi^x(v,t)\}, \tag{4.58}$$

such that $S(t) \subseteq \mathbb{S}$ and $W(t) \subseteq \mathbb{W}$.

In Chapter 5, given a graph $G = (V,A)$ where each arc $a \in A$ is associated with a positive interval $[l^-_a, l^+_a]$, and an arc (u,v), we address the problems of identifying the set of vertices $F(u,v)$ (resp. $W(u,v)$) such that for each vertex $t \in F(u,v)$ (resp. $t \in F(u,v)$), (u,v) is always (resp. never) on a shortest path from u to t, for any assignment of the arcs lengths into their intervals. We give $O(|A| + |V|\log|V|)$ time algorithms for solving these problems. Since shortest paths and most reliable paths are isomorphic (see Definition Box 4), we can use the positive interval $[-\log(\overline{\pi}_a), -\log(\pi_a)]$ for each arc $a \in A$ to obtain that $F(u,v)$ (resp. $W(u,v)$) is the set of vertices such that, for each vertex $t \in F(u,v)$ (resp. $t \in W(u,v)$), (u,v) is always (resp. never) on a most reliable path from u to t for any possible assignment of the probability of each arc $a \in A$ between π and $\overline{\pi}_a$. Thus, $S(t) = \{(u,v) \in A : t \in F(u,v)\}$ and $W(t) = \{(u,v) \in A : t \in W(u,v)\}$, and we can use our algorithms to compute the sets $S(t)$ and $W(t)$ for all $t \in V$ with a single run of time complexity $O(|A|^2 + |A||V|\log|V|)$.

Remark. When the graph is planar, which is often the case for landscape graphs, the number of arcs is linear in the number of vertices. Therefore, the time complexity for identifying $S(t)$ and $W(t)$ drops to $O(|V|^2 \log|V|)$ which is tractable for graphs with up to 10^5 vertices.

Remove the arcs of $\mathbb{W}(\mathbf{t})$. Let $(u,v) \in \mathbb{W}(t)$, since for all scenarios $x \in \{0,1\}^{\Phi}$, (u,v) does not belong to any shortest path from u to t, its removal does not affect the

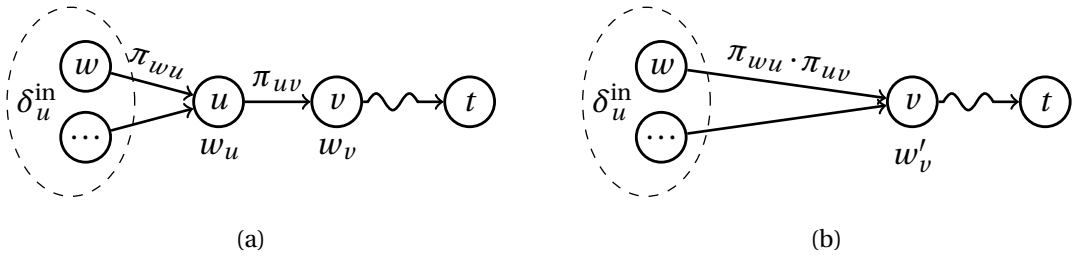
distance from any vertex to t. It is clear, from the definition of PC, that the removal of (u,v) does not affect the contribution of t to PC. Let $f_t^x(G) = \sum_{s\in V} w_s w_t \Pi^x(s,t)$ be the contribution to PC of all the pairs having sink t in the graph G when the probability of connection $\Pi^x(s,t)$ is computed under the scenario x.

Lemma 4.7. *Let $(u,v) \in \mathbb{W}(t)$ and let G' be a graph obtained from G by removing the arc (u,v). Then, for all scenario $x \in \{0,1\}^\Phi$, it holds that $f_t^x(G) = f_t^x(G')$.*

Proof. For every scenario $x \in \{0,1\}^\Phi$, (u,v) does not belong to any shortest path from u to t. Therefore, the removal of (u,v) cannot affect the probability of connection $\Pi(u,t)$ for any vertex t, Thus $f_t^x(G) = f_t^x(G')$. □

Corollary 4.8. *Let $G_t = (V, A')$ be the graph obtained from G by removing all the arcs of $\mathbb{W}(t)$. Then, given a vertex $u \in V$, for any simple ut-path P of G_t whose probability can be higher than $\Pi(u,t)$ in some scenario, i.e., $\prod_{a\in P} \overline{\pi}_a \geq \Pi(u,t)$, it exists a scenario $x \in \{0,1\}^\Phi$ such that P is a most reliable ut-path of G, i.e., $\Pi_G^x(u,t) = \prod_{a\in P} \pi_a^x$.*

Contraction of an arc in $\mathbb{S}(t)$. Now, assume that $(u,v) \in \mathbb{S}(t)$ and $\pi_{uv} = \overline{\pi}_{uv}$. Since (u,v) is on a most reliable path from u to t in any possible scenario, all the flow arriving at u in the generalized flow problem to the vertex t can always be forwarded to v via the arc (u,v). The contraction of (u,v) consists in replacing every arc $(w,u) \in \delta_u^{\text{in}}$ by an arc (w,v) of length $\pi'_{wv} = \pi_{wu} \cdot \pi_{uv}$ and by removing the vertex u and all its outgoing arcs, see Figure 4.8. Let G' be the graph obtained from G by contracting (u,v). The weight of u in G is transferred to the weight of v in G'. Namely, the weight of v in the new graph is $w'_v = w_v + \pi_{uv} \cdot w_u$. The next lemma establishes that the contribution of t to the *PC* in G is equal to its contribution in G'.



(a) (b)

Figure 4.8: (a) A graph G before contraction of an arc (u,v). (b) A graph G' obtained from G by contracting the arc (u,v). The weight w'_v of v in G' is equal to $w_v + \pi_{uv} \cdot w_u$.

Algorithm 2: Contract t-strong arc

Input: $G = (V, A, w, \pi)$, Φ, $(w)_{u\in V}^{i\in\Phi}$, $(\pi)_{a\in A}^{i\in\Phi}$, $(u,v) \in A$
foreach $(w,u) \in \delta_u^{in}$ **do**
  $A \leftarrow A \cup \{(w,v)\}$
  $\pi_{wv} \leftarrow \pi_{wu} \cdot \pi_{uv}$
  foreach $i \in \Phi$ **do**
    $\pi^i_{wv} \leftarrow \pi^i_{wu} \cdot \pi_{uv}$
foreach $i \in \Phi$ **do**
  $w^i_v \leftarrow w^i_v + \pi_{uv} \cdot w^i_u$
$G \leftarrow G[V \setminus \{u\}]$

Lemma 4.9. *Let $(u,v) \in \mathbb{S}(t)$ and let G' be the graph obtained from G by contracting the arc (u,v) and modifying accordingly the weight of w_v. For every scenario $x \in \{0,1\}^\Phi$, $f^x_t(G) = f^x_t(G')$.*

Proof. Let s be a vertex of G and x be a scenario in $\{0,1\}^\Phi$. We denote $f^x_{st}(G)$ the contribution to PC of the pair st with scenario $x \in \{0,1\}^\Phi$ in G. If $s \notin \{u,v\}$, it is easy to check that, for any $x \in \{0,1\}^\Phi$ the length of the shortest path from s to t in G is equal to the length of the shortest path from s to t in G'. Moreover, the weight of s is the same in G and in G'. Therefore, by the definition of PC, $f^x_{st}(G) = f^x_{st}(G')$, for any $x \in \{0,1\}^\Phi$. On the other hand, the contributions of the pairs ut and vt in G sum to the contribution of v in G'. Indeed,

$$\begin{aligned}
f^x_{ut}(G) + f^x_{vt}(G) &= \Pi^x(u,t) \cdot w_u + \Pi^x(v,t) \cdot w_v && (4.59)\\
&= \pi_{uv} \cdot \Pi^x(v,t) \cdot w_u + \Pi^x(v,t) \cdot w_v && (4.60)\\
&= \Pi^x(v,t) \cdot (\pi_{uv} \cdot w_u + w_v) && (4.61)\\
&= \Pi^x(v,t) \cdot w'_v && (4.62)\\
&= f^x_{vt}(G'). && (4.63)
\end{aligned}$$

We conclude that, for any $x \in \{0,1\}^\Phi$,

$$\begin{aligned}
f^x_t(G) &= \sum_{s\in V} f^x_{st}(G) && (4.64)\\
&= f^x_{ut}(G) + f^x_{vt}(G) + \sum_{s\in V\setminus\{u,v\}} f^x_{st}(G) && (4.65)\\
&= f^x_{vt}(G') + \sum_{s\in V\setminus\{u,v\}} f^x_{st}(G') && (4.66)\\
&= f^x_t(G'). && (4.67)
\end{aligned}$$

□

Remove the vertices that can't carry flow. Let G' be the graph obtained removing all the arcs of $\mathbb{W}(t)$. By Corollary 4.8, a vertex u can carry flow in an optimal

solution of the generalized flow problem to t only if it is on a simple path from a vertex v with $w_v > 0$. Let $T = \{u \in V : w_u > 0\}$ denote the set of vertices with positive weights. Thus, u can carry flow in an optimal solution only if $w_u \in T$ or if it exists a vertex $s \in T$ such that u is on an st-path of G'. Let R be the set of vertices not fulfilling these conditions, all the vertices of R can be removed from the generalized flow formulation without affecting its optimal value. Furthermore, R can be computed with simple graph traversal algorithms in $O(|V| + |A|)$ time. For this, it is only necessary to run a traversal algorithm (either Breadth First Search of Depth First Search) from each of the nodes of T, all in a row. At each traversal, the set K of the visited vertices increases, and we repeat the operation with a vertex $s \in (T \setminus K)$ until $T \subseteq K$. Then R is exactly the set of vertices that have not been visited after all these traversals, i.e., $R = V \setminus K$.

4.5.2 Improving linear relaxation bounds

DEFINITION BOX 14: BRANCH AND BOUND ALGORITHMS

Branch-and-bound (BnB) is an exact method for solving hard optimization problems whose space of feasible solutions is non-convex. Although it was quickly extended to other domains, this method was originally developed to tackle mixed integer linear programs [76] and is still the basis for modern MILP solvers. Generally speaking, BnB consists in recursively enumerating all the feasible solutions and use a pruning strategy to avoid the enumeration of solutions that cannot be optimal. In the worst case, this method therefore takes an exponential amount of time if the pruning strategy is not able to prevent the exploration of non-optimal solutions, but it is often very efficient in practice. Let P be a MILP maximization problem on n variables. For the sake of simplicity, we assume that all the variables of P are constrained to be integers between 0 and 1.

$$(\mathrm{P}) \begin{cases} \text{maximize} & c^\intercal x & (4.68) \\ \text{subject to} & Ax \leq b & (4.69) \\ & x \in \{0,1\}^n & (4.70) \end{cases}$$

Since $\{0,1\}^n \subseteq [0,1]^n$, the optimum value of P can be bounded as follows

$$\max_{\substack{x \in \{0,1\}^n \\ Ax \leq b}} c^\intercal x \leq \max_{\substack{x \in [0,1]^n \\ Ax \leq b}} c^\intercal x, \tag{4.71}$$

where $\max_{x \in [0,1]^n : Ax \leq b} c^\intercal x$ can be computed in polynomial time since it is the optimum value of the linear program obtained by relaxing the integrity constraints of the variables of P.

A basic branch and bound algorithm for solving P can be stated as follows

Algorithm 3: Branch and Bound method for P

Input : $c \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$
Output: s^* such that $As^* \leq b$ and $c^\intercal s^*$ is maximum
$s \leftarrow \{0\}^n$
$s^* \leftarrow \{0\}^n$
Procedure `recurse(`k`)`
  if $Q_{s,k}$ *is unfeasible* **then**
    return
  end
  if $i < n$ **then**
    Let y be an optimal solution for $Q_{s,k}$
    if $c^\intercal y < c^\intercal s^*$ **then**
      return
    end
    $s_k \leftarrow 1$; `recurse(`$k+1$`)`
    $s_k \leftarrow 0$; `recurse(`$k+1$`)`
  else
    if $c^\intercal s > c^\intercal s^*$ **then**
      $s* \leftarrow s$
    end
  end
`recurse(0)`
return s^*

where s represents the solution being recursively constructed by the procedure $recurse$, s^* is the best know solution at any point of the algorithm and $Q_{s,k}$ is the linear program obtain by fixing the k first variables of P to their corresponding values in s, i.e., adding the constraint $x_i = s_i$ for each $i \in \{1, \ldots, k\}$. Like with (4.71), the optimum value of $Q_{s,i}$, $c^\intercal y$, can then be interpreted as an upper bound on the value of any feasible solution sharing its i first values with s. Thus, if $c^\intercal y < c^\intercal s^*$ then s^* is a better solution than any solution based on the i first values of s and there is no need to enumerate more of these solutions. It is clear that if the inequality $c^\intercal y < c^\intercal s^*$ is always false, this algorithm results in the enumeration of all feasible solutions, i.e., every $x \in \{0,1\}^n$ such that $Ax \leq b$. This small example illustrates why the efficiency of branch and bound algorithms heavily depends on the fact that the optimum value of the relaxed problem is as small as possible, or, equivalently, that the optimal values of the original problem and of its relaxed version are as close as possible.

As pointed out in Definition Box 3, the efficiency of the methods for solving a mixed integer linear program (MILP) relies on the fact that the optimum value of its linear relaxation is as close as possible to its own optimum value. In our process of formalizing the BC-PC-Opt problem into a MILP, we regularly use big-M constraints

(i.e., McCormick linearizations [85]), in order to get rid of quadratic terms. This method of linearizing quadratic terms is known to degrade the quality of the linear relaxation, i.e., increase the gap between the values of optimal integer solutions and the relaxed ones. The common way of mitigating this effect is to choose the smallest constant M possible for every big-M constraint [12]. Indeed, the role of a big-M constraint is to prevent the value of a variable x from being greater than 0 if the corresponding integer variable y has not been set to 1, i.e.,

$$x \leq M \cdot y, \tag{4.72}$$

with M being greater that any possible value of x. While this inequality suffices to enforce that $x = 0$ if $y = 0$ when $y \in \{0,1\}$, it is not in the relaxed problem where $y \in [0,1]$. Indeed, if M is too big with respect to the possible values of x, e.g., 100 times bigger than the maximum value of x, then even a small positive value for y, e.g., $y = 0.01$, is enough to bypass the constraint and allow x to take any possible value. Thus the goal is to chose the smallest possible upper bound M on the values of x.

In our formulation, big-M constants appear in the constraints (4.46) and (4.47). Given a vertex $t \in V$ and an option $i \in \Phi$, constraints (4.46) in conjonction with constraints (4.45) enforce the variable f_t^i to be equal to 0 if $y_i = 0$ and take a value between 0 and f_t if $y_i = 1$. Here M must then be an upper bound on the values of f_t, i.e., the maximum value of the flow that can arrive in the vertex t in it's generalized flow formulation for any scenario. Similarly, given a vertex $t \in V$, an arc $a = (u,v) \in A$ and an option $i \in \Phi$, constraints (4.47) with constraints of (4.43) enforce that $\phi_a^i = 0$ if $y_i = 0$, i.e., no flow pass through the arc a^i if the option i is not purchased, and, if $y_i = 1$, ϕ_a^i can take any value between 0 and the maximum flow that can arrive at u.

Thus, in both cases the constant M corresponds to the maximum value of the flow that can arrive to a particular vertex u in the generalized flow formulation to a vertex t. The exact upper bound on the value of this flow in any optimal solution can then be expressed as

$$\mathscr{M}_u^t = \max_{x \in \{0,1\}^\Phi} \left(\sum_{s \in \mathscr{I}_{t,u}^x} w_s^x \cdot \Pi^x(s,u) \right), \tag{4.73}$$

where $\mathscr{I}_{t,u}^x$ is the set of vertices s for which it exists a most reliable st-path passing by the vertex u under the scenario x, i.e., $\mathscr{I}_{t,u}^x = \{s \in V : \Pi^x(s,t) = \Pi^x(s,u) \cdot \Pi^x(u,t)\}$. This bound is not trivial to compute as the algorithm that consists in enumerating all the scenarios of $\{0,1\}^\Phi$ is obviously not tractable. Moreover, M_u^t should be computed for each vertex $u \in V$ of each generalized flow to a vertex $t \in V$, i.e., $O(|V|^2)$ times. Since for any pair of vertices $s, t \in V$ and any scenario $x \in \{0,1\}^\Phi$, $\Pi(s,t) \leq \Pi^x(s,t) \leq \Pi^+(s,t)$, we can define the set

$$I_{t,u} = \{s \in V : \Pi(s,t) \leq \Pi^+(s,u) \cdot \Pi^+(u,t)\} \tag{4.74}$$

such that, for any scenario $x \in \{0,1\}^\Phi$, $\mathscr{I}_{t,u}^x \subseteq I_{u,t}$. Thus, we can easily calculate an

upper bound for $\mathscr{M}_u^t$ in the graph G as

$$M_{u,G}^t = \sum_{s \in I_{u,t}} w_s^+ \cdot \Pi_G^+(s,u)\,, \tag{4.75}$$

which can be accomplished in $O(|E|+|V|\log|v|)$ time by running the Dijkstra algorithm from the vertex u on the reversed graph.

Since the graph G_t is obtained from G by deleting all the arcs of $W(t)$, it contains less arcs than G while preserving all the most reliable paths from each vertex $u \in V$ to the vertex t in any scenario. By the fact that $\Pi_{G_t}^+(s,u) \leq \Pi_G^+(s,u)$, and by Lemma 4.7, we have

$$\mathscr{M}_u^t \leq M_{u,G_t}^t \leq M_{u,G}^t\,. \tag{4.76}$$

Therefore, M_{u,G_t}^t is an upper for the maximum flow that can enter the vertex u that is both easy to compute and benefits from the removal of the arcs of $W(t)$.

Remark. In the case where each improvement option correspond to exactly one arc (like for SAO-BC-PC-Opt), we conjecture that M_{u,G_t}^t is the best possible upper bound in the sense that $\mathscr{M}_u^t = M_{u,G_t}^t$, or, equivalently, that it exists a scenario $x \in \{0,1\}^A$ where the quantity of flow reaching u is M_{u,G_t}^t.

5 A Preprocessing algorithm for shortest paths problems with interval data

Table of contents

5.1 Introduction and state of the art

The *Shortest Path Problem* (SPP) is a fundamental and very well studied combinatorial optimization problem. Formally, given a directed graph $G = (V, A)$ where each arc $a \in A$ has a positive length l_a, and two vertices $s, t \in V$, it consists in finding an st-path P that minimizes $\sum_{a \in P} l_a$, i.e., a set of consecutive arcs linking s to t whose sum of the lengths is minimum. This problem arises in numerous applications that are not limited to the fields of telecommunications and transportation with, for example, landscape ecology. It is long known that this problem can be very efficiently solved with the Dijkstra algorithm [35]. However, the SPP can be encountered as a subroutine for larger and more difficult problems like network designing or robust shortest path. In these problems, the length of each arc $a \in A$ is either subject to change or uncertain and is only known to lie in a positive interval $[l_a^-, l_a^+]$. Like in this thesis, these kinds of optimization problems are often solved using *Mixed Integer Linear Programming* (MILP). Identifying whether an arc a is never or always part of a shortest path from a vertex s to a vertex t for any assignment of the arcs lengths into their intervals then becomes interesting. Indeed, this information can be used to reduce the search space by definitely setting the arc a as being part of the shortest st-path or not.

In [71], the authors study the *Robust Deviation Shortest Path Problem* (RDSPP) using MILP. In order to solve their MILP more efficiently, they propose a preprocessing step that requires identifying if an arc a can be part of a shortest st-path for some assignment of the arcs lengths into their intervals. Unfortunately they showed that

identifying such arcs is NP-Complete in the general case, but still give a polynomial algorithm when the graph is restricted to have very particular properties, i.e., when it is a directed acyclic, layered graph with a small width. We are interested in a special case of this problem where $s = u$, i.e., our problem is to identify whether an arc (u, v) is never or always on a shortest path from u to a given vertex t.

Given a graph $G = (V, A)$ where each arc $a \in A$ is associated with a positive interval of lengths $[l_a^-, l_a^+]$:
We say that an arc (u, v) is **t-strong** if it is **always** on a shortest path from u to t for any assignment of the lengths of the arcs into their intervals.
We say that an arc (u, v) is **t-useless** if it is **never** on a shortest path from u to t for any assignment of the lengths of the arcs into their intervals.

The problem of identifying if an arc (u, v) is t-strong or t-useless has also received attention in the context of robust shortest paths problems. In [21], given a vertex $n \in V$, the authors investigate the identification of n-strong and n-useless arcs, which they respectively call T-1-persistent and T-0-persistent arcs, in order to preprocess their MILP formulations for the *Minimax regret Single-Source Shortest Path Problem* (MSS-SPP). The MSS-SPP is the single-source variant of the RDSPP [133]. They provide sufficient (but not necessary) conditions for an arc (u, v) to be t-strong or t-useless and use these conditions to design $O(|A| + |V| \log |V|)$ time algorithms that, given an arc (u, v) and a vertex t, can in most cases identify if (u, v) is t-strong or t-useless. More recently, the authors of [79] also studiedin the problem of identifying t-useless arcs for a given t in order to reduce the size of MILP formulations embedding shortest path computations. They improve the result of [21] by providing a sufficient and necessary condition for an arc (u, v) to be t-useless. In their article, they call t-weak an arc that is not t-useless, i.e., that is on a shortest path from u to t for some assignment of the arcs lengths in their interval, and provide an $O(|V| \cdot (|A| + |V| \log |V|))$ time algorithm that allows to identify if an arc (u, v) is t-useless (non-t-weak in their terms) for a given vertex t.

In Section 5.2, we give a more formal definition of the notion of t-strongness. Then, in Section 5.3, given an arc (u, v) of G, we show how to adapt the Dijkstra's algorithm to compute in $O(|A| + |V| \log |V|)$ time the set of all vertices $t \in V$ such that (u, v) is t-strong. This improves the results of [21] by providing a necessary and sufficient condition for (u, v) to be t-strong, i.e., we can identify all the t-strong arcs while the algorithm of [21] identifies only a subset of it. This also reduces the time complexity for computing the sets of t-strong arcs for all t by a factor $|V|$ as we only need to run the algorithm on each arc instead of each pair of arc and vertex. In Section 5.4, we show how to modify our algorithm to compute, with the same complexity the set of all vertices $t \in V$ such that (u, v) is t-useless. This also improves the results of [79] since our algorithm has a time complexity that is smaller by a factor $|V|$. Finally, in Section 5.5, we show preliminary results on how to extend the notions of t-strongness and t-uselessness to constrained shortest paths.

5.2 Definitions

Given a graph $G = (V, A)$ where each arc $a \in A$ is associated with a positive interval of lengths $[l_a^-, l_a^+]$, we call a *continuous scenario* a vector $[0,1]^A$ that associate to each arc $a \in A$ a coefficient x_a such that the length of the arc a under the scenario x is

$$l_a^x = l_a^+ - x_a \cdot (l_a^+ - l_a^-). \tag{5.1}$$

We define $d^x(s,t)$ to be the length of a shortest path, i.e., the distance, from s to t under the scenario x:

$$d^x(s,t) = \min_{P\,:\,st\text{-path}} l^x(P), \tag{5.2}$$

where $l^x(P) = \sum_{a \in P} l_a^x$ is the length of the path P under the scenario x.

Lemma 5.1. *Given two vertices $s, t \in V$ and an st-path P, it exists a continuous scenario x such that P is a shortest st-path if and only if P is a shortest path in the scenario q where $q_a = 1$ if $a \in P$ and $q_a = 0$ otherwise.*

Proof. Since q is a particular case of continuous scenario, we only need to prove that if P is a shortest under a scenario x it remains a shortest path under the scenario q. The lengths of the arcs of P are guaranteed to be smaller or equal in the scenario q than in the scenario x, since $q_a \geq x_a$ for each arc $a \in P$, whereas the others arcs lengths are guaranteed to be greater or equal since $q_a \leq x_a$ for each arc $a \in V \setminus P$. Thus, among all the st-paths, P is the path whose length is the most reduced by passing from the scenario x to the scenario q. □

Corollary 5.2. *To prove that an arc (u,v) is t-strong or t-useless, we only need to check the property for the $2^{|A|}$ scenarios of $\{0,1\}^A$.*

To state our following results, we introduce the notion of fiber.

The *fiber* of an arc $(u,v) \in A$ under the scenario $x \in \{0,1\}^A$, that we denote $F^x(u,v)$, is the set of vertices such that (u,v) belongs to a shortest path from u to t for each $t \in F^x(u,v)$ when arc lengths are set according to x, i.e.

$$F^x(u,v) = \{t \in V : d^x(u,t) = l_{uv}^x + d^x(v,t)\}. \tag{5.3}$$

Let $F(u,v)$ be the intersection of the fibers of the arc (u,v) over all possible scenarios $x \in \{0,1\}^A$, i.e.

$$F(u,v) = \bigcap_{x \in \{0,1\}^A} F^x(u,v). \tag{5.4}$$

By definition and by Lemma 5.1, an arc (u,v) is t-strong if and only if t belongs to $F^x(u,v)$ for every scenario $x \in \{0,1\}^A$, i.e., if t belongs to $F(u,v)$.

5.3 Computing $F(u,v)$

Let $y \in \{0,1\}^A$ be the following scenario :

$$y_{wt} = \begin{cases} 0 & w \in F(u,v) \textbf{ or } (w,t) = (u,v) \\ 1 & \text{otherwise} \end{cases} \tag{5.5}$$

Lemma 5.3. *The intersection $F(u,v)$ of the fibers of $(u,v) \in A$ over all possible scenarios is the fiber of (u,v) under the scenario y, i.e., $F(u,v) = F^y(u,v)$.*

Proof. Since, by definition, $F(u,v) \subseteq F^y(u,v)$, it remains to prove that $F^y(u,v) \subseteq F^x(u,v)$ for any scenario $x \in \{0,1\}^A$. By way of contradiction, let $t \in F_y(u,v)$ be a vertex at minimum distance from u in the scenario y such that it exists a scenario x with $t \notin F^x(u,v)$. Let P be a shortest ut-path containing (u,v) under the scenario y. For every vertex w of P, $d^y(u,w) \leq d^y(u,t)$ and $w \in F^y(u,v)$. Hence, by the choice of t, w belongs to $F^x(u,v)$ for every scenario x. Therefore, w belongs to $F(u,v)$ and the length of every arc of P is set to its upper bound in the scenario y, i.e., $l^y(P) = l^+(P)$. Now, let Q be a shortest ut-path in the scenario x. If the path Q contains a vertex $z \in F(u,v)$ then $d^x(u,t) = d^x(u,z) + d^x(z,t) = l^x_{uv} + d^x(v,z) + d^x(z,t) = l^x_{uv} + d^x(v,t)$, a contradiction with $t \notin F^x(u,v)$. Therefore, the vertices of Q do not belong to $F(u,v)$ and thus all the arcs of Q are set to their lower bound in y, i.e., $l^y(Q) = l^-(Q)$. Since P is a shortest path in y, $l^y(P) \leq l^y(Q)$. We deduce that $l^x(P) \leq l^+(P) = l^y(P) \leq l^y(Q) = l^-(Q) \leq l^x(Q)$, which contradicts $t \notin F^x(u,v)$. □

We describe a $O(|A| + |V|\log|V|)$ time algorithm that, given an arc (u,v), computes simultaneously the scenario y defined by (5.5) and the fiber $F^y(u,v)$ of arc (u,v) with respect to this scenario. The algorithm is an adaptation of the Dijkstra's shortest path algorithm that assigns colors to vertices. We prove that it colors a vertex w in blue if w belongs to $F^y(u,v)$ and in red otherwise. At each step, we consider a subset of vertices $S \subseteq V$ whose colors have been already computed. Before the first iteration, $S = \{u\}$ and the length of every arc leaving u is set to its lower bound except the length of (u,v) which is set to its upper bound according to scenario y. At each step, since the color of every vertex of S is known, the length, under scenario y, of every arc (w,t) with $w \in S$ is also known. Therefore, it is possible to find a vertex $t \in V \setminus S$ at minimum distance from u, under scenario y, in the subgraph $G[S \cup \{t\}]$ induced by $S \cup \{t\}$. Following Dijkstra's algorithm analysis, we know that the distance under scenario y from u to t in $G[S \cup \{t\}]$ is in fact the distance between u and t in G. This allows us to determine if there exists a shortest path from u to t in the scenario y passing via (u,v) and to color the vertex t accordingly. We end-up the iteration with a new vertex t whose color is known and that can be added to S before starting the next iteration. The algorithm terminates when $S = V$.

Algorithm 4: Computes the set S of vertices t such that (u,v) is t-strong

Input : $G=(V,A)$, $[l_a^-, l_a^+]_{a\in A}$, $(u,v)\in A$
Output: $F(u,v)=\{t\in V : (u,v) \text{ is } t\text{-strong}\}$
$d(u)\leftarrow 0$
$\gamma(u)\leftarrow red$
foreach $(u,w)\in\delta_u^{out}\setminus\{(u,v)\}$ **do**
  $d(w)\leftarrow l_{uw}^-$
  $\gamma(w)\leftarrow red$
$d(v)\leftarrow l_{uv}^+$
$\gamma(v)\leftarrow blue$
$S\leftarrow\{u\}$
while $S\neq V$ **do**
  Pick $t\in V-S$ with smallest $d(t)$ breaking tie by choosing a vertex t such that $\gamma(t)=blue$ if it exists
  if $\gamma(t)$ *is* $blue$ **then**
    foreach $(t,w)\in\delta_t^{out}$ **such that** $d(t)+l_{tw}^+\leq d(w)$ **do**
      $d(w)\leftarrow d(t)+l_{tw}^+$
      $\gamma(w)\leftarrow blue$
  else
    foreach $(t,w)\in\delta_t^{out}$ **such that** $d(t)+l_{tw}^-< d(w)$ **do**
      $d(w)\leftarrow d(t)+l_{tw}^-$
      $\gamma(w)\leftarrow red$
  $S\leftarrow S\cup\{t\}$
return $\{t\in V:\gamma(t)=blue\}$

For every vertex $w\in V\setminus S$, the estimated distance $d(w)$ is the length of a shortest uw-path under the scenario y in the subgraph $G[S\cup\{w\}]$. An arc a is blue if $a=uv$ or if its origin is blue, the other arcs are red, i.e., a is blue if $y_a=0$ and red if $y_a=1$. The estimated color $\gamma(w)$ of w is blue if there exists a blue uw-path of length $d(w)$ in $G[S\cup\{w\}]$ and red otherwise. The correctness of Algorithm 4 follows from the following Lemma.

Lemma 5.4. *For every vertex $t\in S$, $\gamma(t)$ is blue if and only if $t\in F^y(u,v)$.*

Proof. We proceed by induction on the number of vertices of S. When $S=\{u\}$, the property is verified. Now, suppose the property true before the insertion in S of the vertex t such that $d(t)$ is minimum. By induction hypothesis, the lengths of arcs having their source in S are set according to y. Therefore, following Dijkstra's algorithm analysis, we deduce that $d(t)$ is the length of the shortest path from u to t in the graph G under scenario y. If t has been colored blue then there exists a blue vertex $w\in S$ such that $d(t)=d(w)+l_{wt}^+$. By induction hypothesis $w\in F^y(u,v)$ and there exists a shortest ut-path under scenario y containing (u,v), i.e., $t\in F^y(u,v)$. Now, suppose that t has been colored in red. By contradiction, assume there exists a shortest ut-path under scenario y that contains (u,v). This path cannot contain a vertex outside S except t because otherwise, since arc lengths are non-negative, its length according

to y would be greater than $d(t)$ by the choice of t. Therefore, the predecessor w of t in this path belongs to S. Since w belongs to a shortest ut-path passing via (u,v), by induction, it was colored blue. But in this case, there exists a blue vertex w such that $d(w) + l^+_{wt} = d(t)$, and t was colored blue as well, a contradiction. □

We are now ready to state the main result of this section.

Proposition 5.5. *Given a graph $G = (V, A)$ where each arc $a \in A$ is associated with a positive interval of lengths $[l^-_a, l^+_a]$, and an arc (u,v), Algorithm 4 computes in $O(|A| + |V|\log|V|)$ time the set of vertex t such that (u,v) is t-strong.*

Proof. The Lemma 5.3 shows that given an arc (u,v) the set of vertices S such that $t \in S$ if and only if (u,v) is t-strong is a fiber for a specific scenario. The Lemma 5.4 shows that the algorithm compute this fiber. Therefore, the Algorithm 4 is correct. Analogously to the Dijkstra's algorithm, Algorithm 4 can be implemented to run in $O(|A| + |V|\log|V|)$ time by using a Fibonacci heap as priority queue. □

5.4 Extension to the computation of t-useless arcs

An adaptation of Algorithm 4 can compute, given an arc (u,v) the set of vertices W such that $t \in W$ if and only if (u,v) is t-usless. Before describing this adaptation, we explain how the two problems are related. For that, we first introduced a strengthening of the notion of t-strongness. We will say that an arc (u,v) is strictly t-strong if it belongs to all shortest ut-paths for every scenario $x \in \{0,1\}^A$. Recall that t-strongness requires only the existence of a shortest ut-path passing via (u,v) for every scenario $x \in \{0,1\}^A$. Algorithm 4 can be easily adapted to compute for every arc (u,v) the set of vertex t such that (u,v) is strictly t-strong. It suffices to change the way the algorithm breaks the tie between a red path and a blue path when it chooses the next vertex t to visit and when it updates the colors of the neighbors of t. Namely, in the first internal loop, the condition for coloring w in blue becomes $d(w) > d(t) + l^+_{tw}$ while the condition for coloring w in red in the second internal loop becomes $d(w) \geq d(t) + l^-_{tw}$. Moreover, when we choose t such that $d(t)$ is minimal, we break tie by choosing a red vertex if it exists. Clearly, these small changes exclude the existence of a red path of length $d(w)$ between u and a blue vertex w. Therefore, (u,v) belongs to every path of length $d(w)$ and (u,v) is strictly w-strong whenever w is blue. We call the resulting algorithm the strict version of Algorithm 4. The next step is to extend the notion of strict strongness to a subset of arcs having the same source. For any vertex $u \in V$, a subset $\delta \subseteq \delta^{\text{out}}_u$ of arcs is strictly t-strong if, for every scenario $x \in \{0,1\}^A$, all shortest ut-paths intersect δ. By definition, an arc (u,v) is t-useless if and only if $\delta^{\text{out}}_u \setminus \{(u,v)\}$ is strictly t-strong. Indeed, every ut-path avoiding (u,v) intersects $\delta^{\text{out}}_u \setminus \{(u,v)\}$ and, conversely, (u,v) belongs to every ut-path avoiding $\delta^{\text{out}}_u \setminus \{(u,v)\}$. Hence, computing the set of vertex t such that (u,v) is t-useless amounts to compute the set of vertex t such that $\delta^{\text{out}}_u \setminus \{(u,v)\}$ is strictly t-strong. An algorithm that computes this set of

vertices can be obtained from the strict version of Algorithm 4 by modifying only the initialization step: arc (u,v) is colored in red and its length is set to l^-_{uv} while arcs of $\delta^{\text{out}}_u \setminus \{(u,v)\}$ are colored in blue and their lengths are set to their upper bound. A correctness proof very similar to the one of Algorithm 4 (and that we will not repeat) shows that a vertex is colored blue by Algorithm 5 if and only if (u,v) is t-useless. Since the two algorithms have clearly the same time complexity, we conclude this section with the following result.

Algorithm 5: Computes the set of vertex t such that (u,v) is t-useless

Input : $G=(V,A)$, $[l^-_a, l^+_a]_{a\in A}$, $(u,v)\in A$
Output: $\{t \in V : (u,v)$ is t-useless$\}$
$d(u) \leftarrow 0$
$\gamma(u) \leftarrow blue$
foreach $(u,w) \in \delta^{out}_u \setminus \{(u,v)\}$ **do**
  $d(w) \leftarrow l^+_{uw}$
  $\gamma(w) \leftarrow blue$
$d(v) \leftarrow l^-_{uv}$
$\gamma(v) \leftarrow red$
$S \leftarrow \{u\}$
while $S \neq V$ **do**
  Pick $t \in V - S$ with smallest $d(t)$ breaking tie by choosing a vertex t such that $\gamma(t) = red$ if it exists
  if $\gamma(t)$ *is* $blue$ **then**
    foreach $(t,w) \in \delta^{out}_t$ **such that** $d(t) + l^+_{tw} < d(w)$ **do**
      $d(w) \leftarrow d(t) + l^+_{tw}$
      $\gamma(w) \leftarrow blue$
  else
    foreach $(t,w) \in \delta^{out}_t$ **such that** $d(t) + l^-_{tw} \leq d(w)$ **do**
      $d(w) \leftarrow d(t) + l^-_{tw}$
      $\gamma(w) \leftarrow red$
  $S \leftarrow S \cup \{t\}$
return $\{t \in V : \gamma(t) = blue\}$

Proposition 5.6. *Given a graph $G=(V,A)$ where each arc $a \in A$ is associated with a positive interval of lengths $[l^-_a, l^+_a]$, and an arc (u,v), Algorithm 5 computes in $O(|A| + |V|\log|V|)$ time the set of vertex t such that (u,v) is t-useless.*

Remark. For all the presented algorithms, if at the beginning of an iteration, all the unvisited vertices $V \setminus S$ have the same estimated color, i.e., they are all red or all blue, then we can immediately return since these colors will not change throughout the rest of the execution.

5.5 Extension to constrained shortest path problems

In this section, we present preliminary results about the extension of the notion of t-strongness and t-uselessness, and our recognition algorithms, to constrained shortest paths. Indeed, in practice, not all scenarios can be encountered, for example, in our problem BC-PC-Opt or even in some robust shortest path problems [100], a budget constraint restricts the number of arcs that can be at their lower bound length. In these cases, being t-strong is a stronger requirement than necessary for an arc (u, v) to be always on a shortest ut-path.

DEFINITION BOX 15: CONSTRAINED SHORTEST PATH AND LAGRAGIAN RELAXATION

The *(budget) Constrained Shortest Path Problem* (CSPP) [69] is a generalization of the Shortest Path Problem (SPP) where the shortest path must also satisfy a budget constraint. Given a directed graph $G = (V, A)$ where each arc $a \in A$ is associated with a length $l_a \geq 0$ and a cost $c_a \geq 0$, two vertices $s, t \in V$ and a budget β, the CSPP is to find a st-path P whose cost $c(p) = \sum_{a \in P} c_a$ is at most β and whose length $l(P) = \sum_{a \in P} l_a$ is minimum. Interestingly, the only addition of this constraint renders the problem NP-Hard [60] while the SPP can be solved very efficiently. Constrained shortest paths arise in a variety of network problems such as mission planning but also as a subproblem for many operational research methods like column generation [67].
A common approach for solving the CSPP is to explore the search space, i.e., the set of all st-paths, and use the Lagrangian relaxation technique [60, 3] to find lower bounds and prevent testing paths that cannot possibly be the optimal solution. This is very similar to the Branch and Bound approach used to solve MILP. The principle behind Lagrangian relaxations is to include the additional constraints into the objective function in the form of penalties. Here the idea is to penalize expensive arcs by defining a length function that account for their cost:

$$l_a^\mu = l_a + \mu \cdot c_a, \tag{5.6}$$

where μ is a constant called *Lagrange multiplier*. The value of μ is either chosen arbitrarily or by using heuristics that depends on the graph [3]. The shortest path distance from s to t using l_a^μ as the length of each arc a,

$$d^\mu(s,t) = \min_{P\,:\,st\text{-path}} \left(\sum_{a \in P} l_a^\mu \right) = \min_{P\,:\,st\text{-path}} \left(l(P) + \mu \cdot c(P) \right), \tag{5.7}$$

can then be used to derive a lower bound on the length of an optimal path P^*,

$$l(P^*) = \min_{\substack{P : st\text{-path} \\ c(P) \leq \beta}} l(P). \tag{5.8}$$

Indeed, the following inequalities hold:

$$\min_{P : st\text{-path}} \big(l(P) + \mu \cdot c(P)\big) \leq \min_{\substack{P : st\text{-path} \\ c(P) \leq \beta}} \big(l(P) + \mu c(P)\big) \leq l(P^*) + \mu \cdot \beta. \tag{5.9}$$

The first inequality of (5.9) is due to the budget constraint, since the minimum length among a subset of the st-paths can only be greater or equal than those for all the st-paths. The second inequality of (5.9) is obtained from (5.8) and the fact that $\mu \cdot c(P) \leq \mu \cdot \beta$ for all st-paths P with $c(P) \leq \beta$, since $\mu \cdot \beta$ is a constant term, we can move it outside the minimum function. As a consequence, $\big(d^{\mu}(s,t) - \mu \cdot \beta\big)$ is a lower bound on the length of any optimal solution path $l(P^*)$. Since this lower bound depends on the value of the Lagrange multiplier μ, it is often interesting to test many values for μ and retain the highest lower bound obtained.

We extend the notion of t-strongness to constrained shortest paths as follows. Given a graph $G = (V, A)$ where each arc a is associated with a length l_a^+ that can be improved to $l_a^- \leq l_a^+$ by paying a cost c_a, we call scenario a vector $x \in \{0,1\}^A$ such that the length of the arc a under the scenario x, that we denote l_a^x, is equal to l_a^- if $x_a = 1$ and equal to l_a^+ otherwise. The cost of a scenario $x \in \{0,1\}^A$,

$$c(x) = \sum_{a \in A} x_a \cdot c_a, \tag{5.10}$$

corresponds to the cost for improving the length of the arcs $a \in A$ for which $x_a = 1$. Now, given a budget β, an arc (u,v) is said to be t-β-strong if it is on a shortest path from u to t in every scenario $x \in \{0,1\}^A$ whose cost is at most β, i.e., $c(x) \leq \beta$. Clearly, any t-strong arc is t-β-strong for any β, but the opposite is not true. Let $\mathbb{F}^{\beta}(u,v)$ be the intersection of the fibers of the arc (u,v) over all the scenarios whose costs are at most β, i.e.

$$\mathbb{F}^{\beta}(u,v) = \bigcap_{\substack{x \in \{0,1\}^A \\ c(x) \leq \beta}} F^x(u,v). \tag{5.11}$$

By definition, an arc (u,v) is t-β-strong if and only if t belongs to $\mathbb{F}^{\beta}(u,v)$. Computing the set $\mathbb{F}^{\beta}(u,v)$ is a *NP*-Hard problem. Indeed, even the problem of deciding whether v belongs to $\mathbb{F}^{\beta}(u,v)$ or not is *NP*-Hard since the decision version of the CSPP can be reduced to it. More precisely, given a graph $G = (V, A)$ where each arc $a \in A$ has a length l_a and a cost c_a, two vertices $s, t \in V$, a budget β and a length l^*, deciding if it exists a path P from s to t such that $c(P) \leq \beta$ and $l(P) < l^*$ can be reduced to deciding if $t \in \mathbb{F}^{\beta}(s,t)$ in the graph $G' = (V, A \cup \{(s,t)\}, l^-, l^+, c)$ where $l_a^- = l_a$, $l_a^+ = +\infty$

and $l^+_{st} = l^-_{st} = l^*$. Indeed, $t \notin \mathbb{F}^\beta(s,t)$ if and only if it exists an st-path P with $c(P) \leq \beta$ and $l(P) < l^*$ that do not contain the arc (s,t). Since computing $\mathbb{F}^\beta(u,v)$ is NP-Hard, we focus on how to use Lagrange multipliers to compute in polynomial time a subset S such that $F(u,v) \subseteq S \subseteq \mathbb{F}^\beta(u,v)$.

Given a Lagrange multiplier μ, let λ^μ_a denote the modified length of the arc a such that

$$\lambda^\mu_a = \min\left(l^-_a + \mu \cdot c_a, l^+_a\right). \tag{5.12}$$

Then, let $F^x_{\mu,T}(u,v)$, with $T \subseteq V$, denote the fiber of the arc (u,v) in the graph $H = (V, A')$ obtained from G by removing the incoming arcs of T except (u,v), i.e., $A' = A \setminus \{(w,t) \in A : w \notin T, t \in T, (w,t) \neq (u,v)\}$ and where each arc $a \in A'$ is associated to the interval $[\lambda^\mu_a, \lambda^+_a]$ with $\lambda^+_{uv} = l^+_{uv} + \mu \cdot \beta$ and $\lambda^+_a = l^+_a$ for $a \neq (u,v)$, such that

$$F^x_{\mu,T}(u,v) = \{t \in V : d^x_H(u,t) = \lambda^x_{uv} + d^x_H(v,t)\}. \tag{5.13}$$

Here, $d^x_H(st)$ denotes the length of a shortest st-path in the graph H using λ^x_a as the length of each arc $a \in A'$ with $\lambda^x_a = \lambda^\mu_a$ if $x_a = 1$ and $\lambda^x_a = \lambda^+_a$ otherwise. We denote the intersection of all the fibers of (u,v) in this graph by

$$F_{\mu,T}(u,v) = \bigcap_{x \in \{0,1\}^{A'}} F^x_{\mu,T}(u,v). \tag{5.14}$$

Lemma 5.7. *Given an arc (u,v), a budget β, a subset $T \subseteq \mathbb{F}^\beta(u,v)$ and a Lagrange multiplier μ, $T \subseteq F_{\mu,T}(u,v) \subseteq \mathbb{F}^\beta(u,v)$.*

Proof. Since we removed the incoming arcs of T except (u,v) to obtain the graph H, the inclusion $T \subseteq F_{\mu,T}(u,v)$ is trivial because all the remaining paths from u to a vertex of T must contain (u,v). It then remains to prove that $F_{\mu,T}(u,v) \subseteq \mathbb{F}^\beta(u,v)$. We proceed by contradiction. Suppose that $F_{\mu,T}(u,v) \setminus \mathbb{F}^\beta(u,v) \neq \emptyset$ and let t be the vertex of $F_{\mu,T}(u,v) \setminus \mathbb{F}^\beta(u,v)$. By definition, since $t \notin \mathbb{F}^\beta(u,v)$, it exists a scenario $x \in \{0,1\}^A$ with $c(x) \leq \beta$ under which (u,v) do not belong to any shortest ut-path of the graph G, i.e.

$$d^x_G(u,t) < l^x_{uv} + d^x_G(v,t) \leq l^+_{uv} + d^+_G(v,t). \tag{5.15}$$

We chose the vertex t such that $d^x_G(u,t)$ is minimum. Let P be a shortest ut-path realizing the distance $d^x_G(u,t)$ in the graph G, i.e., $l^x(P) = d^x_G(u,t)$. Since $c(x) \leq \beta$, the cost of P under the scenario x, $c^x(P) = \sum_{a \in P} c_a \cdot x_a$, is also at most β. By our hypothesis, P can neither contain (u,v) nor any vertex $w \in T$ since $T \subseteq \mathbb{F}^\beta(u,v)$ and, by definition, there would be a shortest uw-path Q passing by (u,v) with $l^x(P) = l^x(Q) + d^x_G(w,t)$, a contradiction. Therefore, P can only contain arcs of A', and is also a ut-path in the graph H. By Lemma 5.3, $t \in F_{\mu,T}(u,v)$ is equivalent to $t \in F^y_{\mu,T}(u,v)$ where y is the scenario defined as

$$y_{wt} = \begin{cases} 0 & w \in F_{\mu,T}(u,v) \textbf{ or } (w,t) = (u,v) \\ 1 & \text{otherwise} \end{cases}, \tag{5.16}$$

and the distance from u to t in this scenario is

$$d_H^y(u,t) = \lambda_{uv}^+ + d_H^+(v,t) = l_{uv}^+ + d_H^+(v,t) + \mu \cdot \beta. \tag{5.17}$$

We can deduce that all the arcs of P are at their lower bound in the scenario y, otherwise P should contain a vertex $w \in F_{\mu,T}(u,v)$ that either leads to the same contradiction as previously if $w \in \mathbb{F}^\beta(u,v)$ or contradicts our choice of t minimizing $d_G^x(u,t)$ if $w \notin \mathbb{F}^\beta(u,v)$. Thus, the length of P in the scenario y is

$$\lambda^y(P) = \sum_{a \in P} \lambda_a^\mu = l^-(P) + \mu \cdot c(P). \tag{5.18}$$

Since in the scenario y all the arcs of P are at their lower bound and the cost of P in the scenario x is at most β, we have

$$\lambda^y(P) \leq \lambda^x(P) \leq l^x(P) + \mu \cdot \beta, \tag{5.19}$$

with $\lambda^x(P) = l^x(P) + \mu \cdot c^x(P)$. Finally, we obtain the following inequalities

$$\lambda^y(P) \leq l^x(P) + \mu \cdot \beta < l_{uv}^+ + d_G^+(v,t) + \mu \cdot \beta \leq d_H^y(u,t). \tag{5.20}$$

The first inequality of (5.20) comes from (5.19), the second comes from the fact that $l^x(P) = d_G^x(u,t)$ and (5.15), and the last one is derived from (5.17) and $d_G^+(v,t) \leq d_H^+(v,t)$. This leads to a contraction since, according to (5.20), in the graph H and under the scenario y, the ut-path P must be shorter than the distance from u to t. □

We describe an $O(|A| + |V|\log|V|)$ time algorithm that, given an arc (u,v) of G, a subset $T \subseteq \mathbb{F}^\beta(u,v)$ and a Lagrange multiplier μ, computes $F_{\mu,T}(u,v)$. This algorithm is clearly a generalization of Algorithm 4, since for $\mu = 0$ and $T = \emptyset$, $F_{\mu,T}(u,v) = F(u,v)$. It amounts to running Algorithm 4 on the graph H, thus, its correctness follows from the correctness of Algorithm 4 and Lemma 5.7.

Algorithm 6: Computes the set $F_{\mu,T}(u,v) \subseteq \mathbb{F}^{\beta}(u,v)$

Input : $G=(V,A)$, $(l_a^-, l_a^+, c_a)_{a\in A}$, $\beta \in \mathbb{R}^+$, $(u,v)\in A$, $T \subseteq \mathbb{F}^{\beta}(u,v)$, $\mu \in \mathbb{R}^+$
Output: $F_{\mu,T}(u,v)$
$d(u) \leftarrow 0$
$\gamma(u) \leftarrow red$
foreach $(u,w) \in \delta_u^{out} \setminus \{(u,v)\}$ **do**
  $d(w) \leftarrow \lambda_{uw}^{\mu}$
  $\gamma(w) \leftarrow red$
$d(v) \leftarrow l_{uv}^+ + \mu \cdot \beta$
$\gamma(v) \leftarrow blue$
$S \leftarrow \{u\}$
while $S \neq V$ **do**
  Pick $t \in V - S$ with smallest $d(t)$ breaking tie by choosing a vertex t such that $\gamma(t) = blue$ if it exists
  if $\gamma(t)$ *is* $blue$ **then**
    foreach $(t,w) \in \delta_t^{out}$ **such that** $d(t) + l_{tw}^+ \leq d(w)$ **do**
      $d(w) \leftarrow d(t) + l_{tw}^+$
      $\gamma(w) \leftarrow blue$
  else
    foreach $(t,w) \in \delta_t^{out}$ **such that** $w \notin T$ *and* $d(t) + \lambda_{tw}^{\mu} < d(w)$ **do**
      $d(w) \leftarrow d(t) + \lambda_{tw}^{\mu}$
      $\gamma(w) \leftarrow red$
  $S \leftarrow S \cup \{t\}$
return $\{t \in V : \gamma(t) = blue\}$

Proposition 5.8. *Given a graph $G=(V,A)$ where each arc a is associated with a length l_a^+ that can be improved to $l_a^- \leq l_a^+$ by paying a cost c_a, a budget β, an arc $(u,v)\in A$, a subset $T \subseteq \mathbb{F}^{\beta}(u,v)$ of the β-strong vertices for (u,v) and a Lagrange multiplier μ, Algorithm 6 computes in $O(|E|+|V|\log|V|)$ the set $F_{\mu,T}(u,v)$ such that $T \subseteq F_{\mu,T}(u,v) \subseteq \mathbb{F}^{\beta}(u,v)$.*

The use case for Algorithm 4 is to start with the biggest known subset T of $\mathbb{F}^{\beta}$, typically, at the beginning $T = F(u,v)$, and try a value for the Lagrange multiplier μ in order to increase T with vertices of $\mathbb{F}^{\beta}$. The obtained subset is then guaranteed to be at least T, and we can repeat the operation with it and another value for μ. There is no need to test the same value for μ two consecutive times since we can easily show that $F_{\mu,(F_{\mu,T}(u,v))}(u,v) = F_{\mu,T}(u,v)$. This approach seems quite promising since, given k different values for μ, running this procedure is only k times more expensive in time than identifying the set $F(u,v)$, and the benefits seem to increase when the budget gets smaller. Indeed, for any budget value β,

$$F^{+\infty}(u,v) \subseteq \mathbb{F}^{\beta}(u,v) \subseteq F^{0}(u,v), \tag{5.21}$$

where $F^{+\infty}(u,v) = F(u,v)$, and $F^{0}(u,v) = \{t \in V : d^+(u,t) = l_{uv}^+ + d^+(v,t)\}$. However,

since $F_{\mu,T}(u,v)$ is only guaranteed to be a subset of $\mathbb{F}^{\beta}(u,v)$, numerical experiments are still needed to assess the benefits of running such procedure on practical instances and to help develop a strategy for choosing the values for μ. For now, we only know that μ must be comprised between 0 and

$$\max_{a \in A} \frac{(l_a^+ - l_a^-)}{c_a}, \tag{5.22}$$

since above this value, $\lambda_a^{\mu} = l_a^+$ for each arc $a \in A$.

Similarly to Algorithm 4, Algorithm 6 can be modified to identify a subset of the β-useless vertices of an arc (u,v) by following the same rationale as the one presented in Section 5.4. Let $\mathbb{W}^{\beta}(u,v)$ denote the set of vertices t such that (u,v) is t-β-useless i.e.

$$\mathbb{W}^{\beta}(u,v) = \{t \in V : \forall x \in \mathcal{X}^{\beta}, d^x(u,t) < l_{uv}^x + d^x(v,t)\}, \tag{5.23}$$

where $\mathcal{X}^{\beta} = \{x \in \{0,1\}^A : c(x) \leq \beta\}$ is the set of scenarios whose costs are at most β.

Algorithm 7: Computes the set $W_{\mu,T}(u,v) \subseteq \mathbb{W}^{\beta}(u,v)$

Input : $G = (V,A)$, $(l_a^-, l_a^+, c_a)_{a \in A}$, $\beta \in \mathbb{R}^+$, $(u,v) \in A$, $T \subseteq \mathbb{F}^{\beta}(u,v)$, $\mu \in \mathbb{R}^+$
Output: $W_{\mu,T}(u,v)$

```
d(u) ← 0
γ(u) ← blue
foreach (u,w) ∈ δ_u^out \ {(u,v)} do
    d(w) ← l_uw^+ + μ·β
    γ(w) ← blue
d(v) ← λ_uv^μ
γ(v) ← red
S ← {u}
while S ≠ V do
    Pick t ∈ V − S with smallest d(t) breaking tie by choosing a vertex t such that
      γ(t) = red if it exists
    if γ(t) is blue then
        foreach (t,w) ∈ δ_t^out such that d(t) + l_tw^+ < d(w) do
            d(w) ← d(t) + l_tw^+
            γ(w) ← blue
    else
        foreach (t,w) ∈ δ_t^out such that w ∉ T and d(t) + λ_tw^μ ≤ d(w) do
            d(w) ← d(t) + λ_tw^μ
            γ(w) ← red
    S ← S ∪ {t}
return {t ∈ V : γ(t) = blue}
```

Proposition 5.9. *Given a graph $G = (V,A)$ where each arc a is associated with a length l_a^+ that can be improved to $l_a^- \leq l_a^+$ by paying a cost c_a, a budget β, an arc $(u,v) \in A$,*

a subset $T \subseteq \mathbb{F}^{\beta}(u,v)$ *of the* β*-useless vertices for* (u,v) *and a Lagrange multiplier* μ*, Algorithm 7 computes in* $O(|E|+|V|\log|V|)$ *the set* $W_{\mu,T}(u,v)$ *such that* $T \subseteq W_{\mu,T}(u,v) \subseteq \mathbb{W}^{\beta}(u,v)$*.*

6 Greedy Algorithms

Table of contents

In the lack of an efficient method to compute optimal solutions of BC-PC-Opt, ecologists often use greedy algorithms to compute suboptimal solutions [61]. In Section 6.1, we present four commonly used *greedy algorithms* for BC-PC-Opt and highlight their pathological cases in Section 6.2. We will compare the quality of the solutions obtained with these greedy algorithms with the optimal solution on four case studies in Chapter 7. Then, in Section 6.3, we show results for improving the complexity and execution time of the Incremental Greedy algorithm that were obtained as part of Nicolas Yaverian's Bachelor's degree internship that we co-supervised [132].

DEFINITION BOX 16: GREEDY ALGORITHM

A greedy algorithm is an algorithm designed to construct solutions (not necessarily optimal ones) for a combinatorial optimization problem [111] by only making locally optimal choices, for example, by starting from the empty solution and iteratively adding to it the element that offers the largest increase in the objective function, as long as such element exists. Since greedy algorithms never reconsider their previous decisions, they do not always produce optimal solutions for many problems. However, they do provide optimal solutions when the space of solutions for the considered optimization problem has a *matroidal* structure. They can also provide constant factor approximations for some problems. We refer to [28] for an in-depth introduction.

6.1 Definition of the algorithms

The *Incremental Greedy* (IG) algorithm starts from the graph with no improved elements. At each step k, the algorithm selects an option $i \in \Phi$ with the greatest ratio Δ_i^k / c_i until no more options fit in the budget. Here, Δ_i^k denotes the difference between

the value of PC with and without the improvements of the option i at the step k. As usual, c_i is the cost of the option i. This IG algorithm is for example used in [46]. The *Decremental Greedy* (DG) algorithm, similar to the Zonation Algorithm [89], starts from the graph with all improvements performed and iteratively removes the improvement of the option i that have the smallest ratio Δ_i^k / c_i. DG finishes with incremental steps to ensure there is no free budget left. These algorithms perform at most $|\Phi|$ steps and at each step k need to compute Δ_i^k for each option $i \in \Phi$. It is easy to implement IG and DG in $O(|\Phi|^2 \cdot |V|^3)$ time by using an all-pair shortest path algorithm, for example, the Floyd-Warshall one [44], to compute in $O(|V|^3)$ the initial value of the PC and the value of Δ_i^k / c_i for each option i and step k. These complexities are already too large for the practical instances handled by ecologists which can have few thousands of patches. Most of the studies relying on the PC or ECA indicators use simpler algorithms that we call *Static Increasing* (SI) and *Static Decreasing* (SD). These algorithms are variants of the greedy algorithms which do not recompute the ratio Δ_i / c_i of each option i at each step and thus are faster but do not account for cumulative effects nor redundancies. The SI and SD have thus a time complexity of $O(|\Phi| \cdot |V|^3)$.

We will see how to improve the time complexity of the SI and IG algorithm in Section 6.3.

Let G^P, with $P \subseteq \Phi$, denote the landscape graph obtained from G by realizing the improvements corresponding the options of P. Below, we give the pseudocode of each of the algorithms we have described to eliminate any ambiguity.

Algorithm 8: Static Incremental

Input : $G = (V, A, w, \pi)$, Φ, $c \in (\mathbb{R}^+)^\Phi$, $\beta \in \mathbb{R}^+$
Output: $P \subseteq \Phi : c(P) \leq \beta$
Let $S = (i_1, \ldots, i_n)$ be the sequence of the options of Φ in decreasing order of their benefit-cost ratio, where $\frac{PC(G^{\{i_k\}}) - PC(G)}{c(i_k)}$ is the ratio of the option $i_k \in \Phi$
$P \leftarrow \emptyset$
for $k \leftarrow 1$ **to** n **do**
    if $c(P) + c(i_k) \leq \beta$ **then**
        $P \leftarrow P \cup \{i_k\}$
return P

Algorithm 9: Static Decremental

Input : $G = (V, A, w, \pi)$, Φ, $c \in (\mathbb{R}^+)^\Phi$, $\beta \in \mathbb{R}^+$
Output: $P \subseteq \Phi : c(P) \leq \beta$
Let $S^{\text{dec}} = (i_1, \ldots, i_n)$ be the sequence of the options of Φ in increasing order of their benefit-cost ratio, where $\frac{PC(G^\Phi) - PC(G^{\Phi \setminus \{i_k\}})}{c(i_k)}$ is the ratio of the option $i_k \in \Phi$
$P \leftarrow \Phi$
for $k \leftarrow 1$ **to** n **do**
  if $c(P) > \beta$ **then**
    $P \leftarrow P \setminus \{i_k\}$
Let $S^{\text{inc}} = (j_1, \ldots, j_m)$ be the sequence of the options of $\Phi \setminus P$ in decreasing order of their benefit-cost ratio, where $\frac{PC(G^{\{j_k\}}) - PC(G)}{c(j_k)}$ is the ratio of the option $j_k \in \Phi$
for $k \leftarrow 1$ **to** m **do**
  if $c(P) + c(j_k) \leq \beta$ **then**
    $P \leftarrow P \cup \{j_k\}$
return P

Algorithm 10: Incremental Greedy

Input : $G = (V, A, w, \pi)$, Φ, $c \in (\mathbb{R}^+)^\Phi$, $\beta \in \mathbb{R}^+$
Output: $P \subseteq \Phi : c(P) \leq \beta$
$D \leftarrow \{j \in \Phi : c(j) \leq \beta\}$
$P \leftarrow \emptyset$
while $D \neq \emptyset$ **do**
  Pick $i \in D$ maximizing the ratio $\frac{PC(G^{P \cup \{i\}}) - PC(G^P)}{c(i)}$
  $P \leftarrow P \cup \{i\}$
  $D \leftarrow \{j \in D : j \neq i, c(j) \leq \beta - c(P)\}$
return P

Algorithm 11: Decremental Greedy

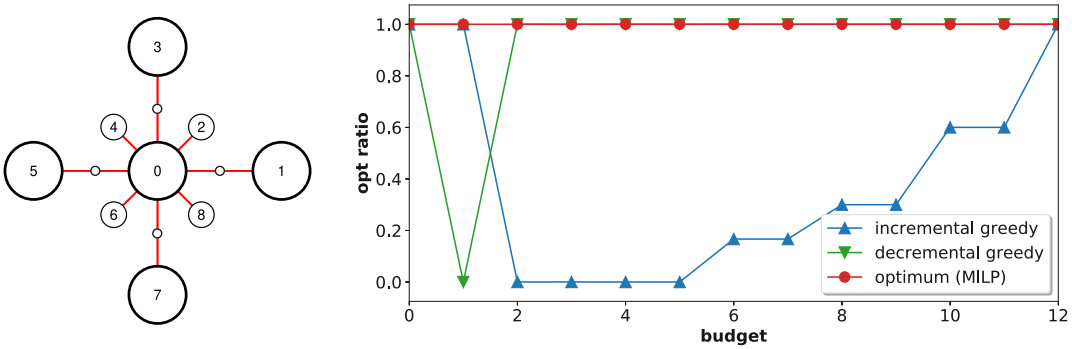
Input : $G = (V, A, w, \pi)$, Φ, $c \in (\mathbb{R}^+)^\Phi$, $\beta \in \mathbb{R}^+$
Output: $P \subseteq \Phi : c(P) \leq \beta$
$P \leftarrow \Phi$
$R \leftarrow \emptyset$
while $c(P) \geq \beta$ **do**
  Pick $i \in P$ minimizing the ratio $\frac{PC(G^P) - PC(G^P \setminus \{i\})}{c(i)}$
  $P \leftarrow P \setminus \{i\}$
  $R \leftarrow R \cup \{i\}$
$R \leftarrow \{j \in R : c(j) \leq \beta - c(P)\}$
while $R \neq \emptyset$ **do**
  Pick $i \in R$ maximizing the ratio $\frac{PC(G^{P \cup \{i\}}) - PC(G^P)}{c(i)}$
  $P \leftarrow P \cup \{i\}$
  $R \leftarrow \{j \in R : j \neq i, c(j) \leq \beta - c(P)\}$
return P

6.2 Arbitrary bad cases

Below, we provide instances on which IG and DG perform poorly compared to an optimal solution. On these instances, it is easy to check that the solutions returned by SI and SD are not better than the solutions returned by IG and DG. In the following instances, all arcs have a probability 1 if improved and 0 otherwise and have unitary costs. Recall that a spider is a tree consisting of several paths glued together on a central vertex (Figure 6.1a, 6.2a and 6.3a).

Bad case for IG: The graph is a spider with $2k$ branches: k long branches with two edges, an intermediate node of weight 0 and a leaf node of weight 1, and k short branches consisting of a single edge with a leaf of very small weight $\epsilon > 0$, see Figure2 (a). All branches are connected to a central node of weight 1. IG performs poorly on this instance. Indeed, IG is tricked into selecting short branches with very small PC improvement because selecting an edge of a long branch alone does not increase PC at all. An optimal solution results in larger value of PC by improving pairs of arcs of long branches.

In this case, IG does not perform well while DG finds an optimal solution computed by the MILP solver except when the budget is 1. In this case, the reverse occurs: DG performs badly while IG is optimal. Indeed, DG realizes that the budget is not sufficient to improve two arcs of a long branch only after removing the improvements of all short branches.

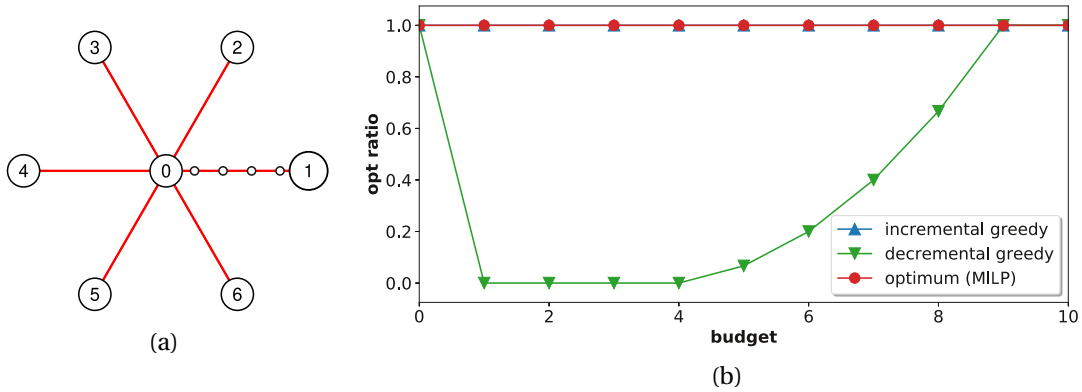


(a) (b)

Figure 6.1: An instance on which Incremental Greedy fails. (a) the graph of the IG bad case with $k = 4$, (b) ratio of the increase in PC between the solutions returned by IG and DG and an optimal solution for several budgets.

Bad case for DG: The graph is obtained from a star with $k+1$ branches by replacing one branch by a path of length k. The central node and all leaves except the leaf of the path have a weight of 1. The leaf of the path has a weight of $1+\epsilon$. The internal nodes of the path have a weight of 0. DG performs poorly on this instance because it removes one by one the branches of the star for which $\Delta_e / c_e = 1$ before removing

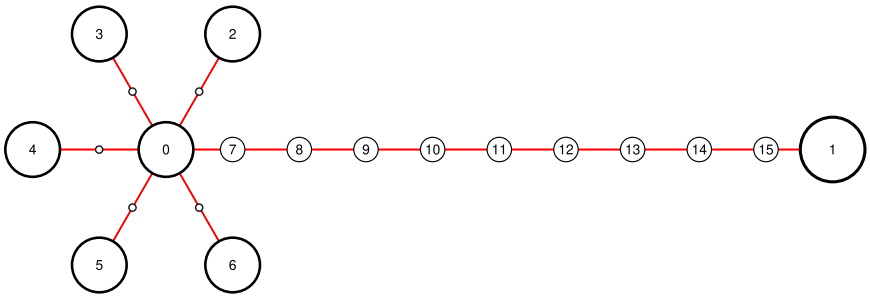
an edge of the path for which $\Delta_e / c_e = 1 + \epsilon$. When the budget is at least 2, an optimal solution removes all the edges of the path before removing an edge of another branch.



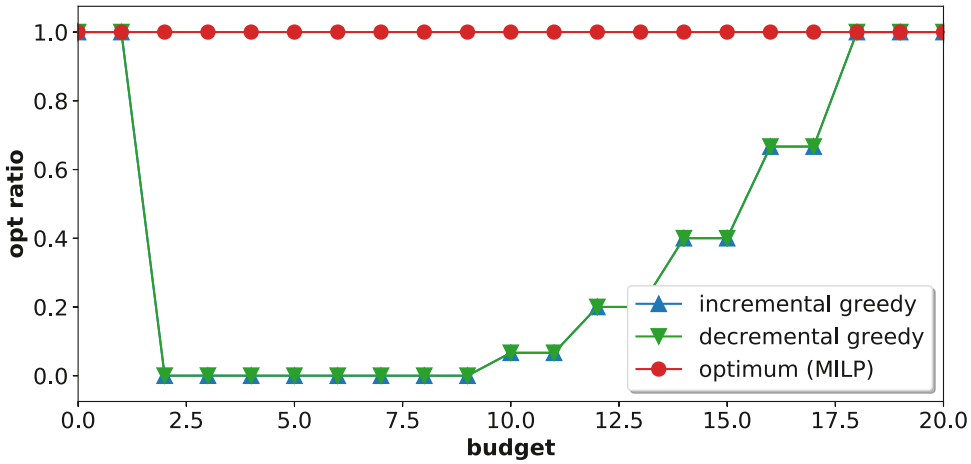
(a) (b)

Figure 6.2: An instance on which Decremental Greedy fails. (a) the graph of the DG bad case with $k = 5$, (b) ratio of the increase in PC between the solutions returned by IG and DG and an optimal solution for several budgets.

Bad case for IG and DG: The graph is a spider with $k+1$ branches. All branches except one are paths of length 2 with an internal node of weight 0 and a leaf of weight 1. The last branch is a path of length $2k$ with internal nodes of weight $\epsilon > 0$ and a leaf of weight $1 + \epsilon$. All branches intersect in a central node whose weight is 1. In this case, both Incremental and Decremental Greedy fail. On one hand, IG selects the edges of the path of length $2k$ one by one and does not realize that by taking two edges of a short branch it could improve much more PC. On the other hand, DG removes first the edges of the short branch because the weight of leaf of a long branch is $1 + \epsilon$ while the weight of the leaf of a short branch is 1. Hence, DG and IG return the same low quality solution.



(a)



(b)

Figure 6.3: An instance on which both Incremental and Decremental Greedy algorithms fail. (a) the graph of the IG and DG bad case with $k = 5$, (b) ratio of the increase in PC between the solutions returned by IG and DG and an optimal solution for several budgets.

The case of Figure 6.3 illustrates the fact that IG and DG do not provide any constant approximation guarantee (even for trees, that are usually simpler cases), i.e. for any constant $0 < \alpha < 1$ there exists an instance of BC-PC-Opt such that $ALG < \alpha OPT$ where ALG is the value of PC for the best solution among those returned by IG and DG and OPT is the value of PC for an optimal solution.

6.3 Incremental Greedy with dynamic update

As described in Section 6.1, each time the greedy algorithms select an option, they need to recompute the all-pair shortest paths for each remaining option i to assess Δ_i which leads to a time complexity of $O(|\Phi|^2 \cdot |V|^3)$, using the Floyd-Warshall algorithm [44]. We can indeed drop this complexity to $O(|\Phi|^2 \cdot |V|^{2.37287})$ by using advanced matrix multiplication algorithms [6], but the hidden multiplicative constant is so high

that it would not be practical for any landscape graph of reasonable size. Consequently, static algorithms are often preferred when the landscape graph has more than a few thousand of vertices and a few hundred of options. In this section we present how to improve the complexity of the IG algorithm to $O(|V|^3 + |\Phi|^2 \cdot k \cdot |V|^2)$ where k is the maximum number of vertices and arcs that can be improved by a single option.

It is long known that when we decrease the length (resp. increase the probability) of an arc we can update the matrix of distances (resp. probabilities) in $O(|V|^2)$ time [33]. In our case, let Π be the matrix of the probabilities of connection and (u, v) be the arc whose probability is about to be increased from π_{uv} to π'_{uv}. The matrix Π' corresponding to the probabilities of connection after the improvement of (u, v) can then be computed from Π as follows:

$$\Pi'_{st} = \max\left(\Pi_{st}, \Pi_{su} \cdot \pi'_{uv} \cdot \Pi_{vt}\right). \tag{6.1}$$

Thus, given an option i than improves the probability of k arcs, we can compute the updated the matrix of probabilities Π^i in $O(k \cdot |V|^2)$ and obtain the new value of the PC in $O(|V|^2)$ as

$$\sum_{s,t \in V} (w_s + w_s^i) \cdot (w_t + w_t^i) \cdot \Pi_{st}^i. \tag{6.2}$$

Recently, the authors of [123] proposed an improved algorithm for updating the matrix of distances (resp. probabilities) upon the decrease of the length (resp. increase of the probability) of an arc (u, v). Their algorithm still runs in $O(|V|^2)$ but exhibits a much better average runtime, which is applicable to our case. The idea is to focus on the set $\mathscr{P}$ of the pairs of vertices whose probability of connection will strictly increase with the improvement of the arc (u, v), i.e.,

$$\mathscr{P} = \{(s, t) \in V^2 : \Pi(s, u) \cdot \pi'(u, v) \cdot \Pi(v, t) > \Pi(s, t)\}. \tag{6.3}$$

Since any subpath of a most reliable path is also a most reliable path, $\mathscr{P}$ can be viewed as a subset of the Cartesian product $S \times T$ where S (resp. T) is the set of vertices whose probability of connection to v (resp. from u) is strictly increased by the improvement of the arc (u, v), i.e.,

$$\mathscr{P} \subseteq S \times T \tag{6.4}$$

with

$$S = \{s \in V : \Pi(s, u) \cdot \pi'_{uv} > \Pi(s, v)\}, \tag{6.5}$$

$$T = \{t \in V : \pi'_{uv} \cdot \Pi(v, t) > \Pi(u, t)\}. \tag{6.6}$$

The sets S and T have the advantage of being computable in $O(|V|)$ time, as shown in Algorithm 12, while also being smaller than V, on average. The matrix of probabilities can thus be updated in $O(|V| + |S| \cdot |T|)$ which is still $O(|V|^2)$, but faster in practice.

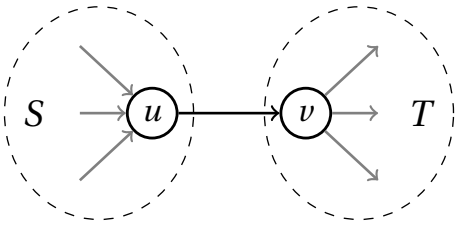


Figure 6.4: Illustration of the set S and T with respect to the improved arc (u,v).

Algorithm 12: Compute-S-T

Input : A probability matrix Π and an arc (u,v) with improved probability π'_{uv}
Output: The pair of sets (S,T)

```
S ← ∅
T ← ∅
for w ∈ V do
    if Π(w,u) · π'_uv > Π(w,v) then
        S ← S ∪ {w}
    if π'_uv · Π(v,w) > Π(u,w) then
        T ← T ∪ {w}
return (S,T)
```

We still need to compute Δ_i, i.e., the amount by which the PC is increased by the option i. Recall that π^i_{uv} corresponds to the probability of (u,v) when it is improved by the option i and that w^i_u corresponds to the amount by which the option i increases the weight of u. Let F_i and W_i be the sets of arcs and vertices whose weights are strictly increased by the option i. If $W_i = \emptyset$ we can easily compute Δ_i by iteratively updating the probability matrix Π for each improved arc in $O(|F_i| \cdot |V|^2)$ time, c.f. Algorithm 13.

Algorithm 13: Compute-Δ_i-arcs-only

Input : A probability matrix Π and an option $i \in \Phi$ with $W_i = \emptyset$
Output: Δ_i

```
Δ_i ← 0
Π' ← Π
for (u,v) ∈ F_i do
    (S,T) ← Compute-S-T(Π', (u,v), π^i_uv)
    for s ∈ S do
        for t ∈ T do
            p ← Π'_su · π^i_uv · Π'_vt
            if p > Π'_st then
                Δ_i ← Δ_i + w_s · w_t · (p − Π'_st)
                Π'_st ← p
return Δ_i
```

However, when $W_i \neq \emptyset$ we need to account for the increased contribution to the PC of the pairs of vertices of $V \times W_i$ and $W_i \times V$. One way to achieve this efficiently is to maintain a vector $f \in \mathbb{R}^V$ that associates to each vertex t the quantity of flow arriving at t, i.e., $f_t = \sum_{s \in S} w_s \cdot \Pi_{st}$. When increasing the weights of the vertices of W_i, the new vector f' can be calculated from f in $O(|W_i| \cdot |V|)$ time such that, for each vertex t, $f'_t = f_t + \sum_{s \in W_i} w^i_s \cdot \Pi_{st}$. The incurred increase in PC can then be computed in $O(|V|)$ time as

$$\left(\sum_{t \in V} w_t \cdot (f'_t - f_t)\right) + \left(\sum_{t \in W_i} w^i_t \cdot f'_t\right). \tag{6.7}$$

The first sum of (6.7) correspond to the gain obtained by increasing the weights of the source vertices while the second sum correspond to the gain achieved by increasing the weights of the target vertices. Thus, the whole procedure of computing Δ_i can be implemented in $O(|F_i| \cdot |V|^2 + |W_i| \cdot |V|)$, as described by Algorithm 14.

Algorithm 14: Compute-Δ_i

Input : A probability matrix Π, the vector $f \in \mathbb{R}^V$ and an option $i \in \Phi$
Output: Δ_i

$\Delta_i \leftarrow 0$
$\Pi' \leftarrow \Pi$
$f' \leftarrow f$
for $(u, v) \in F_i$ **do**
  $(S, T) \leftarrow$ Compute-S-$T(\Pi', (u, v), \pi^i_{uv})$
  for $s \in S$ **do**
    for $t \in T$ **do**
      $p \leftarrow \Pi'_{su} \cdot \pi^i_{uv} \cdot \Pi'_{vt}$
      if $p > \Pi'_{st}$ **then**
        $\Delta_i \leftarrow \Delta_i + w_s \cdot w_t \cdot (p - \Pi'_{st})$
        $f'_t \leftarrow f'_t + w_s \cdot (p - \Pi'_{st})$
        $\Pi'_{st} \leftarrow p$
if $W_i \neq \emptyset$ **then**
  $f'' \leftarrow f'$
  for $s \in W_i$ **do**
    for $t \in V$ **do**
      $f''_t \leftarrow f''_t + w^i_s \cdot \Pi'_{st}$
  $\Delta_i \leftarrow \Delta_i + \left(\sum_{t \in V} w_t \cdot (f''_t - f'_t)\right) + \left(\sum_{t \in W_i} w^i_t \cdot f''_t\right)$
return Δ_i

6.4 A bound for improving IG on SAO-BC-PC-Opt

In the case when each option $i \in \Phi$ improves only one arc (u, v) of the graph, i.e., the case of an instance of the SAO-BC-PC-Opt problem, we can also consider a bounding

method to avoid computing the real value of Δ_i if i cannot be the option with the greatest ratio Δ_i / c_i. Indeed, in this case, Δ_i can be expressed as

$$\Delta_i = \sum_{(s,t)\in\mathscr{P}} w_s \cdot w_t \cdot (\Pi'(s,t) - \Pi(s,t)), \tag{6.8}$$

where Π' is the probability matrix obtained by improving the arc (u,v), and we can then bound Δ_i as follows

$$\begin{aligned}
\Delta_i &\leq \sum_{(s,t)\in\mathscr{P}} w_s \cdot w_t \cdot \Pi(s,u) \cdot \big(\pi'(u,v) - \pi(u,v)\big) \cdot \Pi(v,t) && (6.9)\\
&\leq \sum_{s\in S}\sum_{t\in T} w_s \cdot w_t \cdot \Pi(s,u) \cdot \big(\pi'(u,v) - \pi(u,v)\big) \cdot \Pi(v,t) && (6.10)\\
&\leq \left(\sum_{s\in S} w_s \cdot \Pi(s,u)\right) \cdot \big(\pi'(u,v) - \pi(u,v)\big) \cdot \left(\sum_{t\in T} w_t \cdot \Pi(v,t)\right). && (6.11)
\end{aligned}$$

The first bound (6.9) is obtained from the fact that, for every pair of vertices $(s,t) \in \mathscr{P}$, $\Pi'(s,t) = \Pi(s,u) \cdot \pi'_{uv} \cdot \Pi(v,t)$, and $\Pi(s,u) \cdot \pi_{uv} \cdot \Pi(v,t) \leq \Pi(s,t)$, giving

$$\Pi'(s,t) - \Pi(s,t) \leq \Pi(s,u) \cdot \big(\pi'(u,v) - \pi(u,v)\big) \cdot \Pi(v,t). \tag{6.12}$$

The second bound (6.10) derives from $\mathscr{P} \subseteq S \times T$ and the fact that all the terms of the sum are positive. Then, (6.11) is a reformulation of (6.10) showing that this bound can be computed in $O(|V|)$ time together with the sets S and T. Thus, at a given step k of the IG algorithm, when computing Δ_i^k for each option i, we can use the bound M_i given by (6.11) to avoid computing the exact value of Δ_i^k for an option i if we already found an option j with a greater ratio, i.e., such that

$$\frac{\Delta_j^k}{c_j} \geq \frac{M_i}{c_i}. \tag{6.13}$$

Experiments conducted during the internship of Nicolas Yaverian have shown that very interesting speedups can be obtained over the naive implementation of the IG algorithm by using the methods presented in Sections 6.3 and 6.4. As the gains obtained are highly instance-dependent, with observed speedups between 4 and 100 times, it would be interesting to evaluate these algorithms on a larger set of instances.

7 Software and numerical experiments

Table of contents

7.1 Software production

An important part of this thesis has been devoted to the development of C++ tools in order to provide practical and efficient implementations of the data structures and algorithms needed to perform our optimization method. Consequently, quite a bit of time has been spent in learning to produce safe, portable and efficient C++ code using the last standards of C++, i.e. C++17, 20 and 23, and state-of-the-art building tools like CMake and Conan.

The experiments presented in this chapter are part of our article [59] whose code is accessible at `https://gitlab.lis-lab.fr/francois.hamonic/landscape_opt_networks_submission`. This first tool is implemented in C++17 uses Gurobi Optimizer [55] for solving MILP formulations, the graph library LEMON [34] for managing graph algorithms, and the library TBB [101] for multithreading the preprocessing and greedy algorithms. Since then, our codebase evolved, we use now C++20 and included our own graph library, MELON `https://github.com/fhamonic/melon`, and our own linear solver interface, MIPpp `https://github.com/fhamonic/mippp`. These two libraries are at very early stages of development but contain all the functionalities needed for implementing our optimization method. This new implementation is available at `https://github.com/fhamonic/landscape_opt`. Our graph library, MELON, offers between 25% and 50% faster execution times for the Dijkstra algorithm than the LEMON [34] and Boost.Graph [24] implementations, depending on hardware

specificities. We refer to https://github.com/fhamonic/melon_benchmark for up-to-date benchmarks. Moreover, the implementation of static digraphs in MELON is totally thread safe and easy to manipulate thanks to the extensive use of C++20 ranges and concepts. On the other hand, our linear solver interface, MIPpp, allows to concisely and efficiently instantiate mixed integer linear programs through the use of operator overloading and template metaprogramming, thanks again to C++20 functionalities. For the moment, MIPpp allows to interface either the Gurobi Optimizer [55] or Cbc [48] solvers.

7.2 Case studies

We have selected four case studies that cover a wide range of potential applications for conservation practitioners (Fig. 7.1), including conservation and restoration problems focused on habitat patches or corridors for different types of organisms (mammal, fish, bird, amphibian) in different types of ecosystems (terrestrial or freshwater). These case studies reflect different types of graphs or topologies that are classically used in connectivity conservation studies and described in Section 3.3. The case studies №1,2 and 4 are representing different patch based models with, respectively, the tree graph (typically used for river systems), the Minimum Planar Graph (MPG) and the Complete Thresholded Graph (CTG), and the case study №3 is a hexagonal raster model. These case studies relate to real conservation or restoration problems, but the results presented here and the parameterizing of the models aim to be illustrative. Application-oriented results would require the implication of static stakeholders, which was beyond the scope of this thesis.

Case study №1 consists in identifying among a set of 15 dams present on the Aude river (France) those that need to be equipped with fish passes in order to restore the river connectivity for trouts [106]. The river is modelled by a graph of 45 vertices of which 34 represent the river segments obtained by cutting the river tributaries at each confluence point and each dam while the 11 remaining vertices represent the confluence points [39, 112]. Each stretch is associated with a weight representing its length, as an approximation for its area. Two adjacent segments u and v are connected by reciprocal arcs (u,v) and (v,u) which represents the ability for an individual to move from the center of one stretch to the center of the other in upstream and downstream directions. Each arc (u,v) is associated with a probability π_{uv} representing the feasibility for an individual to make the corresponding movement. If the segments u and v are separated by a dam we fix this probability to zero, i.e., $\pi_{uv} = 0$, otherwise we compute it according to the negative exponential model, considering a median dispersal range of 10 kilometers [30], i.e., $\pi_{uv} = \exp\left(-\frac{\log(0.5)}{10000} \cdot d(u,v)\right)$ where $d(u,v)$ is the distance in meters between the centers of the segments u and v along the river course. The installation of a fish pass on a dam is modeled by increasing the probability of the corresponding arcs from 0 to 80% of the probability computed with the negative exponential model, and we assume that all fish passes have the same construction cost.

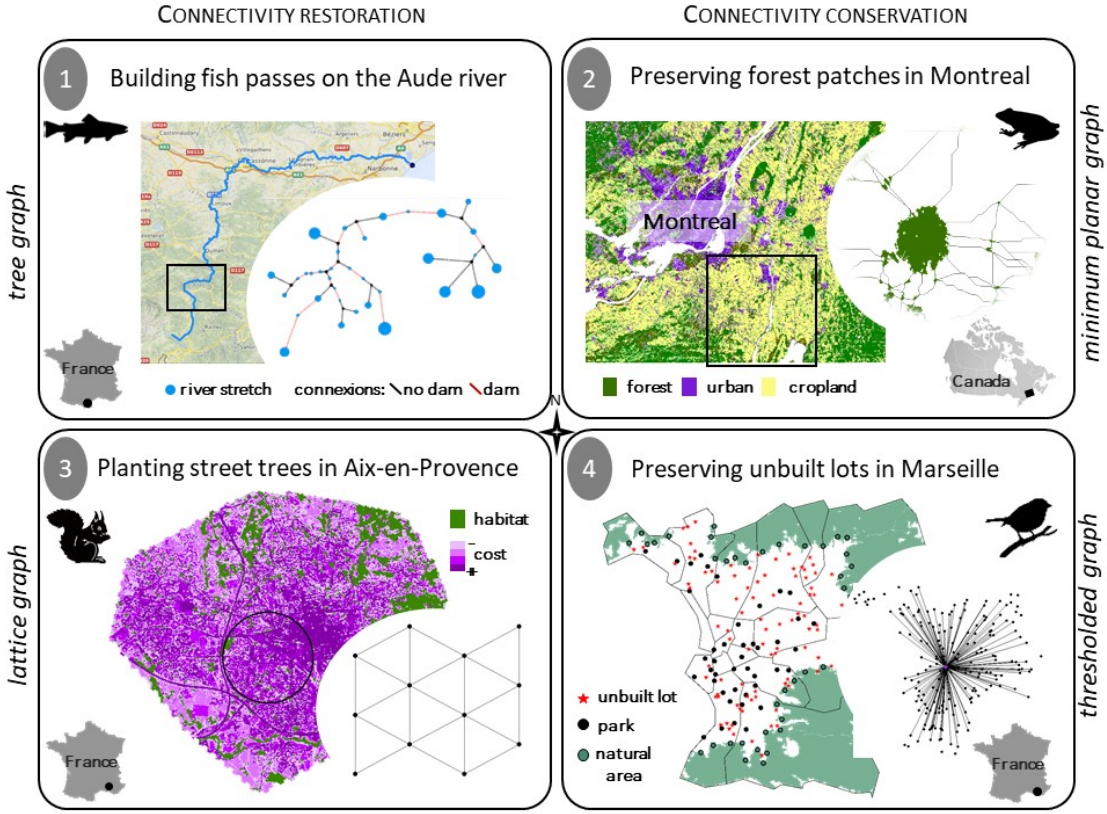


Figure 7.1: Case studies.

Case study №2 consists in identifying the remnant forest patches that need to be preserved from deforestation in the Montreal neighborhood (Canada) to guaranty habitat connectivity for the wood frog [5]. Here the 518-vertices graph is a minimum planar graph [41] with 1814 arcs. Each arc is weighted with a probability of movement which is a negative exponential function of the least-cost path length among patches, with a median dispersal of 300 meters. For each vertex u, we have an estimate of the area that the corresponding habitat patch could lose by 2050 due to agriculture or urbanization ('business as usual' scenario, [5]). A total of 260 vertices could be reduced in size ($w_u^i > 0$), of which 80 could disappear entirely if not protected ($w_u^i = w_u$). For partially-threatened patches we consider that individuals can continue to move across adjacent arcs with unchanged probabilities. However, for each fully-threatened patch, we replace its corresponding vertex u by two vertices u_1 and u_2 such that u_1 receive the incoming arcs of u and u_2 the outgoing arcs, i.e., $\delta^{\text{in}}_{u_1} = \delta^{\text{in}}_u$ and $\delta^{\text{out}}_{u_2} = \delta^{\text{out}}_u$, and add an arc (u_1, u_2) whose probability is forced to be 1, if the patch is protected and 0 otherwise. Fully-threatened patches that are not protected thus cannot be part of most reliable paths. Without loss of generality, we report the weight of u on u_1. We assume that the protection cost of each patch is proportional to its potential area loss by 2050.

Case study №3 consists in identifying street sections in which planting trees can

improve the connectivity of the urban canopy for the European red squirrel in the city of Aix-en-Provence. The landscape is here modelled with a raster-based model of 6565 hexagonal cells of 187 m^2. Each cell is associated with a quality weight of 1 if it contains mostly treed areas and 0 otherwise, and probability μ_u (probability that an individual succeeds when moving through it) that depends on the underlying land cover: 1 for habitat, 0.97 for partially treed areas, 0.7 for low vegetated areas, 0.45 for non vegetated areas, 0.25 for roads and 0 for buildings. The probability of each arc (u,v) is computed as $\pi_{uv} = (\mu_u \cdot \mu_v)^{\frac{1}{2}}$. We assume that planting trees along a street section increases by 6 the permeability of the arcs its crosses, i.e., $\pi^i_{uv} = (\pi_{uv})^{\frac{1}{6}}$, and that the cost of these actions is proportional to the number of crossed hexagons. For illustration, we chose here a set of 47 candidate street sections for which we implemented PC optimization. To make the instance more tractable, we contract adjacent cells that correspond to habitat, as described in Section 3.3.3. The resulting graph contains then 6186 vertices, of which 60 represent habitat patches, and 27818 arcs.

Case study №4 consists in identifying unbuilt lots that need to be preserved from artificialization to maintain connectivity among urban parks and the surrounding natural massifs in the city of Marseille for songbirds (e.g., Eurasian blackcap). Unbuilt lots, that are mainly present in the city periphery, can indeed act like stepping stones between massifs and parks. The baseline graph is composed of 196 vertices of which 42 models the frontier of massifs (20 ha each), 43 represents smaller parks (1 ha), 11 larger parks (5 ha), and 100 unbuilt lots (0.1 ha). We use a negative exponential function of the border-to-border distance between patches, with a median dispersal distance of 3000 meters [95] to estimate the probability of arcs. The graph is a complete thresolded graph with a thresolded probability of 0.135, i.e., it contains all arcs except those whose probability would be less than 0.135 which correspond approximately to an inter-patch distance of 8600 meters. When an unbuilt lot is not selected for conservation, its area becomes zero, and it no longer contributes to any shortest path (probability of moving along adjacent arcs set to zero). When selected, its attributes and those of its adjacent arcs do not change.

7.3 Numerical experiments

In this section, we report on our computational experiments in order to demonstrate the combined benefits of our MILP formulation and preprocessing step. We performed the numerical experiments on a desktop computer equipped with an Intel(R) Core(TM) i7-8700k 4.8 gigahertz and 32 gigabytes of memory and running Manjaro Linux release 21.2.4 with GCC version 10.2 and the libstdc++ that comes with it. At the moment, we used Gurobi Optimizer [55] version 9.1.1 with default settings for solving MILP formulations, the graph library LEMON [34] version 1.3.1 for managing graph algorithms, and the library TBB [101] version 2020.3 for multithreading the preprocessing and greedy algorithms. Code and data are available at `https://gitlab.lis-lab.fr/francois.hamonic/landscape_opt_networks_submission`.

7.3.1 Scalability and benefits of the preprocessing

In order to address the scalability of our approach and the benefits of our preprocessing step, we execute our method with and without preprocessing on about one hundred instances obtained from our four case studies by varying the budget between 0 (meaning no option is selected) and 100% (meaning all the options are selected). Our goal is to understand how the computing times and the size of the MILP model vary with respect to (i) the budget available, (ii) the number of binary decision variables, (iii) the presence of the preprocessing, (iv) and the percentage of improvable arcs compared to the total number of arcs in the graph.

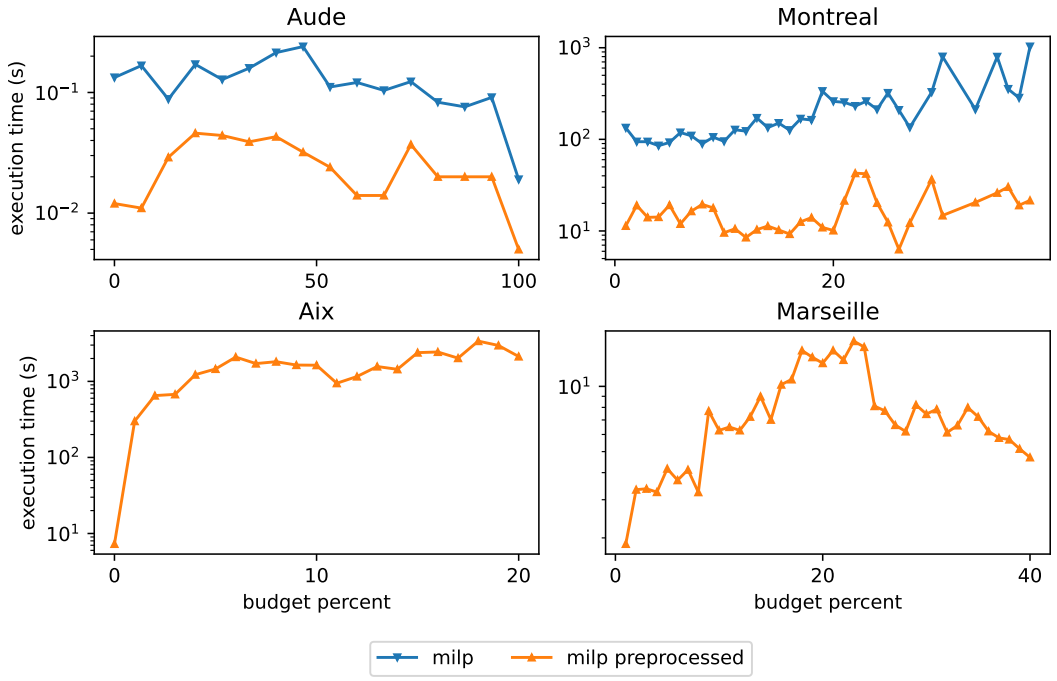


Figure 7.2: Execution times on the four case studies as a function of the budget (missing points correspond to instances that do not finish within 10 hours)

For the Aude and Montreal cases, the preprocessing reduces the resolution time by about a factor of 10 (Figure 7.2). Without preprocessing, the Aix and Marseille instances are not solved by the optimizer within 10 hours, whereas with preprocessing they become solvable in about half an hour and half a minute respectively. In most unfinished instances, the optimizer reaches the optimal solution but is not able to prove its optimality since it did not complete the exploration of the search space in the allotted time. Some instances of the Montreal case cannot be solved without preprocessing. The average computation times shown in Table 7.2 do not take these instances into account.

The preprocessing represents a small portion of the total computation time for all case studies (Table 7.1). The number of variables of the model is reduced by about 75% in the Aude case, 60% in the Montreal case, 70% in the Aix case and 99% in the

case	MILP			preprocessed MILP			
	#var	#const	time	#var	#const	p. time	time
Aude	4061	2551	120 ms	1069	1055	3 ms	20 ms
Montreal	830530	262445	4 mins	318848	167153	0.26 s	19 s
Aix	1748708	624841	–	555124	295010	3 s	1600 s
Marseille	4949825	78410	–	41676	22465	0.9 s	7 s

Table 7.1: Comparison of the MILP and the preprocessed MILP according to the number of variables (#var), the number of constraints (#const), the preprocessing time (p. time) and the average computation time (time).

Marseille case. This last number is explained by the fact that the Marseille graph is near complete and a large proportion of its arcs are t-useless for some vertex t, i.e., they never belong to any most reliable path to t and can thus be removed. Regarding the constraints, the reduction is 60% for the Aude case, 33% for the Montreal one, 53% for the Aix case and about 70% for the Marseille case.

#unbuilt lots	MILP			preprocessed MILP			DG
	#var	#const	time	#var	#const	time	time
20	345775	18410	12 s	10716	7197	< 1 s	< 1 s
50	717901	36410	2 mins	34613	21485	2 s	7 s
80	1306597	59810	32 mins	76281	42039	13 s	30 s
110		-		132225	66759	1 min	1 min 30 s
140		-		207999	96600	3 mins	3 mins
170		-		308355	132701	28 mins	7 mins

Table 7.2: Comparison of the MILP, the preprocessed MILP and DG according to the number of variables (#var), the number of constraints (#const) and the time (on average with 20 different budget values) it takes to solve the Marseille instance with different numbers of unbuilt lots

For the case of Marseille, the one with the largest number of variables, we also explored how the number of potential options influences the number of constraints and the computation time. We see in Table 7.2 that the computation time of the MILP without preprocessing increases very quickly. It takes more than 30 minutes on average for instances with 80+ unbuilt lots whereas the preprocessed ones can be solved in less than 30 minutes with up to 170 unbuilt lots. This is due to the preprocessing step that significantly reduces the time required to solve the linear relaxation by reducing the number of variables, constraints and non-zero entries of the mixed integer program. The preprocessed MILP is faster than the greedy algorithm for instances with at most 140 unbuilt lots (the preprocessing step was not used for greedy algorithms).

To show the efficiency of the preprocessing on the number of constraints in the problems (which is related to computation time), we run our preprocessing on 400

randomly generated instances from a model of the landscape around Montreal for snowshoe hares of 8733 vertices and 18422 arcs [5] and study the impact of the preprocessing on the MILP formulation size. For building these instances, we take 20 connected subgraphs of 500 vertices and for each graph we create 20 instances by randomly picking a percentage of arcs whose probability could be increased from π to $\sqrt{\pi}$. Sampling in the very large graph representing the Montreal region gives rise to different types of instances in terms of shapes, arc density, etc. This is why we decided to use the Montreal case for these experiments.

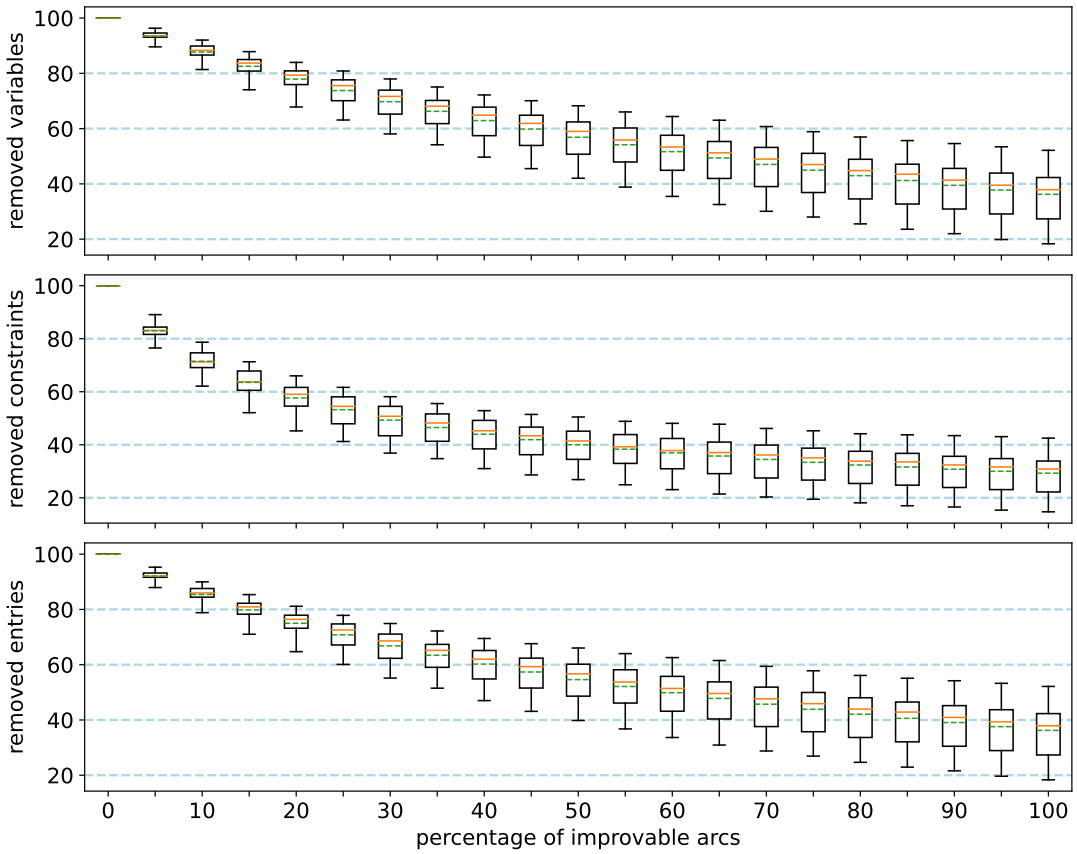


Figure 7.3: Box plots showing the percentage of constraints, variables and non-zero entries removed by the preprocessing as a function of the percentage of improvable arcs. The red line is the median, the dashed green line is the mean, the box represents the values between the 25th and 75th percentiles and the whiskers the min and max values.

Figure 7.3 shows a box plot of the size reduction of the MILP formulation in terms of constraints, variables and non-zero entries with respect to the percentage of arcs that could be improved compared to the total number of arcs in the graph. The preprocessing removes almost all the elements of the MILP formulation when the number of improvable arcs arrives close to zero. This reduction decreases with the

number of arcs that can be improved. When 20% of the arcs can be improved, the preprocessing removes on average 80% and at least 70% of the model's variables, on average 60% and at least 45% of the model's constraints, and on average 75% and at least 65% of the model's non-zero entries. Even when 100% of the arcs can be improved, the preprocessing reduces on average by 35% the model's size. The reductions achieved in terms of variables, constraints and entries seem to be robust since, for any percentage of improvable arcs, the gaps between the 25th and 75th percentiles do not exceed 17%. Furthermore, these results appear to be consistent with those of Table 7.1. Indeed, in the case of Aix, in which about 15% of the arcs can be improved, our preprocessing reduces the number of variables by 70% and the number of constraints by 65%.

7.3.2 Quality of the solutions

In order to compare the quality of the solutions returned, we run the four simple algorithms described in Chapter 6 and our optimization method on a total of about 100 instances obtained from our four case studies by varying the budget value.

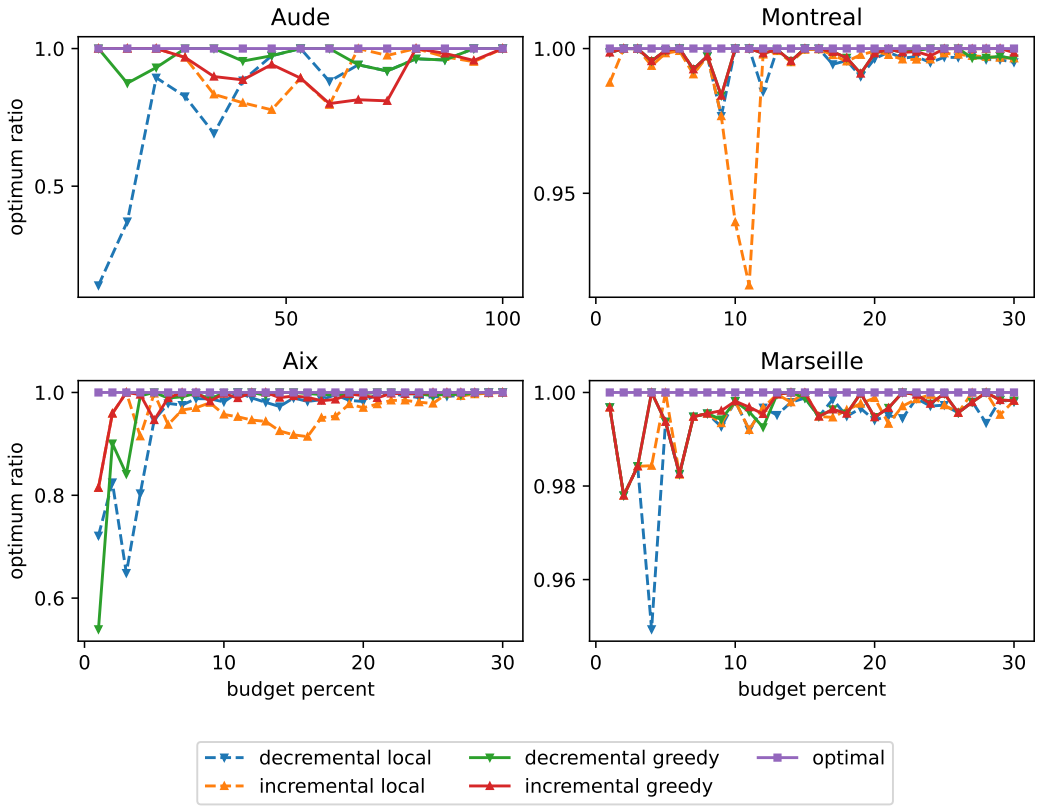


Figure 7.4: Percentage of gain in PC achieved by the solutions of the different algorithms compared to the optimal solution for different budget values.

	IL		DL		IG		DG	
	min.	avg.	min.	avg.	min.	avg.	min.	avg.
Aude	77.7 %	93.1 %	13.9 %	82.9 %	80 %	93 %	87.4 %	96.8 %
Montreal	91.8 %	99.2 %	97.7 %	99.6 %	98.4 %	99.8 %	98.4 %	99.8 %
Aix	81.5 %	96.1 %	64.9 %	95.6 %	81.5 %	98.6 %	53.9 %	97.3 %
Marseille	97.8 %	99.5 %	94.9 %	99.3 %	97.8 %	99.6 %	97.8 %	99.6 %

Table 7.3: Minimum and average optimilaty ratio for each algorithm and case study.

For each case study and each of the four algorithms, there is at least one budget value for which the quality of the solution is significantly lower than the quality of the optimal solution, the greatest departures being observed at lower budget values (Figure 7.4, Table 7.3). Greedy versions of incremental and decremental algorithms perform on average better than their static counterpart (Table 7.3). The minimum and average optimality ratio in the Aude and Aix cases is lower than in the other cases, for all algorithms (Table 7.3). This can be explained by the fact that the improvements seem to have a stronger impact on the distances between patches in the case of Aude and Aix. Static and greedy algorithms are generally quite close to the optimal solution (5% lower on average). However, all algorithms, whether static or greedy, incremental or decremental, provide poor quality solutions for some budget values (Table 7.3).

7.3.3 Execution Times

In order to evaluate the trade-offs between optimality and resolution time, we also recorded the execution times of all algorithms on the instances of the previous section.

For all the cases, the IL algorithm is the fastest one, taking milliseconds for the Aude case and of the order of seconds for the other cases (Fig. 7.5). The DL algorithm follows with a calculation time 2 to 4 times longer than the IL algorithm. The IG and DG algorithms computing time are respectively linearly increasing and linearly decreasing with the budget value, with some outliers in Montreal case. The IG algorithm takes a maximum of 10 milliseconds in the Aude case and a maximum of 30 seconds in the other cases. The DG algorithm takes a maximum of 20 milliseconds in the Aude case and few minutes in the others. The optimal solution takes in the order of 100 milliseconds for the Aude case, few minutes for the Montreal and Marseille cases and one hour for the Aix case.

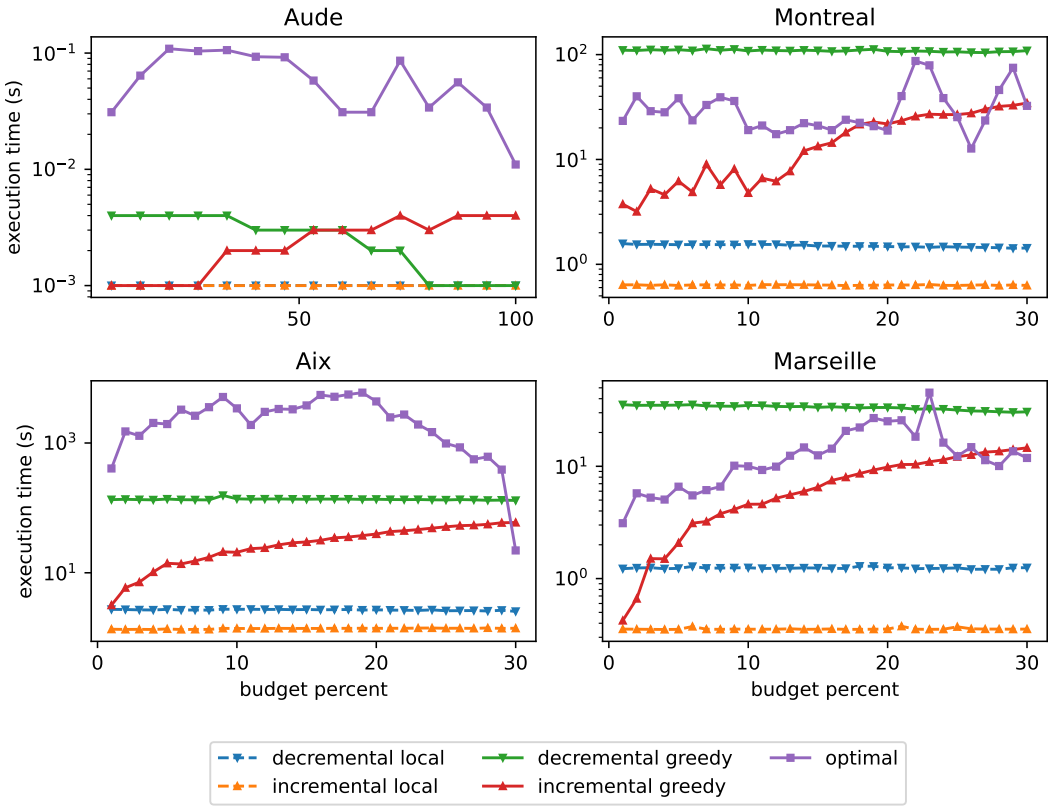


Figure 7.5: Execution time of the algorithms as a function of budget (log scale)

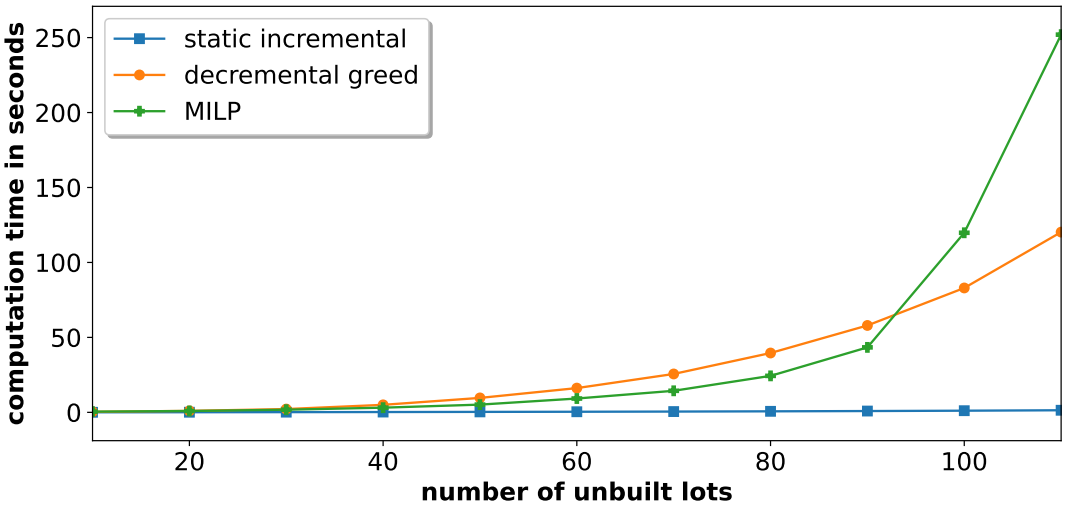


Figure 7.6: Computation times to solve the MILP on the Marseille case grow exponentially with the number of options while computation times of suboptimal algorithms grow only polynomially

7.4 Discussion

The mixed integer optimization algorithm always finds the best solution for a given budget. As expected, both the static and greedy algorithms lead to solutions that are mostly less good than the optimal solution, i.e., the options chosen lead to a lower PC value. As expected, the greedy algorithms also lead on average to solutions that are closer to the optimum than the static algorithms, because they partly account for the cumulated effect that can occur. All algorithms, whether static or greedy, incremental or decremental, deviate from the optimal solution for some budget values, which means that they do not keep pace with the performance of the optimal algorithm as the budget increases. These deviations mean that at some point, past decisions were not the best decisions given the new decisions that can be made as the budget increases. There are two possible and non-exclusive explanations to these deviations. On the one hand, the deviations could be due to the suboptimality of the greedy and static algorithms for the knapsack problem, a problem in combinatorial optimization. The knapsack problem seeks to determine, given a set of objects with a weight and a value, which objects should be chosen to fill a knapsack so that the total weight is less than or equal to its load limit and the total value is as large as possible [92]. It is similar to our problem of resource allocation, where the decision-makers have to choose from a set of non-divisible projects (options) under a fixed budget. Thus, even if there are no cumulative effects between the chosen options, static and greedy algorithms do not guarantee an optimal budget allocation [36]. However, this suboptimality issue only arises when the costs of the options are not all the same. Otherwise, i.e. when the options all have equal costs, the knapsack can easily be filled optimally by choosing the options with the highest values first. Unlike the other three cases, in the Aude case, all options have a similar cost, so we know that the observed deviations cannot be explained by the suboptimality of the greedy algorithm for the knapsack problem. On the other hand, these deviations could be due to synergies or redundancies of certain combinations of options, i.e., cumulated effects. The fact that the four simpler algorithms perform poorly in some arbitrary and unpredictable cases is due to the way they work sequentially, and their inability to question previous decisions based on novel ones. We illustrated trivial cases for which each of the static and greedy algorithms fail to reach good solutions in Chapter 6. Since in the Aude case all options have equal weights, we know that, in this case, the suboptimality is not related to the suboptimality of greedy algorithms for the knapsack problem. We thus assume that the observed deviations between the simpler algorithms and the optimal algorithm is due to the interactions among the potential options and the order in which they are chosen, similarly to the bad cases exposed in Chapter 6. Indeed, since the river system is represented by a tree (dendritic) graph, this case has some similarities with our schematic example above for the bad cases. In particular, sequential selections may overlook the large potential effects of selecting multiple options along a single branch if each individual option provides only poor improvements to the PC. In the other three cases, we assume that the observed deviations may be due to both explanations, the size and density of the graphs makes it difficult to understand the effect of each

option or budget level on the whole graph and its PC value. Computation time varies by several orders of magnitude between case studies and between algorithms. The computation time of the different algorithms depends on several elements, including: the search procedure, the size of the graph, the number of options, the budget, and the complexity of the problem (existence of synergies/redundancies among options), all being not necessarily independent. For both static algorithms, the computation time depends only on the number of possible options and the time needed to calculate PC (so the graph size), but only once for each option. That is why they are quick to compute, and that also explains why they are currently largely used in applied conservation problems. Here, decremental (less used) is a bit longer than incremental due to implementation procedure which implies more calculation steps for small budget values. For both greedy algorithms, the computation time depends on the number of options, but also largely on the size of the graph, as the time required to compute PC directly depends on it, and PC needs to be recalculated $N-k$ times for each k^{th} additional option. The computation time of the incremental greedy algorithm increases with the value of the budget, this is due to the fact that a larger budget allows for the inclusion of more options and therefore requires the addition of the computation of the effects of these additional options on top of those already chosen. For the optimal algorithm, the computation time depends on the space that needs to be explored on the decision tree of options, which is at most a function of 2^N (N being the number of options), but the optimization process allows a large reduction of this time by pruning the search space to remove non-optimal solutions, like the branch and bound method seen in Definition-Box 3. For this reason, the computation time of the optimal algorithm is shorter for smaller problems (smaller graphs with fewer options) but also when the problem is simpler (fewer complex interactions between the potential solutions) because more branches can be pruned earlier. Above a budget value the marginal gain on PC may be very small or even null. In the case of Aix, for instance, this arises around 30% of the budget (or 80% in the case of Aude), because the highest possible PC has been reached. We have chosen to illustrate our experiments by demonstrating the method on cases on which the four algorithms run easily and thus quantify the differences in the orders of magnitude required by the different algorithms. Obviously, no one is afraid to run a calculation for a few minutes, hours, or even days if they want to arrive at an efficient and cost-effective solution to their conservation problem. However, months or years quickly become problematic. As shown in Table 7.2 and Figure 7.6, the computation time needed to solve the mixed integer program grows exponentially with the number of options. When this computation time reaches several minutes, the maximum size of the instances that can be solved optimally in a reasonable time is almost reached. Indeed, with this exponential growth, adding a few more options would explode the computation time to several days or even months (see Fig. 7.6). Combining and comparing different conservation scenarios with different budgets, or different sets of possible options, would also only multiply the computation time by the number of different scenarios chosen.

In our four case studies, if the mixed integer optimization algorithm always finds

the best solution for a given budget, and if the quality of the solution obtained with the different algorithms depends on how much they account for the cumulated effects (optimal > greedy > static), the static and greedy algorithms are generally quite close to the optimal solution (5% lower on average). The largest departures from the optimum are observed at low budget values and, beyond a certain budget value all algorithms stay very close to the optimal solution. This convergence between the algorithms, especially for higher budget values, tends to show that in our cases the cumulated effects have low impact. However, our results also show some discrepancies between the cases, the solutions being on average better for Marseille and Montreal cases than for Aix and Aude cases (Fig. 7.4). These discrepancies could be explained by the density of these graphs (how close to complete they are). A higher (resp. lower) density, as in the case of Marseille (resp. Aude), leads to reach the maximum value of PC more quickly (resp. less quickly). Alternatively, the maximum value of PC is also reached more quickly when fewer options strongly impact the most probable paths between multiple vertices. This maximum value of PC is reached when the probabilities of connection between all habitat patches have reached their maximum value, or equivalently the distances among patches have reached their minimum. After this point, additional options are only redundant. This emerging property actually corresponds to something we believe is a shortcoming of the PC indicator which only takes into account the shortest path distance among pairs of vertices and not the number of short paths: redundancies taken into account by circuit connectivity measure [88] can lead to more robust networks. As we have illustrated our work with four very contrasted and complementary case studies, we believe that our results are rather robust and are not related to the modelling of the landscape (graph types) nor to the type of problem. We thus conclude that the optimal resolution is to be preferred when the landscape model is small enough (few hundred of habitat patches) to obtain a solution in a reasonable time. Our results show that the benefits of optimal resolution are greater on instances that are weakly connected or whose options have a strong impact on the most probable paths between multiple nodes.

Conclusion

In the face of the nature crisis that this planet is experiencing – while the solution is probably to address the anthropogenic causes [17, 94, 25] – landscape connectivity has been identified as a key element to mitigate biodiversity losses resulting from habitat loss and fragmentation. Consequently, decision-makers need reliable tools to select the most efficient and cost-effective actions to preserve or enhance biodiversity habitats. Indeed, current methods for prioritizing connectivity conservation/restoration actions are suboptimal because they largely ignore cumulated effects [104].

In this thesis we addressed the optimization of the Probability of Connectivity (PC) indicator [109], which is currently the most widely used indicator for assessing the priority of connectivity conservation and restoration actions [72]. We introduced the Budget-Constrained PC Optimization problem (BC-PC-Opt) that can capture both connectivity conservation and restoration problematics. We then proved that this problem is NP-Hard and inapproximable with a constant factor greater than $(1 + \frac{1}{e})$, and gave a mixed integer formulation for solving it efficiently. Our experiments show the scalability of our approach that now allows solving to optimality instances with few hundred of habitat patches whereas the previous models were limited to about 30 patches [131]. In addition to provide an optimization method for real-world instances of modest size, this contribution highlights the fact that the algorithms used in practice are suboptimal, even on realistic instances larger than the trivial bad cases, and now allows measuring how bad this suboptimality is. In our four case studies, although our mixed integer linear program always finds the best solution, the local and greedy algorithms are indeed quite close to the optimal solution (5% lower on average). In a broader context, we improved existing [21, 79] and introduced new preprocessing steps for shortest path problems on graphs whose arcs lengths are only known to lie in positive intervals. Our experiments show that these preprocessing steps play a key role in the scalability of our approach by significantly reducing the size of our MILP formulations. Furthermore, the problems of identifying t-strong and t-useless arcs, addressed by our algorithms, were previously studied for robust shortest path problems [71, 21] and have undoubtedly many other applications. For example, they would be of similar interest for optimizing another connectivity indicator based on shortest path computations.

On a practical side, it would be important to experiment our approaches on other practical instances of the BC-PC-Opt problem arising from different ecological contexts. Implementing the extension of our algorithms to constrained shortest path problems, described in Section 5.5, would address the added benefits of accounting for the budget limit while identifying arcs that can or cannot be part of an optimal solution. It would also be interesting to look for existing problems that could benefit

from our preprocessing techniques and conduct experiments to evaluate these benefits. At the same time, it seems very important to pursue the software development in order to provide a tool for practitioners to easily use our optimization methods.

On a theoretical side, since we showed the greedy algorithms can perform arbitrarily bad on some instance, it would be natural to look for some constant approximation algorithms. A first approach to tackle this question would be to find reasonable assumptions on the instances such that a greedy algorithm could a have a constant approximation ratio. If these assumptions are fulfilled by the real instances that we considered, this would explain our experimental observations. Moreover, since a polynomial time approximation scheme has been given in the case of trees [129], it could also be interesting to know for which larger classes of graphs constant factor approximation algorithms for this problem exist. Another interesting question is to determine whether good solutions could be obtained by decomposing geographically the problem, solving independently the subproblems of each region and then reassembling the solutions. In this case, a notion of fairness could help to allocate the budget among the regions so that each region can enhance its own *internal connectivity* keeping a part of the budget to enhance the connectivity between the regions. Another line of research would be to investigate the optimization of other connectivity indicators based on circuit theory [88]. Indeed, while not being used for prioritizing connectivity conservation and restoration actions, these indicators are gaining in popularity and the notion of effective resistance intimately relates to Semidefinite Programming (SDP), a generalization of linear programming [53].

Historically, the interaction between computer science and ecology has long been one-sided, with ecologists adapting pre-existing notions such as graph theory to their case studies. We hope that this thesis can serve as an example to encourage interdisciplinary exchanges which are much more interesting from a practical and theoretical point of view as they also stimulate the emergence of structural, metric and algorithmic questions that have sounded practical applications.

Bibliography

[1] Adriaensen, F., Chardon, J.P., De Blust, G., et al. “The application of ‘least-cost’ modelling as a functional landscape model”. In: *Landscape and Urban Planning* 64.4 (2003), pp. 233–247. DOI: 10.1016/S0169-2046(02)00242-6 (cit. on pp. 27, 36).

[2] Ahuja, Ravindra K., Magnanti, Thomas L., and Orlin, James B. “Generalized flows”. In: *Network Flows: Theory, Algorithms, and Applications*. Prentice-Hall, Inc., 1993. Chap. 15, pp. 566–597. ISBN: 978-0136175490 (cit. on p. 53).

[3] Ahuja, Ravindra K., Magnanti, Thomas L., and Orlin, James B. “Lagrangian relaxation and network optimization”. In: *Network Flows: Theory, Algorithms, and Applications*. Prentice-Hall, Inc., 1993. Chap. 16, pp. 598–648. ISBN: 978-0136175490 (cit. on p. 74).

[4] Ahuja, Ravindra K., Magnanti, Thomas L., and Orlin, James B. *Network Flows: Theory, Algorithms, and Applications*. Prentice-Hall, Inc., 1993. ISBN: 978-0136175490 (cit. on p. 52).

[5] Albert, Cécile H., Rayfield, Bronwyn, Dumitru, Maria, and Gonzalez, Andrew. “Applying network theory to prioritize multispecies habitat networks that are robust to climate and land-use change”. In: *Conservation Biology* 31 (2017), pp. 1383–1396. DOI: 10.1111/cobi.12943 (cit. on pp. 13, 27, 33, 36, 40, 93, 97).

[6] Alman, Josh and Williams, Virginia Vassilevska. “A Refined Laser Method and Faster Matrix Multiplication”. In: *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 522–539. DOI: 10.1137/1.9781611976465.32 (cit. on p. 86).

[7] R.E.A. Almond, M. Grooten, D. Juffe Bignoli, and T. Peterson, eds. *Living Planet Report 2020 - Building a nature- positive society*. World Wildlife Fund, 2022. ISBN: 978-2-88085-316-7. URL: https://wwflpr.awsassets.panda.org/downloads/lpr_2022_full_report.pdf (cit. on p. 12).

[8] Andren, Henrik. “Effects of habitat fragmentation on birds and mammals in landscapes with different proportions of suitable habitat: a review”. In: *Oikos* (1994), pp. 355–366. DOI: 10.2307/3545823 (cit. on p. 25).

[9] Arora, Sanjeev, Karger, David, and Karpinski, Marek. “Polynomial Time Approximation Schemes for Dense Instances of NP-Hard Problems”. In: *Journal of Computer and System Sciences* 58.1 (1999), pp. 193–210. ISSN: 0022-0000. DOI: 10.1006/jcss.1998.1605 (cit. on p. 47).

[10] Awade, Marcelo, Boscolo, Danilo, and Metzger, Jean Paul. "Using binary and probabilistic habitat availability indices derived from graph theory to model bird occurrence in fragmented forests". In: *Landscape Ecology* 27 (2012), pp. 185–198. DOI: 10.1007/s10980-011-9667-2 (cit. on p. 32).

[11] Beier, Paul, Spencer, Wayne, Baldwin, Robert F, and McRAE, BRAD H. "Toward best practices for developing regional connectivity maps". In: *Conservation Biology* 25.5 (2011), pp. 879–892. DOI: 10.1111/j.1523-1739.2011.01716.x (cit. on p. 13).

[12] Belotti, Pietro, Bonami, Pierre, Fischetti, Matteo, et al. "On handling indicator constraints in mixed integer programming". In: *Computational Optimization and Applications* 65.3 (2016), pp. 545–566. DOI: 10.1007/s10589-016-9847-8 (cit. on pp. 56, 65).

[13] Bhaskara, Aditya, Charikar, Moses, Chlamtac, Eden, Feige, Uriel, and Vijayaraghavan, Aravindan. "Detecting High Log-Densities: An O(N¼) Approximation for Densest k-Subgraph". In: *Proceedings of the Forty-Second ACM Symposium on Theory of Computing*. STOC '10. Association for Computing Machinery, 2010, pp. 201–210. DOI: 10.1145/1806689.1806719 (cit. on p. 47).

[14] Bodin, Örjan and Saura, Santiago. "Ranking individual habitat patches as connectivity providers: Integrating network analysis and patch removal experiments". In: *Ecological Modelling* 221.19 (2010), pp. 2393–2405. ISSN: 0304-3800. DOI: 10.1016/j.ecolmodel.2010.06.017 (cit. on pp. 30, 42).

[15] Bose, Prosenjit and Smid, Michiel. "On plane geometric spanners: A survey and open problems". In: *Computational Geometry* 46.7 (2013). EuroCG 2009, pp. 818–830. ISSN: 0925-7721. DOI: 10.1016/j.comgeo.2013.04.002 (cit. on p. 37).

[16] Box, George EP. "Robustness in the strategy of scientific model building". In: *Robustness in statistics*. Elsevier, 1979, pp. 201–236. DOI: 10.1016/B978-0-12-438150-6.50018-2 (cit. on p. 32).

[17] Brondizio, E. S., Settele, J., Díaz, S., and Ngo, H. T. *Global assessment report on biodiversity and ecosystem services of the Intergovernmental Science- Policy Platform on Biodiversity and Ecosystem Services*. IPBES, Bonn, Germany, 2019. ISBN: 978-3-947851-13-3. DOI: 10.5281/zenodo.3831673 (cit. on pp. 12, 104).

[18] Bunn, A.G, Urban, D.L, and Keitt, T.H. "Landscape connectivity: A conservation application of graph theory". In: *Journal of Environmental Management* 59.4 (2000), pp. 265–278. DOI: 10.1006/jema.2000.0373 (cit. on pp. 25, 26).

[19] Burel, Françoise. "Hedgerows and Their Role in Agricultural Landscapes". In: *Critical Reviews in Plant Sciences* 15.2 (1996), pp. 169–190. DOI: 10.1080/07352689.1996.10393185 (cit. on p. 24).

[20] Calabrese, Justin M. and Fagan, William F. "A comparison-shopper's guide to connectivity metrics". In: *Frontiers in Ecology and the Environment* 2.10 (2004), pp. 529–536. DOI: 10.1890/1540-9295(2004)002[0529:ACGTCM]2.0.CO;2 (cit. on p. 26).

[21] Catanzaro, Daniele, Labbé, Martine, and Salazar-Neumann, Martha. "Reduction approaches for robust shortest path problems". In: *Computers & OR* 38 (2011), pp. 1610–1619. DOI: 10.1016/j.cor.2011.01.022 (cit. on pp. 14, 15, 68, 104).

[22] Chvatal, Vasek, Chvatal, Vaclav, et al. *Linear programming*. Macmillan, 1983. ISBN: 9780716715870 (cit. on p. 52).

[23] Cobham, Alan. "The Intrinsic Computational Difficulty of Functions". In: *Logic, Methodology and Philosophy of Science: Proceedings of the 1964 International Congress (Studies in Logic and the Foundations of Mathematics)*. Ed. by Yehoshua Bar-Hillel. North-Holland Publishing, 1965, pp. 24–30. DOI: 10.2307/2270886 (cit. on p. 44).

[24] community, Boost. *Boost C++ libraries*. 2022. URL: https://www.boost.org (cit. on p. 91).

[25] Contribution of Working Group II to the Sixth Assessment Report of the Intergovernmental Panel on Climate Change : Pörtner, Hans-Otto, Roberts, Debra C, Adams, H, et al. *Climate Change 2022: Impacts, Adaptation and Vulnerability*. Cambridge University Press. Cambridge University Press, 2022. DOI: 10.1017/9781009325844 (cit. on pp. 12, 104).

[26] Cook, Stephen. "The importance of the P versus NP question". In: *Journal of the ACM (JACM)* 50.1 (2003), pp. 27–29. DOI: 10.1145/602382.602398 (cit. on p. 45).

[27] Cook, Stephen A. "The Complexity of Theorem-Proving Procedures". In: *Proceedings of the Third Annual ACM Symposium on Theory of Computing*. STOC '71. Association for Computing Machinery, 1971, pp. 151–158. ISBN: 9781450374644. DOI: 10.1145/800157.805047 (cit. on p. 44).

[28] Cormen, Thomas H, Leiserson, Charles E, Rivest, Ronald L, and Stein, Clifford. *Introduction to algorithms*. MIT press, 2009. ISBN: 978-0-262-53305-8 (cit. on pp. 22, 81).

[29] Cornwall, Christopher E., Comeau, Steeve, Kornder, Niklas A., et al. "Global declines in coral reef calcium carbonate production under ocean acidification and warming". In: *Proceedings of the National Academy of Sciences* 118.21 (2021), e2015265118. DOI: 10.1073/pnas.2015265118 (cit. on p. 12).

[30] Crook, David A. "Is the home range concept compatible with the movements of two species of lowland river fish?" In: *Journal of Animal Ecology* 73.2 (2004), pp. 353–366. DOI: 10.1111/j.0021-8790.2004.00802.x (cit. on p. 92).

[31] Crooks, Kevin R. and Sanjayan, M. *Connectivity conservation*. Vol. 14. Cambridge University Press, 2006. ISBN: 978-0521673815 (cit. on p. 13).

[32] Dantzig, George. *Linear Programming and Extensions*. Princeton University Press, 1963. ISBN: 9781400884179. DOI: 10.1515/9781400884179 (cit. on p. 52).

[33] Demetrescu, Camil and Italiano, Giuseppe F. "A new approach to dynamic all pairs shortest paths". In: *Journal of the ACM (JACM)* 51.6 (2004), pp. 968–992. DOI: 10.1145/1039488.1039492 (cit. on p. 87).

[34] Dezs, Balázs, Jüttner, Alpár, and Kovács, Péter. "LEMON - an Open Source C++ Graph Template Library". In: *Electron. Notes Theor. Comput. Sci.* 264.5 (2011), pp. 23–45. DOI: 10.1016/j.entcs.2011.06.003 (cit. on pp. 91, 94).

[35] Dijkstra, Edsger W et al. "A note on two problems in connexion with graphs". In: *Numerische mathematik* 1.1 (1959), pp. 269–271. DOI: 10.1007/BF01386390 (cit. on pp. 21, 28, 67).

[36] Dilkina, Bistra, Lai, Katherine J., and Gomes, Carla P. "Upgrading Shortest Paths in Networks". In: *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*. Springer Berlin Heidelberg, 2011, pp. 76–91. DOI: 10.1007/978-3-642-21311-3_9 (cit. on p. 101).

[37] Dunning, John B, Danielson, Brent J, and Pulliam, H Ronald. "Ecological processes that affect populations in complex landscapes". In: *Oikos* (1992), pp. 169–175. DOI: 10.2307/3544901 (cit. on p. 23).

[38] Enderton, Herbert B. *Elements of set theory*. Academic press, 1977. ISBN: 978-0-12-238440-0 (cit. on p. 18).

[39] Erős, Tibor and Lowe, Winsor H. "The landscape ecology of rivers: from patch-based to spatial network analyses". In: *Current Landscape Ecology Reports* 4.4 (2019), pp. 103–112. DOI: 10.1007/s40823-019-00044-6 (cit. on p. 92).

[40] Fahrig, Lenore. "Effects of habitat fragmentation on biodiversity". In: *Annual review of ecology, evolution, and systematics* (2003), pp. 487–515. DOI: 10.1146/annurev.ecolsys.34.011802.132419 (cit. on pp. 24, 25).

[41] Fall, Andrew, Fortin, Marie-Josee, Manseau, Micheline, and O'Brien, Dan. "Spatial graphs: principles and applications for habitat connectivity". In: *Ecosystems* 10.3 (2007), pp. 448–461. DOI: 10.1007/s10021-007-9038-7 (cit. on pp. 36–38, 93).

[42] Feige, Uriel. "A Threshold of Ln n for Approximating Set Cover". In: *J. ACM* 45.4 (July 1998), pp. 634–652. DOI: 10.1145/285055.285059 (cit. on p. 50).

[43] Feige, Uriel, Peleg, David, and Kortsarz, Guy. "The dense k-subgraph problem". In: *Algorithmica* 29.3 (2001), pp. 410–421. DOI: 10.1007/s004530010050 (cit. on p. 47).

[44] Floyd, Robert W. "Algorithm 97: Shortest Path". In: *Commun. ACM* 5.6 (1962). DOI: 10.1145/367766.368168 (cit. on pp. 82, 86).

[45] Foley, Melissa M., Mease, Lindley A., Martone, Rebecca G., et al. "The challenges and opportunities in cumulative effects assessment". In: *Environmental Impact Assessment Review* 62 (2017), pp. 122–134. ISSN: 0195-9255. DOI: 10.1016/j.eiar.2016.06.008 (cit. on p. 13).

[46] Foltête, Jean-Christophe, Girardet, Xavier, and Clauzel, Céline. "A methodological framework for the use of landscape graphs in land-use planning". In: *Landscape and Urban Planning* 124 (2014), pp. 140–150. DOI: 10.1016/j.landurbplan.2013.12.012 (cit. on pp. 42, 43, 82).

[47] Forman, Richard TT and Baudry, Jacques. "Hedgerows and hedgerow networks in landscape ecology". In: *Environmental management* 8.6 (1984), pp. 495–510. DOI: 10.1007/BF01871575 (cit. on p. 24).

[48] Forrest, John, Ralphs, Ted, Santos, Haroldo Gambini, et al. *coin-or/Cbc: Release releases/2.10.8*. 2022. DOI: 10.5281/zenodo.6522795 (cit. on p. 92).

[49] Gács, Peter and Lovász, Laszlo. "Khachiyan's algorithm for linear programming". In: *Mathematical Programming at Oberwolfach*. Springer Berlin Heidelberg, 1981, pp. 61–68. DOI: 10.1007/BFb0120921 (cit. on p. 52).

[50] Galpern, Paul, Manseau, Micheline, and Fall, Andrew. "Patch-based graphs of landscape connectivity: A guide to construction, analysis and application for conservation". In: *Biological Conservation* 144.1 (2011), pp. 44–55. DOI: 10.1016/j.biocon.2010.09.002 (cit. on p. 37).

[51] García-Feced, C., Saura, S., and Elena-Rosselló, R. "Improving landscape connectivity in forest districts: A two-stage process for prioritizing agricultural patches for reforestation". In: *Forest Ecology and Management* 261.1 (2011), pp. 154–161. DOI: 10.1016/j.foreco.2010.09.047 (cit. on pp. 42, 43).

[52] Garey, Michael R and Johnson, David S. "Computers and intractability". In: *A Guide to the* (1979) (cit. on p. 45).

[53] Ghosh, Arpita, Boyd, Stephen, and Saberi, Amin. "Minimizing Effective Resistance of a Graph". In: *SIAM Review* 50.1 (2008), pp. 37–66. DOI: 10.1137/050645452 (cit. on p. 105).

[54] Grech, Alana, Hanert, Emmanuel, McKenzie, Len, et al. "Predicting the cumulative effect of multiple disturbances on seagrass connectivity". In: *Global change biology* 24.7 (2018), pp. 3093–3104. DOI: 10.1111/gcb.14127 (cit. on p. 13).

[55] Gurobi Optimization, LLC. *Gurobi Optimizer Reference Manual*. 2022. URL: https://www.gurobi.com (cit. on pp. 91, 92, 94).

[56] Haberl, Helmut, Wiedenhofer, Dominik, Virág, Doris, et al. "A systematic review of the evidence on decoupling of GDP, resource use and GHG emissions, part II: synthesizing the insights". In: *Environmental Research Letters* 15.6 (June 2020), p. 065003. DOI: 10.1088/1748-9326/ab842a (cit. on p. 12).

[57] Haddad, Nick M, Brudvig, Lars A, Clobert, Jean, et al. “Habitat fragmentation and its lasting impact on Earth’s ecosystems”. In: *Science advances* 1.2 (2015), e1500052. DOI: 10.1126/sciadv.1500052 (cit. on p. 24).

[58] Hallmann, Caspar A., Sorg, Martin, Jongejans, Eelke, et al. “More than 75 percent decline over 27 years in total flying insect biomass in protected areas”. In: *PLOS ONE* 12.10 (Oct. 2017), pp. 1–21. DOI: 10.1371/journal.pone.0185809 (cit. on p. 12).

[59] Hamonic, François, Albert, Cécile, Couëtoux, Basile, and Vaxès, Yann. “Optimizing the ecological connectivity of landscapes”. In: *Networks* (2022). DOI: 10.1002/net.22131 (cit. on pp. 3, 91).

[60] Handler, Gabriel Y. and Zang, Israel. “A dual algorithm for the constrained shortest path problem”. In: *Networks* 10.4 (1980), pp. 293–309. DOI: 10.1002/net.3230100403 (cit. on p. 74).

[61] Hanson, Jeffrey O, Schuster, Richard, Strimas-Mackey, Matthew, and Bennett, Joseph R. “Optimality in prioritizing conservation projects”. In: *Methods in Ecology and Evolution* 10.10 (2019), pp. 1655–1663. DOI: 10.1111/2041-210X.13264 (cit. on p. 81).

[62] Hashemi, Rastegar and Darabi, Hassan. “The Review of Ecological Network Indicators in Graph Theory Context: 2014-2021”. In: *International Journal of Environmental Research* 16 (2022). DOI: 10.1007/s41742-022-00404-x (cit. on pp. 27, 32).

[63] Heller, Nicole E. and Zavaleta, Erika S. “Biodiversity management in the face of climate change: A review of 22 years of recommendations”. In: *Biological Conservation* 142.1 (2009), pp. 14–32. DOI: 10.1016/j.biocon.2008.10.006 (cit. on p. 13).

[64] Hodgson, Jenny A., Moilanen, Atte, Wintle, Brendan A., and Thomas, Chris D. “Habitat area, quality and connectivity: striking the balance for efficient conservation”. In: *Journal of Applied Ecology* 48.1 (2011), pp. 148–152. DOI: 10.1111/j.1365-2664.2010.01919.x (cit. on p. 33).

[65] Howey, Meghan CL. “Multiple pathways across past landscapes: circuit theory as a complementary geospatial method to least cost path for modeling past movement”. In: *Journal of Archaeological Science* 38.10 (2011), pp. 2523–2535. DOI: 10.1016/j.jas.2011.03.024 (cit. on p. 31).

[66] Institut national de l’information géographique et forestière. *Remonter le temps*. 2022. URL: https://remonterletemps.ign.fr/comparer/basic?x=-0.777181&y=49.208000&z=15&layer1=ORTHOIMAGERY.ORTHOPHOTOS.1950-1965&layer2=ORTHOIMAGERY.ORTHOPHOTOS2000-2005&mode=doubleMap (cit. on p. 24).

[67] Irnich, Stefan and Desaulniers, Guy. "Shortest Path Problems with Resource Constraints". In: *Column Generation*. Ed. by Guy Desaulniers, Jacques Desrosiers, and Marius M. Solomon. Springer US, 2005, pp. 33–65. DOI: 10.1007/0-387-25486-2_2 (cit. on p. 74).

[68] Jaeger, Jochen AG. "Landscape division, splitting index, and effective mesh size: new measures of landscape fragmentation". In: *Landscape ecology* 15.2 (2000), pp. 115–130. DOI: 10.1023/A:1008129329289 (cit. on pp. 25, 26).

[69] Joksch, H.C. "The shortest route problem with constraints". In: *Journal of Mathematical Analysis and Applications* 14.2 (1966), pp. 191–197. DOI: 10.1016/0022-247X(66)90020-5 (cit. on p. 74).

[70] Justeau-Allaire, Dimitri. "Planification systématique de la conservation basée sur les contraintes, une approche générique et expressive : application à l'aide à la décision pour la conservation des forêts de Nouvelle-Calédonie". PhD thesis. Université Montpellier, 2020. URL: https://tel.archives-ouvertes.fr/tel-03329679 (cit. on p. 14).

[71] Karaşan, OE, Pinar, MÇ, and Yaman, H. "The robust shortest path problem with interval data". In: 2001. URL: https://optimization-online.org/2001/08/361/ (cit. on pp. 15, 67, 104).

[72] Keeley, Annika T.H., Beier, Paul, and Jenness, Jeff S. "Connectivity metrics for conservation planning and monitoring". In: *Biological Conservation* 255 (2021), p. 109008. DOI: 10.1016/j.biocon.2021.109008 (cit. on pp. 14, 27, 31, 104).

[73] Keil, J. Mark and Gutwin, Carl A. "Classes of Graphs Which Approximate the Complete Euclidean Graph". In: *Discrete Comput. Geom.* 7.1 (Dec. 1992), pp. 13–28. DOI: 10.5555/2805869.2805935 (cit. on p. 37).

[74] Khot, Subhash. "Ruling Out PTAS for Graph Min-Bisection, Dense k-Subgraph, and Bipartite Clique". In: *SIAM Journal on Computing* 36.4 (2006), pp. 1025–1071. DOI: 10.1137/S0097539705447037 (cit. on p. 47).

[75] King, S. and O'Hanley, J. R. "Optimal Fish Passage Barrier Removal–Revisited". In: *River Research and Applications* 32.3 (2016), pp. 418–428. DOI: 10.1002/rra.2859 (cit. on p. 43).

[76] Land, A. H. and Doig, A. G. "An Automatic Method of Solving Discrete Programming Problems". In: *Econometrica* 28.3 (1960), pp. 497–520. ISSN: 00129682, 14680262. DOI: 10.2307/1910129 (cit. on p. 63).

[77] Lawler, Joshua J. "Climate change adaptation strategies for resource management and conservation planning". In: *Annals of the New York Academy of Sciences* 1162.1 (2009), pp. 79–98. DOI: 10.1111/j.1749-6632.2009.04147.x (cit. on p. 13).

[78] Levin, Leonid. "Universal search problems (in Russian)". In: vol. 9. 3. 1973, pp. 115–116. ISBN: 9781450374644 (cit. on p. 44).

[79] Lindner, Niels, Maristany de las Casas, Pedro, and Schiewe, Philine. "Optimal Forks: Preprocessing Single-Source Shortest Path Instances with Interval Data". In: Open Access Series in Informatics (OASIcs) 96 (2021), 7:1–7:15. DOI: 10.4230/OASIcs.ATMOS.2021.7 (cit. on pp. 15, 68, 104).

[80] Lovász, L., Pelikán, J., and Vesztergombi, K. "Euler's Formula". In: *Discrete Mathematics: Elementary and Beyond*. Springer New York, 2003, pp. 189–196. ISBN: 978-0-387-21777-2. DOI: 10.1007/0-387-21777-0_12 (cit. on p. 37).

[81] Magris, Rafael A, Treml, Eric A, Pressey, Robert L, and Weeks, Rebecca. "Integrating multiple species connectivity and habitat quality into conservation planning for coral reefs". In: *Ecography* 39.7 (2016), pp. 649–664. DOI: 10.1111/ecog.01507 (cit. on p. 13).

[82] Mardani, Abbas, Streimikiene, Dalia, Cavallaro, Fausto, Loganathan, Nanthakumar, and Khoshnoudi, Masoumeh. "Carbon dioxide (CO2) emissions and economic growth: A systematic review of two decades of research from 1995 to 2017". In: *Science of The Total Environment* 649 (2019), pp. 31–49. ISSN: 0048-9697. DOI: https://doi.org/10.1016/j.scitotenv.2018.08.229 (cit. on p. 12).

[83] Matoušek, Jiří and Gärtner, Bernd. *Understanding and using linear programming*. Vol. 33. Springer, 2007. ISBN: 978-3-540-30697-9. DOI: 10.1007/978-3-540-30717-4 (cit. on p. 52).

[84] McClure, Meredith L, Hansen, Andrew J, and Inman, Robert M. "Connecting models to movements: testing connectivity model predictions against empirical migration and dispersal data". In: *Landscape Ecology* 31.7 (2016), pp. 1419–1432. DOI: 10.1007/s10980-016-0347-0 (cit. on p. 31).

[85] McCormick, Garth P. "Computability of global solutions to factorable nonconvex programs: Part I - Convex underestimating problems". In: *Mathematical Programing* 10.1 (1976), pp. 147–175. DOI: 10.1007/BF01580665 (cit. on pp. 56, 65).

[86] McGarigal, Kevin and Marks, Barbara J. "Spatial pattern analysis program for quantifying landscape structure". In: *Gen. Tech. Rep. PNW-GTR-351. US Department of Agriculture, Forest Service, Pacific Northwest Research Station* (1995), pp. 1–122. DOI: 10.2737/PNW-GTR-351 (cit. on p. 25).

[87] McRae, BH, Shah, Viral, and Edelman, Alan. "Circuitscape: modeling landscape connectivity to promote conservation and human health". In: *The Nature Conservancy* 14 (2016). DOI: 10.13140/RG.2.1.4265.1126 (cit. on p. 31).

[88] McRae, Brad H., Dickson, Brett G., Keitt, Timothy H., and Shah, Viral B. "Using circuit theory to model connectivity in ecology, evolution, and conservation". In: *Ecology* 89 (2008), pp. 2712–2724. DOI: 10.1890/07-1861.1 (cit. on pp. 13, 27, 31, 103, 105).

[89] Moilanen, Atte, Franco, Aldina MA, Early, Regan I, Fox, Richard, Wintle, Brendan, and Thomas, Chris D. "Prioritizing multiple-use landscapes for conservation: methods for large multi-species planning problems". In: *Proceedings of the Royal Society B: Biological Sciences* 272.1575 (2005), pp. 1885–1891. DOI: 10.1098/rspb.2005.3164 (cit. on p. 82).

[90] Morin, Dana J., Fuller, Angela K., Royle, J. Andrew, and Sutherland, Chris. "Model-based estimators of density and connectivity to inform conservation of spatially structured populations". In: *Ecosphere* 8.1 (2017), e01623. DOI: 10.1002/ecs2.1623 (cit. on pp. 27, 29, 34, 36).

[91] Narasimhan, G. (Giri) and Smid, Michiel. *Geometric spanner networks*. Jan. 2007. ISBN: 978-0-521-81513-0. DOI: 10.1017/CBO9780511546884 (cit. on p. 37).

[92] Nemhauser, George and Wolsey, Laurence. *Integer and Combinatorial Optimization*. John Wiley & Sons, Ltd, 1988, pp. 1–26. ISBN: 9781118627372. DOI: 10.1002/9781118627372 (cit. on p. 101).

[93] Nonner, Tim. "PTAS for Densest-k-Subgraph in Interval Graphs". In: *Algorithmica* 74.1 (2016), pp. 528–539. DOI: 10.1007/s00453-014-9956-7 (cit. on p. 47).

[94] Otero, Iago, Farrell, Katharine N., Pueyo, Salvador, et al. "Biodiversity policy beyond economic growth". In: *Conservation Letters* 13.4 (2020), e12713. DOI: https://doi.org/10.1111/conl.12713 (cit. on pp. 12, 104).

[95] Paradis, Emmanuel, Baillie, Stephen R, Sutherland, William J, and Gregory, Richard D. "Patterns of natal and breeding dispersal in birds". In: *Journal of Animal ecology* 67.4 (1998), pp. 518–536. DOI: 10.1046/j.1365-2656.1998.00215.x (cit. on p. 94).

[96] Pascual-Hortal, Lucía and Saura, Santiago. "Comparison and development of new graph-based landscape connectivity indices: Towards the priorisation of habitat patches and corridors for conservation". In: *Landscape Ecology* 21 (2006), pp. 959–967. DOI: 10.1007/s10980-006-0013-z (cit. on pp. 11, 13, 26).

[97] Paul Chew, L. "There are planar graphs almost as good as the complete graph". In: *Journal of Computer and System Sciences* 39.2 (1989), pp. 205–219. ISSN: 0022-0000. DOI: 10.1016/0022-0000(89)90044-5 (cit. on p. 37).

[98] Pereira, Juliana, Saura, Santiago, and Jordán, Ferenc. "Single-node vs. multi-node centrality in landscape graph analysis: Key habitat patches and their protection for 20 bird species in NE Spain". In: *Methods in Ecology and Evolution* 8 (2017), pp. 1458–1467. DOI: 10.1111/2041-210X.12783 (cit. on pp. 14, 42).

[99] Pereira, Miguel, Segurado, Pedro, and Neves, Nuno. "Using spatial network structure in landscape management and planning: A case study with pond turtles". In: *Landscape and Urban Planning* 100 (2011), pp. 67–76. ISSN: 0169-2046. DOI: 10.1016/j.landurbplan.2010.11.009 (cit. on p. 32).

[100] Pérez-Galarce, Francisco, Candia-Véjar, Alfredo, Maculan, Guido, and Maculan, Nelson. "Improved robust shortest paths by penalized investments". In: *RAIRO Oper. Res.* 55 (2021), pp. 1865–1883. DOI: 10.1051/RO\%2F2021086 (cit. on p. 74).

[101] Pheatt, Chuck. "Intel® threading building blocks". In: *Journal of Computing Sciences in Colleges* 23.4 (2008), pp. 298–298. DOI: 10.5555/1352079.1352134 (cit. on pp. 91, 94).

[102] Pinto, Naiara and Keitt, Timothy H. "Beyond the least-cost path: evaluating corridor redundancy using a graph-theoretic approach". In: *Landscape Ecology* 24.2 (2009), pp. 253–266. DOI: 10.1007/s10980-008-9303-y (cit. on p. 32).

[103] Poli, Caroline, Hightower, Jessica, and Fletcher Jr., Robert J. "Validating network connectivity with observed movement in experimental landscapes undergoing habitat destruction". In: *Journal of Applied Ecology* 57.7 (2020), pp. 1426–1437. DOI: 10.1111/1365-2664.13624 (cit. on p. 31).

[104] Rubio, Lidón, Bodin, Örjan, Brotons, Lluís, and Saura, Santiago. "Connectivity conservation priorities for individual patches evaluated in the present landscape: How durable and effective are they in the long term?" In: *Ecography* 38 (2015), pp. 782–791. DOI: 10.1111/ecog.00935 (cit. on pp. 14, 42, 104).

[105] Rudnick, Deborah A., Ryan, Sadie J, Beier, Paul, et al. "The role of landscape connectivity in planning and implementing conservation and restoration priorities". In: *Issues in Ecology* 16 (2012), pp. 1–23. ISSN: 1092-8987 (cit. on p. 26).

[106] Saint-Pé, Keoni. "In situ quantification of brown trout movements". PhD thesis. Université Paul Sabatier-Toulouse III, 2019. URL: https://tel.archives-ouvertes.fr/tel-02942828/document (cit. on p. 92).

[107] Saura, Santiago, Bastin, Lucy, Battistella, Luca, Mandrici, Andrea, and Dubois, Grégoire. "Protected areas in the world's ecoregions: How well connected are they?" In: *Ecological Indicators* 76 (2017), pp. 144–158. DOI: 10.1016/j.ecolind.2016.12.047 (cit. on pp. 27, 29).

[108] Saura, Santiago, Estreguil, Christine, Mouton, Coralie, and Rodríguez-Freire, Mónica. "Network Analysis to Assess Landscape Connectivity Trends: Application to European Forests (1990-2000)". In: *Ecological Indicators* 11 (2011), pp. 407–416. DOI: 10.1016/j.ecolind.2010.06.011 (cit. on pp. 11, 27–29).

[109] Saura, Santiago and Pascual-Hortal, Lucía. "A new habitat availability index to integrate connectivity in landscape conservation planning: Comparison with existing indices and application to a case study". In: *Landscape and Urban Planning* 83 (2007), pp. 91–103. ISSN: 0169-2046. DOI: 10.1016/j.landurbplan.2007.03.005 (cit. on pp. 13, 26, 30, 32, 33, 36, 104).

[110] Saura, Santiago and Rubio, Lidón. "A common currency for the different ways in which patches and links can contribute to habitat availability and connectivity in the landscape". In: *Ecography* 33.3 (2010), pp. 523–537. DOI: 10.1111/j.1600-0587.2009.05760.x (cit. on p. 30).

[111] Schrijver, Alexander. *Combinatorial Optimization*. Springer Berlin, Heidelberg, 2003. ISBN: 978-3-540-44389-6 (cit. on pp. 21, 81).

[112] Segurado, Pedro, Branco, Paulo, and Ferreira, Maria T. "Prioritizing restoration of structural connectivity in rivers: a graph based approach". In: *Landscape Ecology* 28.7 (2013), pp. 1231–1238. DOI: 10.1007/s10980-013-9883-z (cit. on p. 92).

[113] Shi, Qinru, Gomes-Selman, Jonathan M., García-Villacorta, Roosevelt, Sethi, Suresh, Flecker, Alexander S., and Gomes, Carla P. "Efficiently Optimizing for Dendritic Connectivity on Tree-Structured Networks in a Multi-Objective Framework". In: *Proceedings of the 1st ACM SIGCAS Conference on Computing and Sustainable Societies*. COMPASS '18. Association for Computing Machinery, 2018. DOI: 10.1145/3209811.3209878 (cit. on p. 43).

[114] Sotirov, Renata. "On solving the densest k-subgraph problem on large graphs". In: *Optimization Methods and Software* 35.6 (2020), pp. 1160–1178. DOI: 10.1080/10556788.2019.1595620 (cit. on p. 47).

[115] Tambosi, Leandro R, Martensen, Alexandre C, Ribeiro, Milton C, and Metzger, Jean P. "A framework to optimize biodiversity restoration efforts based on habitat amount and landscape connectivity". In: *Restoration ecology* 22.2 (2014), pp. 169–177. DOI: 10.1111/rec.12049 (cit. on pp. 34, 42).

[116] Tarabon, Simon, Bergès, Laurent, Dutoit, Thierry, and Isselin-Nondedeu, Francis. "Maximizing habitat connectivity in the mitigation hierarchy. A case study on three terrestrial mammals in an urban environment". In: *Journal of environmental management* 243 (2019), pp. 340–349. DOI: 10.1016/j.jenvman.2019.04.121 (cit. on p. 13).

[117] Taylor, Philip D., Fahrig, Lenore, Henein, Kringen, and Merriam, Gray. "Connectivity Is a Vital Element of Landscape Structure". In: *Oikos* 68 (1993), pp. 571–573. DOI: 10.2307/3544927 (cit. on pp. 13, 25, 32).

[118] Taylor, Philip D., Fahrig, Lenore, and With, Kimberly A. "Landscape connectivity: a return to the basics". In: *Connectivity Conservation*. Ed. by Kevin R. Crooks and M. Sanjayan. Vol. 14. Cambridge University Press, 2006, pp. 29–43. DOI: 10.1017/CBO9780511754821.003 (cit. on pp. 25, 38).

[119] Tischendorf, Lutz and Fahrig, Lenore. "On the usage and measurement of landscape connectivity". In: *Oikos* 90.1 (2000), pp. 7–19. DOI: 10.1034/j.1600-0706.2000.900102.x (cit. on p. 25).

[120] Turner, Monica Goigel. "Landscape ecology: the effect of pattern on process". In: *Annual review of ecology and systematics* (1989), pp. 171–197. DOI: 10.1146/annurev.es.20.110189.001131 (cit. on p. 23).

[121] Urban, Dean and Keitt, Timothy. "Landscape connectivity: A graph-theoretic perspective". In: *Ecology* 82 (2001), pp. 1205–1218. DOI: 10.1890/0012-9658(2001)082[1205:LCAGTP]2.0.CO;2 (cit. on pp. 13, 25, 26, 42).

[122] Vazirani, Vijay V. *Combinatorial Optimization*. Springer Berlin, Heidelberg, 2003. ISBN: 978-3-540-65367-7. DOI: 10.1007/978-3-662-04565-7 (cit. on p. 45).

[123] Verbel, Arturo, Rodriguez, Nestor, and Rojas–Galeano, Sergio. "A Simple Yet Effective Algorithm to Compute Incremental All-Pairs Shortest Distances". In: *Applied Computer Sciences in Engineering*. Springer International Publishing, 2020, pp. 222–229. DOI: 10.1007/978-3-030-61834-6_19 (cit. on p. 87).

[124] Vitikainen, Arvo. "An overview of land consolidation in Europe". In: *Nordic Journal of Surveying and real Estate research* 1.1 (2004), pp. 25–43. ISSN: 1459-5877 (cit. on p. 23).

[125] Ward, Michelle, Saura, Santiago, Williams, Brooke, et al. "Just ten percent of the global terrestrial protected area network is structurally connected via intact land". In: *Nature communications* 11.1 (2020), pp. 1–10. DOI: 10.1038/s41467-020-18457-x (cit. on pp. 27, 29).

[126] Watts, Kevin and Handley, Phillip. "Developing a functional connectivity indicator to detect change in fragmented landscapes". In: *Ecological Indicators* 10.2 (2010), pp. 552–557. DOI: 10.1016/j.ecolind.2009.07.009 (cit. on pp. 27, 29).

[127] Whitehead, Amy L, Kujala, Heini, and Wintle, Brendan A. "Dealing with cumulative biodiversity impacts in strategic environmental assessment: A new frontier for conservation planning". In: *Conservation letters* 10.2 (2017), pp. 195–204. DOI: 10.1111/conl.12260 (cit. on p. 13).

[128] Wiens, John A. "Spatial scaling in ecology". In: *Functional ecology* 3.4 (1989), pp. 385–397. DOI: 10.2307/2389612 (cit. on p. 23).

[129] Wu, Xiaojian, Sheldon, Daniel, and Zilberstein, Shlomo. "Stochastic Network Design in Bidirected Trees". In: *Proceedings of the Twenty-Eighth Neural Information Processing Systems Conference*. 2014, pp. 882–890. URL: http://rbr.cs.umass.edu/shlomo/papers/WSZnips14.html (cit. on pp. 14, 43, 105).

[130] Xia, Ge. "The Stretch Factor of the Delaunay Triangulation Is Less than 1.998". In: *SIAM Journal on Computing* 42.4 (2013), pp. 1620–1659. DOI: 10.1137/110832458 (cit. on p. 37).

[131] Xue, Yexiang, Wu, Xiaojian, Morin, Dana, et al. "Dynamic Optimization of Landscape Connectivity Embedding Spatial-capture-recapture Information". In: *31st AAAI Conference on Artificial Intelligence*. Vol. 31. 2017, pp. 4552–4558. URL: https://ojs.aaai.org/index.php/AAAI/article/view/11175 (cit. on pp. 14, 15, 44, 104).

[132] Yaverian, Nicolas. *Optimisation de la connectivité des paysages écologiques*. May 2021 (cit. on p. 81).

[133] Yu, Gang and Yang, Jian. “On the Robust Shortest Path Problem”. In: *Computers & Operations Research* 25.6 (1998), pp. 457–468. DOI: 10.1016/S0305-0548(97)00085-3 (cit. on p. 68).