UNIVERSIDADE FEDERAL DO ESTADO DO RIO DE JANEIRO CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA ESCOLA DE INFORMÁTICA APLICADA

GRAPH WALKER: Um Simulador Extensível de Passeios Aleatórios

Vivian Bittencourt Poetzscher Yuan

Orientador

Carlos Eduardo Ribeiro de Mello

RIO DE JANEIRO, RJ - BRASIL
Junho de 2021

GRAPH WALKER: Um Simulador Extensível de Passeios Aleatórios

Vivian Bittencourt Poetzscher Yuan

Projeto de Graduação apresentado à Escola de Informática Aplicada da Universidade Federal do Estado do Rio de Janeiro (UNIRIO) para obtenção do título de Bacharel em Sistemas de Informação.

Aprovada por:

_____

Carlos Eduardo Ribeiro de Mello, Ph.D. - UNIRIO

_____

Sydney Cunha de Lucena, D.Sc. - UNIRIO

RIO DE JANEIRO, RJ - BRASIL
Junho de 2021

Catalogação informatizada pela autora

B94 Bittencourt Poetzscher Yuan, Vivian
DESENVOLVIMENTO DA FERRAMENTA GRAPH WALKER / Vivian Bittencourt Poetzscher Yuan. -- Rio de Janeiro, 2021.
37

Orientador: Carlos Eduardo Ribeiro de Mello.
Trabalho de Conclusão de Curso (Graduação) - Universidade Federal do Estado do Rio de Janeiro, Graduação em Sistemas de Informação, 2021.

1. Ferramenta de Simulação. 2. Passeio aleatório. 3. Simulação discreta. 4. Grafos planares. I. Ribeiro de Mello, Carlos Eduardo, orient. II. Título.

À minha amada avó Carmen (*in memoriam*) que,
com seu exemplo, me ensinou a melhor forma de viver.

Agradecimentos

À minha mãe, que sempre acreditou em mim e me ajudou a superar todos os desafios.

Às minhas irmãs do coração, Kayla (*in memoriam*) e Heidi, que me deram mais amor e alegria do que eu achava possível.

Ao meu pai, que me ensinou o significado de dedicação ao trabalho.

Ao meu melhor amigo Bruno, que compartilhou comigo seu conhecimento e paixão por computação.

Ao meu professor e orientador Carlos Eduardo, por me ensinar e me guiar nessa importante etapa da minha formação.

RESUMO

Diante da complexidade e da infinidade de cenários de tomada de decisão que envolvem o mundo real, o uso de modelos computacionais de simulação ganha cada vez mais proeminência, sobretudo por conta do aumento da capacidade de processamento através da Computação em Nuvem e do amadurecimento de ferramentas de análise, visualização e tratamento de dados. Neste contexto, as ferramentas de simulação computacional tornam-se chave na análise de risco, produção de diagnósticos e do estabelecimento de medidas prescritivas em cenários simulados. O presente trabalho propõe uma ferramenta de simulação para modelos baseados em passeio aleatório simples e tendencioso em grafos planares, acessível e extensível, de modo a permitir a sua utilização em diversas aplicações conforme os modelos definitos pelo usuário. A ferramenta Graph Walker visa representar a movimentação de objetos em ambientes modelados através de grafos planares, tais como de trânsito de veículos e pedestres. Neste trabalho, apresentamos um estudo de caso de aplicação da ferramenta proposta no cenário de simulação de movimentação de automóveis em regiões da cidade do Rio de Janeiro, identificando os principais gargalos que impactam o fluxo.

Palavras-chave: Ferramenta de simulação, passeio aleatório, simulação discreta, grafos planares.

ABSTRACT

In view of the complexity and multitude of decision making situations that involve the real world, the understanding and use of computer models have even more importance, especially due to the increase of processing capacity of cloud computing and the advancement of analysis tools, data visualization and management. In this context, computer simulation tools are essential to risk analysis, diagnostic information production and prescriptive measures in simulated scenarios. The present work aims to develop a discrete simulation tool of simple and biased random walk on planar graphs, accessible and extensible, in order to allow its broad application based on the models defined by its user. The Graph Walker tool aims to represent the flow of objects in environments modelled through planar graphs, such as vehicle and pedestrian traffic. In this work, a case study is introduced, where we identify the bottlenecks that affect the flow of vehicles in regions of the city of Rio de Janeiro.

Keywords: Simulation tool, discrete simulation, random walk, planar graphs.

Sumário

Lista de Figuras

Lista de Tabelas

1. Introdução

Diante da complexidade e da infinidade de cenários de tomada de decisão que envolvem o mundo real, o uso de modelos computacionais de simulação ganha cada vez mais proeminência, sobretudo por conta do aumento da capacidade de processamento através da Computação em Nuvem e do amadurecimento de ferramentas de análise, visualização e tratamento de dados. Neste contexto, ferramentas de simulação computacional tornam-se chave na análise de risco, produção de diagnósticos e do estabelecimento de medidas prescritivas em cenários simulados. O modelo definido reproduz o comportamento lógico do sistema real em determinado período de tempo, gerando resultados que permitem melhor compreender os diversos aspectos do cenários em questão. Adicionalmente, permite ainda testar eventuais hipóteses específicas a partir de modificações no modelo.

Um modelo amplamente utilizado no projeto de simulações é o “passeio aleatório", isto é, um caminho formado pela sucessão de passos aleatórios [1]. No início do século XX foram observados alguns fenômenos naturais sob essa perspectiva, tais como o comportamento molecular ou a difusão gasosa [9], desde então diversos trabalhos vem sendo realizados utilizando este modelo.

Graças à evolução dos recursos computacionais e das linguagens de programação de computadores ocorrida nas últimas décadas, simulações cada vez mais sofisticadas vem sendo desenvolvidas. Estas geram uma grande quantidade de dados, que graças ao aumento da capacidade de processamento, podem ser analisados, gerando conclusões acerca dos sistemas reais relativamente rápido e com baixo custo. Isto representa uma grande vantagem em diversas áreas de domínio, onde avaliação de riscos e cenários é peça-chave na tomada de decisão.

1.1 Motivação

Atualmente existem vários programas de simulação disponíveis [5], com aplicações em diversas áreas, tais como sistemas de transporte, produção, financeiros, saúde, construção civil e manufatura, dentre outros. Contudo, não identificamos uma ferramenta extensível, flexível e de código-aberto que possa ser utilizada para modelos de simulação de passeio aleatório em grafos planares.

Os modelos de simulação executados na ferramenta proposta neste trabalho fornecerão informações sobre sistemas que são impactados pelo movimento, tais como o fluxo de veículos e pedestres, permitindo identificar gargalos e a testar rotas alternativas, contribuindo para a melhor gestão de sistemas de trânsito e configuração urbana, com benefícios diretos para a comunidade.

1.2 Objetivo

O presente trabalho objetivou desenvolver um programa básico de simulação de eventos discretos em grafo planar, acessível e extensível de modo a permitir a sua utilização em diversas aplicações conforme o modelo definido pelo usuário.

Trata-se de uma ferramenta de simulação de modelos de passeio aleatório (random walk) em grafos planares, representando a movimentação de objetos em diferentes topologias que podem representar abstrações variadas, tais como bairros, cidades, países e etc.

Ao executar simulações de sistemas que são afetados pelo movimento, tais como os de trânsito de veículos e pedestres, o modelo gera dados que permitem avaliar a concentração de fluxos e rotas de maior movimento, como por exemplo os indesejáveis gargalos na malha rodoviária da cidade ou o melhor ponto para a instalação de um quiosque de vendas em um shopping center. Outra aplicação relevante é no estabelecimento de rotas de evacuação a serem seguidas na ocorrência de desastres naturais, incêndios, atentados, etc.

Desta forma, o objetivo a ser alcançado pelo uso da ferramenta de simulação Graph Walker é contribuir positivamente para a análise e gestão de diferentes cenários que envolvam objetos móveis e seus movimentos em diferentes esquemas topológicos.

1.3 Escopo

A ferramenta Graph Walker implementa um modelo de simulação de eventos discretos que possui em seu núcleo um algoritmo de passeios aleatórios para reproduzir um sistema dinâmico, em que as entidades mudam a partir da ocorrência instantânea de eventos e tem a sua evolução representada ao longo do tempo.

Além destas funcionalidades, a ferramenta disponibiliza variáveis editáveis para que o usuário possa fazer simples alterações no comportamento e na quantidade de objetos. Dentre elas, a inserção de 'pontos de atração' no grafo para os quais os objetos tenderão a ir (passeio aleatório tendencioso).

Adicionalmente, o usuário disporá de quatro formas de visualização do programa em tempo discreto, tanto da movimentação dos objetos quanto das arestas relativamente mais utilizadas.

A ferramenta de simulação Graph Walker não se propõe a fazer análises probabilísticas do resultado, e sim apresentar a simulação dos objetos se movimentando em tempo discreto em um grafo, de modo que o usuário possa visualizar os deslocamentos ocorridos e obter dados relevantes para a análise do sistema.

Outros modelos de simulação de eventos discretos, tais como fila de eventos ou autômatos celulares, estão fora do escopo da proposta da ferramenta.

1.4 Organização do texto

O presente trabalho está estruturado em quatro capítulos, além desta introdução. O capítulo II de Fundamentação Teórica apresenta referencial teórico utilizado no desenvolvimento da ferramenta, descrevendo conceitos básicos de modelos de simulação em tempo discreto, processos estocásticos, processos de Markov e passeio aleatório. O capítulo III desenvolve a proposta da ferramenta Graph Walker e como são implementados os conceitos mencionados no capítulo anterior. Ainda nesse capítulo, encontram-se detalhes do funcionamento da ferramenta, a arquitetura proposta e como o usuário poderá interagir com ela. Em sequência, o capítulo IV de Resultados Simulados apresenta todas as variáves de entrada que o usuário tem como opção. Também são descritos diversos experimentos com a ferramenta executando diferentes exemplos, incluindo topologias similares aos casos reais de malha rodoviária. O capítulo V conclui o trabalho apresentando as considerações finais e sugere possibilidades de aprofundamento posterior.

2. Fundamentação Teórica

Neste capítulo são apresentados os componentes fundamentais para o desenvolvimento da ferramenta e sobre os conceitos de simulação em tempo discreto, processos estocásticos, processos de Markov e passeios aleatórios.

2.1 Modelos de Simulação

Simulação pode ser definida como uma "imitação"de um processo do mundo real [5]. Modelos de simulação permitem estudar fenômenos, e assim fazer previsões e associações à realidade. Com o auxílio da computação, é estabelecido um modelo que interage com variáveis e formulações matemáticas para simular determinado comportamento, que permite coletar dados para análise e obter suporte à tomada de decisão.

O ambiente controlado de modelos de simulação geralmente traz vantagens que compensam o custo para o seu desenvolvimento, pois muitos sistemas do mundo real são demasiado complexos para serem analisados sem o auxílio de uma ferramenta de simulação devido à complexidade das relações que os compõem. Portanto, para que seja possível estudar tais sistemas, são utilizados modelos de simulação para estimar as características de interesse sobre o fenômeno no mundo real.

São diversas as aplicações das simulações, como, por exemplo, no planejamento e gerenciamento de trânsito, aeroportos, metrôs e portos. Também, na análise de sistemas financeiros, no planejamento de sistemas hospitalares, de correio, fabricação, sistemas de emergência e evacuação, entre outros. [5]

Os modelos de simulação podem ser classificados em [6]:

- **Estático | Dinâmico** - um modelo estático representa um momento específico, onde o resultado independe do tempo. Por exemplo, a análise de uma venda diária, que

não é influenciada pela venda do dia anterior. Já o modelo dinâmico depende do tempo e representa a evolução do sistema, como por exemplo a simulação de uma linha de montagem de veículos.

- **Determinístico | Estocástico** - um modelo determinístico não possui componentes probabilísticos (isto é, componentes aleatórios). Um exemplo determinístico é um sistema complexo de equações diferenciais representando uma reação química, em que o resultado é sempre o mesmo, uma vez que as variáveis de entrada e as relações foram especificadas. Por outro lado, um modelo estocástico depende de variáveis aleatórias, ou seja, dependem de alguma distribuição de probabilidade [14]. Este é o caso da maioria dos sistemas. Por produzir um resultado aleatório, modelos estocásticos de simulação devem ser tratados como uma estimativa do sistema real [13]. Como exemplos, podemos citar simulações de redes de comunicação e serviços para clientes.

- **Contínuo | Discreto** - se o modelo é contínuo, as variáveis se alteram de forma gradativa e normalmente são descritas por equações diferenciais. Podemos citar o crescimento de uma planta ou variação no nível da água de um reservatório. Em oposição, modelos de simulação de tempo ou eventos discretos tratam momentos instantâneos no tempo nos quais eventos ocorrem. É possível dizer que o sistema muda a partir de pontos finitos no tempo.

É importante destacar que uma simulação serve apenas de apoio à tomada de decisão e, portanto, não é uma previsão do futuro, otimização ou modelo matemático [7].

A ferramenta Graph Walker, proposta nesta dissertação, implementa um modelo dinâmico, estocástico e discreto.

2.2 Variáveis Aleatórias e Processos Estocásticos

Um experimento aleatório é um processo em que o resultado não se pode saber com certeza [3]. O conjunto de todos os possíveis resultados de um experimento é chamado de *espaço amostral* e é denotado por S. Os resultados possíveis são chamados de *pontos amostrais*. Assim, se um experimento consiste em jogar uma moeda, então S = {Cara, Coroa}.

Uma *variável aleatória*, portanto, é uma função que mapeia um número real (qualquer n maior que -∞ e menor que +∞) para cada evento no espaço amostral S [6]. Dessa

forma, uma variável aleatória é definida como *discreta* se esta mapeia eventos em um espaço amostral de eventos discretos [13], ou seja, uma quantidade enumerável de resultados no espaço amostral, que será o caso que abordaremos nesta dissertação.

Um *processo estocástico* pode ser tratado como uma extensão da definição de variável aleatória. Dessa forma, processo estocástico é uma coleção de variáveis aleatórias ordenadas no tempo, em que todas estão definidas no mesmo espaço amostral [12]. Em outras palavras, é a descrição de um fenômeno aleatório que varia com o tempo. Portanto, ao invés de pensar na variável aleatória X que mapeia um evento w $\in$ S, onde S é o espaço amostral, para um número X(w), passamos a pensar como que a variável aleatória mapeia o evento para diferentes números em diferentes momentos. Assim, ao invés de lidarmos com X(w), lidamos com X(t, w), onde t $\in$ T e T é chamado de conjunto parâmetro do processo (normalmente um conjunto de momentos) [3].

Podemos citar diversos exemplos de fenômenos que podem ser representados por processos estocásticos, tais como, o comportamento de uma partícula de gás, as variações no preço de ações, a evolução do número de desempregados em um país, a falha de um equipamento. Da mesma forma que a variável aleatória, o processo estocástico pode ser classificado como discreto ou contínuo, em relação ao estado de valores que o processo pode assumir ou em relação ao tempo. Estado discreto significa que o espaço amostral que as variáveis aleatórias podem assumir é enumerável. Já o tempo discreto implica que o tempo no processo é finito ou igualmente enumerável [8].

2.3 Processo de Markov

Os processos de Markov são utilizados amplamente em diversas áreas, desde a engenharia à ciência social [3]. Sistemas que têm memória limitada do passado são ideais para o uso desse processo. Por exemplo, considere uma sequência de jogos onde o jogador ganha 1 ponto para cada vitória e perde 1 ponto para cada derrota. A quantidade de pontos que o jogador vai acumular depois de n + 1 jogos é determinada pela quantidade de pontos que ele tem depois de n jogos. Qualquer outra informação é irrelevante para essa predição. O mesmo se aplica no exemplo da análise de crescimento de uma população, que depende apenas da população atual e possivelmente das últimas gerações.

Dado o estado atual do processo, o estado futuro é independente do passado [8]. Esta é a *propriedade Markov*. Em processos de Markov de segunda ordem, o estado futuro depende do estado atual e do último estado, e assim por diante para processos de maior ordem. Igualmente à definição de processo estocástico, o processo Markov se classifica

em discreto e contínuo de acordo com seu conjunto de estados ou de acordo com o tempo. Um processo Markov é representado por uma cadeia de Markov[11].

A seguir falaremos de passeio aleatório, em que sua definição se relaciona com as mencionadas anteriormente.

2.4 Passeio Aleatório

Passeio aleatório ou *Random walk*, é processo estocástico originalmente proposto por Pearson [2] em 1905. Este descreve um problema onde um objeto se move em uma sucessão de passos discretos aleatórios em um espaço matemático. Na figura 2.2 vemos um exemplo simples de passeio aleatório, que é a movimentação do objeto em uma linha. As duas possibilidades de passos nesse caso são apenas +1 e -1, ambas com igual probabilidade de escolha e que possui a soma de seus passos tendendo a 0.

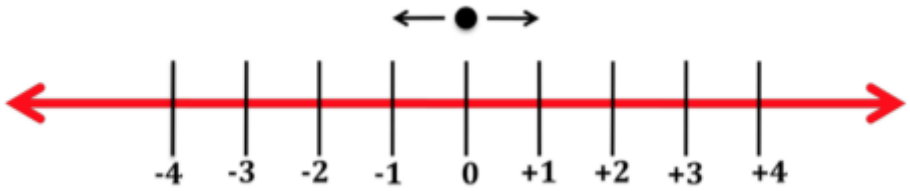


Figura 2.1: Passeio Aleatório no espaço matemático unidimencional

Esse processo foi amplamente estudado em inúmeras dimensões, devido à sua aplicabilidade em diversos fenômenos que aparentam ter comportamento desordenado [3]. Podemos citar, como exemplo, uma molécula se movimentando em um líquido ou gás, cenário no qual foi desenvolvida a formulação matemática conhecida como Movimento Browniano, uma generalização do processo de Markov [15]. Outros exemplos são a disseminação de vírus em uma rede de computadores ou a situação financeira de um apostador[16]. Assim, o passeio aleatório possui a capacidade de modelar matematicamente diferentes acontecimentos nas áreas de física, química, ciência da computação, engenharia, entre outros[1].

Analogamente, o passeio aleatório possui diversas variações, como o *biased random walk*, ou passeio aleatório tendencioso [10]. Sua abordagem é que cada caminho possível de escolha tem probabilidades distintas dado a um critério específico. A ferramenta desenvolvida neste trabalho inclui o passeio aleatório simples e o passeio aleatório tendencioso em grafos planares. Mais informações sobre tipos de passeios aleatórios podem ser encontradas em [2].

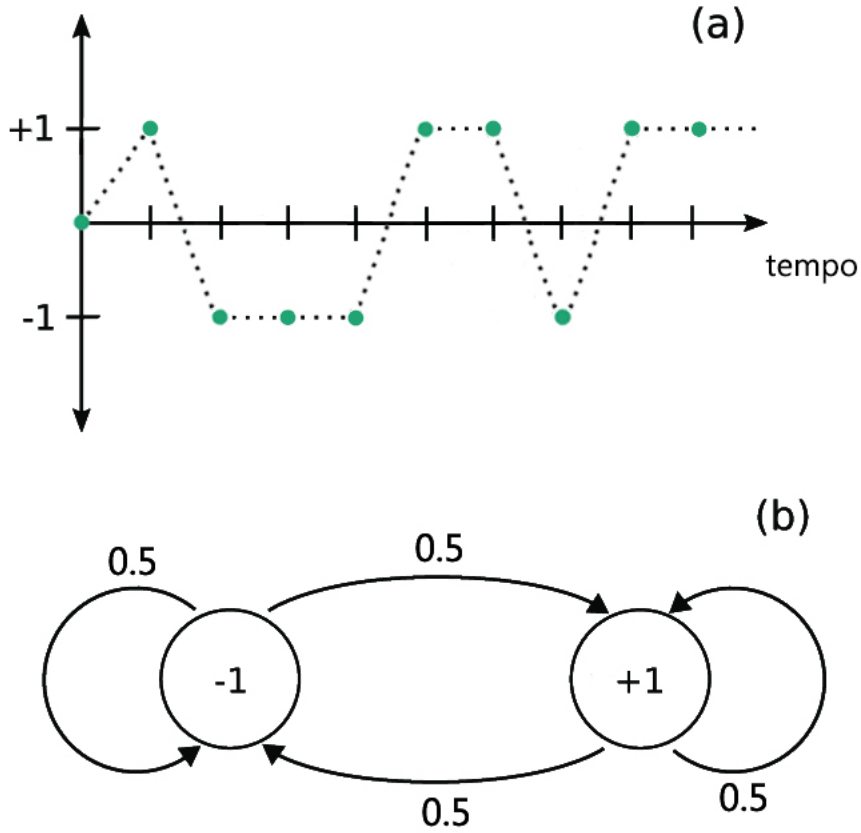


Figura 2.2: (a) Exemplo de um algoritmo de passeio aleatório simples onde os pontos verdes representam eventos, e podem apenas tomar os valores +1 e -1. (b) Mesmo exemplo representado na cadeia de Markov.

A simulação de passeio aleatório em grafos planares de diferentes topologias é interessante para o estudo de diversos cenários. Por exemplo, identificação de pontos de maior tráfego em uma determinada topologia de trânsito, ou a previsão e gerenciamento de evacuação de um local público. Logo, neste trabalho daremos exemplos da execução e aplicabilidade da ferramenta proposta em diferentes topologias de trânsito. Uma simulação permite a verificação de fluxo tanto aleatório simples, como aleatório tendencioso de carros em grafos que representam topologias reais de mapas urbanos. Neste caso, isto é aplicável para projetos de melhoria de fluxo de tráfego e tópicos relacionados.

3. Proposta da Ferramenta

Neste capítulo apresentamos como a ferramenta de simulação de passeios aleatórios em grafos planares, que chamamos de Graph Walker, está desenvolvida e como o usuário poderá interagir com ela de forma a chegar no resultado desejado.

Na primeira parte, uma descrição do funcionamento da ferramenta é realizada considerando possíveis parâmetros que o usuário poderá definir ao utilizá-la.

3.1 A Ferramenta

A ferramenta Graph Walker tem como proposta simular passeios aleatórios em grafos planares, representando objetos e sua movimentação em tempo discreto. A ferramenta permite inserir diferentes dados e variáveis para adequar a saída conforme a necessidade da simulação.

Graph Walker foi desenvolvido na linguagem de programação Python[1] , utilizando Jupyter Notebook[2]. A primeira célula do Notebook, ou a primeira parte do código, contém as variáveis editáveis pelo usuário. Falaremos mais sobre Jupyter Notebook na seção 4.1. Em um grafo, será possível inserir uma quantidade de objetos e movimentá-los tanto em passeio aleatório simples, como de forma customizada (passeio aleatório tendencioso). Uma das características dessa ferramenta é permitir que o usuário determine o comportamento dos objetos para visualizá-los em tempo de simulação enquanto se movimentam.

[1]Python: https://www.python.org/
[2]Jupyter Notebook: https://jupyter.org/

3.2 Arquitetura

A arquitetura da ferramenta Graph Walker está organizada em por uma função principal *main* que chamará um conjunto de funções. Estas gerenciam todo o comportamento da ferramenta, gerando o grafo a partir do arquivo de entrada, atualizando variáveis de controle, fazendo o sorteio aleatório para qual caminho cada objeto deve seguir e, finalmente, exibindo o gráfico. Toda a arquitetura das funções implementadas e seus relacionamentos podem ser vistos na Figura 3.1.

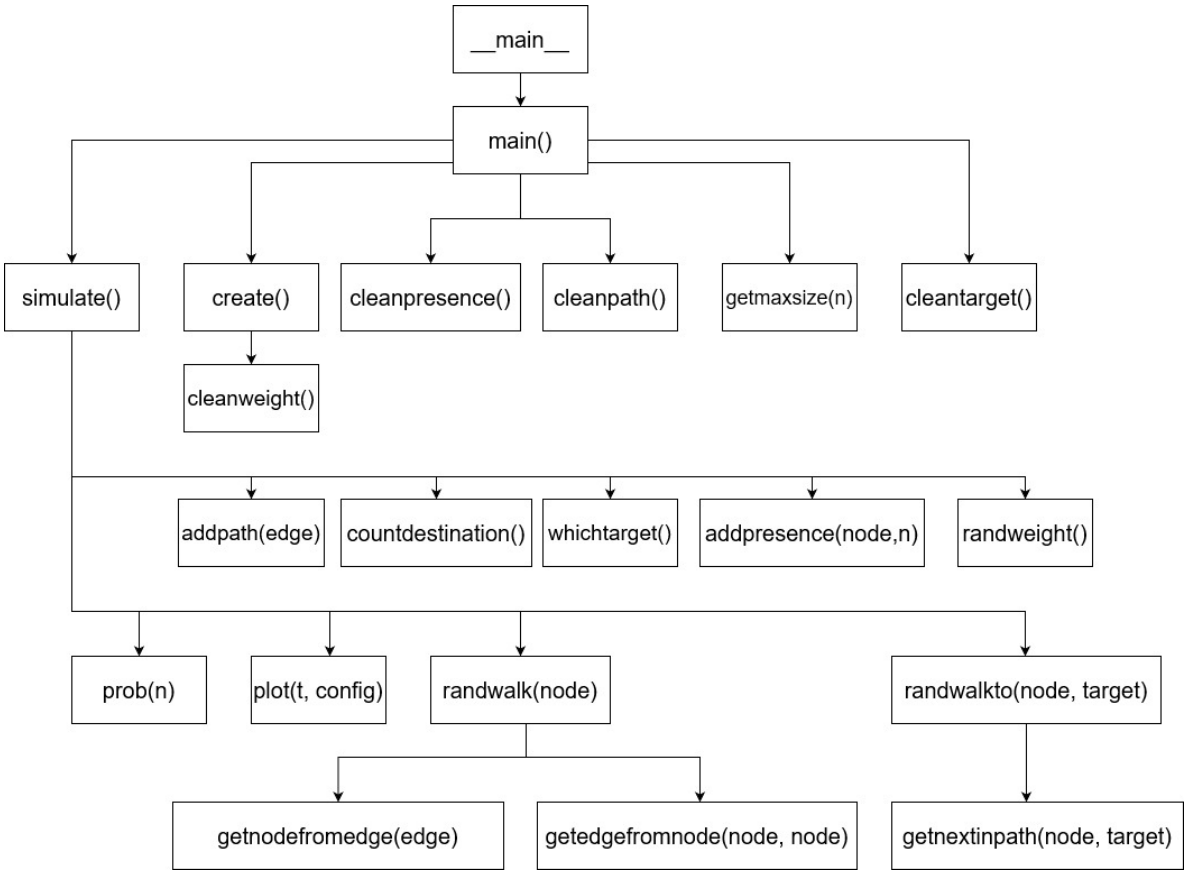


Figura 3.1: Arquitetura da Ferramenta GraphWalker

Na sequência, abordaremos o papel de cada uma das funções utilizadas pela função *main*:

- A função *create* gera o grafo a partir das opções do usuário (com ou sem arquivo de entrada).
- A função *getmaxsize* fornece o tamanho do grafo para melhor desenhá-lo.
- As funções *cleanweight*, *cleanpresence*, *cleanpath* e *cleantarget* são funções para iniciar a variáveis de controle dos objetos.

- A função *simulate* gerencia os atributos associados a cada objeto e as funções para movimentação destes.
- A função *whichtarget* distribui aleatóriamente um sumidouro (dentre as opções fornecidas pelo usuário) para cada objeto.
- A função *addpath* atualiza a variável de controle quando um objeto passar por uma aresta, e será a base para visualização do uso de cada caminho.
- A função *countdestination* atualiza o parâmetro de controle de quanto objetos chegaram ao seu destino.
- A função *addpresence* atualiza quantos objetos estão presentes em cada vértice.
- A função *randweight* gera pesos aleatórios para as arestas do grafo.
- A função *prob* define a probabilidade do objeto permanecer no mesmo vértice, considerando o número de objetos no mesmo.
- A função *plot* é encarregada de atualizar o grafo a cada momento para visualização do usuário.
- A função *randwalk* escolhe o caminho para cada objeto aleatóriamente (passeio aleatório simples).
- A função *randwalkto* é similar à *randwalk*, porém cada objeto terá uma chance de se direcionar para seu sumidouro (passeio aleatório tendencioso).
- A função *getnextinpath* define qual aresta pertence ao caminho mais curto em direção ao sumidouro do objeto.
- As funções *getnodefromedge* e *getedgefromnode* retornam, respectivamente, os vértices compondo uma aresta específica e a aresta composta por dois vértices, caso exista.

3.3 Customização

Há diversas formas de customizar o comportamento da simulção na ferramenta. A seguir discutiremos quais são essas opções e como elas funcionam.

A primeira opção é escolher entre grafos predefinidos (uma treliça simples, de tamanho definido pelo usuário, e uma treliça ligada a outra por meio de uma única ponte) ou

edge_id	x1	y1	x2	y2	weight
edge 1	0	12	1	12	1
edge 2	1	12	2	12	2
edge 3	2	12	3	12	3
edge 4	4	12	4	12	3
edge 5	5	12	7	12	5

Tabela 3.1: Exemplo de Tabela de Importação

importar um arquivo .csv representando um grafo planar específico. Este arquivo deverá incluir as colunas **edge_id**, o identificador da aresta; **x1** e **y1**, sendo a posição do primeiro nó do vértice; **x2** e **y2**, a posição do segundo vértice; e **weight**, o peso dessa aresta. O peso influencia diretamente a escolha de caminho do objeto quando utilizado o passeio aleatório tendencioso. Para que o objeto possa se direcionar ao seu destino, o caminho mais curto deve ser calculado, que é composto pelas arestas com menor peso entre ele e o objeto.

A tabela 3.1 mostra um exemplo de como um arquivo a ser importado deve ser organizado.

Em seguida, o usuário escolhe entre quatro diferentes visualizações do grafo. A primeira possibilidade é ver apenas o grafo simples, sem objetos. Na segunda, os objetos podem ser visualizados se movendo a cada instante da simulação representados em vermelho. A intensidade da cor no vértice aumenta conforme a quantidade de objetos naquela posição. A terceira opção é similar à segunda, apenas com o adendo de números para informar a quantidade de objetos em cada vértice. Finalmente, a última visualização permite apresentar as arestas dos grafos em laranja, com a intensidade da cor representando o fluxo total de objetos relativos à quantidade total de passos.

O máximo de tempo discreto que a ferramenta executa a simulação é outro parâmetro configurável e o tempo atual está disposto na visualização.

Uma das principais funcionalidades da ferramenta Graph Walker é a possível definição de um ou mais destinos para onde os objetos tenderão a se dirigir. Chamaremos estes destinos de "sumidouros". Cada objeto, ao analisar as arestas disponíveis para seu próximo passo, terá uma chance, definida pelo usuário, de escolher a aresta que leva ao menor caminho para seu respectivo destino. Ao chegar no destino, o objeto irá desaparecer do grafo, tendo concluído sua rota.

O ponto inicial dos objetos pode ser definido manualmente e/ou gerados de forma aleatória. Caso sejam definidos manualmente, e se houver sumidouros, seus respectivos destinos também deve ser definidos pelo usuário.

Adicionalmente, o usuário pode escolher pesos nas arestas do grafo para influenciar ou não a rota dos objetos. Se escolher usar os pesos, é possível gerá-los aleatoriamente ou utilizar os pesos pré-definidos através dp arquivo de importação.

Finalmente, há outro adicional de ajuste de comportamento, que é a probabilidade do objeto permanecer no mesmo estado. O algoritmo que define essa probabilidade pode ser configurado facilmente, porém utiliza-se de forma padrão uma função sigmoide simples. A função sigmoide utilizada está descrita na equação 3.1, onde x é o número de objetos no vértice e o a representa o quanto a sigmoide deve ser deslocada.:

$$S(x) = \frac{1}{1 + e^{-x+a}} \tag{3.1}$$

Portanto, a função que determina esse comportamento deriva a probabilidade de um objeto parar de forma proporcional à quantidade de objetos contida no mesmo vértice.

Nesta seção vimos a proposta da ferramenta Graph Walker e como será possível customizar a entrada para atingir o resultado desejado. Na próxima seção serão abordados os resultados da ferramenta.

4. Resultados Simulados

Neste capítulo serão apresentados os resultados obtidos pela ferramenta e como o comportamento dos objetos pode ser personalizado através dos parâmetros de entrada, além da sua aplicabilidade em diferentes topografias de grafos semelhantes a casos urbanos reais.

4.1 Parâmetros de Entrada

Nas seguintes tabelas 4.1 e 4.2 descreveremos todas as possíveis entradas do usuário na ferramenta, seus possíveis valores e suas respectivas descrições.

Parâmetro	**Valores**	**Descrição**
criarmalha	*Boolean*	Gerar grafo simples ou importar grafo customizado.
largurax	Z*+	Tamanho do grafo gerado no eixo x.
alturay	Z*+	Tamanho do grafo gerado no eixo y.
ponte	*Boolean*	Gerar um grafo malha simples ou com topologia de ponte.
docimport	r'C:\..\arquivo.csv'	Caminho do arquivo csv no computador.
delimiter	',' ou ';'	Delimitador para leitura do arquivo csv.
plot	0, 1, 2, 3	Opções de visualização do grafo ilustrado na figura 4.2.
plottime	Q+	Intervalo de tempo de atualização do grafo para visualização.
tmax	Z*+	Tempo discreto máximo que a ferramenta rodará.
sumidouro	*Boolean*	Utilizar ou não destino(s) para qual(is) os objetos tenderão a ir.
target	*Array* (ex.: [('1-0'), ...]	Vértice(s) que representará(ão) o(s) sumidouro(s).
targetchance	Porcentagem	Chance do objeto escolher a aresta que leva ao caminho mais curto para o sumidouro
nobj	Z+	Número de objetos a serem inseridos no mapa aleatoriamente
usestart	*Boolean*	Utilizar ou não objetos inseridos manualmente.
start	*Array* (ex.: [('1-0'), ...]	Posição do(s) objeto(s) inserido(s) manualmente.
targetstart	*Array* (ex.: [(0), ...]	Referência a variável target, definindo qual sumidouro cada objeto inserido manualmente focará.
usesweight	*Boolean*	Utilizar ou não o peso das arestas, e.g. tem mais chance de escolher o caminho com menor peso baseado no targetchance
usestop	*Boolean*	Utilizar ou não o comportamento de parar de acordo com uma probabilidade. Pré-definido como a função de Poisson de densidade acumulada de forma a ser inversamente proporcional a quantidade de objetos no mesmo vértice.
randweight	*Boolean*	Gerar ou não pesos aleatórios para cada uma das arestas. (Irá sobrescrever se houver pesos no arquivo importado)

Tabela 4.1: Parâmetros de entrada na ferramenta Graph Walker

Variável	Possíveis valores	Descrição
maxprob	Porcentagem	Definir o valor máximo de probabilidade que o objeto pare no lugar
minprob	Porcentagem	Definir o valor mínimo de probabilidade que o objeto pare no lugar
mid	Z+	O quanto a sigmoide será deslocada (Representa a quantidade de objetos no vértice onde haverá 5% de chance de parar no lugar)
valorfuncao	Chamada da função desejada	Definir qual a função que será utilizada (recebe o número de objetos no vértice e mid)

Tabela 4.2: Parâmetros de entrada associados à movimentação dos objetos

4.2 Plataforma

Para o desenvolvimento da ferramenta Graph Walker utilizamos a plataforma web de código-fonte aberto Jupyter Notebook [1]. Este fornece um ambiente de desenvolvimento especializado para construção e execução de códigos Python.

O Jupyter Notebook funciona como um servidor web hospedado no ambiente local onde está sendo executado o código, podendo ser acessado, por padrão, pela URL http://localhost:8888/.

Um dos motivos pela escolha deste ambiente de desenvolvimento e compilação é a facilidade e disponibilidade de como os arquivos podem ser transportados entre diferentes dispositivos. Dessa forma, é possível obter o arquivo de extensão .ipynb, que se trata de um arquivo estruturado em formato JSON que possui, além de todo o código python da ferramenta sendo desenvolvida, a estruturação para execução de pequenos trechos do código, chamadas de "células", como visto na figura 4.1. Essa funcionalidade auxilia no desenvolvimento e em testes de partes isoladas. Caso queiramos apenas o código Python para execução em algum *host* que não possua o Jupyter Notebook instalado, também é possível exportá-lo isoladamente.

Adicionalmente, essa plataforma permite que cada célula possua um tipo, como *markdown* e de código. A célula de *markdown* permite escrever texto ou inserir imagens no meio do programa. Essa divisão em células facilita a compreensão do programa como um todo. Assim, nas primeiras células de código podemos encontrar todas as variáveis descritas anteriormente, para que o usuário tenha fácil acesso a elas.

[1] Jupyter Notebook: https://jupyter.org/

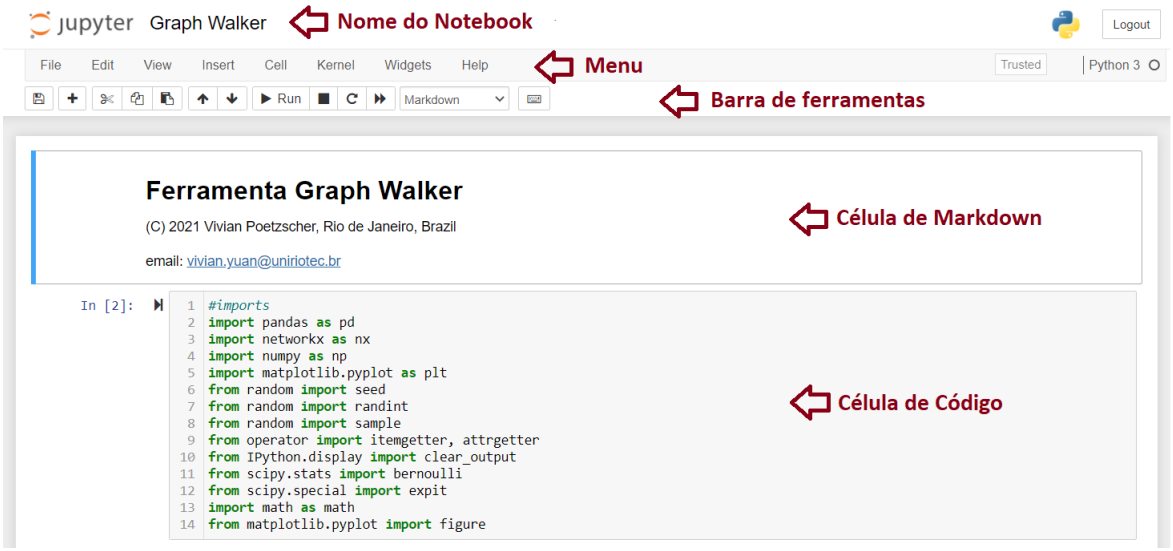


Figura 4.1: Interface do Jupyter Notebook

4.3 Opções de visualização

Nessa seção veremos como as opções de visualização da ferramenta mostram o grafo para o usuário e os objetos se movimentando em tempo discreto. Na figura 4.2 é possível ver o *output* de diferentes valores da variável **plot**. O grafo escolhido foi uma malha simples gerada automaticamente pela ferramenta.

Na opção onde há variável **plot** = 0, vemos apenas o grafo simples sem objetos, em que as arestas estão em cinza e os vértices em verde. Possui o simples objetivo de verificar a topologia do grafo de entrada.

Em seguida, na opção **plot** = 1, os objetos se encontram no grafo em vermelho. Em cima do grafo, o valor de T representa o tempo discreto que o grafo está sendo mostrado, e é atualizado a cada iteração de movimento dos objetos.

Já quando o valor da variável **plot** = 2, podemos ver a exata quantidade de objetos em cada vértice graças a um pequeno número ao lado. Esta opção é bastante similar à opção 1.

Por fim, a opção **plot** = 3 permite visualizar o quanto cada aresta foi utilizada pelos objetos. A intensidade da cor representa o a quantidade relativa de objetos que passou por aquele caminho.

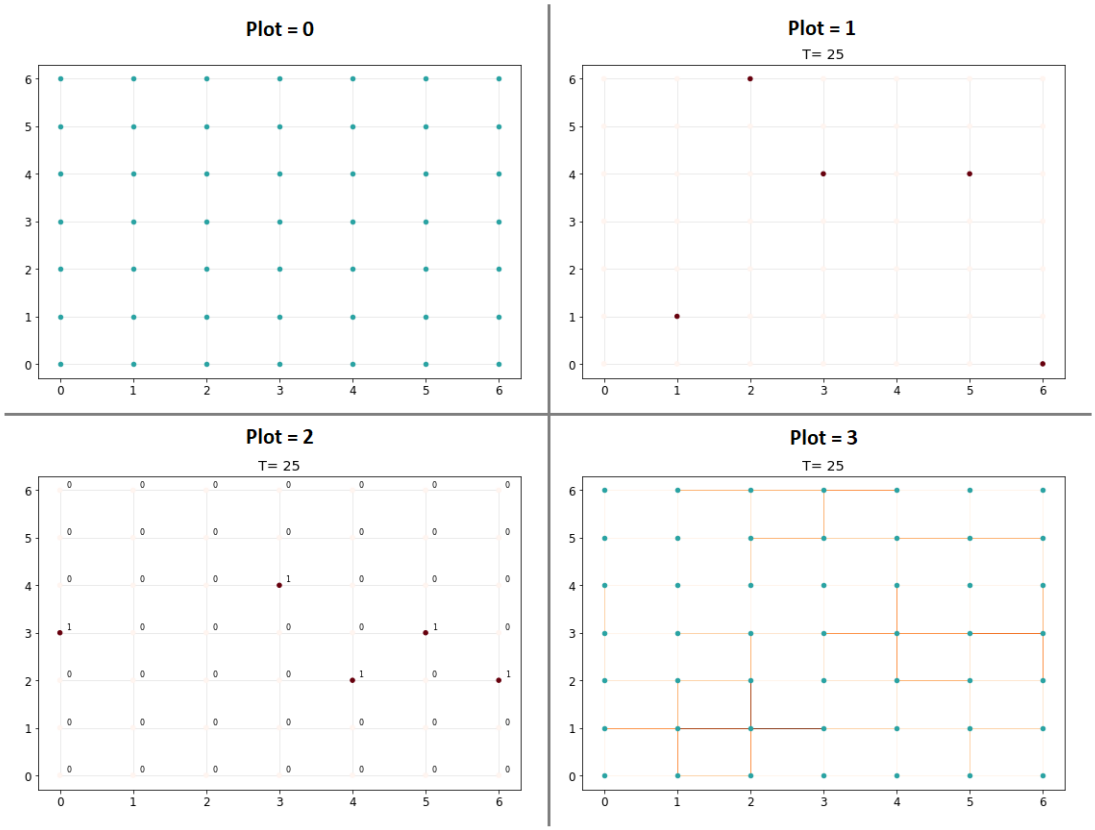


Figura 4.2: Quatro diferentes tipos de visualização

4.4 Exemplos Topológicos

A seguir veremos três exemplos topológicos baseados na malha rodoviária. Estes exemplos ilustram o funcionamento da ferramenta e como o passeio aleatório tendencioso é capaz de aproximar comportamentos reais do fluxo de veículos. Os grafos apresentados foram inspirados em pontos-chave da cidade do Rio de Janeiro e criados manualmente para a simulação. Para os exemplos da visualização 1 utilizaremos 10 objetos inseridos aleatoriamente, sendo possível parar no mesmo local. Para os de visualização 3, teremos 300 objetos, com igual possibilidade de parar à visualização 1. Em ambas situações os objetos estarão se movimentando pelo grafo com 70% de chance de escolher o menor caminho possível (considerando os pesos das arestas) em direção ao seu respectivo destino (sumidouro).

Na figura 4.3 é possível ver quatro momentos consecutivos da ferramenta executando, sendo utilizada a visualização 1. Este exemplo foi baseado na topologia de ponte, onde a maioria dos objetos tem como objetivo chegar ao outro lado.

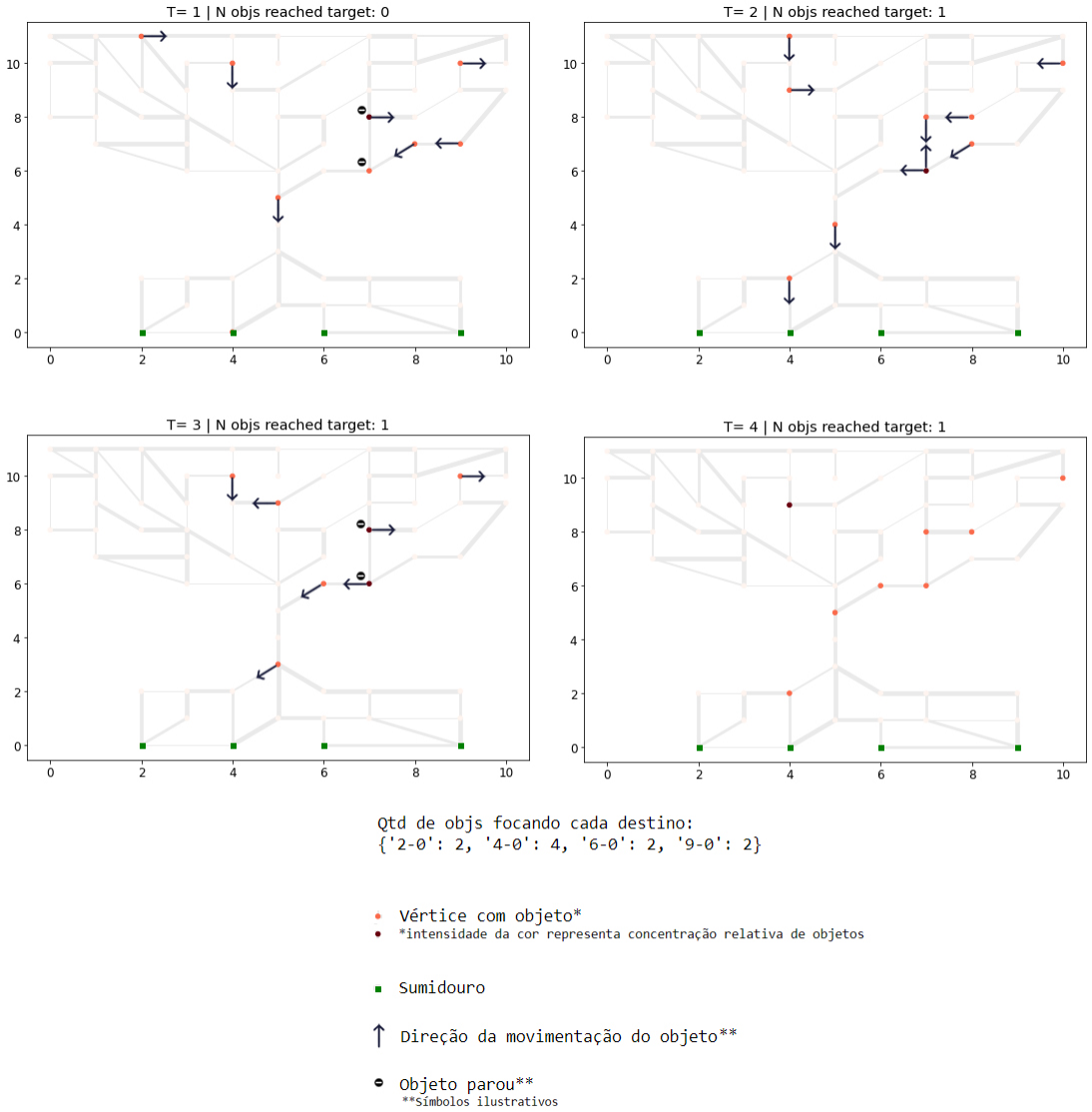
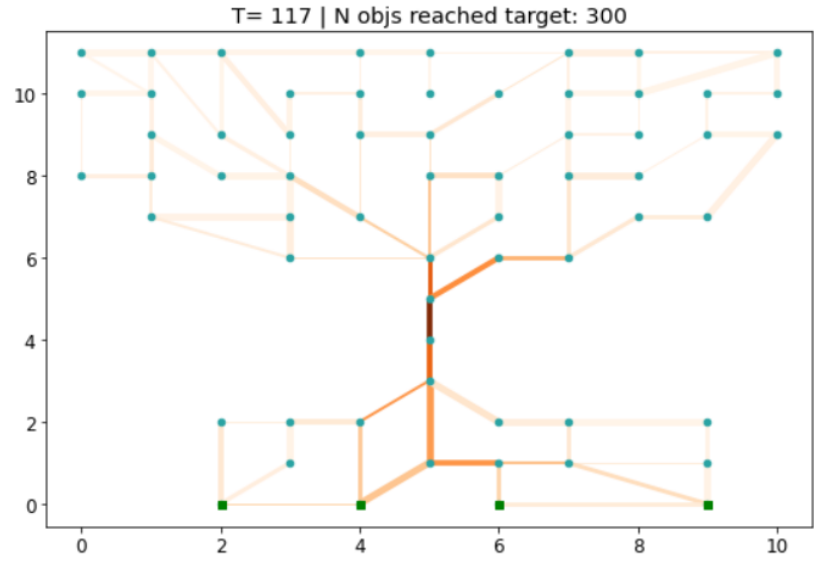


Figura 4.3: Exemplo Topolológico de Ponte

Em seguida a figura 4.4 apresenta o mesmo exemplo topológico baseado em ponte, porém na visualização 3. Aqui vemos claramente o alto fluxo na região da ponte e área adjacente, podendo ser comparado à um "funil". Um caso real que podemos citar semelhante a esta topologia é a ponte Rio-Niterói no estado do Rio de Janeiro. Dessa forma, podemos dizer que o resultado da simulação se comportou como esperado.

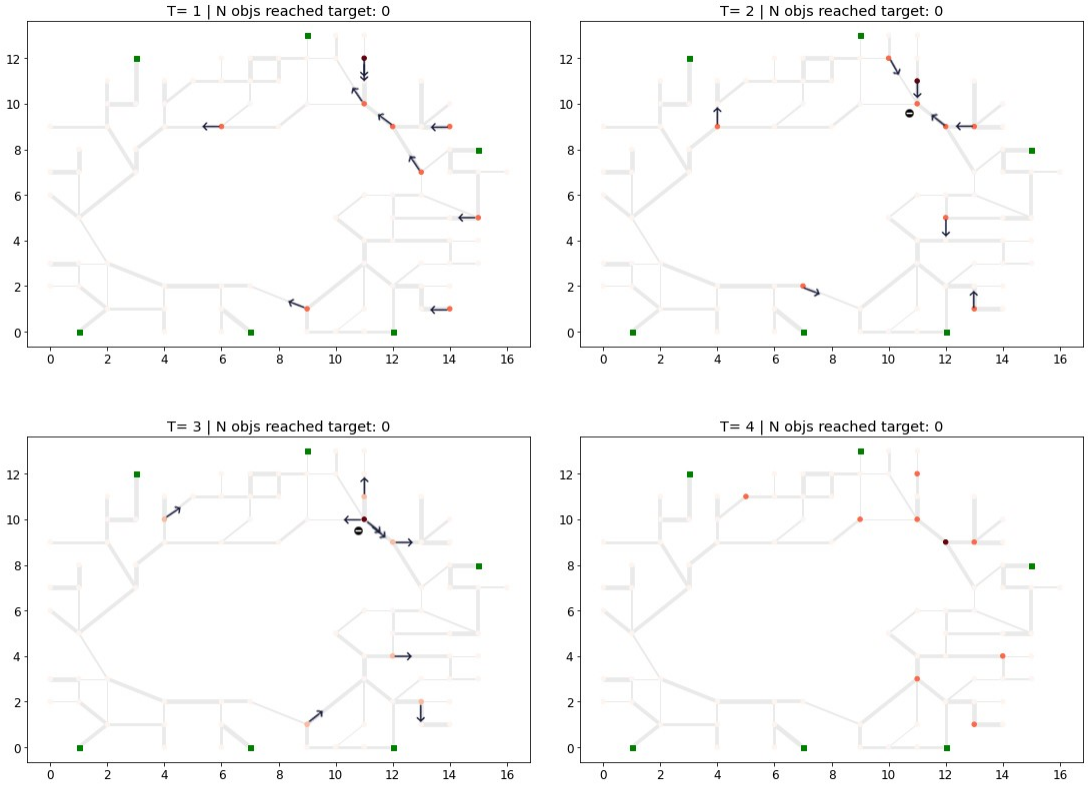



```
Finalizado em 117 passos.

Qtd de objs focando cada destino:
{'2-0': 77, '4-0': 77, '6-0': 70, '9-0': 76}
```

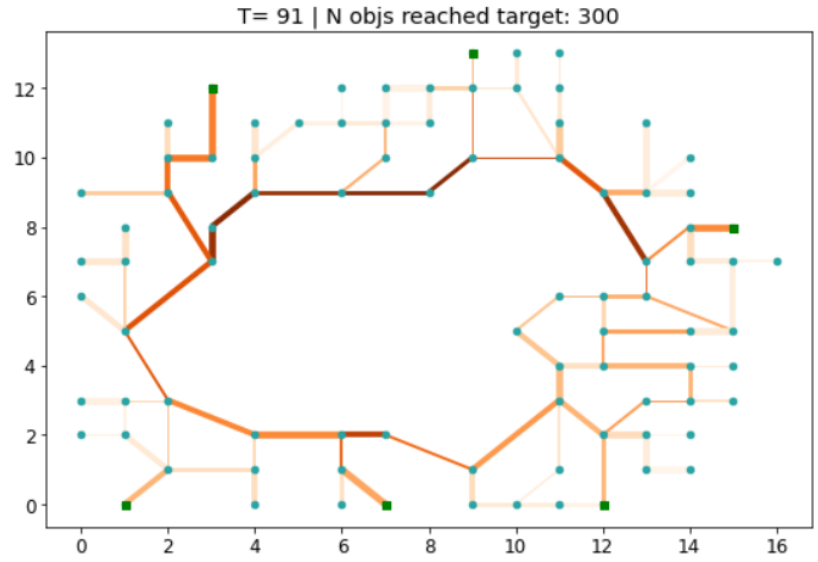
Figura 4.4: Exemplo topolológico de ponte com indicação de fluxo

O próximo exemplo, ilustrado nas figuras 4.5 e 4.6, utiliza uma topologia comparável a um lago no meio da malha rodoviária. Assim como ocorre na Lagoa Rodrigo de Freitas no Rio de Janeiro, esta disposição de ruas circundando a lagoa é comum e gera alto fluxo de veículos diretamente na área adjacente à lagoa.




```
Qtd de objs focando cada destino:
{'1-0': 1, '12-0': 2, '15-8': 3, '9-13': 2, '3-12': 1, '7-0': 1}
```

Figura 4.5: Exemplo Topolológico de Lago




```
Finalizado em 91 passos.

Qtd de objs focando cada destino:
{'1-0': 50, '12-0': 48, '15-8': 54, '9-13': 47, '3-12': 53, '7-0': 48}
```

Figura 4.6: Exemplo topolológico de lago com indicação de fluxo

Por fim, o seguinte exemplo de topologia (figuras 4.7 e 4.8) se baseia em um rio cruzando o meio da cidade, com 4 pontes ligando ambos os portos. Podemos observar na figura 4.8 como o fluxo se divide para atravessar o rio e chegar aos destinos.

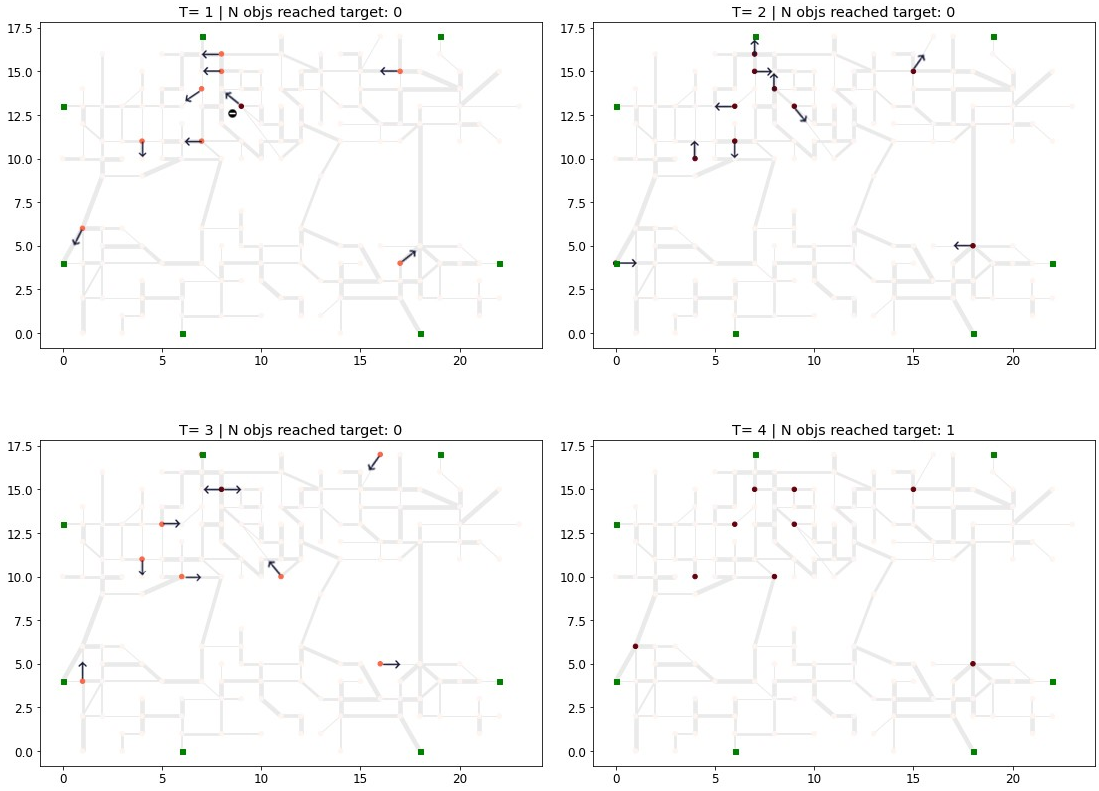
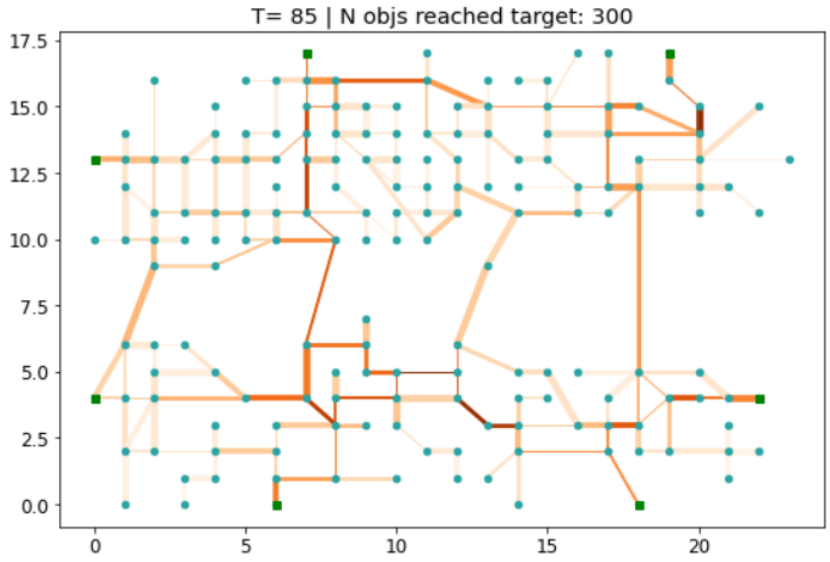


Figura 4.7: Exemplo Topolológico de Rio




```
Finalizado em 85 passos.

Qtd de objs focando cada destino:
{'6-0': 48, '0-4': 38, '18-0': 44, '22-4': 45, '19-17': 39, '7-17': 52, '0-13': 34}
```

Figura 4.8: Exemplo topológico de rio com indicação de fluxo

Em conclusão, a ferramenta Graph Walker se comportou como esperado nas situações anteriores e comprovou sua aplicabilidade para a aproximação de comportamento de objetos em grafos planares de diferentes topologias.

Todo o código fonte da Graph Walker é aberto e pode ser encontrado em: github.com/vivibpy/GraphWalker.

5. Conclusão

Falaremos sobre as conclusões do trabalho apresentado neste capítulo, fazendo uma breve retrospectiva sobre seu objetivo e como foi aplicado o estudo de caso. Também abordaremos os trabalhos futuros que poderão ser originados do presente trabalho.

5.1 Considerações gerais

O trabalho apresentado teve como objetivo o desenvolvimento de uma ferramenta extensível de simulação discreta de passeio aleatório simples e tendencioso em grafos planares. Desta forma, Graph Walker foi desenvolvida sobre os conceitos de processos estocásticos, simulação discreta, processos de Markov e passeio aleatório, de forma a facilitar o usuário na análise da movimentação dos objetos no grafo de escolha.

Para validar o comportamento da ferramenta foram analisadas três topografias distintas, facilmente relacionáveis com cenários reais da malha rodoviária.

Os resultados obtidos apresentaram uma boa aproximação da realidade, e seus resultados foram satisfatórios em todos as topografias executadas. Como esperado, foi possível identificar os pontos de maior fluxo nas pontes e adjacente, como no caso do primeiro e do terceiro exemplo (topologia de ponte e de rio respectivamente), e ao redor do lago, no segundo exemplo.

Além disso, a execução na plataforma Jupyter foi adequada e forneceu uma visão didática e organizada para facilitar o uso da ferramenta.

Portanto, Graph Walker simulou com sucesso o comportamento de objetos se movimentando em passeio aleatório simples e tendencioso. Sua aplicabilidade e facilidade de uso em diferentes topologias a torna uma ferramenta benéfica a melhor gestão de cenários de tomada de decisão impactados pelo movimento de objetos.

5.2 Trabalhos futuros

Oportunidades de continuidade da ferramenta foram identificadas para sua aplicabilidade em diversos cenários. Podemos citar a expansão da ferramenta para uso nas áreas gestão de emergências e de controle de doenças contagiosas, como também a produção de um arquivo de resultados gerados pela simulação.

Situações de emergência como acidentes e desastres naturais geram caos, e tornam a segurança e os procedimentos de evacuação das pessoas um desafio. Neste contexto, a ferramenta tem potencial para gerar caminhos alternativos com a inserção de "bloqueios"no grafo (equiparáveis a acidentes), possibilitando estudar áreas onde o desvio do fluxo causaria trânsito elevado e auxiliando na tomada de decisão para o melhor gerenciamento da situação.

Adicionalmente, na área de saúde podemos citar cenários onde há o contágio através do contato entre as pessoas. A ferramenta poderia ser expandida para atribuir aos objetos diversos estados, para que certos atributos passassem de um para o outro de acordo com a proximidade. Assim, o desenvolvimento da condição contagiosa poderia ser melhor estudada em ambientes com um alto número de pessoas em movimento em topologias sofisticadas.

A ferramenta Graph Walker também pode ser expandida para que os dados gerados pela simulação sejam registrados em um arquivo externo. Dado a arquitetura da ferramenta, todos os dados são mantidos em variáveis durante a simulação. Estes, portanto, estão acessíveis para que sejam facilmente exportados para um arquivo e aproveitados por outras aplicações.

Referências Bibliográficas

[1] Lawler, Gregory & Limic, Vlada. (2010). Random Walk: A Modern Introduction. 10.1017/CBO9780511750854.

[2] Pearson, K. "The problem of the random walk", Nature 72, 294; 318; 342 (1905).

[3] Ibe, O. Elements of Random Walk and Diffusion Processes. 1st ed. Massachusetts, Wiley Publishing.

[4] Pimentel-Souza F., 1992. Efeitos da poluição sonora no sono e na saúde em geral - ênfase urbana. Revista Brasileira de Acústica e Vibrações, 10: 12-22.

[5] Miyagi, Paulo E. Introdução à Simulação Discreta. USP – SP, 2006.

[6] Law, A. M. Simulation modeling and analysis, volume 4. McGraw-Hill New York (2007)

[7] Almeida, João Flávio de Freitas. (2015). Simulação por eventos discretos. Disponível em: <http://cursos.unipampa.edu.br/cursos/engenhariadeproducao/files/2016/08/apostila-sim-simulacao-por-eventos-discretos.pdf>

[8] Ferreira Filho, V. J. M. (2005). Processos estocásticos e teoria de filas, Programa de Engenharia de Produção, Universidade Federal do Rio de Janeiro, Rio de Janeiro, RJ

[9] Carazza, B. The history of the random-walk problem: considerations on the interdisciplinarity in modern physics. Riv. Nuovo Cim. 7, 419–427 (1977). https://doi.org/10.1007/BF02747280

[10] Patlak, C. S. (1953). Random walk with persistence and external bias. The Bulletin of Mathematical Biophysics, 15(3), 311–338. doi:10.1007/bf02476407

[11] Ames, Charles. "The Markov Process as a Compositional Model: A Survey and Tutorial."Leonardo, vol. 22 no. 2, 1989, p. 175-187. Project MUSE muse.jhu.edu/article/602251.

[12] Ross, Sheldon M., et al. Stochastic processes. Vol. 2. New York: Wiley, 1996.

[13] Melsa, James L., and Andrew P. Sage. An introduction to probability and stochastic processes. Courier Corporation, 2013.

[14] Bartlett, Maurice Stevenson. An introduction to stochastic processes: with special reference to methods and applications. CUP Archive, 1978.

[15] Kac, M. (1947). Random Walk and the Theory of Brownian Motion. The American Mathematical Monthly, 54(7), 369. doi:10.2307/2304386

[16] Wang, X.-Z., Zhai, J.-H., & Zhang, S.-F. (2008). A model of finite-step random walk with absorbent boundaries. International Journal of Computer Mathematics, 85(11), 1685–1696. doi:10.1080/00207160701543400