utad UNIVERSIDADE DE TRÁS-OS-MONTES E ALTO DOURO

Computação Natural

Mestrado em Engenharia Informática - 1.º ano

José Paulo Barroso de Moura Oliveira
Eduardo José Solteiro Pires

Trabalho Prático

Simulação de Incêndios Florestais Baseada em Agentes

Diogo Medeiros (70633) João Santos (68843)

março 2023

Resumo

Neste trabalho, será desenvolvido um sistema computacional baseado em agentes racionais, utilizando a ferramenta NetLogo, para simular a propagação de incêndios florestais, considerando diversos fatores ambientais. Pretende-se entender melhor o comportamento do fogo em diferentes situações e determinar as melhores estratégias de combate e prevenção. Para alcançar esse objetivo, serão simulados quatro cenários distintos, variando a inclinação do terreno e a direção e intensidade do vento.

Através deste estudo, espera-se demonstrar a importância e complexidade da simulação de incêndios florestais e destacar o valor das abordagens baseadas em agentes para aprofundar a compreensão das interações e comportamentos envolvidos. O trabalho servirá como base para futuras investigações e melhorias no campo da prevenção e combate a incêndios florestais, procurando modelos mais refinados e eficazes que reduzam o impacto dos incêndios florestais nas pessoas e nos ecossistemas.

Palavras-chave – simulação de incêndios florestais, agentes racionais, modelos baseados em agentes, NetLogo, fatores ambientais, topografia, direção e intensidade do vento, prevenção e combate a incêndios

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

Capítulo 1

Introdução

No âmbito da Unidade Curricular de Computação Natural, foi solicitado um trabalho prático que consiste no desenvolvimento de um sistema computacional com agentes racionais usando a ferramenta NetLogo, com o propósito de simular a propagação de um incêndio florestal com base em diferentes fatores ambientais.

1.1 Agentes Racionais

Segundo Russel e Norvig [1], um agente é qualquer coisa que perceciona o seu ambiente através de sensores e atua sobre esse ambiente através de atuadores.

Um agente racional é aquele que toma a decisão correta, sendo necessário definir o contexto da decisão, o que considera certo e o que considera errado. Quando colocado num ambiente, o agente gera uma sequência de ações de acordo com estímulos (ou perceções) [1].

Esta desencadeia uma sequência de estados no ambiente que, quando favorável, determina o sucesso do agente, avaliado de acordo com uma medida de desempenho.

A definição de racionalidade de um agente tem por base:

- A medida de desempenho que define o critério de sucesso;
- O conhecimento prévio do agente sobre o ambiente;
- As ações que o agente pode executar;
- A sequência de perceções do agente até à data.

1.2 Incêndios Florestais

Nos sistemas de simulação de incêndios florestais, os agentes racionais podem desempenhar um papel crucial na prevenção e controlo destes desastres. Podem ser programados para recolher informações sobre as condições ambientais, como temperatura, humidade e direção do vento, e tomar decisões sobre a melhor estratégia de combate ao fogo.

Além disso, estes agentes podem ser utilizados para simular o comportamento do fogo e a propagação do incêndio, permitindo que os gestores do sistema tomem decisões informadas sobre como agir.

Em Viegas [2], o autor apresenta uma visão geral sobre a propagação de incêndios florestais e destaca a importância do estudo e gestão desses desastres devido à sua crescente incidência em anos recentes. O autor discute os processos físicos envolvidos na propagação do fogo, especialmente em condições extremas, com ênfase na segurança e proteção de vidas humanas durante a propagação do incêndio.

Em Papadopoulos, et al. [3], os autores reforçam a importância da previsão da propagação de incêndios florestais e apresentam uma avaliação comparativa de diferentes modelos de simulação existentes na literatura, concluindo que o FARSITE é o mais indicado para esse tipo de previsão. Já em Singh et al. [4], os autores apresentam um estudo realizado na Floresta Taradevi, em Himachal Pradesh, Índia, no qual dados de deteção remota e SIG foram utilizados gerar a entrada necessária para a modelação de incêndios florestais recorrendo ao FARSITE.

Diferentes ferramentas e metodologias têm sido desenvolvidas para simplificar e melhorar estes processos, desde métodos inovadores de autómatos celulares (CA) integrados com Máquinas de Aprendizagem Extrema (ELM) [5], ferramentas 'Web' interativas como a FLogA (Fire Logic Animation), que permitem aos seus utilizadores simular incêndios em florestas reais com base em diferentes condições climáticas [6], e até mesmo ambientes de simulação integrados [7], por exemplo, o DEVS-FIRE [8], baseado na especificação de sistemas de eventos discretos (DEVS) e na utilização de modelos de espaço celular para simular a propagação do fogo e modelos de agentes para simular a contenção do fogo.

Os modelos de simulação de incêndios florestais são amplamente utilizados por especialistas em incêndios e combustíveis nos EUA para apoiar decisões táticas e estratégicas relacionadas com a mitigação do risco de incêndios florestais [9].

A aplicação desses modelos é o resultado do desenvolvimento de um algoritmo de propagação de incêndios de tempo mínimo (MTT), desenvolvido por Finney [9], que torna computacionalmente viável simular milhares de incêndios e gerar mapas de probabilidade e intensidade de queima em grandes áreas. A pesquisa de Finney, et al. [9, 10], demonstra uma abordagem prática para o uso de simulações de incêndios florestais em escalas muito amplas para fins de planeamento operacional e, possivelmente, pesquisa ecológica.

Capítulo 2

Problema

A simulação de incêndios florestais é um desafio importante para os profissionais que trabalham na prevenção e combate a incêndios. Essa tarefa é essencial para entender o comportamento dos incêndios em diferentes situações, e determinar as melhores táticas de combate. Para realizar essa análise de forma eficiente, é necessário considerar diversos fatores que afetam o comportamento do fogo, como topografia, vento, temperatura, fagulhas e tipo de árvores presentes na região.

Para realizar essas simulações com precisão e rapidez, o uso de sistemas de agentes inteligentes tem se mostrado uma opção promissora. Esses sistemas permitem a criação de modelos complexos, que consideram uma abundante de variáveis, e podem simular o comportamento do fogo em tempo real. Além disso, é possível utilizar algoritmos de aprendizado de máquina para otimizar a tática de combate, e reduzir o risco de acidentes ou danos à fauna e flora local.

O modelo de simulação construído deverá apresentar um gráfico com o número de árvores queimadas ao longo do tempo, além de armazenar esses dados em ficheiro. O algoritmo deverá ser executado múltiplas vezes para cada cenário, de modo a garantir a veracidade dos resultados.

Capítulo 3

Metodologia

A ferramenta NetLogo foi escolhida para resolver o problema proposto devido às suas características que a tornam uma linguagem de programação ideal para a modelação de sistemas complexos baseados em agentes.

Algumas suposições tiveram de ser feitas para simplificar o modelo ou torná-lo mais viável, já que nem todos os fatores que afetam o problema podem ser levados em consideração, nomeadamente:

- A floresta tem uma dimensão fixa de 33x33 células (ou *patches*);
- Há dois tipos de árvores na floresta: pinheiros e carvalhos;
- Cada árvore tem uma probabilidade de ser incendiada pelo fogo;
- Cada árvore tem uma probabilidade de criar fagulhas, que podem incendiar outras árvores;
- O terreno tem uma inclinação que afeta a propagação do fogo, assumindo que o fogo sobe;
- O fogo espalha-se para as células adjacentes, com uma probabilidade de se propagar;
- O vento pode soprar o fogo e as fagulhas numa direção específica, independente da inclinação;
- Cada célula tem uma temperatura inicial fixa, que varia com a propagação do incêndio.

3.1 Agentes

A modelação de incêndios florestais com o uso de sistemas de agentes inteligentes envolve a criação de agentes que representem os intervenientes deste cenário, nomeadamente, as árvores da floresta, os fogos, e as fagulhas.

Esses agentes interagem entre si e com o ambiente, permitindo a criação de um modelo dinâmico que simula o comportamento dos incêndios florestais.

Assim, foram definidas as seguintes "raças" (*breeds*) de agentes:

- Árvores (*trees*)
- Fogos (*fires*)
- Fagulhas (*sparks*)

Começando pelos fogos, estes foram definidos com um propósito meramente ilustrativo, de forma a poder visualizar com maior detalhe a propagação do incêndio, pelo que têm associada uma só propriedade: `life-in-ticks`, que representa o seu tempo de vida até os fogos serem destruídos.

Já no caso das fagulhas, estas representam as fagulhas soltas pelas árvores incendiadas, que podem deslocar-se por grandes distâncias, dependendo da direção do vento. Assim, cada fagulha possui duas propriedades inicializadas na sua criação: `final-xcor` e `final-ycor`, que representam a posição final da fagulha onde esta poderá incendiar outras árvores.

Por fim, temos os agentes de maior relevância na simulação, responsáveis pela propagação do fogo, cujo comportamento está dependente das condições climáticas iniciais e atuais da floresta: as árvores, ou *trees*. Dada a complexidade destes agentes, consideramos relevante sumariar as suas propriedades na Tabela 3.1.

Nome	**Tipo**	**Descrição**
kind	string	tipo de árvore
ticks-since-spark	int	n.º de *ticks* desde a última fagulha
is-burning	bool	se a árvore está a arder
is-burnt	bool	se a árvore já ardeu
burning-speed	float	velocidade de queima
spark-probability	float	probabilidade de gerar fagulhas

Tabela 3.1: Propriedades das árvores

Dependendo do tipo escolhido, a árvore pode demorar mais tempo a arder, como é o caso do carvalho, ou gerar fagulhas com maior facilidade, no caso do pinheiro devido às agulhas. Assim, estas propriedades deverão ter valores iniciais fixos associados ao tipo de árvore plantada, como se pode ver na Tabela 3.2.

Propriedade / Tipo	**Pinheiros**	**Carvalhos**
kind, shape	pine-tree	oak-tree
burning-speed	0.3	0.1
spark-probability	0.15	0.05

Tabela 3.2: Tipos de árvore

3.2 Ambiente

O ambiente de simulação, tal como é possível observar na Fig. 3.1, consiste na floresta populada de pinheiros e carvalhos, representada por uma grade de células (ou *patches*), em que cada célula pode conter uma ou mais árvores, um espaço vazio ou um local queimado pelo fogo.

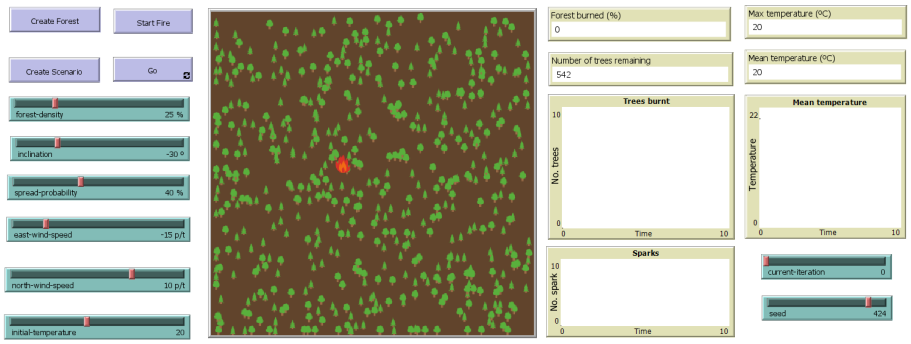

Figura 3.1: Ambiente de simulação no NetLogo

Além disso, a fim de poder retratar e avaliar diferentes cenários de incêndio, foi definido um conjunto de variáveis globais, com o recurso aos elementos gráficos do NetLogo, para representar quer propriedades da floresta em si, quer fatores ambientais que podem afetar a propagação do fogo. Entre outros, o modelo prevê os seguintes parâmetros:

- `forest-density` - a densidade da floresta, entre 0 e 100%;
- `inclination` - a inclinação do terreno, entre $-60°$ e $60°$, responsável pela altitude de cada célula;
- `east-wind-speed` - a velocidade do vento na direção leste (negativa se o vento soprar para o oeste), entre -25 e 25 p/t:
- `north-wind-speed` - a velocidade do vento na direção norte (negativa se o vento soprar para o sul), entre -25 e 25 p/t;
- `initial-temperature` - a temperatura inicial para cada célula do mundo, entre 0 °C e 45 °C;
- `spark-frequency` - a frequência de ocorrência das fagulhas em ticks, por omissão, 150 ticks;
- `spread-probability` - a probabilidade de uma árvore ser incendiada pelo fogo, entre 0 e 100%;
- `forest-seed` - a semente para o gerador de números aleatórios, entre 0 e 500;

- `current-run` - o número da *run* atual de um dado cenário, entre 0 e 10;
- `iterations` - a lista de entradas contendo um *snapshot* do modelo no final de cada `tick`.

Por fim, os patches também apresentam propriedades únicas, nomeadamente, `temperature` e `altitude`, que representam, respetivamente, a temperatura em cada célula à medida que o incêndio se propaga, e a altitude determinada pela inclinação do terreno.

3.3 Algoritmo

O objetivo principal deste algoritmo é fornecer uma ferramenta eficiente para modelar e prever o comportamento dos incêndios florestais, permitindo a avaliação dos efeitos de diferentes cenários na propagação do fogo. A simulação baseada em agentes é uma abordagem que considera a interação entre os elementos do ambiente e os agentes que nele atuam, permitindo a modelação de comportamentos complexos e dinâmicos.

O algoritmo descrito aqui considera fatores como topografia, vento, temperatura, a estrutura da comunidade vegetal e outros elementos relevantes para o comportamento do fogo. A seguir, serão apresentados detalhes do funcionamento do algoritmo e como ele foi implementado para atender aos objetivos.

Algoritmo 1: Criação da floresta (`createForest`)

```
sparkFrequency ← 150;
seed ← forestSeed;
for patch ∈ patches do
    altitude ← calcAltitude(pxcor);
    temperature ← initialTemperature;
    if random(0, 100) < forestDensity then
        plantTree(pxcor, pycor);
    end
end
```

Em Alg. 1, encontra-se detalhado o processo de população da floresta. Este começa por definir a frequência, em ticks, de criação de fagulhas, seguido da atribuição da *seed* responsável pela distribuição de árvores, bem como pela criação do fogo. A cada célula é atribuída uma altitude dependente da sua posição, bem como uma temperatura inicial escolhida pelo utilizador. Também, segundo a densidade da floresta, são plantadas árvores nos diferentes patches.

Com a floresta plantada, resta criar o fogo inicial, tal como se pode ver em Alg. 2. Começa-se por escolher aleatoriamente um 'patch', no qual é criada a primeira chama. A seguir, são salvas as configurações do cenário em ficheiro YAML, e reiniciada a semente. O modelo entra num ciclo condicional, responsável por salvar o estado do modelo em cada iteração e de propagar o incêndio. Após o fim do ciclo, são guardados em ficheiro CSV os resultados das iterações.

Algoritmo 2: Criação do fogo inicial (`startFire`)

```
patch ← random(patches);
ignite(patch);
saveConfig();
seed ← random(seeds);
while anyTreesBurning() do
    saveIteration();
    fire();
end
saveIterations();
```

Algoritmo 3: Evolução do incêndio (`fire`)

```
for tree ∈ trees do
    if isBurning then
        if color < "yellow" then
            spreadFire(neighbors, altitude);
        end
        if color < "brown" then
            if random(0, 1) < sparkProbability and
             ticksSinceSpark > sparkFrequency then
                create(spark);
                ticksSinceSpark ← 0;
            end
            else
                ticksSinceSpark ← ticksSinceSpark + 1;
            end
        end
    end
end
for spark ∈ sparks do
    if position ≠ finalPosition then
        forward(0.1);
    end
    else
        ignite(patch-here);
    end
end
fadeEmbers();
tick();
```

O Algoritmo 3 representa o algoritmo responsável pela evolução do incêndio ao longo da execução do modelo. Em cada tick, as árvores a arder espalham o fogo para as células vizinhas e geram fagulhas, segundo uma certa probabilidade, consoante a cor das suas folhas - esta representa o estado da árvore, mais ou menos queimada. Já as fagulhas movem-se em direção à sua posição final, na qual incendeiam as árvores presentes. Por fim, o estado de queima das árvores é atualizado com o procedimento `fadeEmbers`, responsável por alterar as propriedades da árvore, em particular a sua cor.

Algoritmo 4: Ignição do fogo (`ignite`)

```
create(fire);
for tree ∈ trees−here do
    if not (isBurning and isBurnt) then
        isBurning ← true;
    end
end
```

Para incendiar as árvores, recorremos ao procedimento `ignite`, definido em Alg. 4. Este algoritmo é responsável por criar um agente *fire* na célula atual, bem como de incendiar quaisquer árvores que não estejam a arder na mesma.

Algoritmo 5: Propagação do fogo (`spreadFire`)

```
Input: fireAltitude
probability ← spreadProbability;
direction ← towards(this);
switch direction do
    case 0° do
        probability ← probability − northWindSpeed;
    end
    case 90° do
        probability ← probability − eastWindSpeed;
    end
    case 180° do
        probability ← probability + northWindSpeed;
    end
    case 270° do
        probability ← probability + eastWindSpeed;
    end
end
meanTemp ← meanTemperature(neighbors);
probability ← probability + ln²(meanTemp + 1);
if fireAltitude > altitude then
    probability ← probability · (1 + |tan(inclination/3)|);
end
if random(0, 100) < probability then
    ignite(patch-here);
end
```

Por fim, temos o algoritmo responsável pela propagação do fogo em si, retratado em Alg. 5. Começa-se por determinar a direção do fogo em relação à célula atual, e por inicializar a probabilidade de propagação com o valor definido pelo utilizador no início do programa. A seguir, consoante a direção do vento, ajusta-se adequadamente esta, tal que o mesmo possa contribuir, positiva ou negativamente para essa propagação. Segue-se o cálculo da temperatura média nas células vizinhas, sendo igualmente usada para modificar novamente a propriedade (quanto mais quente estiver o ambiente em redor, melhor se propaga o fogo). Por fim, e porque o fogo tende a subir, quando enfrenta terrenos íngremes, a probabilidade é alterada uma última vez para refletir a inclinação do terreno. Termina-se por incendiar a célula presente, segundo o valor final da probabilidade de propagação.

Todos os detalhes da implementação do modelo de simulação em NetLogo, bem como do módulo desenvolvido em Python para o processamento dos resultados, podem ser consultados no Apêndice A.

Capítulo 4

Resultados

Neste capítulo serão apresentados e analisados os resultados da simulação de um incêndio florestal, em quatro cenários distintos. Estes cenários apresentam configurações de modelo semelhantes, diferindo apenas nos seguintes parâmetros: velocidade do vento no sentido norte e este, e inclinação do terreno. Todos os valores podem ser consultados na Tab. 4.1.

Propriedades	**Cenário 1**	**Cenário 2**	**Cenário 3**	**Cenário 4**
Densidade	25%	-	-	-
Carvalhos/pinheiros	271/272	-	-	-
Prob. propagação	0.4	-	-	-
Temperatura inicial	20 °C	-	-	-
Inclinação	30°	−30°	30°	−30°
Vento Norte	-10	-10	10	10
Vento Este	15	15	15	15

Tabela 4.1: Configurações dos diferentes cenários

Cada cenário foi executado onze vezes, e para cada execução foram com guardados em CSV os seguintes parâmetros do modelo: tick, número de árvores ardidas - absoluto, percentual e para cada tipo de árvore, número de fagulhas na floresta, temperatura média e máxima.

4.1 Cenário 1

O primeiro cenário apresenta uma inclinação positiva do terreno, de 30°, com vento sudeste de intensidade $(-10, 15)\ \mathrm{m\,s^{-1}}$, ou seja, a favor da propagação do incêndio. O modelo terminou com 71.3% da floresta ardida, após 3377 ticks, ou 387 árvores em termos absolutos - 205 carvalhos e 182 pinheiros.

De início, o fogo propaga-se rápido, como se pode notar pelas curvas acentuadas das Fig. 4.1 e 4.2, acabando por abrandar por volta dos 2000 ticks e estabilizar pouco tempo depois. As fagulhas, dado a sua aleatoriedade e condições específicas de geração, apresentam uma evolução mais errática, evidente pela Fig. 4.3, só começando a espalhar-se a partir dos 400 ticks, e atingindo um pico de 15 fagulhas por tick por volta dos 1500 ticks. Dado que a floresta é pouco densa, a temperatura média não ultrapassou os 330 °C, já a temperatura máxima, atingida após 800 ticks (Fig. 4.4), foi muito superior, cerca de 1870 °C.

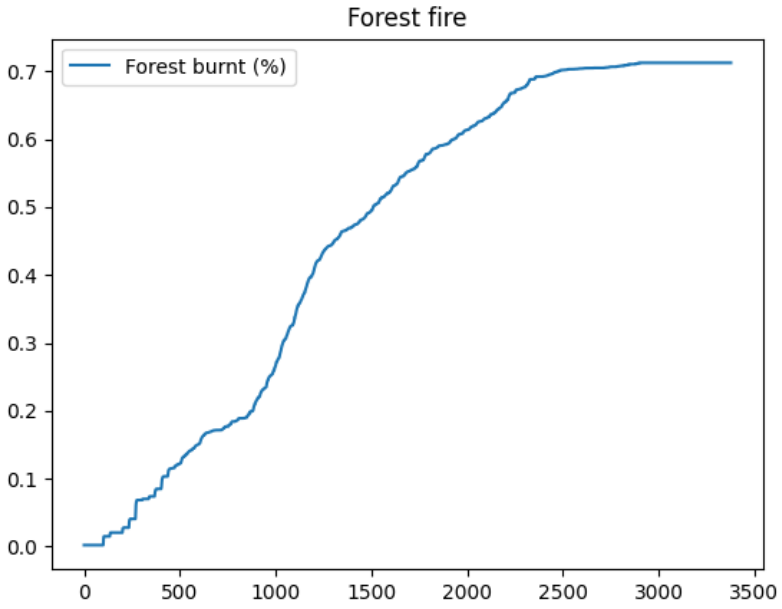

Figura 4.1: Cenário 1 - Total de floresta ardida (em percentagem)

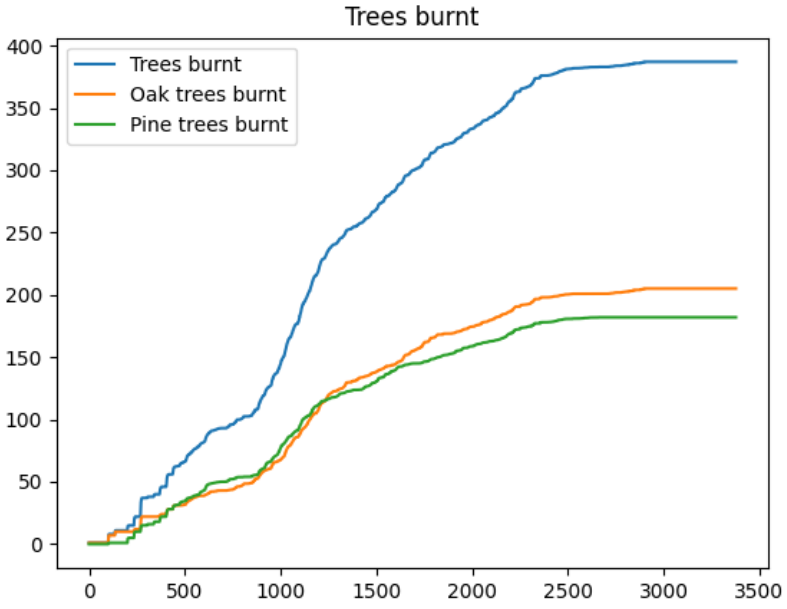

Figura 4.2: Cenário 1 - Total de árvores ardidas

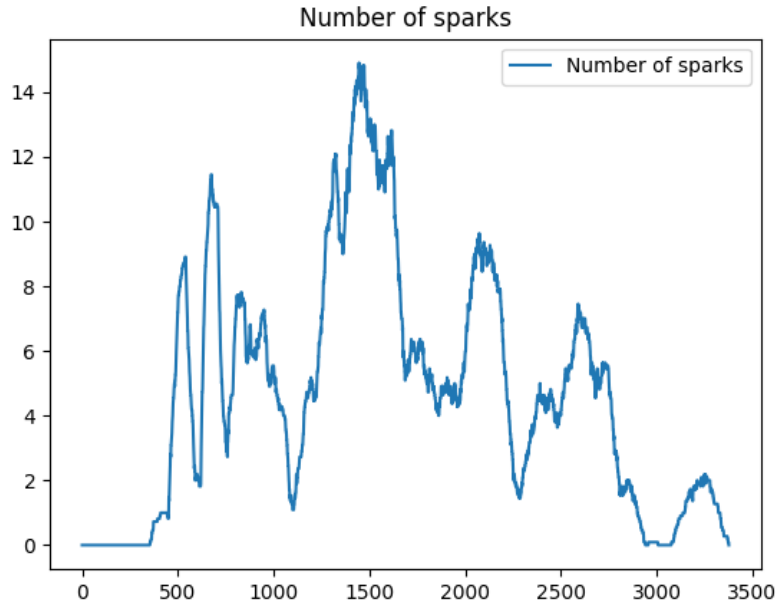

Figura 4.3: Cenário 1 - Evolução do número de fagulhas

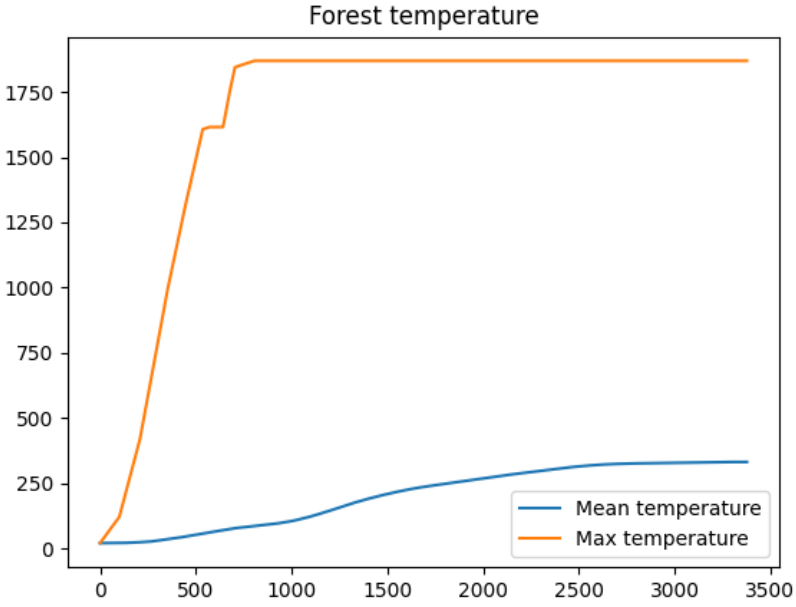

Figura 4.4: Cenário 1 - Evolução da temperatura média e máxima

4.2 Cenário 2

O segundo cenário apresenta uma inclinação negativa do terreno, de $-30°$, com vento sudeste de intensidade $(-10, 15)\ \mathrm{m\,s^{-1}}$, ou seja, contra a propagação do incêndio. O modelo terminou com 71.4% da floresta ardida, após 3359 ticks, ou 388 árvores em termos absolutos - 206 carvalhos e 182 pinheiros.

Apesar das condições adversas, o fogo propagou-se rapidamente, à semelhança do que aconteceu em 4.1. Quando analisadas as curvas das Fig. 4.5 e 4.6, torna-se evidente que o modelo teve uma evolução quase idêntica, porventura graças à alta probabilidade de propagação (40%). Novamente, as fagulhas, apresentam uma evolução mais instável, evidente pela Fig. 4.7, atingindo um pico de 16 fagulhas por tick por volta dos 1400 ticks. Por fim, a temperatura média ficou-se igualmente pelos 330 °C, já a temperatura máxima, atingida após 800 ticks (Fig. 4.8), foi à volta de 1870 °C.

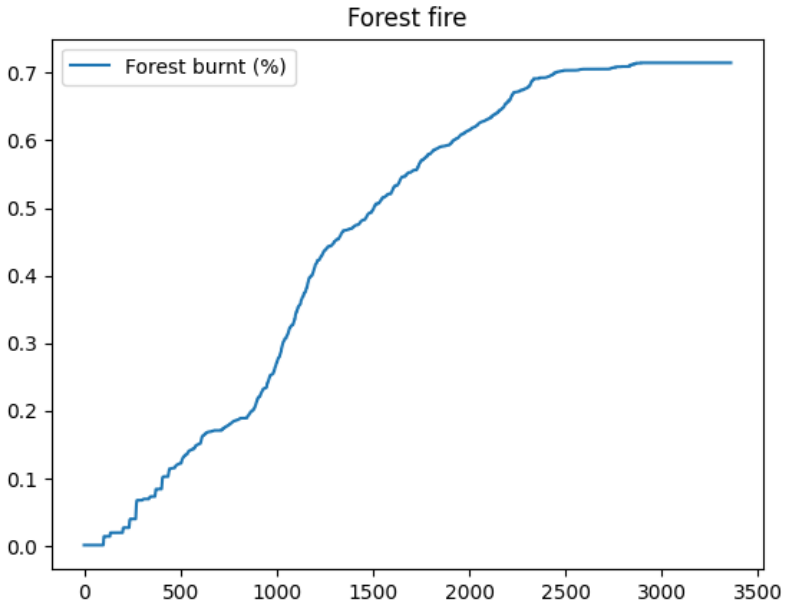

Figura 4.5: Cenário 2 - Total de floresta ardida (em percentagem)

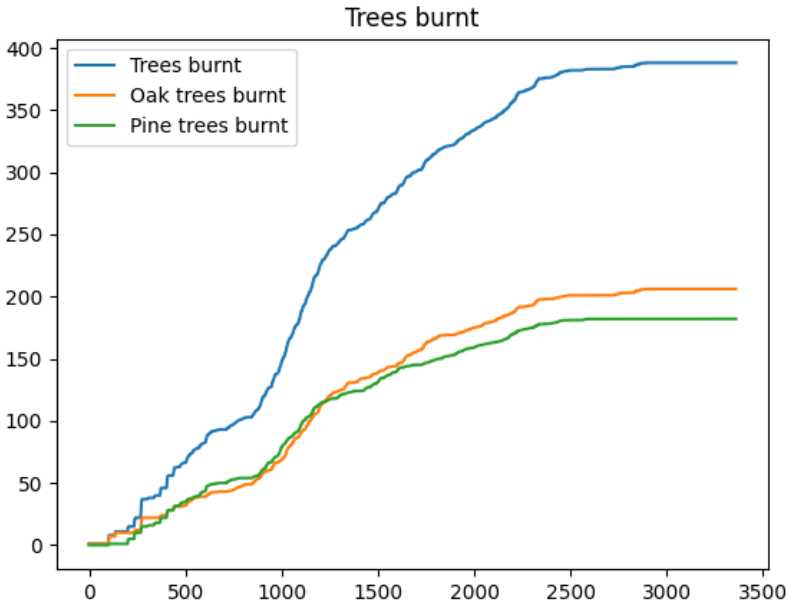

Figura 4.6: Cenário 2 - Total de árvores ardidas

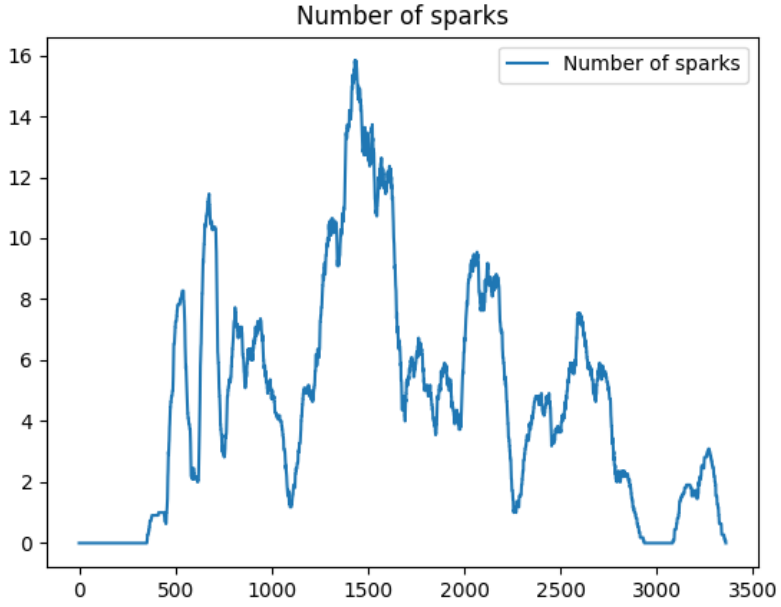

Figura 4.7: Cenário 2 - Evolução do número de fagulhas

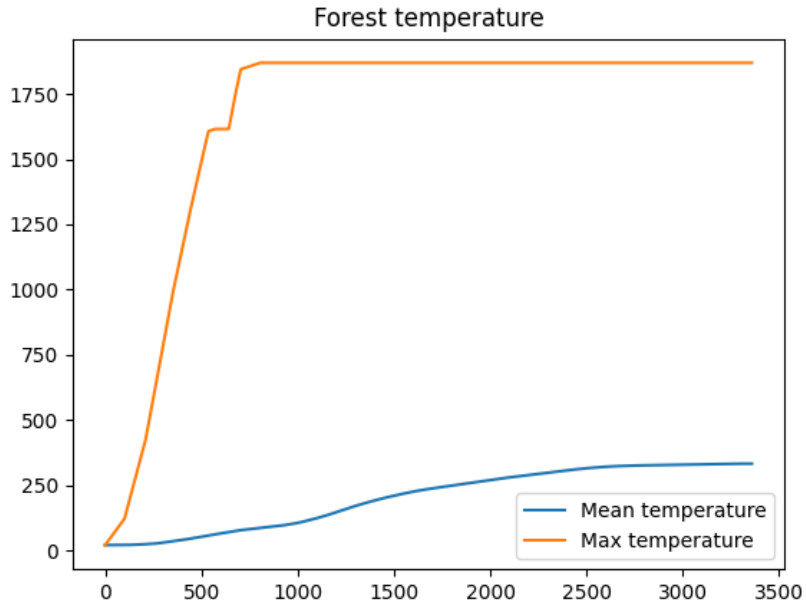

Figura 4.8: Cenário 2 - Evolução da temperatura média e máxima

4.3 Cenário 3

O terceiro cenário apresenta uma inclinação positiva do terreno, de 30°, com vento noroeste de intensidade $(10, -15)$ $\mathrm{m\,s^{-1}}$, ou seja, contra da propagação do incêndio. O modelo terminou com 93.4% da floresta ardida, após 3042 ticks, ou 508 árvores em termos absolutos - 250 carvalhos e 258 pinheiros.

O incêndio conseguiu destruir uma maior área da floresta, visível em Fig. 4.9, em menos tempo que os cenários anteriores. Ao nível do tipo de árvores ardidas, dada a sua idêntica distribuição pelo terreno, não houve variações significativas entre pinheiros e carvalhos, como se pode constatar pela Fig. 4.10. Também ao nível das fagulhas criadas, é possível notar um aumento considerável, com o modelo a atingir um pico de 18,5 fagulhas por tick por volta 1500 ticks (Fig. 4.11). A temperatura média final foi ligeiramente superior, na ordem dos 412 °C, já a temperatura máxima, atingida após 800 ticks (Fig. 4.12), manteve-se nos 1870 °C.

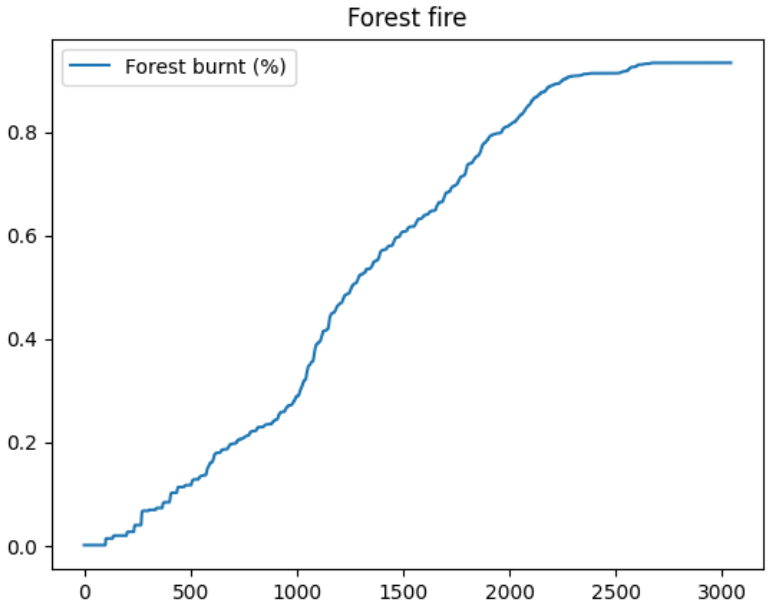

Figura 4.9: Cenário 3 - Total de floresta ardida (em percentagem)

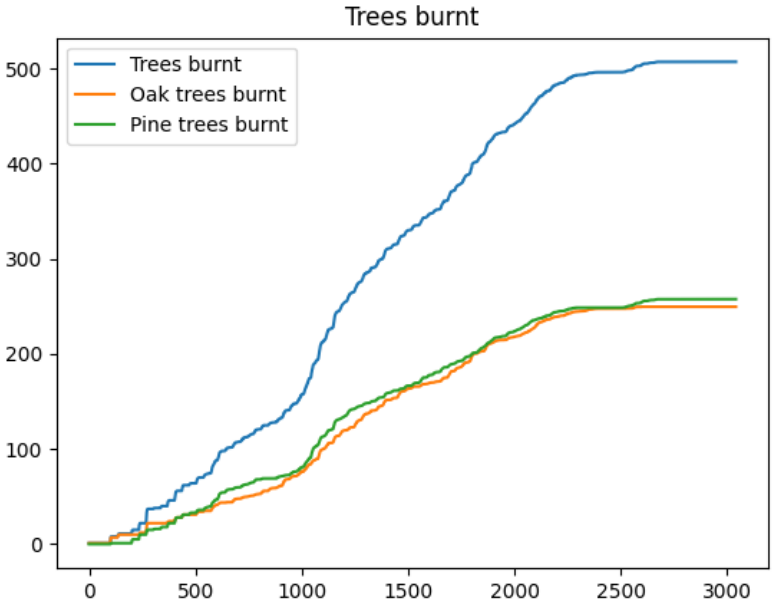

Figura 4.10: Cenário 3 - Total de árvores ardidas

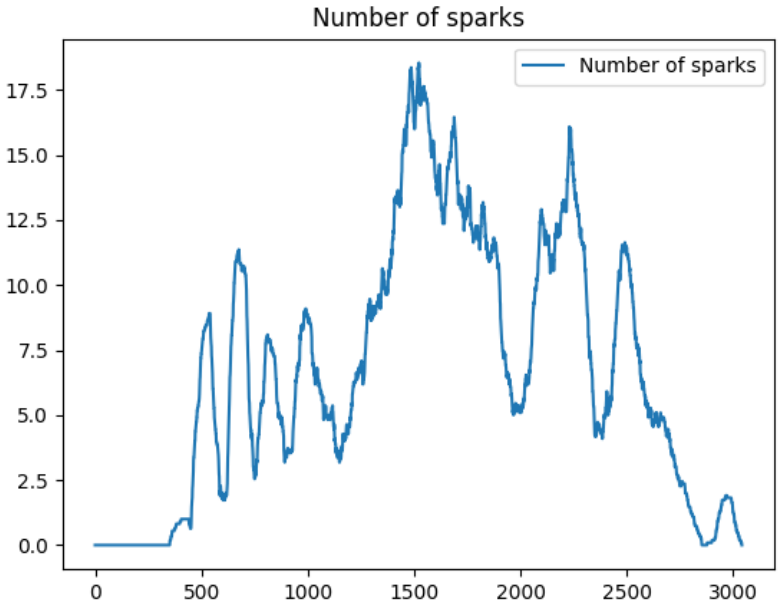

Figura 4.11: Cenário 3 - Evolução do número de fagulhas

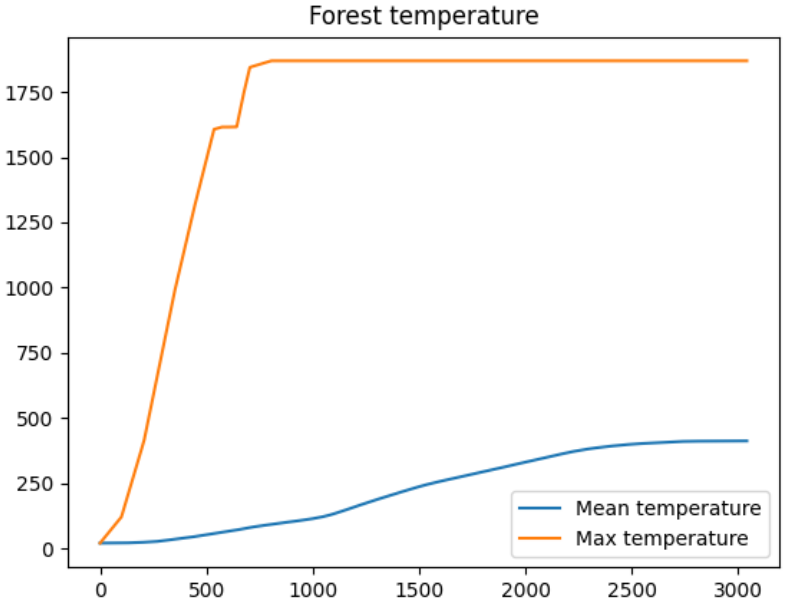

Figura 4.12: Cenário 3 - Evolução da temperatura média e máxima

4.4 Cenário 4

O quarto e último cenário apresenta uma inclinação negativa do terreno, de $-30°$, com vento noroeste de intensidade $(10, -15)$ $\mathrm{m\,s^{-1}}$, ou seja, a favor da propagação do incêndio. O modelo terminou com 93.3% da floresta ardida, após 3103 ticks, ou 507 árvores em termos absolutos - 250 carvalhos e 257 pinheiros.

Este cenário apresenta condições de vento idênticas ao descrito em 4.3, pelo que o modelo em si também evidenciou uma evolução do incêndio similar, quer ao nível da floresta ardida (Fig. 4.13), quer ao nível do tipo de árvores ardidas (Fig. 4.14). No que concerne as fagulhas, o modelo atingiu um pico marginalmente superior, de 18,5 fagulhas por tick por volta 1500 ticks (Fig. 4.15). A temperatura média final foi por volta de 411 °C, já a temperatura máxima, atingida após 800 ticks (Fig. 4.16), manteve-se nos 1870 °C.

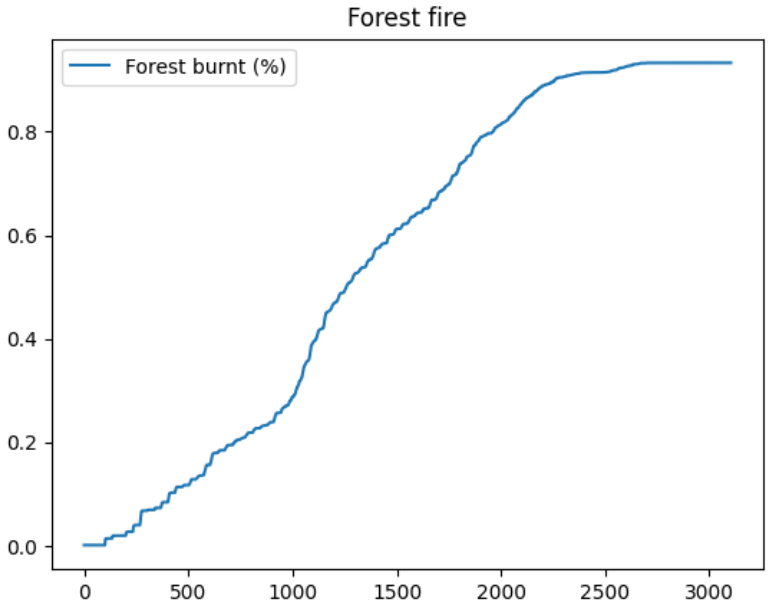

Figura 4.13: Cenário 4 - Total de floresta ardida (em percentagem)

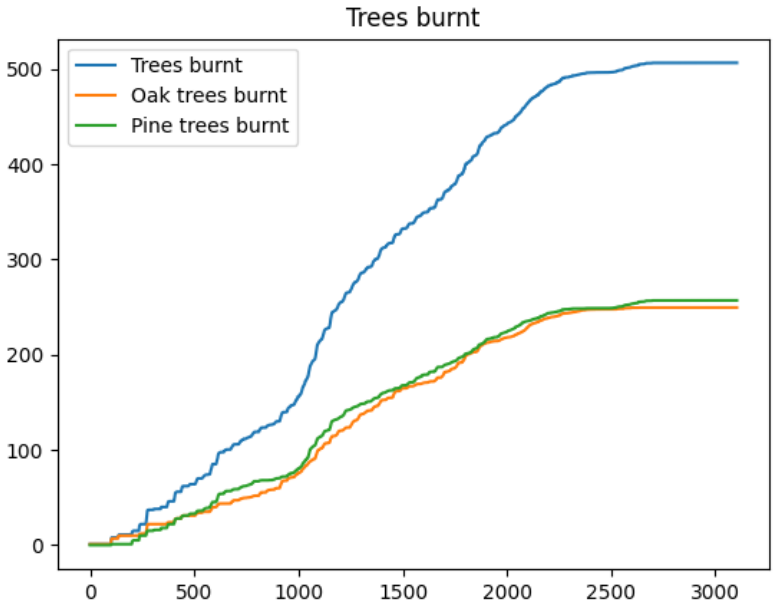

Figura 4.14: Cenário 4 - Total de árvores ardidas

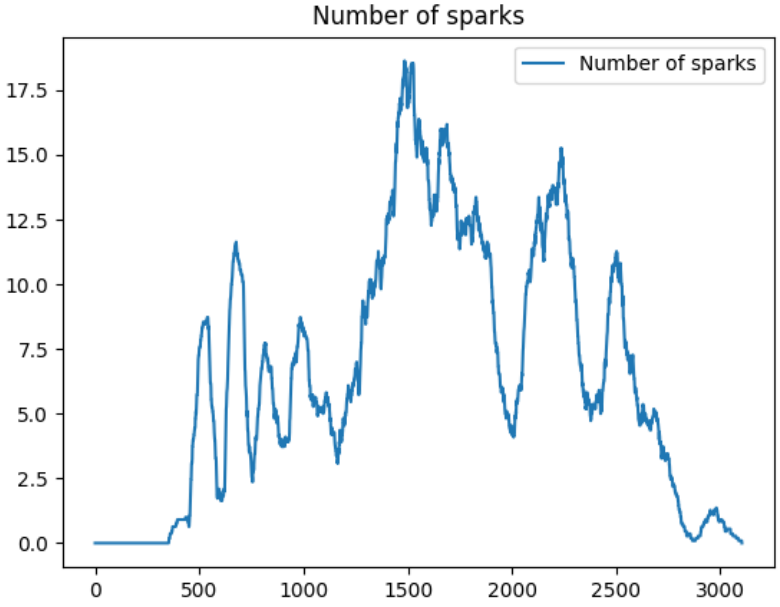

Figura 4.15: Cenário 4 - Evolução do número de fagulhas

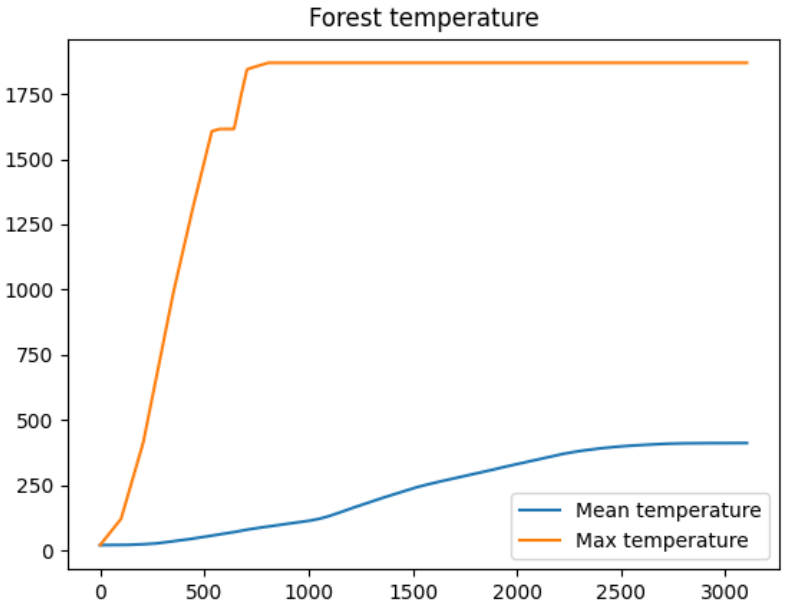

Figura 4.16: Cenário 4 - Evolução da temperatura média e máxima

Capítulo 5

Conclusões

Neste trabalho, desenvolveu-se um sistema computacional com agentes racionais utilizando a ferramenta NetLogo, visando simular a propagação de um incêndio florestal considerando diferentes fatores ambientais. Foram analisados quatro cenários distintos, variando a inclinação do terreno e a direção do vento.

Os resultados obtidos não foram conforme o esperado, uma vez que os cenários com vento a favor e contra a inclinação apresentaram valores contraditórios. As nossas expectativas eram de que a propagação do incêndio seria maior nos casos em que o vento estivesse a favor da inclinação, e menor nos casos em que o vento estivesse contra a inclinação. No entanto, o que se observou foi que a percentagem da floresta ardida e outros parâmetros analisados não seguiram essa lógica.

Uma possível explicação para os resultados contraditórios pode ser a simplificação de alguns elementos do modelo, bem como a aleatoriedade de fatores como a geração de fagulhas. Outro fator que poderá ter contribuído para essa contradição é a alta probabilidade de propagação utilizada, que pode ter mascarado o efeito esperado da direção do vento e da inclinação do terreno.

A partir das conclusões deste trabalho, é possível sugerir melhorias no modelo de simulação para investigação futura. Estas melhorias podem incluir a consideração de outros fatores ambientais, como a humidade do ar e do solo, e uma análise mais aprofundada das probabilidades de propagação. Além disso, a realização de mais testes com diferentes configurações de parâmetros pode ajudar a compreender melhor as discrepâncias observadas.

Concluímos que a simulação baseada em agentes é uma ferramenta valiosa no estudo de incêndios florestais, permitindo aos investigadores compreender melhor as complexas interações e comportamentos envolvidos. Essas simulações têm sido usadas para explorar tópicos como estratégias de combate a incêndios, impacto do clima e topografia, e fatores humanos na gestão de incêndios. A melhor compreensão destes fatores pode levar a decisões mais informadas e práticas mais eficazes, tornando a simulação baseada em agentes uma ferramenta essencial na redução do impacto dos incêndios florestais. Embora os resultados não tenham sido os esperados, o trabalho realizado neste projeto servirá como base para futuras investigações e melhorias no campo da prevenção e combate a incêndios florestais.

Apêndice A

Anexos dos ficheiros fonte

A.1 Especificação do modelo de simulação em NetLogo

```
extensions [csv]
globals [spark-frequency iterations]

patches-own [altitude temperature]
trees-own [
  burning-speed spark-probability
  is-burning is-burnt
  kind ticks-since-spark
]
sparks-own [final-xcor final-ycor]
fires-own [life-in-ticks]

breed [sparks spark]
breed [trees tree]
breed [fires fire]

to create-forest
  clear-all
  ; define csv header
  set iterations [[
    "tick"
    "forest_burnt"
    "forest_burnt_perc"
    "oak_trees_burnt"
    "pine_trees_burnt"
    "no_sparks"
    "mean_temperature"
    "max_temperature"
  ]]
  ; setup patches
  set spark-frequency 150
  ask patches [
    set pcolor 33
```

```
    set altitude calc-altitude pxcor
    set temperature initial-temperature
  ]
  random-seed forest-seed
  ask patches with [random-float 100 < forest-density] [
    plant-tree pxcor pycor
  ]
  reset-ticks
end

to plant-tree [x y]
  let th 1
  set x (random-pcor x th "x")
  set y (random-pcor y th "y")
  let tree-type random-tree-type
  sprout-trees 1 [
    setxy x y
    set color green
    set shape tree-type
    set kind tree-type
    set is-burning false
    set is-burnt false
    ifelse tree-type = "pine-tree" [
      set burning-speed 0.3
      set spark-probability 0.15
    ] [
      set burning-speed 0.1
      set spark-probability 0.05
    ]
  ]
end

to-report random-pcor [pcor th dir]
  let min-pcor min-pxcor
  let max-pcor max-pxcor
  if dir = "y" [
    set min-pcor min-pycor
    set max-pcor max-pycor
  ]
  set pcor pcor + (random-float th) - th
  set pcor median (list min-pcor pcor max-pcor)
  report pcor
end

to start-fire
  let fire-started false
  while [not fire-started] [
    ask n-of 1 patches [ ignite ]
    if count fires > 0 [ set fire-started true ]
  ]
```

```
  save-config
  random-seed new-seed
end

to go
  ; save iteration
  save-iteration
  ; stop running if no tree is burning
  if not any? (turtle-set trees with [is-burning] sparks) [
    save-iterations
    stop
  ]
  ask trees with [is-burning] [
    let fire-altitude [altitude] of patch-here
    ; spread fire if tree is yellow
    if color < yellow [
      ask neighbors with [any? trees-here] [ spread-fire
↪   fire-altitude ]
    ]
    ; create sparks
    if color < brown [
      ifelse random-float 1 < spark-probability
      and ticks-since-spark > spark-frequency [
        ask patch-here [
          sprout-sparks 1 [
            set shape "fire"
            set size 0.7
            set final-xcor (spark-final-cor pxcor "x")
            set final-ycor (spark-final-cor pycor "y")
            facexy final-xcor final-ycor
          ]
        ]
        set ticks-since-spark 0
      ] [ set ticks-since-spark ticks-since-spark + 1 ]
    ]
  ]
  ; move sparks
  ask sparks [
    ifelse (distancexy final-xcor final-ycor) > 0.1 [
      fd 0.1
    ] [
        ask patch-here [ ignite ]
        die
    ]
  ]
  ; 'kill' fire turtles, to not block the view
  ask fires [
    set life-in-ticks life-in-ticks - 1
    if life-in-ticks <= 0 [ die ]
  ]
```

```
  ; burn trees
  fade-embers
  tick
end

to save-config
  if current-run = 0 [
    set-current-directory user-directory
  ]
  let config-path "config.yaml"
  if file-exists? config-path [ stop ]
  file-open config-path
  file-print (word "forest-density: " forest-density)
  file-print (word "inclination: " inclination)
  file-print (word "spread-probability: " spread-probability)
  file-print (word "east-wind-speed: " east-wind-speed)
  file-print (word "north-wind-speed: " north-wind-speed)
  file-print (word "initial-temperature: " initial-temperature)
  file-print (word "no-trees: " count trees)
  file-print (word "no-oak-trees: " count trees with [kind =
↪   "oak-tree"])
  file-print (word "no-pine-trees: " count trees with [kind =
↪   "pine-tree"])
  file-close
end

to save-iteration
  let iteration (list
    ticks
    count-trees-burnt ""
    (count-trees-burnt "" / count trees)
    count-trees-burnt "oak-tree"
    count-trees-burnt "pine-tree"
    count sparks
    mean-temperature
    max-temperature
  )
  set iterations (insert-item (length iterations) iterations
↪   iteration)
end

to save-iterations
  let csv-path (word "run" current-run ".csv")
  csv:to-file csv-path iterations
  ; reset iteration counter
  ifelse current-run = 10 [
    set current-run 0
  ] [
    set current-run current-run + 1
  ]
```

```
end

to-report spark-final-cor [pcor dir]
  let wind-speed east-wind-speed
  let min-pcor min-pxcor
  let max-pcor max-pxcor
  if dir = "y" [
    set wind-speed north-wind-speed
    set min-pcor min-pycor
    set max-pcor max-pycor
  ]
  set wind-speed (round wind-speed / 2)
  set pcor pcor + wind-speed
  set pcor median (list min-pcor pcor max-pcor)
  report pcor
end

to spread-fire [fire-altitude]
  let probability spread-probability
  ; compute the direction from you (the green tree) to the
↪  burning tree
  let direction towards myself
  if direction = 0 [
    set probability probability - north-wind-speed
  ]
  if direction = 90 [
    set probability probability - east-wind-speed
  ]
  if direction = 180 [
    set probability probability + north-wind-speed
  ]
  ; burning tree is west -> the west wind aids the spread
  if direction = 270 [
    set probability probability + east-wind-speed
  ]
  ; increase probability according to temperature
  let mean-temp (mean [temperature] of neighbors)
  set probability probability + (ln (mean-temp + 1)) ^ 2
  ; keep probability within boundaries
  set probability median (list 0 probability 100)
  ; decrease probability if fire is at lower altitude
  let altitude-diff fire-altitude - altitude
    if altitude-diff > 0 [
    set probability probability * (1 + abs (tan inclination / 3))
  ]
  ; ignite if probable
  if random 100 < probability [
    ignite
  ]
end
```

```
to ignite
  sprout-fires 1 [
    set shape "fire"
    set size 2
    let green-trees trees-here with [not (is-burning or
↪   is-burnt)]
    ifelse any? green-trees [
      set life-in-ticks 10
    ] [ die ]
    ask green-trees [
      set is-burning true
    ]
  ]
end

;; achieve fading color effect for the fire as it burns
to fade-embers
  ask trees with [is-burning] [
    set color color - burning-speed  ;; make red darker
    ask patch-here [ set temperature temperature + 1 ]
    ask neighbors [ set temperature temperature + 0.25 ]
    if color < red - 3.5 [ ;; are we almost at black?
      ask patch-here [
        set pcolor 2
      ]
      set is-burning false
      set is-burnt true
      set shape "charred-ground"
    ]
  ]
end

to-report random-tree-type
  let tree-types ["oak-tree" "pine-tree"]
  report one-of tree-types
end

to-report count-trees-burnt [tree-type]
  report count trees with [
    (kind = tree-type or tree-type = "")
    and (is-burning or is-burnt)
  ]
end

to-report calc-altitude [x]
  let b max-pxcor - min-pxcor
  let h b * abs (tan inclination)
  let alt (x - min-pxcor) * tan inclination
  if inclination < 0 [ set alt alt + h ]
```

```
  report alt
end

to-report mean-temperature
  report mean [temperature] of patches
end

to-report max-temperature
  let max-temp 0
  ask patches with-max [temperature] [
    set max-temp temperature
  ]
  report max-temp
end
```

A.2 Especificação do módulo de processamento de dados em Python

A.2.1 Programa principal

```python
from pathlib import Path

from forestfire.scenario import Scenario


def main() -> None:
    runs_folder = Path("runs")
    # Load all scenarios
    scenarios = [Scenario.from_yaml(path) for path in
    ↪  runs_folder.iterdir()]
    # Save the average run and metrics plots for each scenario
    for i, scenario in enumerate(scenarios, start=1):
        scenario_path = runs_folder / f'scenario{i}'
        # get the average run
        avg_run = scenario.average_run()
        # Save the average run
        avg_run.to_csv(scenario_path / "average.csv")
        # Save the metrics plots
        avg_run.plot_metrics(save_path=scenario_path)


if __name__ == "__main__":
    main()
```

A.2.2 Definição da classe Scenario

```python
from dataclasses import dataclass, field
from pathlib import Path
from typing import Self
```

```python
import numpy as np
import yaml

from forestfire.run import Run


@dataclass
class Scenario:
    forest_density: int
    inclination: int
    spread_probability: int
    east_wind_speed: int
    north_wind_speed: int
    initial_temperature: int
    no_trees: int
    no_oak_trees: int
    no_pine_trees: int
    runs: list[Run] = field(default_factory=list)

    @classmethod
    def from_yaml(cls, path: Path) -> Self:
        """
        Load a scenario from a folder containing a config.yaml
↪   file and csv files for each run.
        :param path: Path to the folder.
        :return: The scenario.
        """
        config_path = path / "config.yaml"
        with open(config_path, "r") as f:
            config: dict[str, int] = yaml.safe_load(f)
        runs = [Run.from_csv(path / f"run{i}.csv") for i in
        ↪  range(11)]
        # replace keys with snake case
        config = {k.replace("-", "_"): v for k, v in
        ↪  config.items()}
        return cls(**config, runs=runs)

    def __post_init__(self) -> None:
        self.runs.sort(key=lambda run: run.tick[-1])
        self.extend_runs()

    def extend_runs(self) -> None:
        """
        Extend all runs in the scenario to the same length.
        :return: None
        """
        max_ticks = self.runs[-1].tick[-1]
        for run in self.runs:
            run.extend(max_ticks - run.tick[-1])
```

```
    def average_run(self) -> Run:
        """
        Calculate the average run. This is done by averaging the
↪   values of each run.
        :return: The average run.
        """
        return Run(
            tick=self.runs[0].tick,
            forest_burnt=np.mean([run.forest_burnt for run in
            ↪   self.runs], axis=0),
            forest_burnt_perc=np.mean([run.forest_burnt_perc for
            ↪   run in self.runs], axis=0),
            oak_trees_burnt=np.mean([run.oak_trees_burnt for run
            ↪   in self.runs], axis=0),
            pine_trees_burnt=np.mean([run.pine_trees_burnt for
            ↪   run in self.runs], axis=0),
            no_sparks=np.mean([run.no_sparks for run in
            ↪   self.runs], axis=0),
            mean_temperature=np.mean([run.mean_temperature for
            ↪   run in self.runs], axis=0),
            max_temperature=np.mean([run.max_temperature for run
            ↪   in self.runs], axis=0),
        )
```

A.2.3 Definição da classe Run

```
from dataclasses import dataclass
from pathlib import Path
from typing import Self

import numpy as np
import pandas as pd
from matplotlib import pyplot as plt


@dataclass
class Run:
    tick: np.ndarray[int]
    forest_burnt: np.ndarray[int]
    forest_burnt_perc: np.ndarray[float]
    oak_trees_burnt: np.ndarray[int]
    pine_trees_burnt: np.ndarray[int]
    no_sparks: np.ndarray[int]
    mean_temperature: np.ndarray[float]
    max_temperature: np.ndarray[float]

    @classmethod
    def from_csv(cls, path: Path) -> Self:
        """
```

```
        Load a run from a csv file.
        :param path: Path to the csv file.
        :return: The run.
        """
        df = pd.read_csv(path)
        return cls(**dict(zip(df.columns, df.to_numpy().T)))

    def extend(self, no_ticks: int) -> None:
        """
        Extend the run with the given number of ticks.
        :param no_ticks: Number of ticks to extend the run with.
        :return: None
        """
        no_ticks = int(no_ticks)
        self.tick = np.append(self.tick, np.arange(self.tick[-1]
        ↪  + 1, self.tick[-1] + no_ticks + 1))
        keys = [key for key in self.__dict__.keys() if key !=
        ↪  "tick"]
        for key in keys:
            self.__dict__[key] = np.append(
                self.__dict__[key],
                np.full(no_ticks, self.__dict__[key][-1])
            )

    def plot_metrics(self, save_path: Path | None = None) ->
    ↪  None:
        """
        Plot all metrics. This is a convenience function.
        :param save_path: The path where to save the plots. If
↪  None, the plots are shown.
        :return: None
        """
        self.plot_forest_burnt(save_path)
        self.plot_trees_burnt(save_path)
        self.plot_sparks(save_path)
        self.plot_temperature(save_path)

    def plot_forest_burnt(self, save_path: Path | None = None) ->
    ↪  None:
        """
        Plot the forest burnt over time.
        :param save_path: The path where to save the plot. If
↪  None, the plot is shown.
        :return: None
        """
        plt.plot(self.tick, self.forest_burnt_perc, label="Forest
        ↪  burnt (%)")
        plt.title("Forest fire")
        plt.legend()
        if save_path is not None:
```

```
            plt.savefig(save_path / "forest_fire.png")
            plt.close()
        else:
            plt.show()

    def plot_trees_burnt(self, save_path: Path | None = None) ->
    ↪  None:
        """
        Plot the number of trees burnt over time. This includes
↪   oak and pine trees.
        :param save_path: The path where to save the plot. If
↪   None, the plot is shown.
        :return: None
        """
        plt.plot(self.tick, self.forest_burnt, label="Trees
        ↪  burnt")
        plt.plot(self.tick, self.oak_trees_burnt, label="Oak
        ↪  trees burnt")
        plt.plot(self.tick, self.pine_trees_burnt, label="Pine
        ↪  trees burnt")
        plt.title("Trees burnt")
        plt.legend()
        if save_path is not None:
            plt.savefig(save_path / "trees_burnt.png")
            plt.close()
        else:
            plt.show()

    def plot_temperature(self, save_path: Path | None = None) ->
    ↪  None:
        """
        Plot the temperature over time. This includes the mean
↪   and max temperature.
        :param save_path: The path where to save the plot. If
↪   None, the plot is shown.
        :return: None
        """
        plt.plot(self.tick, self.mean_temperature, label="Mean
        ↪  temperature")
        plt.plot(self.tick, self.max_temperature, label="Max
        ↪  temperature")
        plt.title("Forest temperature")
        plt.legend()
        if save_path is not None:
            plt.savefig(save_path / "temperature.png")
            plt.close()
        else:
            plt.show()

    def plot_sparks(self, save_path: Path | None = None) -> None:
```

```
        """
        Plot the number of sparks over time.
        :param save_path: The path where to save the plot. If
↪   None, the plot is shown.
        :return: None
        """
        plt.plot(self.tick, self.no_sparks, label="Number of
        ↪  sparks")
        plt.title("Number of sparks")
        plt.legend()
        if save_path is not None:
            plt.savefig(save_path / "sparks.png")
            plt.close()
        else:
            plt.show()

    def to_csv(self, path: Path) -> None:
        """
        Save the run to a csv file.
        :param path: Path to the csv file.
        :return: None
        """
        df = pd.DataFrame(self.__dict__)
        df.to_csv(path, index=False)
```

Bibliografia

[1] S. Russel and P. Norvig, "Intelligent Agents," in *Artificial Intelligence: A Modern Approach, 3rd ed.*, 3rd ed. New Jersey: Pearson Education. Inc., 2010, pp. 34–63.

[2] D. Viegas, "Overview of Forest Fire Propagation Research," *Fire Safety Science*, vol. 10, pp. 95–108, 1 2011. [Online]. Available: https://publications.iafss.org/publications/fss/10/95/view/fss_10-95.pdf

[3] G. D. Papadopoulos and F. N. Pavlidou, "A comparative review on wildfire simulators," *IEEE Systems Journal*, vol. 5, no. 2, pp. 233–243, 6 2011.

[4] S. K. Singh and S. Kanga, "Forest fire simulation modeling using remote sensing & GIS," *International Journal of Advanced Research in Computer Science*, vol. 8, pp. 326–332, 2017.

[5] 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, 3 2017.

[6] N. Bogdos and E. S. Manolakos, "A tool for simulation and geo-animation of wildfires with fuel editing and hotspot monitoring capabilities," *Environ. Model. Softw.*, vol. 46, pp. 182–195, 8 2013.

[7] M. A. Finney, I. C. Grenfell, C. W. McHugh, R. C. Seli, D. Trethewey, R. D. Stratton, and S. Brittain, "A Method for Ensemble Wildland Fire Simulation," *Environmental Modeling & Assessment*, vol. 16, no. 2, pp. 153–167, 2011.

[8] L. Ntaimo, X. Hu, and Y. Sun, "DEVS-FIRE: Towards an Integrated Simulation Environment for Surface Wildfire Spread and Containment," *Simulation*, vol. 84, no. 4, pp. 137–155, 2008.

[9] A. A. Ager and M. A. Finney, "Application of wildfire simulation models for risk analysis," *Geophysical Research Abstracts. 11: EGU2009-5489*, 2009. [Online]. Available: https://www.fs.usda.gov/research/treesearch/42278

[10] M. A. Finney, C. W. McHugh, I. C. Grenfell, K. L. Riley, and K. C. Short, "A simulation of probabilistic wildfire risk components for the continental United States," *Stochastic Environmental Research and Risk Assessment*, vol. 25, no. 7, pp. 973–1000, 10 2011. [Online]. Available: https://link.springer.com/article/10.1007/s00477-011-0462-z