الجمهورية الجزائرية الديمقراطية الشعبية
People's Democratic Republic of Algeria
وزارة التعليم العالي والبحث العلمي
Ministry of Higher Education and Scientific Research

N Réf :……

University Center Abdelhafid Boussouf - Mila
Institute of Mathematics and Computer Science
Department of Computer Science

Dissertation prepared with a view to obtaining a Master's degree

In computer science

Specialty: Artificial intelligence and its applications (I2A)

Applying Cellular Automata and Optimization Techniques to Develop and Calibrate a Forest Fire Simulation Model

Prepared by:

HADDAD Takoua

Jury:

Supervised by	Dr. BENHAMMADA Sadek	MCA	C.U. Mila
President	Dr. GUEMRI Oualid	MCB	C.U. Mila
Examiner	Dr. KHALFI Souheila	MCA	C.U. Mila

Academic year: 2023 - 2024

Acknowledgment

I would like to thank Almighty God for giving me the strength and patience to complete this modest work.

I would especially like to thank Dr. Benhammeda Sadek, my supervisor, for his presence, his help and his invaluable advice, which greatly assisted me in the completion of this project.

I would also like to thank the members of the jury for their interest in this work, by agreeing to examine it and enrich it with their proposals.

I would also like to thank all the people who have helped me, in one way or another, in the realization of this work, in particular my parents and friends who have encouraged and supported me throughout my studies.

Finally, I would like to thank all the students of the I2A graduating class and all those who collaborated in any way in the development of this work.

HADDAD Takoua

Dedication

بسم الله الرحمن الرحيم

من قال أنا لها "نالها"

وأنا لها إن أبت رغما عنها أتيت بها.

الحمد لله الذي ما نجحنا وما علونا ولا تفوقنا إلا برضاه الحمد لله الذي ما اجتزنا دربا ولا تخطينا جهدا إلا بفضله وإليه ينسب الفضل والكمال والإكمال؛

إلى من زين اسمي بأجمل الألقاب, من دعمني بلا حدود وأعطاني بلا مقابل سندي وقوتي وملاذي بعد الله. فخري واعتزازي؛

والدي"الطيب"حفظك الله لي؛

إلى من جعل الله الجنة تحت أقدامها, من سهلت لي الشدائد بدعائها, سر قوتي ونجاحي. جنتي

والدتي "نور الهدى" متعها الله بالصحة و العافية؛

إلى ضلعي الثابت وأمان أيامي إلى قرة عيني

إخواني "عبد الرؤوف , طه" و أختي "ياسمين"

إلى عائلتي

إلى من كان عونا وسندا في هذا الطريق..... للأصدقاء الأوفياء ورفقاء السنين لأصحاب الشدائد والأزمات؛

إلا من ضاقت السطور عن ذكرهم فوسعهم قلبي؛

إلى أصدقائي, عائلتي الثانية, من جعلتم هذه الرحلة جميلة ومثمرة؛

اهديكم هذا الإنجاز وثمرة نجاحي الذي لطالما تمنيته ,ها أنا اليوم أتممت أول ثمراته بفضل من الله عز وجل؛

فالحمد لله على ما وهبني وأن يعينني ويجعلني مباركة أينما كنت.

تقوى حداد

Abstract

Forest fires are a chronic and recurring phenomenon, with their intensity increasing. Developing reliable tools for simulating forest fires will allow for a better understanding and prediction of their spread, and consequently, better management of this scourge. Therefore, the simulation of forest fires has been of interest to the scientific community for decades, with a variety of models proposed to study this phenomenon.

This thesis aims to develop and calibrate a forest fire spread simulator utilizing cellular automata and metaheuristics. The simulator will be based on an existing model and optimized using Simulated Annealing. It will feature configurable simulation settings and include visualization of results through graphs.

Keywords: Simulation, Cellular Automata, Metaheuristics, Simulated Annealing.

Résumé

Les incendies de forêt sont un phénomène chronique et récurrent, dont l'intensité augmente. Développer des outils fiables de simulation des incendies de forêt permettra de mieux comprendre et prévoir leur propagation, et par conséquent, de mieux gérer ce fléau. Par conséquent, la simulation des incendies de forêt intéresse la communauté scientifique depuis des décennies, avec une variété de modèles proposés pour étudier ce phénomène.

Cette thèse vise à développer et calibrer un simulateur de propagation des feux de forêt utilisant des automates cellulaires et des métaheuristiques. Le simulateur sera basé sur un modèle existant et optimisé à l'aide d'un recuit simulé. Il comportera des paramètres de simulation configurables et inclura la visualisation des résultats sous forme de graphiques.

Mots clés: Simulation, Automates Cellulaires, Méta euristique, Recuit Simulé.

ملخص

تعتبر حرائق الغابات ظاهرة مزمنة ومتكررة. إن تطوير أدوات موثوقة لمحاكاة حرائق الغابات من شأنه أن يسمح بفهم أفضل والتنبؤ بانتشارها، وبالتالي إدارة هذه الآفة بشكل أفضل. لذلك، ظلت محاكاة حرائق الغابات موضع اهتمام المجتمع العلمي منذ عقود، مع وجود نماذج متنوعة مقترحة لدراسة هذه الظاهرة.

تهدف هذه الأطروحة إلى تطوير ومعايرة محاكي انتشار حرائق الغابات باستخدام الآلي الخلوية والميتاهيريستكس. سيعتمد جهاز المحاكاة على نموذج موجود ويتم تحسينه باستخدام محاكاة التلدين. سيضم إعدادات محاكاة قابلة للتكوين ويتضمن تصور النتائج من خلال الرسوم البيانية.

كلمات البحث: محاكاة, الآلي الخلوي, الميتاهيريستكس , محاكاة التلدين.

Table of contents

List of table

List of figure

List of Abbreviations

Abbreviations	Definition
CA	Cellular Automata
DES	Discrete Time Simulation
ECA	Elementary Cellular Automata
PCA	Probabilistic Cellular Automata
PSO	Particle Swarm Optimization
SA	Simulated Annealing
TS	Tabu Search

General introduction

Forest fires, which occur frequently in different regions of the world, represent a serious threat to ecosystems, human life and material goods. Such fires can spread quickly and unexpectedly, resulting in significant losses and significant environmental disturbances. Understanding and modelling the spread of forest fires is therefore critical to preventing and managing these natural disasters.

Modelling the spread of forest fires has become an important area of research due to the increasing impact of fires on the environment and human communities. Many mathematical and computer solution have been developed to understand and predict the spread of fires.

To overcome these limitations, alternative approaches such as cellular automata (CA) have been proposed. Cellular automata, introduced by von Neumann in 1966, provide a more detailed and localized representation of complex systems, such as forest fires. Cellular automata model the phenomena of proliferation using a network of cells, each of which can adopt different states depending on the rules of transition and the states of neighboring cells. These models make it possible to capture local interactions and observe how they affect the overall behavior of the system, thereby providing a better understanding of the spread of fires.

However, the accuracy and reliability of these models depend on their calibration. Calibration involves adjusting the model parameters to align simulations with observed data, ensuring that the model accurately reflects real-world fire dynamics. This process can be challenging due to the intricate nature of forest fires and the multitude of factors influencing their behavior.

Metaheuristic optimization techniques, such as Simulated Annealing, offer a robust solution for calibrating forest fire simulation models. These techniques are designed to efficiently explore the parameter space and find optimal or near-optimal solutions, making them well-suited for the complex and non-linear nature of fire spread phenomena.

The goal of our work is to develop and calibrate a wildfire simulation model, starting from a prototype to be proposed by the supervisor. The simulator to be realized is based on a probabilistic cellular automaton model, which contains several parameters that we will optimize using simulated annealing algorithm. These simulators will integrate a graphical interface, to visualize the simulation of fire propagation in time and space.

This thesis is organized as follow:

Chapter 1 introduce the basic concepts of modeling and simulation, with a special focus on cellular automata.

Chapter 2 provide an overview of forest fires simulation, discusses how cellular automata can be applied to simulate forest fires and explores the role of metaheuristic techniques in model calibration.

Chapter 3 presents the methodology for developing and implementing the forest fire simulation model. It describes the model's structure, the calibration process using Simulated Annealing, and the results obtained from the simulations.

Finally, the dissertation ends with a general conclusion summarizing the main contributions of our work and proposing perspectives for future research.

Chapter 1. Modelling, Simulation and Cellular Automata

1. Introduction

Modelling and simulation offer powerful tools for predicting the spread and behavior of fires, enabling more effective planning and response strategies. This chapter begins by exploring the foundational concepts of modelling and simulation, providing an overview of the different types of models and the steps involved in the simulation process. A particular emphasis is placed on cellular automata, which are discrete-time systems efficient for simulating complex spatio-temporal phenomena like forest fires. Key concepts such as cells, grids, neighbourhoods, and transition functions are presented. Additionally, the chapter covers elementary and probabilistic cellular automata, highlighting their applicability and importance.

2. Modelling & Simulation

A dictionary definition of simulation is "the technique of imitating the behavior of some situation by means of an analogous situation or apparatus to gain information more conveniently or to train (or entertain) personnel." "Some situation" in the definition corresponds to a source system, and an apparatus is a simulator. As elaborated in the definition, there are two types of simulation objectives: one is to gain information and the other is to train or entertain personnel. The former is often called an analytic simulation and the latter a virtual environment simulation.

The main purpose of an analytic simulation is the quantitative analysis of the source system based on "exact" data. Thus, the simulation should be executed in an as-fast-as possible manner and be able to precisely reproduce the event sequence of the source system. An analytic simulation is often referred [1].

2.1. Basic Concepts

Here are some basic concepts related to simulation:

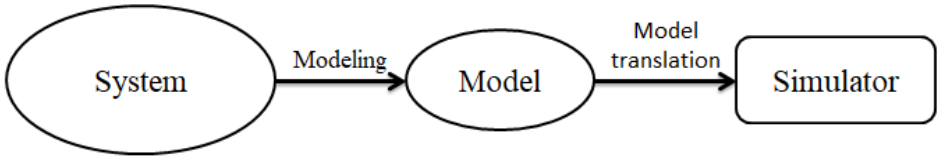

Figure 1.1 Basic concepts related to simulation

2.1.1. System

A system refers to the entity or phenomenon being modeled or simulated. It can be physical, such as a manufacturing plant or a natural ecosystem, or abstract, such as a computer network or a financial market.

2.1.2. Model

A model is a simplified representation of a system that captures its essential features and relationships. It defines the structure and behavior of the system in a mathematical or computational form, allowing researchers to study its dynamics and make predictions.

2.1.3. Simulator

A simulator is a software or hardware tool used to execute a simulation model and generate results. It provides the computational environment necessary to simulate the behavior of the system and analyze its performance under various conditions.

2.1.4. Modeling

Within the realm of simulation and various other disciplines, entails the creation of representations that mirror real-world systems, facilitating the analysis, comprehension, and prediction of their behavior. It serves as a pivotal process whereby intricate systems are condensed into models that encapsulate key attributes for analysis and simulation objectives.

2.1.5. Model translation

Model translation denotes the procedure of converting or adapting models from one format or representation to another. This approach finds extensive application across diverse domains such as performance modeling and machine translation, enabling the adjustment of models to cater to distinct purposes or languages.

2.2. Types of Simulation Models

Simulation models play a crucial role in various fields, offering insights into real-world systems. Here are some common types of simulation models [1]:

2.2.1. Continuous time Simulation

In Continuous Time Simulation approach, the state of the system is updated continuously over time according to a set of equations, typically involving differential equations, that represent the relationships between different variables and how they change continuously (see Figure 1.2). Continuous Time Simulation is particularly used when system changes are smooth and continuous. They are commonly applied in engineering, physics, and environmental studies, such as fluid dynamics simulations, climate and weather prediction, and population dynamics.

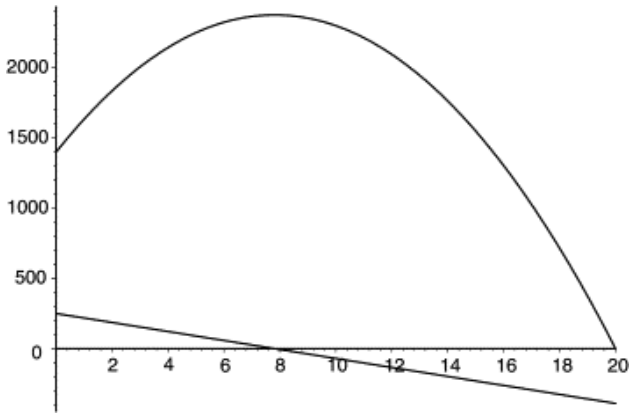

Figure 1.2 Continuous time Simulation [1].

2.2.2. Discrete Event Simulation

Discrete-event simulation focuses on events that happen at specific points in time and change the state of the system, such as the arrival of a customer in a queue, the completion of a task, or the failure of a machine (see Figure 1.3). Discrete event models are suitable for processes with distinct events such as queuing systems in service industries, call centers, hospitals, etc.

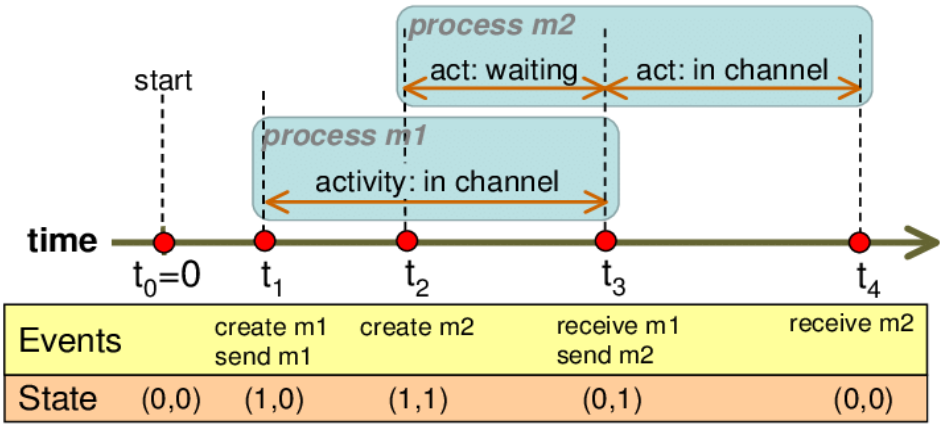

Figure 1.3 Discrete Event Simulation [22].

2.2.3. Discrete time Simulation

In Discrete Time Simulation, the state of the system is updated at fixed time intervals (see Figure 1.4). This approach contrasts with Continuous Time Simulation, where variables are monitored continuously, and Discrete Event Simulation (DES), where the system state is updated solely at the occurrence of specific events. Discrete Time Simulation is particularly advantageous for systems characterized by changes that occur at specific intervals or time steps, or where regular, periodic updates are more appropriate or simpler to implement. This method facilitates the modeling of systems in a manner that aligns with their natural temporal resolution, thereby simplifying analysis and implementation.

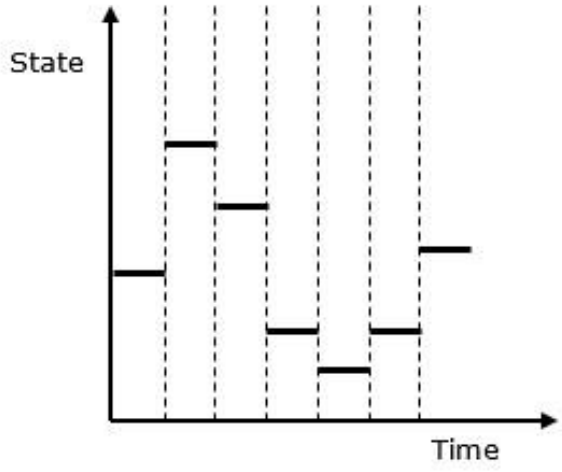

Figure 1.4 Discrete time Simulation [3].

2.2.4. Agent-Based Simulation:

Agent-based simulation extends the capability of entities to allow autonomy and intelligence. The resulting autonomous objects (called agents) are able to proactively interact with other agents and the environment in which they situate. If an agent-based simulation model includes both discrete and continuous state variables, it is a hybrid discrete-continuous simulation model with proactive autonomous entities [2].

Agent-based models are used in systems where individual behaviors and interactions between them are critical to understanding the overall system dynamics, such as social systems modeling, traffic flow simulations, and market dynamics.

2.2.5. Monte Carlo Simulation:

The Monte Carlo simulation is a class of computational algorithms that relies on repeated random sampling to compute the numerical integration of functions arising in engineering and science. These functions are often impossible to evaluate using direct analytical methods, such as reliability analysis of mechanical and electrical systems. In recent years, Monte Carlo simulation has also been utilized as a technique to comprehend the impact of risk and uncertainty in financial, project management, and other forecasting models [1]. Figure 1.5 illustrates an example of applying Monte Carlo simulation to estimate π.

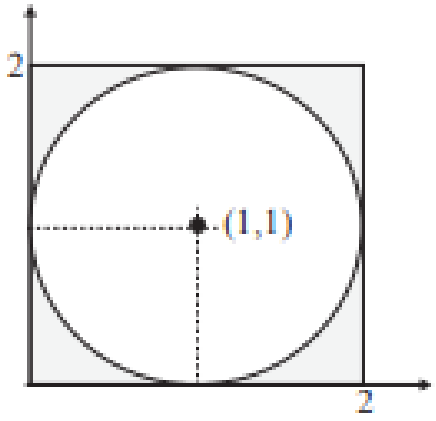

Figure 1.5 A circle of unit radius to compute the value of π via Monte Carlo simulation [1].

2.3. Steps in the Simulation Process

A simulation study involves systematic steps from problem formulation to reporting (see Figure 1.6). Each step is crucial for ensuring that the simulation model is accurate, reliable, and useful for decision-making. The steps in a simulation study are as follows [3]:

2.3.1. Problem formulation

Every study begins with a statement of the problem, provided by policy makers. Analyst ensures it is clearly understood. If it is developed by analyst policy makers should understand and agree with it [3].

2.3.2. Setting of objectives and overall project plan

In this step, we define the objectives of the simulation study, specifying the questions that the simulation aims to answer. The overall project plan should include [3]:

- Alternative systems to be considered.
- Methods for evaluating their effectiveness.
- Staffing plans.
- Study costs.
- Timeline for each phase anticipated results.

2.3.3. Model conceptualization

Create an abstract representation of the system, identifying key components and their interactions. This involves outlining assumptions, key components, their interactions, and the variables of interest. It is best to start with a simple model and build toward greater complexity. However, the model complexity need not exceed that required to accomplish the purposes for which the model is intended.

It is not necessary to have a one-to-one mapping between the model and the real system. Only the essence of the real system is needed. It is advisable to involve the model user in model conceptualization. Involving the model user will both enhance the quality of the resulting model and increase the confidence of the model user in the application of the model [3].

2.3.4. Data collection

This step involves gathering the necessary data to build and validate the model. This includes input data (e.g., arrival rates, service times) and historical data for validation. Accurate and comprehensive data is crucial for creating a realistic model [3].

2.3.5. Model translation

This step consists of converting the conceptual model into computational model using simulation software or programming language. This step involves coding the model logic, defining parameters, and setting up the simulation environment. The translation should faithfully represent the conceptual model [3].

2.3.6. Verification

The aim of this step is to ensure the model is built correctly according to its specifications. We check if the code is functioning as intended and produces the expected outputs for given inputs [3].

2.3.7. Validation

This step confirms that the model accurately reflects the real-world system it represents. You compare the model's outputs with real-world data or the behavior of the actual system. Validation usually is achieved through the calibration of the model, an iterative process of comparing the model against actual system behavior and using the discrepancies between the two, and the insights gained, to improve the model. This process is repeated until model accuracy is judged acceptable [3].

2.3.8. Experimental design

In this step, we define how we will use the simulation to answer research questions. This involves determining input variables, the length of simulation runs, and the number of replications to be made of each run [3].

2.3.9. Production runs and analysis

Conduct simulation runs to according to experimental design, to estimate measures of performance for the system designs that are being simulated [3].

2.3.10. More Runs

Given the analysis of runs that have been completed, the analyst determines whether additional runs are needed and what design those additional experiments should follow [3].

2.3.11. Documentation and reporting

There are two types of documentation: program documentation and process documentation.

- **Program documentation:** Can be used again by the same or different analysts to understand how the program operates. Further modification will be easier. Model users can change the input parameters for better performance.
- **Process documentation:** Gives the history of a simulation project. The result of all analysis should be reported clearly and concisely in a final report. This enables to review the final formulation and alternatives, results of the experiments and the recommended solution to the problem. The final report provides a vehicle of certification [3].

2.3.12. Implementation

Implementation refers to putting the simulation model into use. This might involve integrating it with other systems, deploying it for users, or using it for ongoing analysis and decision-making. The success of the implementation phase depends on how well the previous eleven steps have been performed. If the model user has been thoroughly involved and understands the nature of the model and its outputs, likelihood of a vigorous implementation is enhanced [3].

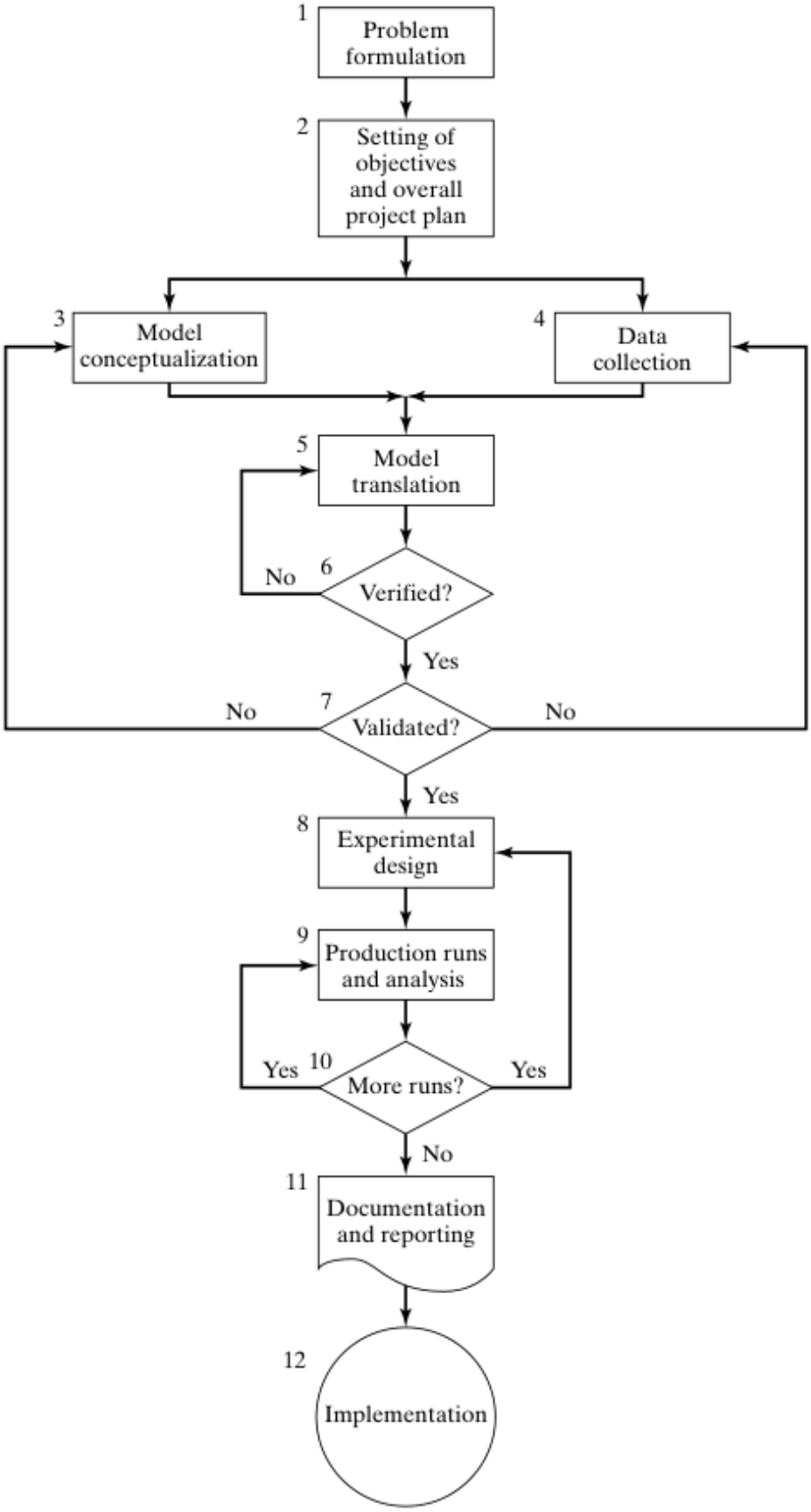

Figure 1.6. Steps in the Simulation Process [3].

3. Cellular automata

Cellular Automata (CA) are mathematical models emerged from the work of J. von Neumann [4], and used to simulate complex systems by means of a regular network of simple uniform entities interacting locally in a synchronous and manner (Figure 1.7). They are widely applied in various fields, from biology to computer science, and even in modelling social systems.

3.1. Definition

A cellular automaton consists of a regular grid of "cells", each containing a "state" chosen from a finite set, which can evolve over time. The state of a cell at time $t + 1$ is a function of the state at time t of a finite number of cells called its "neighborhood". At each new unit of time, the same rules are applied simultaneously to all the cells in the grid, producing a new "generation" of cells entirely dependent on the previous generation. The elements of the cellular automaton are cells, grid, neighbourhood, and transition rules.

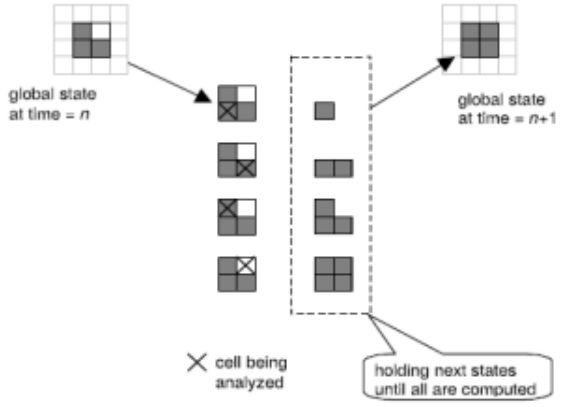

Figure 1.7 Cycle of cellular automaton simulation.

3.2. Cell

The fundamental element of a cellular automaton is the cell. A cell is defined by its coordinates. For example, in a two-dimensional grid, a cell is defined by two integers(i, j), where i is the row number and j is the column number. Each cell can be in a finite number of states [5].

3.3. Grid

The cells are arranged on a regular grid of dimension d [5]. The grid can be in any finite number of dimensions, but the most commonly studied CA are one-dimensional ($d = 1$) and two-dimensional ($d = 2$).

3.4. Neighborhood

The neighborhood of a cell refers to the set of cells that could potentially interact with it at a given time. This neighborhood encapsulates all the information required for updating the cell's state at each time step, with the cell itself being included in its own neighborhood. The most common neighborhood types, as described by Neumann in 1966 [4], include:

- **Von Neumann:** Considers only the immediate North, South, East, and West neighbors of the cell.
- **Moore:** Expands upon the Von Neumann neighborhood by also incorporating the diagonal cells.
- **Extended Moore:** Further extends the neighborhood distance beyond the immediate cells to encompass a wider area.

These neighborhood configurations are illustrated in Figure 1.8 (a), (b), and (c) respectively.

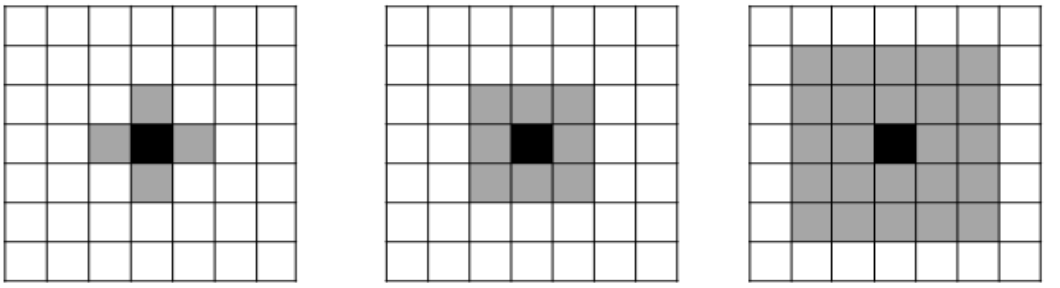

Figure 1.8 The most commonly used neighborhoods [5].

3.5. Transition function

The evolution rule determines the state of each cell at time step $t + 1$ based on the state of the cells in its neighborhood at time t (see Figure 1.9).

As an example, we present the cellular automaton: Conway's Game of Life. It was proposed by John Horton Conway in 1970. The elements of the Game of Life can be outlined as follows:

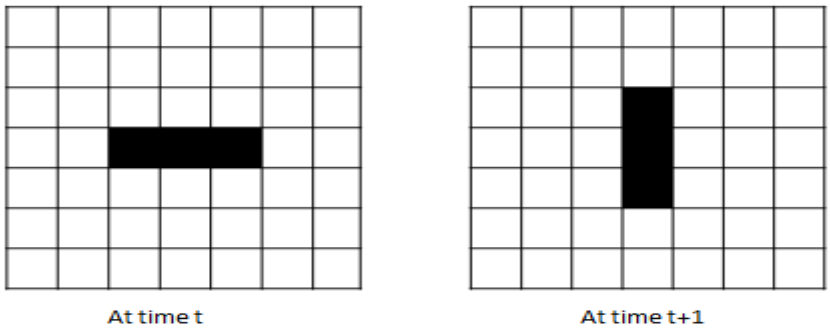

Figure 1.9 Example of applying the rules of the Game of Life [5].

- A cell can take two states: 0 (dead or inactive), and 1 (a live or active).
- The Game of Life evolves on a two-dimensional square grid ($\boldsymbol{d = 2}$).
- The Game of Life uses the Moore neighborhood.
- The evolution rules of the Game of Life are:
    1. An inactive cell surrounded by 3 active cells becomes active;
    2. An active cell surrounded by 2 or 3 active cells remains active;
    3. In all other cases, the cell becomes or remains inactive.

3.6. Formal definition of cellular automata

Formally, a cellular automaton is defined as a 4-tuple (d, Q, N, δ) where:

- $\boldsymbol{d}$ is the dimension of the automaton, and its grid is denoted as $\boldsymbol{Z^d}$, the discrete space of dimension $\boldsymbol{d}$.
- $\boldsymbol{Q}$, is a finite the set of states that the cells can take.
- $\boldsymbol{N \subseteq Z^d}$ is a finite subset of $\boldsymbol{Z^d}$, representing the neighborhood of each cell $\boldsymbol{i}$: $\boldsymbol{N(i) = (N_1, N_2, \dots, N_a) = (N_j \in Z^d;\ 1 \leq j \leq a)}$
- $\boldsymbol{\delta :\ Q^a \rightarrow Q}$ Is a transition rule, where $\boldsymbol{a = |N|}$.

A configuration of cellular automata assigns a state to each cell of the grid: a configuration is a function from Z^d to Q.

3.7. Elementary Cellular Automata

Elementary Cellular Automata (ECA) is a cellular automaton $A = (Z^d, Q, N, \delta)$ where:

- $\boldsymbol{d = 1}$
- $\boldsymbol{Q} = \{\mathbf{0}, \mathbf{1}\}$
- $\boldsymbol{N} = \{-\mathbf{1}, \mathbf{0}, \mathbf{1}\}$: The neighborhood of a cell $\boldsymbol{i}$ consists of the cell itself and its two immediate neighbors (left and right): $\boldsymbol{i}, \boldsymbol{i} - \mathbf{1}, \boldsymbol{i} + \mathbf{1}$.
- Transaction rule $\boldsymbol{\delta}$: The state of each cell in the next time step depends on its current state and the states of its two neighbors. Since there are 3 cells in the neighborhood and each can be in one of 2 states, there are $\mathbf{2^3} = 8$ possible combinations of neighborhood states: **111, 110, 101, 100, 011, 010, 001, 000.**

The transition rule δ associates a state with each configuration of neighborhood. There are $2^8 = 256$ possible transition rules. Each transition rule can be seen as writing an integer between 0 and 255.

Example . The following table corresponds to the rule: $\mathbf{01011010_2 = 90}$

Neighbourhood (q_1, q_2, q_3)	111	110	101	100	011	010	001	000
$\delta(q_1, q_2, q_3)$: next state of central cell	**0**	**1**	**0**	**1**	**1**	**0**	**1**	**0**

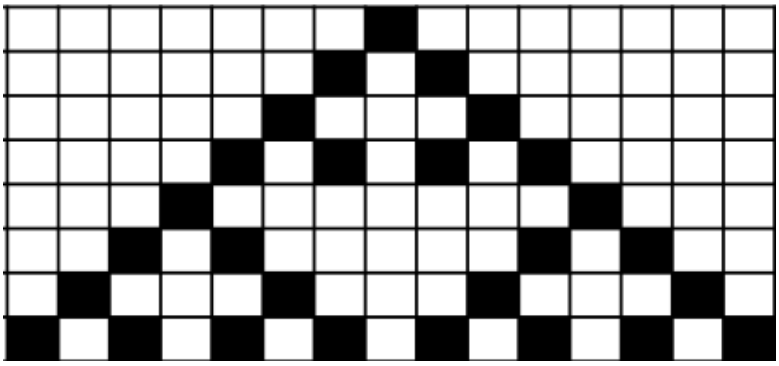

Figure 1.10 Space-time diagram of Rule 90 [5].

3.8. Probabilistic cellular automaton

A probabilistic (or stochastic) cellular automaton (PCA) is a variant of cellular automaton where the state transitions of cells are governed by probabilistic rules rather

than deterministic ones, as seen in traditional cellular automata like Conway's Game of Life [6].

- **Probabilistic Transition Rules:**

In an PCA, each cell's state transition is determined by probabilistic rules. The state of a cell of cell i at time $t+1$ $s_i(t+1)$ in a probabilistic cellular automaton can be represented as a random variable that depends on the state of its neighbourhoods $\mathcal{N}_i(t)$ at time. The transition probability can be written as:

$$P(\, s_i(t+1) = \, S \mid \mathcal{N}_i(t)) = \, N\,)$$

Where S is the new state and N is the neighborhood configuration. The collection of these probabilities for all possible neighborhood configurations defines the probabilistic transition rule.

- **Modeling Complex Systems**

Many real-world systems exhibit inherent randomness. PCA incorporate stochastic elements into their transition rules, making them suitable for modeling systems where outcomes are not deterministic but probabilistic. PCA are used to model and simulate complex systems in physics, biology, ecology, and social sciences:

- **Ecology and Biology:** Modeling forest fires, disease spread, and population dynamics.
- **Physics:** Simulating diffusion, phase transitions, and fluid dynamics.
- **Social Sciences:** Analyzing opinion dynamics, crowd behavior, and other social phenomena influenced by probabilistic interactions.

4. Conclusion

In summary, this chapter has started by explaining basic concepts and types of simulation models, providing a foundation for understanding modelling and simulations. By examining cellular automata in detail, including their components like cells, grids, and transition functions, and probabilistic cellular automata, we've seen their potential for simulating fire spread. These discussions will guide us in the step of developing a forest fire simulation model based cellular automata.

Chapter 2. Forest Fire Simulation and Metaheuristic

1. Introduction

Having presented modelling and simulation in general, and cellular automata in particular, in the previous chapter, we present a forest fire simulation. Understanding the dynamics of forest fires is crucial for effective management and mitigation strategies. This chapter explores the fundamental concepts surrounding forest fires, including the fire triangle and the mechanisms behind ignition and spread. We also discuss the role of wildfire simulation in aiding decision-making processes for fire management. Then, the chapter narrows down to forest fire modelling based on cellular automata, a powerful approach for simulating fire behaviour. Finally, we introduce metaheuristics, a class of optimization algorithms used in calibrating forest fire simulation models.

2. Overview of Forest Fire

2.1. What Is a Fire?

A fire is made up of glowing, red-hot material and often flames. These give off heat and normally produce smoke. Fire happens when a material burns. Burning is a chemical reaction between a material and oxygen, which is one of the gases in the air. Burning is also called combustion [7].

2.2. The fire triangle

For ignition and combustion to occur, three factors need to be present at the same time: **fuel** which can be any material that can burn, an **external heat source** (flame), and **oxygen**, which is needed to fuel the fire. These three elements are represented in the fire triangle (Figure 2.1). The fire triangle is a simple model for understanding the elements required for most fires.

Figure 2.1.The fire triangle [7].

2.3. What Is a Forest Fire?

The sight of a fire sweeping furiously through a forest is both frightening and spectacular. Forest fires (also called wildfires) are devastating events. Their intense heat burns leaves and branches to a crisp before making them explode, roaring and crackling, into flames. Giant flames leap hundreds of feet into the air, along with columns of thick, black smoke. In the biggest fires, spinning tornadoes are created by the rapidly rising heat [7].

Figure 2.2. Forest Fire (Wildfire) [7].

2.4. How Forest Fires Start

Two conditions are needed for a forest fire to start. First, the forest fuel (the trees, other plants, and dead matter on the forest floor) must be dry. Second, there must a source of heat to ignite the fuel and start the fire. Fires are either started by natural events or by people, either accidentally or deliberately. By far the most common natural fire starter is lightning. Lightning hits the earth about 100,000 times every day. It is the major cause of fire in remote areas. Fires are occasionally started by volcanic eruptions. Another cause is the buildup of heat from rotting vegetation on the forest floor, known as spontaneous combustion [7].

Figure 2.3 Lightning [7].

2.5. Forest Fires Spreading

Heat produced by a fire moves by convection (hot air currents) and radiation (heat rays). This makes the surrounding fuel hot enough to catch fire. The fuel burns too, first making flames and then smoldering, until the fuel is used up. In this way, a line of flames called a flame front moves through the forest, spreading in every direction from the source of the fire [7].

Several factors contribute to the spread of wildfires, including fuel density, landscape slope, and wind. Let's delve into each of these spreading factors:

2.5.1. Fuel Density

The density and arrangement of fuel, such as vegetation and organic matter, play a significant role in determining how quickly a fire spreads. Higher fuel densities provide more material for the fire to burn, leading to faster spread rates. Different types of fuel, from fine grasses to dense forests, can influence fire behavior and intensity.

2.5.2. Landscape Slope

The slope of the terrain affects the speed and direction of fire spread. Fires tend to spread more rapidly uphill due to the preheating of fuel ahead of the fire, increased wind speeds, and the stacking effect of flames. Conversely, fires may spread more slowly or even stall on flat terrain or when moving downhill, depending on other factors like wind and fuel moisture.

2.5.3. Wind

Wind is one of the most critical factors influencing fire behavior and spread. Wind can accelerate the rate at which a fire spreads by supplying oxygen to the flames and by carrying embers ahead of the fire front, igniting new spots. Wind can also influence the direction of fire spread, causing fires to move more quickly in the direction of the wind and creating erratic fire behavior. Additionally, strong winds can make firefighting efforts more challenging by spreading embers over long distances, increasing the likelihood of spot fires, and causing fires to behave unpredictably.

2.5.4. Type of Vegetation

The type of vegetation present in an area affects its flammability and the rate at which fire can spread through it. Different vegetation types have varying fuel characteristics, including moisture content, density, and chemical composition, which influence their flammability. Grasslands, for example, typically burn more quickly than forests due to their lower fuel moisture content and greater surface area. Forests with dense vegetation can sustain more intense fires that spread both on the ground and through the canopy, while shrublands may burn more rapidly due to the high oil content in some species. Understanding the types of vegetation present in an area is essential for predicting fire

behavior and implementing effective wildfire management strategies tailored to the specific fuel conditions.

2.6. Wildfire Simulation for Decision-Making

Simulation is essential for comprehending fire behavior and aiding decision-making across various aspects of fire management. Here's why simulation matters:

- **Understanding Fire Behavior:** Simulation models enable researchers and fire managers to explore the complex dynamics of fire behavior in diverse environments. By inputting data such as weather conditions, topography, fuel types, and ignition sources, simulation models can predict how a fire might spread, its speed, and its potential direction. This understanding is crucial for devising effective strategies to combat wildfires and minimize their impact.
- **Risk Assessment:** Simulation models help assess the risk of wildfires in specific areas. By analyzing factors like fuel load, weather patterns, and historical fire data, fire managers can identify areas prone to wildfires and prioritize resources for prevention and mitigation efforts.
- **Resource Allocation:** Simulation optimizes the allocation of firefighting resources. By simulating different scenarios, fire managers can determine the most effective deployment of personnel, equipment, and resources to contain and suppress wildfires. This ensures efficient resource utilization to minimize damage and protect lives and property.
- **Evacuation Planning:** During wildfires, evacuation planning is critical for ensuring the safety of affected residents. Simulation models can simulate real-time fire spread and predict its movement. This information guides decisions about when and where to evacuate people and how to manage evacuation routes to ensure timely and safe evacuations.
- **Training and Education:** Simulation provides a safe environment for training firefighters and emergency responders. Fire behavior simulators allow trainees to experience realistic firefighting scenarios without actual fire risks, helping them develop the skills needed to respond effectively to wildfires.

Overall, simulation enhances our understanding of fire behavior, supports informed decision-making, and improves the effectiveness of wildfire management strategies, ultimately reducing the impact of wildfires on communities and the environment.

3. Forest fire modelling based cellular automata

Forest fire simulator is a computational model designed to predict the spread of fires based on environmental or contextual conditions (weather conditions, topography of the landscape, fire sources, etc.). Forest fire simulation models are importance tools for understanding fire dynamics, predicting fire behaviour in order to implement preventive and operational policies that aim to minimize fire risks and limit damage.

The first models developed were mathematic models, developed based on physics, combustion, fluid dynamics [8], [9]. However, the phenomenon of forest fires depends on many factors, between them there are complex interactions, which makes it particularly difficult to formulate effective and reliable mathematical models, which motivated the emergence of new approaches to model this phenomenon.

Cellular automata (CA) models simulate fire spread by dividing the landscape into a grid of cells. Each cell's state changes based on predefined rules and the states of neighbouring cells. Models based cellular automata can capture complex spatial interactions and heterogeneity of factors affecting the spread of forest fires (vegetation types, weather conditions, and landscape topography) [10], [11], Consequently, many cellular automata-based wildfire models have been proposed [10], [12], [13], [14].

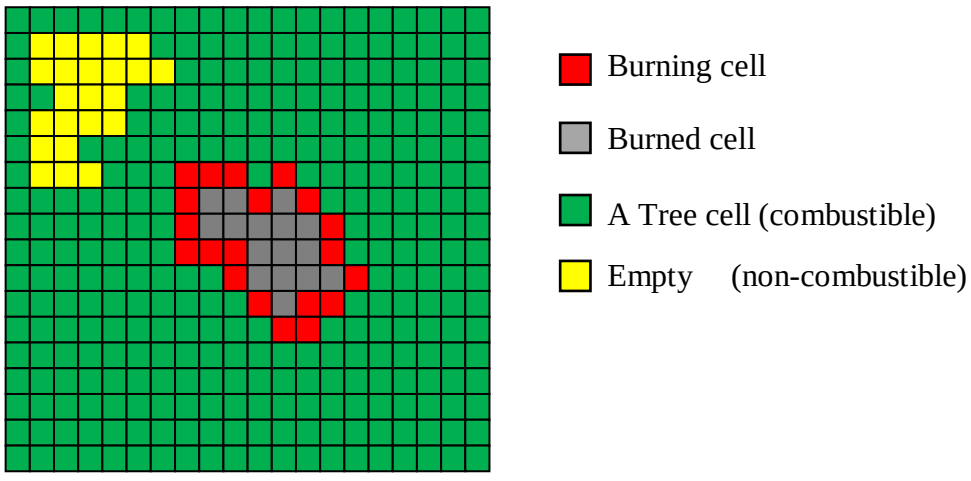

Figure 2.4 Representation of forest area using a grid.

4. Metaheuristics for calibrating forest fire simulation models

In forest fire simulation model based on probabilistic cellular automata, the transition probability of a cell to the burning state, is governed by a formula involving multiple interacting parameters, such as wind speed, ground elevation, vegetation density, and humidity. These parameters exhibit non-linear and potentially stochastic relationships with the model's output, making conventional optimization methods less effective. Metaheuristics, such as Genetic Algorithms, Particle Swarm Optimization, and Simulated Annealing, offer robust solutions for exploring vast solution spaces and identifying optimal or near-optimal configurations within reasonable time frames. In this section, we will review the basic concepts of metaheuristics, and then conclude with the most common metaheuristics algorithms.

4.1. Definition

A metaheuristic is a meta-process or a high-level process that can be used and/or adapted to solve optimization problems. In general, this process has a set of mechanisms and techniques necessary to guide and reinforce a heuristic search procedure, such as: diversification and intensification mechanisms, techniques for escaping local optima etc... These Mechanisms and techniques enable metaheuristics to efficiently exploit and explore the search space and find high-quality solutions [18].

Many classification criteria may be used for metaheuristics:

• **Nature inspired versus non nature inspired**: Many metaheuristics are inspired by natural processes: evolutionary algorithms and artificial immune systems from biology; ants, bees colonies, and particle swarm optimization from swarm intelligence into different species (social sciences); and simulated annealing from physics.

• **Population-based search versus single-solution based search**: Single-solution based algorithms (e.g., local search, simulated annealing) manipulate and transform a single solution during the search while in population-based algorithms (e.g., particle swarm, evolutionary algorithms) a whole population of solutions is evolved. These two families have complementary characteristics: single-solution based metaheuristics are exploitation oriented; they have the power to intensify the search in local regions. Population-based metaheuristics are exploration oriented; they allow a better diversification in the whole search space. In fact, the algorithms belonging to each family of metaheuristics share many search mechanisms.

4.2. Neighborhood principle

The general principle most widely used in the development of metaheuristics is that of neighborhood. Each solution *s* of the problem is associated with a subset N(S) of solutions. A neighborhood method generally starts with an initial configuration s, which is subjected to an iterative process. It seeks to improve the current configuration by replacing it with one of its neighbors, taking into account the objective function. This process stops and returns to the best solution found when the stopping criterion is reached. This stopping condition generally concerns a limit on the number of iterations or on the objective to be achieved. Neighborhood methods differ mainly in the neighborhood used and the strategy for traversing a neighboring solution [12].

4.3. General concepts

Several important concepts related to meta-heuristics can be highlighted:

- Metaheuristics are typically applied to problems that cannot be optimized using traditional mathematical methods. Although they may not perform as well as specialized heuristics designed for specific problems, they offer a versatile approach.
- Metaheuristics are commonly used in combinatorial optimization but can also address continuous or mixed problems involving both discrete and continuous variables.
- Certain metaheuristics are theoretically "convergent" under specific conditions, ensuring that they will find the global optimum within a finite time. Convergence is typically guaranteed under the condition of ergodicity, meaning the algorithm can reach any solution with every move. However, in practice, achieving quasi-ergodicity (the ability to reach any solution within a finite number of moves) is often sufficient and practical.

4.4. Classification of Metaheuristics

Metaheuristic algorithms can be broadly classified into two main categories: population-based methods, known as evolutionary algorithms, which include genetic algorithms and other related techniques, and single-solution methods, such as simulated annealing.

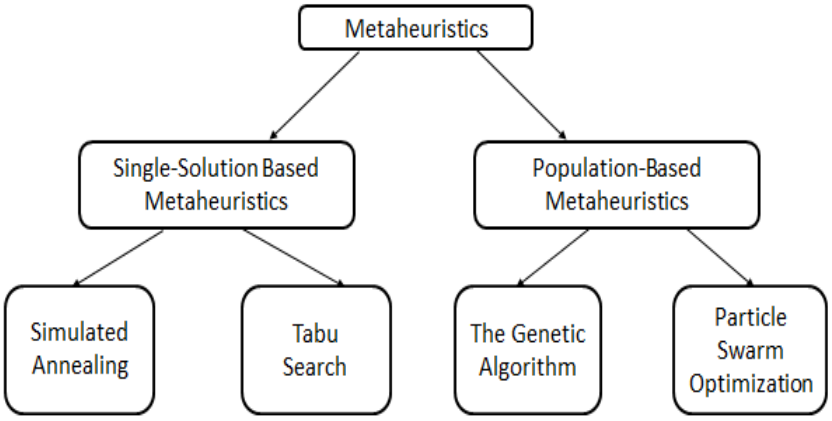

Figure 2.5 Classification of Metaheuristic

4.4.1. Single-Based Metaheuristics

Single-solution-based metaheuristics start the search with a single initial solution. They rely on the concept of neighborhood to improve the quality of the current solution. In fact, the initial solution undergoes a series of modifications based on its neighborhood. The goal of these local modifications is to explore the neighborhood of the current solution in order to progressively improve its quality over the different iterations [13].

4.4.1.1. Tabu Search

A very well-known and widely used metaheuristic in the field of optimization. TS was proposed by Fred Glover in 1986. The basic idea of this metaheuristic is to use a memory called the tabu list which contains the solutions visited during the previous x iterations.

During the search procedure, the algorithm is prohibited from returning to a solution that belongs to the tabu list in order to: (1) avoid cycles, (2) escape local optima and (3) d explore the search space in an efficient manner. The x value is called the size of the tabu list.

Formally, the TS begins with an initial solution s. At each iteration, the TS constructs the neighborhood structure N(s) of the current solution s and chooses the best non-Tabu solution S_n (which does not belong to the tabu list) in N(s). Then the solution was replaced by the new solution $S_n$and declared itself forbidden or taboo. The tabu list is updated by adding s and deleting the oldest solution if the size of L exceeds a maximum fixed at the

beginning. The TS ends when the stopping criterion is satisfied. The best solution found *b* is updated at each iteration [18].

4.4.1.2. Simulated Annealing

The simulated annealing algorithm was developed by Kirkpatrick, Gelatt, and Vecchi. [17].It is inspired by the annealing process in metallurgy, where a metal is heated to a high temperature and then cooled slowly to achieve a defect-free alloy. This process is based on the principles of thermodynamics, where a sudden drop in temperature can lead to a local optimum, whereas a gradual cooling process can result in a global optimum. Metallurgists understand that rapid cooling can introduce microscopic defects, whereas slow cooling can produce a well-ordered structure. This concept is applied in simulated annealing to optimize complex problems by gradually reducing the temperature and accepting worse solutions to avoid local optima and converge to the global optimum.

- **The pseudo code of simulated annealing:**

Begin
Let S be an initial solution and an initial temperature T ;
While (the stopping criterion is not satisfied) Do
Randomly select a solution S' close to S ;
r = a random number between 0 and 1.
Calculate Δ;
If (S' is better than S or $r < e^{-\frac{\Delta}{T}}$) Then
$S = S'$;
If (S is better than S'') Then
$S'' = S$;
End if
End if
Update T ;
return S'' ;
End

- **Some applications of simulated annealing:**
  - Image processing.
  - Scheduling problems.
  - Electronic circuit design (placement and routing problems).
  - Lottery (France) computer network organization.

- Garbage collection.

- **Advantages of the method:**
  - Good quality solutions.
  - General method and easy to program.
  - Flexibility: New constraints can be easily incorporated**.**
- **Disadvantages:**
  - Computation time: excessive in some applications.

4.4.2. Population-Based Metaheuristics

Population-based metaheuristics initiate the search with a diverse set of solutions. They operate on a set of solutions to extract the best one (global optimum), which represents the solution to the problem at hand. The idea of using a set of solutions instead of a single solution enhances the diversity of the search and increases the possibility of discovering high-quality solutions.

4.4.2.1. Genetic Algorithm

The Genetic Algorithm is an optimization method inspired by the natural evolution of species. It uses a population of potential solutions, known as individuals, that evolve from generation to generation through selection, crossover, and mutation operations. The main objective of this algorithm is to identify an approximate solution for a problem, even if not necessarily the optimal one.

The theory of Darwin on the evolution of species provides the foundations of the genetic algorithm, which rests upon three key principles: the principle of variation, the principle of adaptation, and the principle of heredity.

- **Variation:** There are more or less significant differences between individuals in a population (of the same species).
- **Adaptation:** Some individuals possess characteristics that give them a better chance of surviving and reproducing compared to others.
- **Heredity:** The characteristics of individuals in a population are transmitted to their offspring. In other words, these characteristics must be hereditary [14].

This figure, utilizing genetic terminology, facilitates the application of genetic algorithms by defining key concepts:

- **Population:** The entire set of potential solutions.
- **Individual:** Represents a single solution within the population.
- **Chromosome:** A segment of the solution, often a string of genes in genetic algorithms.
- **Gene:** Represents a specific characteristic or trait within an individual or chromosome.

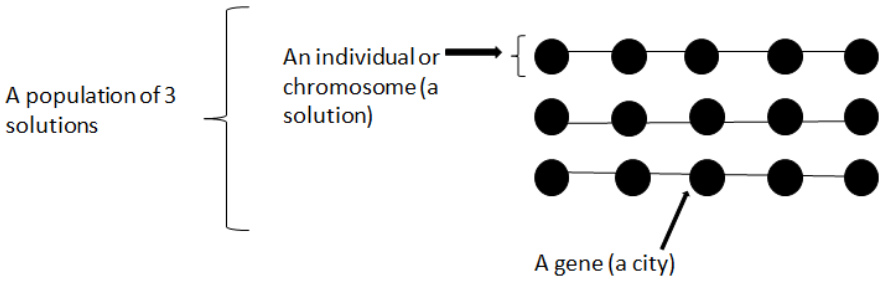

Figure 2.6 Genetic terminology TSP as example [18].

4.4.2.2. Particle Swarm Optimization (PSO)

Particle Swarm Optimization (PSO) is a swarm intelligence algorithm inspired by the dynamic social behavior and emergent behavior that arises in socially organized colonies.

PSO is a population-based algorithm, meaning it exploits a population of individuals to explore promising regions of the search space. In this context, the population is referred to as swarms and the individuals (i.e., the search points) are referred to as particles. Each particle moves with an adaptable velocity within the search space and maintains a memory of the best position it has encountered. In the global variant of PSO, the best position retained by swarm individuals is communicated to other particles. In the local variant, each particle is assigned to a topological neighborhood consisting of a predefined number of particles. In this case, the best position retained by particles within the neighborhood is communicated among them [15].

The movement of each particle is influenced by the following three components:

- **A physical component:** the particle tends to follow its current direction of movement.
- **A cognitive component:** the particle tends to move towards the best site it has encountered.
- **A social component:** the particle tends to move towards the best site reached by its neighbors.

5. Conclusion

In this chapter, we explored the fundamentals of computer modeling and simulation, focusing on simulating forest fires using cellular automata. This approach provided us with the necessary knowledge for our project and helped us choose a suitable simulation model for our application.

Chapter 3. Implementation and Calibration of the Forest Fire Simulation Model

1. Introduction

In this chapter, we detail our approach to calibrate a forest fire simulation model based on cellular automaton using the Simulated Annealing (SA) metaheuristic. The chapter begins with an overview of the basic model description, delineating the fundamental components of the wildfire simulation model. We discuss the grid definition, cell states representing different fire stages, and rules of evolution governing fire spread dynamics. Then, we present the model implementation, detailing the choice of programming language and libraries, also we elucidate the user interface for visualizing and interpreting simulated fire behavior.

We delineate the definition of the objective function, control parameters of the SA algorithm, and data pre-processing steps to prepare input data for the calibration process to enhance the accuracy of the wildfire simulation model.

Finally, we evaluate the calibration results to assess the performance of the calibrated model, by comparing simulated fire spread against observed data and employing various metrics.

2. The basic Model Description

The model we will implement and calibrate is proposed by Alexandridis (2008) [10]. This model uses cellular automata associated with geographic information (land elevation, vegetation, etc.) and meteorological data (wind direction and speed) to predict the evolution of fire fronts in heterogeneous forest landscapes. This section provides a detailed description of the grid, the states of cells, the rules governing their evolution, and the variables influencing fire spread

2.1. Grid Definition

A landscape is represented by a two-dimensional grid, where each cell represents a portion of the surface. These cells are square, allowing for eight possible directions of fire propagation (see Figure 3.1).

2.2. State of cells

Each cell in the grid can exist in one of the following states:

- **Unburned:** The cell contains vegetation that has not yet burned.

- **Burning:** The cell is currently on fire.
- **Burned:** The cell has already burned and no longer contains flammable material.
- **Empty:** The cell contains no vegetation or flammable material.

Transitions between these states are governed by the rules of evolution described in the next section.

2.3. Rules of evolution

At each time step t, the following rules are applied to all cells (i,j) in the grid.

- **Rule 1:** if the state of cell $(\boldsymbol{i,j})$ at time $\boldsymbol{t}$ is *Empty*, then the state does not change at time t+1.
- **Rule 2:** if the state of cell $(\boldsymbol{i,j})$ at time $\boldsymbol{t}$ is *Burning*, then the state of this cell at time $\boldsymbol{t+1}$ becomes Burned.
- **Rule 3:** if the state of cell $(\boldsymbol{i,j})$ at time $\boldsymbol{t}$ is *Burned*, then its state does not change at time $\boldsymbol{t+1}$.
- **Rule 4:** if the state of cell $(\boldsymbol{i,j})$ at time $\boldsymbol{t}$ is Burning, then the state of a neighboring cell $(\boldsymbol{i\pm1, j\pm1})$ becomes Burning at time $\boldsymbol{t+1}$ with probability $\boldsymbol{P_{burn}}$ (equation 1).

Rule 4 implies that when a given cell is in a state of Burning at the current time step, the fire can spread to neighboring cells at the next time step with probability P_{burn}. This probability is a function of various parameters that affect fire propagation, and will be analyzed in the following paragraphs.

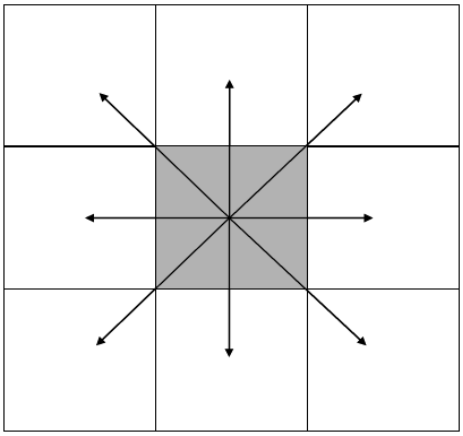

Figure 3.1 Possible directions of fire propagation on a square grid [10].

2.4. Variables affecting fire spread

The methodology proposed by Alexandridis et al. [10] takes into account several variables that can affect both the shape and rate of forest fire propagation: vegetation type, vegetation density, wind speed and direction, and ground elevation.

Variables specific to the terrain, such as vegetation type, vegetation density, and ground elevation, are also encoded in matrices similar to the state matrix. The probability P_{burn} is calculated by:

$$P_{burn} = P_h(1 + P_{veg})(1 + P_{den})P_w P_s \quad (1)$$

Where:

- $\boldsymbol{P_h}$**:** is the constant probability that a cell adjacent to a burning cell, containing a given type of vegetation with a specific density, will catch fire in the next time step without any wind and on flat landscape.
- $\boldsymbol{P_{den}}$**:** is the probability of fire spread depending on vegetation density.
- $\boldsymbol{P_{veg}}$**:** is the probability of fire spread depending on vegetation type.
- $\boldsymbol{P_W}$**:** is the probability of fire spread depending on wind speed.
- $\boldsymbol{P_S}$**:** is the probability of fire spread depending on the slope of the terrain.

Note that these probabilities are multiplied by the constant probability P_h to obtain the adjusted probability that accounts for all the aforementioned factors.

2.4.1. Effect of the type and density of vegetation

The effects of vegetation type and density are represented by the probabilities P_{veg} and P_{den} respectively. Specifically, the type and density of vegetation on the surface are divided into several distinct categories. Vegetation types are grouped into three categories: (1) agricultural, (2) thickets, and (3) HallepoPine trees. Vegetation density is classified into three categories: sparse, normal, and dense. Each vegetation type is assigned a value for P_{veg} (see Table 1), and each density category is assigned a value for P_{den} (see Table 2).

Category	Type	P_{veg}
1	agricultural	-0.4
2	Thickets	0
3	HallepoPine	0.3

Table 1 Different P_{veg} values in different vegetation types.

Category	Density	P_{den}
1	Sparse	-0.4
2	Normal	0
3	Dense	0.3

Table 2 Different P_{den}values at different vegetation density.

2.4.2. Effect of the wind speed and direction

To model the effect of wind direction and speed on fire propagation, the following equations are used:

$$P_w = exp(c_1 V) ft \quad (2)$$

$$ft = exp(V c_2 (cos(\theta) - 1)) \quad (3)$$

Where c_1 and c_2 are constants (see Table 3), θ is the angle between the fire spread direction and the wind direction, and V is the wind speed. Using this formula, the wind direction can take a continuous value between 0 and 360°.

These equations illustrate the general form of the probability P_w as a function of the angle θ for certain arbitrary values of the constants c_1, c_2, and the wind speed V.

Parameter	value
P_h	0.58
a	0.078
c_1	0.045
c_2	0.131

Table 3 The different values of the constant P_h, a, c_1, and c_2.

2.4.3. Effect of the ground elevation

The effect of ground elevation (slope) on fire propagation is modeled by the following equation:

$$R_s = R_{0s} exp(a\ \theta_s) \quad (4)$$

R_{0s} : is the spread rate when the slope angle is zero,

θ_s : is the slope angle.

a: is a constant that can be adjusted based on experimental data.

According to equation (4), the probability of modeling the slope effect is calculated as:

$$P_s = exp\,(a\ \theta_s) \tag{5}$$

It is important to note that due to the square grid, the slope angle is calculated differently depending on whether the two neighboring cells are adjacent or diagonal to the burning cell. Specifically, for adjacent cells, the slope angle is given by:

$$\theta_s = \tan^{-1}\left(\frac{E_1 - E_2}{L}\right) \tag{6}$$

Where $E_1$and $E_2$are the elevations of the two cells, and L is the length of the square side. For diagonal cells, the formula transforms into:

$$\theta_s = \tan^{-1}\left(\frac{E_1 - E_2}{L\sqrt{2}}\right) \tag{7}$$

3. Model calibration using Simulated Annealing

This section outlines the process of calibrating the wildfire simulation model presented in section 2, using the simulated annealing metaheuristic. The calibration aims to optimize model parameters to enhance the accuracy of the simulation results compared to observed wildfire data. The section details the definition of the objective function, the control parameters of simulated annealing, the data pre-processing steps, the calibration process, and the evaluation of the calibration results.

3.1. Definition of Objective function

The objective function in the simulated annealing process quantifies the difference between the simulated and observed wildfire behaviors. The objective function (equation 11) used is calculated using F1-score, which is proposed to measure predictive performance of a binary classification [20], it is particularly relevant in applications where the positive class is rare relative to the negative class. Nowadays, the F1-score is widely used in most application areas, including to measure the performance of forest fire simulations [21].

The F1-score (equation 10) is the harmonic mean of precision (equation 8) and recall (equation 9):

Precision: Measures the proportion of cells identified as burning that actually are burning (accuracy of positive predictions). It's calculated as:

$$Precision = \frac{TP}{(TP + FP)} \tag{8}$$

Recall: Measures the proportion of actual burning cells that are correctly identified by the simulation (completeness of positive predictions). It's calculated as:

$$Recall = \frac{TP}{(TP + FN)} \tag{9}$$

Where:

- **TP (True Positive):** True positives are the number of cells that are correctly classified as belonging to the burned area.
- **TN (True Negative):** True negatives are the number the cells that are correctly classified as not belonging to the burned area.
- **FN (False Negative):** False negatives are the number of cells that are incorrectly classified as not belonging to the burned area.
- **FP (False Positive):** False positives are the number of cells that are incorrectly classified as belonging to the burned area.

F1-score: Combines precision and recall into a single metric, providing a balanced view of the model's performance. It's calculated as:

$$F1 - score = 2 * \frac{(Precision * Recall)}{(Precision + Recall)} \tag{10}$$

The objective function $\boldsymbol{F}(\boldsymbol{P})$ is the average of the F1-scores from the 10 simulation runs. This averaging process helps to smooth out the effects of model stochasticity and provides a more reliable estimate of the model's accuracy:

$$\boldsymbol{F}(\boldsymbol{P}) = \frac{1}{10}\sum_{i=1}^{10} FS_i(\boldsymbol{P}) \tag{11}$$

Where:

- $\boldsymbol{P}$ represents the vector of model parameters to be calibrated (see section 3.2).
- $FS_i(\boldsymbol{P})$ is the F1-score of the i-th simulation run with parameters $\boldsymbol{P}$.

3.2. Model Parameters to Calibrate

The parameters of the model to be calibrated are those which are used to calculate the P_{burn} probability (see equation 1), and determine the impact of the different factors which influence the fire spread (type and density of vegetation, elevation, speed and direction the wind):

- $\boldsymbol{P_h}$ **:** is a constant, the value 0.58.
- $\boldsymbol{a}$ **:** is a constant parameter in the model with a value of 0.078.
- $\boldsymbol{MinP_{veg}}$: the minimum value of $\boldsymbol{P_{veg}}$ (see equation 1).
- $\boldsymbol{MaxP_{veg}}$**:** the maximum value of $\boldsymbol{P_{veg}}$ (see equation 1).
- $\boldsymbol{MinP_{dens}}$**:** the minimum value of $\boldsymbol{P_{dens}}$ (see equation 1).
- $\boldsymbol{MaxP_{dens}}$**:** the maximum value of $\boldsymbol{P_{dens}}$ (see equation 1).
- $\boldsymbol{c_1}$**:** is a constant parameter in the model with a value of 0.045.
- $\boldsymbol{c_2}$**:** is a constant parameter in the model with a value of 0.131.

Table 4 summarizes the parameters of the model and their corresponding ranges used in the simulated annealing process.

Parameter	Value in the basic model	Equation	Table	Range
P_h	0.58	equation 1	Table 3.1	[0.01 , 1.5]
a	0.078	equation 4 (Effect elevation)	Table 3.1	[0,001 , 0.5]
$MinP_{veg}$	-0.4	equation 1 (Type of vegetation effect)	Table 3.1	$[-1. -0.1]$
$MaxP_{veg}$	0.3	equation 1 (Type of vegetation effect)	Table 3.1	[0.1 , 1.5]
$MinP_{dens}$	-0.3	equation 1 (density of vegetation effect)	Table 3.2	$[-1. -0.1]$
$MaxP_{dens}$	0.4	equation 1 (density of vegetation effect)	Table 3.2	[0.1 , 1.5]
c_1	0.045	equation 2 (Effect of wind)	Table 3.3	[0,001 , 0.05]
c_2	0.131	equation 3 (Effect of wind)	Table 3.3	[0,001 , 0.3]

Table 4 Model parameters to calibrate

3.3. Data pre-processing

Data normalization is a crucial step in the pre-processing phase of model calibration. It involves transforming the raw data into a format compatible with the models. For our forest fire simulation model, we need to normalize the vegetation density and type values from the data to match the parameter ranges used in the model.

3.3.1. Normalizing Vegetation type

The vegetation type data in the dataset ranges from 0 to 0.5. However, the model parameter P_{veg} must lie between $MinP_{veg}$ and $MaxP_{veg}$, which are two parameters to calibrate of the model (see Table 3.4).

We apply a linear normalization formula to have a P_{veg} in the range $[MinP_{veg}, MaxP_{veg}]$. The transformation is as follows:

$$P_{veg} = \left(\frac{typeVeg - Min_{typeVeg}}{Max_{typeVeg} - Min_{typeVeg}}\right) \times (MinP_{veg} - MaxP_{veg}) + MinP_{veg} \quad (12)$$

Where:

$typeVeg$: The vegetation type value in the data set

$Min_{typeVeg}$: The minimum value of vegetation type in the data set

$Max_{typeVeg}$: The minimum value of vegetation type in the data set

Given that $Min_{typeVeg} = 0$ and $Max_{typeVeg} = 0.5$, we obtain:

$$P_{veg} = \left(\frac{typeVeg}{0.5}\right) \times (MinP_{veg} - MaxP_{veg}) + MinP_{veg} \quad (13)$$

3.3.2. Normalizing Vegetation Density

The vegetation density data in the dataset ranges from 0 to 100. However, the model parameter P_{dens} must lie between $MinP_{den}$ and $MaxP_{den}$, which are two parameters to calibrate of the model (see Table 4).

We apply a linear normalization formula to have a P_{dens} in the range $[MinP_{den}, MaxP_{dens}]$. The transformation is as follows:

$$P_{den} = \left(\frac{den - Min_{den}}{Max_{den} - Min_{den}}\right) \times (MinP_{dens} - max\_P_{dens}) + MinP_{den} \quad (14)$$

Where:

den: The density value in the data set

Min_{den}: The minimum value of density in the data set

Max_{den}: The minimum value of density in the data set

Given that $Min_{den} = 0$ and $Max_{den} = 100$, we obtain:

$$P_{den} = \left(\frac{den}{100}\right) \times (MinP_{den} - MaxP_{den}) + MinP_{den} \quad (15)$$

3.4. Calibration Process

The calibration aims to enhance the accuracy of the wildfire simulation model. Using the Simulated Annealing (SA) metaheuristic, we systematically adjust the model parameters to minimize the discrepancy between the simulated fire spread and observed data. This section details the steps involved in the calibration process.

3.4.1. Initialization

The calibration process begins with choosing the initial solution, setting the initial temperature and cooling schedule for the Simulated Annealing algorithm.

- **Initial solution:** The parameters to be calibrated are defined in Section. The initial values of these parameters are the values from the original model (see Table 5).

Parameter	Initial value
P_h	0.58
a	0.078
$MinP_{veg}$	-0.4
$MaxP_{veg}$	0.3
$MinP_{dens}$	-0.3
$MaxP_{dens}$	0.4
c_1	0.045
c_2	0.131

Table 5 The initial solution

- **Initial Temperature:** Temperature is a control parameter that starts high and gradually decreases. It determines the probability of accepting worse solutions. In the experiments, we tested different temperature values (see Table 6).

3.4.2. Iterative Optimization

The heart of the calibration process lies in the iterative optimization steps of the SA algorithm.

1. **Perturbation:** Slightly alter the current set of parameters to explore the solution space. We select one of the parameters at random and give it a random number from here range (see Table 4).
2. **Simulation:** Run 10 wildfire simulation model using the perturbed parameters. Calculate the F1-score for each simulation (see equation 10).
3. **Objective Function Evaluation:** Compute the objective function, which is the average of F1-scores of the 10 performed simulations (see equation 11).
4. **Acceptance Criterion:** Determine whether to accept the new parameters based on the Metropolis criterion:

$$\boldsymbol{Probability\ of\ Acceptance} = \begin{cases} \boldsymbol{if}\ \ \Delta > \mathbf{0} \\ \boldsymbol{random\ r} > \boldsymbol{exp}\left(\frac{-\Delta}{\boldsymbol{T}}\right) \end{cases} \quad (12)$$

Where Δ is the change in the objective function, and $\boldsymbol{T}$ the current temperature.

5. **Cooling:** The temperature is decreased at each iteration according to exponential cooling schedule: $T_t = T_0 \times \alpha$. Where T_t the temperature at iteration t, T_0 the initial temperature, and α the cooling rate. We set the cooling rate to $\alpha = 0.9$.
6. **Iteration Termination:** Repeat steps 1-5 until a stopping criterion is met. The stopping criterion that we used is $temperature \leq 1$.

3.4.3. Finalization

Upon completing the iterative optimization, finalize the calibration by recording the best parameters set achieved, that maximizes the objective function.

3.5. Model Implementation

3.5.1. Programming Language and Libraries

❖ **Python Language**

We chose Python to develop our fire simulation application. Its clean syntax makes coding a breeze, and its extensive libraries offer ready-made tools for data handling, control flow, and more, streamlining development. This versatile language transcends platforms, running seamlessly on Windows, Mac OS, Linux, and even embedded systems without rewrites. Finally, Python's object-oriented nature perfectly aligns with the fire

simulation model. Cells become objects, encapsulating their state, behavior, and interaction with the environment. This structured approach simplifies code creation, improves radability, and ensures long-term maintainability.

❖ **Libraries**

The model is implemented in Python using libraries such as:

- **Random** (**random**)**:** This library provides functions for generating random numbers. It's commonly used for simulating random events or creating random data for testing purposes.
- **Math** (**math**)**:** This library offers various mathematical functions, constants, and tools. It includes basic mathematical operations like sine, cosine, square root, as well as more advanced functions like logarithms and factorials.
- **Pygame:** This library is specifically designed for multimedia programming in Python. It provides functionalities for creating graphical user interfaces (GUIs), handling game input (key presses, mouse clicks), rendering graphics (images, shapes, text), and playing sounds.
- **NumPy (**numpy**)**: This library is a fundamental package for scientific computing in Python. It offers powerful tools for working with multi-dimensional arrays and matrices. NumPy provides efficient operations on numerical data, linear algebra functions, and random number generation with better performance compared to the standard random library.
- **Matplotlib** (matplotlib.pyplot): Matplotlib is a plotting library for creating static, animated, and interactive visualizations in Python. pyplot is a module in Matplotlib that provides a MATLAB-like interface.

3.5.2. User interface and model Outputs

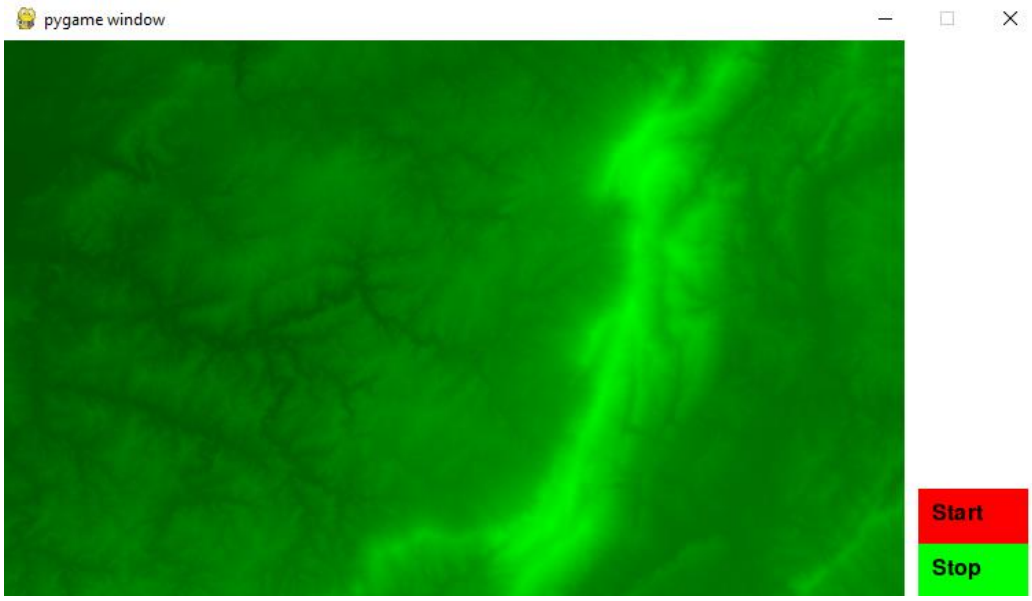

Figure 3.2 Simulation prototype interface

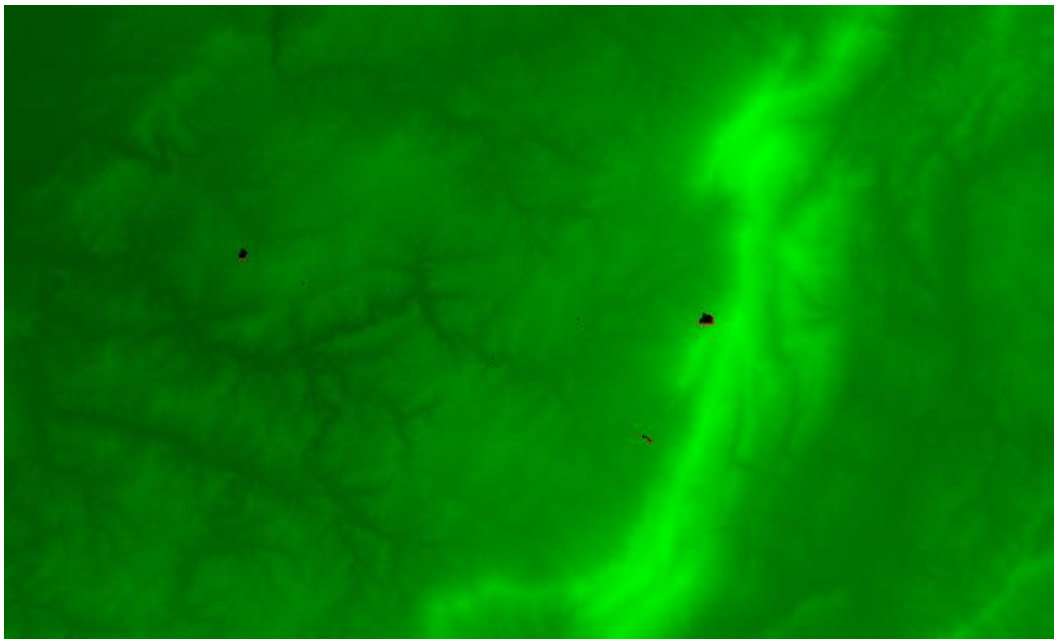

Figure 3.3 A forest fire simulation at the start

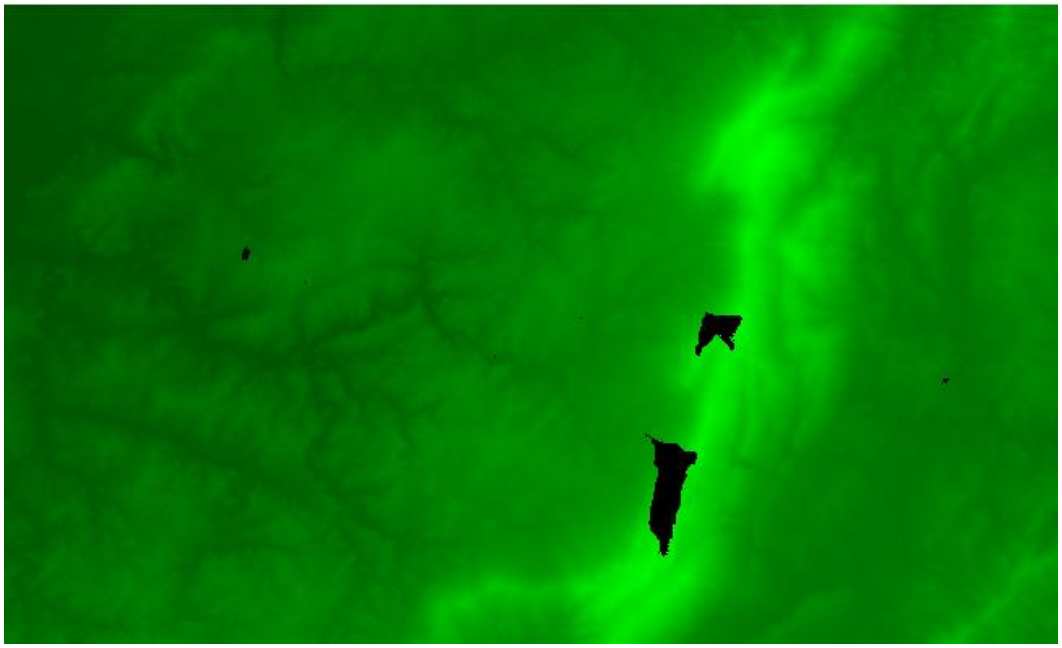

Figure 3.4 End of a forest fire simulation

3.6. Results and Discussion

To investigate the influence of the initial temperature parameter on the calibration process, multiple calibration runs were conducted with varying initial temperaturevalues. The objective function values obtained from each run were compared to assess the quality of the calibration outcomes. Table 6 show the obtained results.

Temperature	Initial Solution	Objective function of solution	Precision	Recall
10000	0,22116842	0,37504769	0,39701437	0,35797556
20000	0,29757718	0,4231854	0,30122987	0,72265271
50000	0,24361159	0,45424251	0,5427411	0,64453752
100000	0,12103862	0,47621206	0,64696614	0,489808
200000	0,19405749	0,5095584	0,37052151	0,827993
500000	0,2722253	0,4035089	0,2996045	0,6190226
1000000	0,2604095	0,44537249	0,42730215	0,7854101
2000000	0,2213882	0,431391	0,3995544	0,76195462

5000000	0,24048865	0,407254	0,3630723	0,46530541
10000000	0,20406134	0,55264836	0,48065506	0,6985689
20000000	0,2581559	0,50353073	0,37167377	0,804188
50000000	0,25732287	0,41452274	0,27560422	0,85849912

Table 6 Model calibration results applying different initial temperatures

The relationship between temperature and objective function of solution: We note that the value of the objective function generally increases with increasing temperature, starting from 0.37504769 at a temperature of 10000 and reaching 0.55264836 at a temperature of 10000000. Which shows In (Figure 3.2) However, there are some fluctuations at different temperatures, such as a decrease in the value of the objective function at 5,000,000 and 50,000,000.

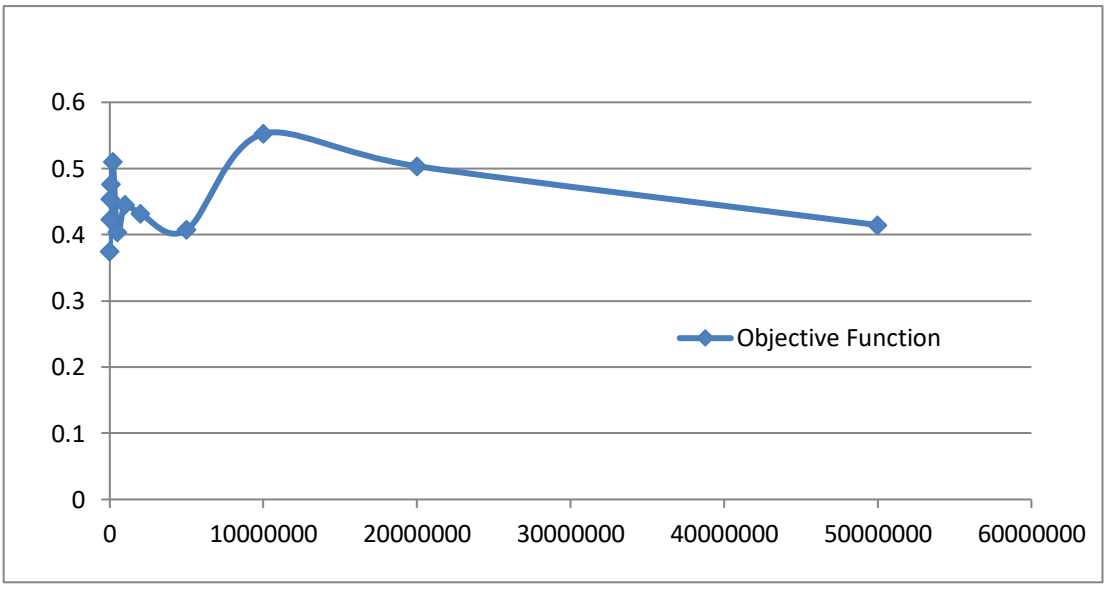

Figure 3.5 Variation of Objective Function with Temperature

The differences between the initial solution and the optimized solution are significant across various performance metrics, indicating substantial improvements through the optimization process. Here's a detailed explanation of each metric:

The difference between the initial solution and the optimized solution is significant, reflecting a substantial improvement in the objective function. For instance, at a

temperature of 10,000, the initial solution value is 0.22116842, while the objective function value is 0.37504769. This significant increase indicates the effectiveness of the optimization process in enhancing the overall performance of the solution.

Although the precision decreases from 0.73179714 in the initial solution to 0.48065506 in the optimized solution, this trade-off is often necessary to improve other metrics. The decrease in precision suggests that the optimized solution makes more positive predictions, including more false positives, but it is part of a broader strategy to improve the overall performance.

The increase in recall from 0.17336823 to 0.6985689 indicates a substantial improvement in the ability of the optimized solution to identify true positives. This significant increase demonstrates the optimization process's success in enhancing the solution’s comprehensiveness in identifying relevant positives.

Objective Function: The significant improvement in the objective function highlights the optimization process's effectiveness in enhancing the overall solution quality.

Precision: The trade-off in precision indicates a shift towards a more inclusive prediction strategy, accepting more false positives to improve recall.

Recall: The marked increase in recall underscores the optimization process's success in identifying a greater number of true positives.

Overall, the table (Table 7) and the histogram (see Figure 3.3) analysis demonstrate that the optimization process has led to a solution with a significantly better objective function and recall, despite a decrease in precision. This reflects the effectiveness of the optimization strategy in improving the overall solution quality.

	Objective function	**Precision**	**Recall**
Initial solution	0,20406134	0,73179714	0,17336823
Optimised solution	0,55264836	0,48065506	0,6985689

Table 7 The difference between the initial solution and the optimized solution

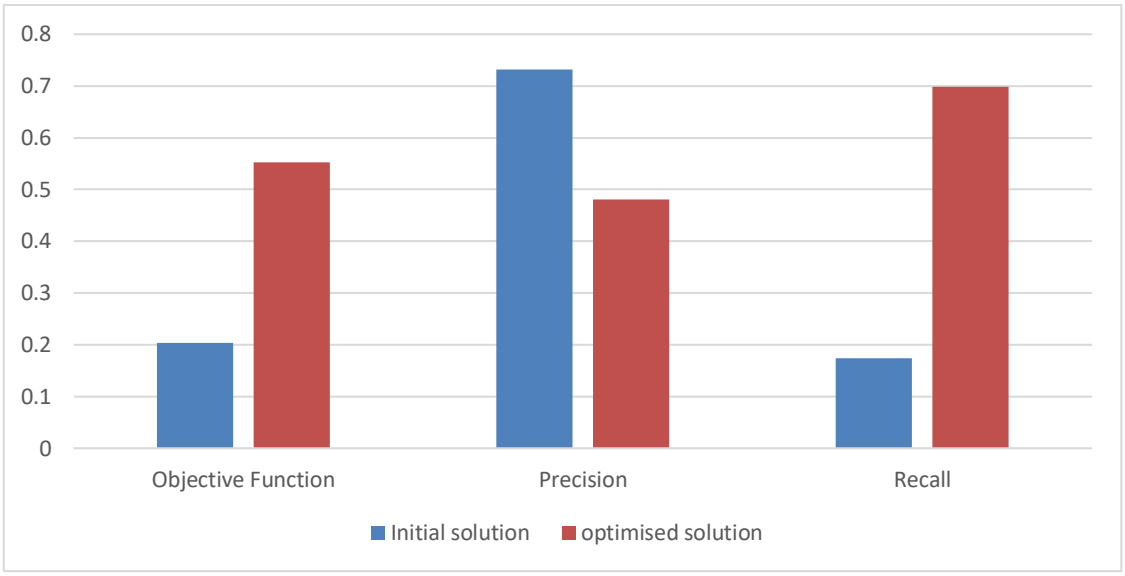

Figure 3.6 Histogram displaying objective function, precision and recall using Initial parameters (Table 5) and optimized parameters (Table 8)

This diagram provides a comprehensive view of how the objective function, recall, and precision metrics vary with changes in temperature during the optimization process. (see Figure 3.4).

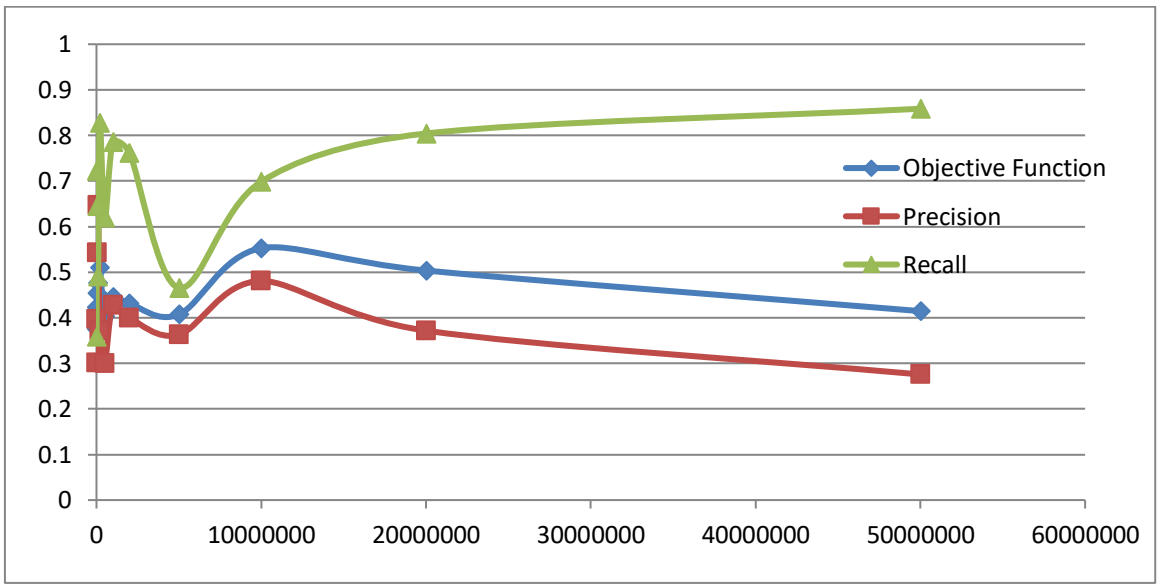

Figure 3.7 Variation of Objective Function , recall, and precision with Temperature

Recall, representing the ability to identify true positives, shows variability with changing temperatures. At low temperatures (e.g., 10,000), recall is relatively low (0.35797556), but it improves significantly at 20,000 (0.72265271), indicating a robust identification of true positives. This trend continues, with recall reaching its highest value of 0.85849912 at 50,000,000. However, there are fluctuations at intermediate temperatures, such as a decrease at 100,000 (0.489808), showing the importance of balancing temperature settings to maintain high recall.

Precision, indicating the accuracy of true positive identification relative to false positives, also varies with temperature. At lower temperatures, precision is moderate, with a value of 0.39701437 at 10,000. It reaches a peak of 0.64696614 at 100,000, indicating a high accuracy of true positive identification. However, at very high temperatures, precision tends to become unstable. For instance, at 50,000,000, precision drops to 0.27560422, reflecting the random search behavior of the algorithm.

Combined Analysis: The combined analysis of the objective function, recall, and precision across different temperatures illustrates the dynamic interplay between these metrics. The optimal temperature of 10,000,000 stands out as it achieves a favorable objective function (0.55264836), high recall (0.6985689), and balanced precision (0.48065506), demonstrating the effectiveness of the algorithm at this setting. This temperature allows the algorithm to effectively explore the solution space without excessive randomness, leading to high-quality solutions.

The best values of parameters that give high objective function value are given in the Table 8.

Parameter	best value
P_h	0.68
a	0.086
$MinP_{veg}$	-0.323
$MaxP_{veg}$	0.473
$MinP_{dens}$	-0.257
$MaxP_{dens}$	0.292
c_1	0.001

c_2	0.035

Table 8 The best parametres values that give high objective function value.

Figure 3.5 illustrates the simulation results using the initial parameters (Table 5), while Figure 3.6 presents the simulation results with the optimized parameters (Table 8). The comparison reveals a significant improvement in the model's ability to identify true positives (TP) after calibration using Simulated Annealing. This enhancement demonstrates the effectiveness of the optimization process in refining the model's accuracy and predictive capability.

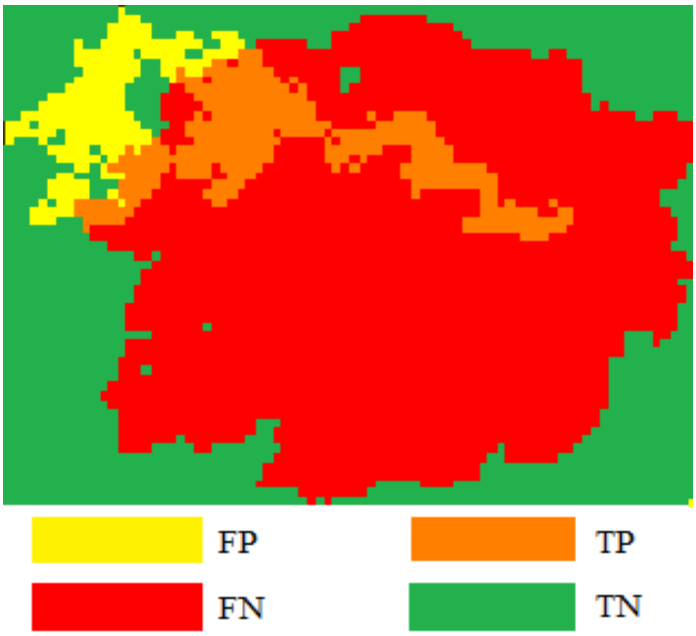

Figure 3.8 Simulation results using the initial parameters (Table 5)

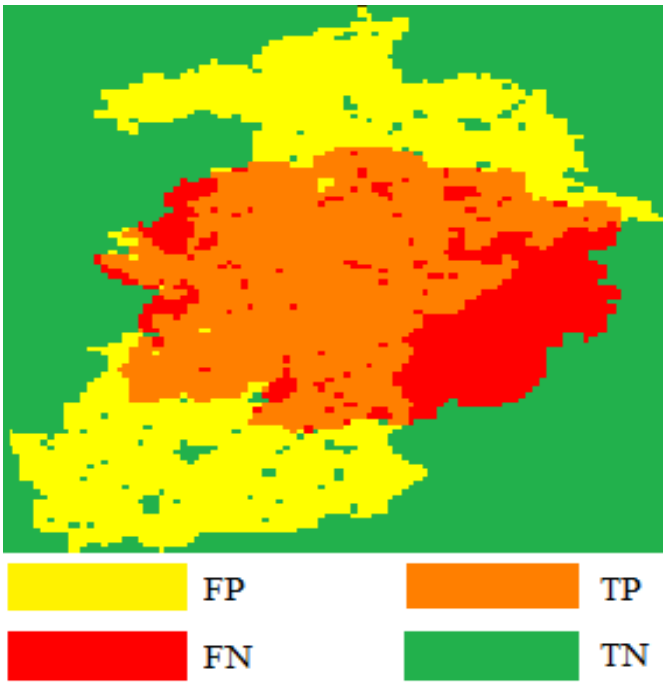

Figure 3.9 Simulation results using the best parameters (Table 8)

4. Conclusion

This chapter has provided a comprehensive overview of our approach conducted to develop and calibrate a wildfire simulation model based on cellular automata and the SA metaheuristic, by systematically outlining the model description, implementation details, calibration process, and evaluation of calibration results.

The calibration process using the SA algorithm has enabled the refinement of model parameters to accurately replicate observed fire behavior.

The calibration process for the wildfire simulation model using the Simulated Annealing (SA) metaheuristic yielded promising results, with notable improvements observed across different temperature value.

General conclusion

Forest fires are a critical environmental challenge, exacerbated by climate change and human activities, demanding sophisticated tools for prediction and management. This master's thesis has explored the development and calibration of a forest fire simulation model using cellular automata and optimization techniques, specifically Simulated Annealing.

The research began with a comprehensive review of modeling and simulation principles, highlighting the suitability of cellular automata for capturing the complex dynamics of forest fire spread. Cellular automata offer a straightforward yet powerful framework, allowing for the simulation of fire behavior through localized interactions within a grid system.

Subsequent chapters detailed the implementation of a forest fire simulation model based on cellular automata. Key components, such as grid definition, cell states, evolution rules, and variables influencing fire spread, were meticulously defined. The model's development aimed to provide a configurable and user-friendly simulation environment, complete with graphical visualization of results.

A significant focus of the thesis was the calibration of the simulation model using Simulated Annealing, a metaheuristic optimization technique. Calibration is essential for aligning the model's output with real-world observations, thereby enhancing its predictive accuracy. Despite the complexity of forest fire dynamics and the numerous variables involved, the application of Simulated Annealing provided a systematic approach to exploring the parameter space and optimizing the model.

The results of the calibration process, while preliminary, demonstrated the potential of combining cellular automata with metaheuristic techniques for forest fire simulation. The calibrated model showed improved alignment with observed fire patterns, though further refinement is needed to achieve higher accuracy and reliability. These preliminary results underscore the challenges inherent in modeling such a complex and stochastic phenomenon but also highlight the progress made in this study.

In conclusion, this thesis has made an interesting contribution to forest fire modeling by developing a cellular automata-based simulation model and employing Simulated Annealing for its calibration. While the results obtained are preliminary, they provide a foundation for future research and improvements. Continued efforts in this direction, including the incorporation of more detailed data and advanced optimization techniques, will further enhance the model's capabilities and its application in forest fire management and mitigation strategies.

Bibliography

[1] 'Modeling and Simulation of Discrete Event Systems | Wiley', Wiley.com. Accessed: May 13, 2024. [Online]. Available: https://www.wiley.com/en-au/Modeling+and+Simulation+of+Discrete+Event+Systems-p-9781118386996

[2] W. Kin and V. W. K. Chan, 'Foundations of Simulation Modeling', 2011. doi: 10.1002/9780470400531.eorms0336.

[3] 'Discrete-Event System Simulation by Banks, Jerry; Nelson, Barry L.; Nicol, David: Good Paperback (2004) | ThriftBooks-Atlanta'. Accessed: May 14, 2024. [Online]. Available: https://www.abebooks.com/9780131446793/Discrete-Event-System-Simulation-Carson-John-0131446797/plp

[4] J. Von Neumann and A. W. Burks, 'Essays on cellular automata', *Urbana, Illinois: University of Illinois Press*, 1966.

[5] 'CHAPITRE II Automates cellulaires.pdf'.

[6] P.-Y. Louis and F. R. Nardi, Eds., *Probabilistic Cellular Automata: Theory, Applications and Future Perspectives*, vol. 27. in Emergence, Complexity and Computation, vol. 27. Cham: Springer International Publishing, 2018. doi: 10.1007/978-3-319-65558-1.

[7] A. Ganeri, *Forest Fire*. Franklin Watts Ltd, 2009.

[8] F. A. Albini, 'A Model for Fire Spread in Wildland Fuels by-Radiation†', *Combustion Science and Technology*, vol. 42, no. 5–6, pp. 229–258, Mar. 1985, doi: 10.1080/00102208508960381.

[9] R. C. Rothermel, 'A mathematical model for predicting fire spread in wildland fuels', *Res. Pap. INT-115. Ogden, UT: U.S. Department of Agriculture, Intermountain Forest and Range Experiment Station. 40 p.*, vol. 115, 1972, Accessed: May 24, 2024. [Online]. Available: https://www.fs.usda.gov/research/treesearch/32533

[10] A. Alexandridis, D. Vakalis, C. I. Siettos, and G. V. Bafas, 'A cellular automata model for forest fire spread prediction: The case of the wildfire that swept through Spetses Island in 1990', *Applied Mathematics and Computation*, vol. 204, no. 1, Art. no. 1, 2008.

[11] J. Ning *et al.*, 'Comparison of Different Models to Simulate Forest Fire Spread: A Case Study', *Forests*, vol. 15, no. 3, p. 563, 2024.

[12] 'A hybrid stochastic Lagrangian – cellular automata framework for modelling fire propagation in inhomogeneous terrains - ScienceDirect'. Accessed: Jun. 03, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1540748922002838

[13] Z. Zheng, W. Huang, S. Li, and Y. Zeng, 'Forest fire spread simulating model using cellular automaton with extreme learning machine', *Ecological Modelling*, vol. 348, pp. 33–43, 2017.

[14] X. Li *et al.*, 'Simulating forest fire spread with cellular automation driven by a LSTM based speed model', *Fire*, vol. 5, no. 1, Art. no. 1, 2022.

[15] S. Mouassa and T. Bouktir, 'Optimisation de l'écoulement de puissance par une méthode métaheuristique (technique des abeilles) en présence d'une source renouvelable (éolienne) et des dispositifs FACTS', 2012.

[16] A. Gherboudj, 'Méthodes de résolution de problèmes difficiles académiques', *Université de Constantine2*, 2013, Accessed: May 29, 2024. [Online]. Available:

http://www.univ-constantine2.dz/files/Theses/Informatique/Doctorat/Amira-Gherboudj.pdf

[17] S. Kirkpatrick, C. Gelatt, and M. Vecchi, 'Simulated annealing, sa', 1982, Accessed: Jun. 10, 2024. [Online]. Available: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=779ffa466fd2e04d5c63cef2443a0b31ee2a25d7

[18] Guemri, O. Cours de résolution de problèmes et optimisation combinatoire, centre universitaire abdelhafid boussouf mila, 2023 - 2024.https://elearning.centre-univ-mila.dz/a2024/course/view.php?id=3116.

[19] 'Leila SAADI.pdf'. Accessed: May 31, 2024. [Online]. Available: http://eprints.univ-batna2.dz/226/1/Leila%20SAADI.pdf

[20] N. Chinchor and B. M. Sundheim, 'MUC-5 evaluation metrics', in *Fifth Message Understanding Conference (MUC-5): Proceedings of a Conference Held in Baltimore, Maryland, August 25-27, 1993*, 1993. Accessed: Jun. 05, 2024. [Online]. Available: https://aclanthology.org/M93-1007.pdf

[21] D. Radke, A. Hessler, and D. Ellsworth, 'FireCast: Leveraging Deep Learning to Predict Wildfire Spread.', in *IJCAI*, 2019, pp. 4575–4581. Accessed: Jun. 05, 2024. [Online]. Available: https://www.ijcai.org/Proceedings/2019/0636.pdf

[22] J. Lara, E. Guerra, A. Boronat, R. Heckel, and P. Torrini, 'Domain-specific discrete event modelling and simulation using graph transformation', *Software & Systems Modeling*, vol. 13, Feb. 2014, doi: 10.1007/s10270-012-0242-3.