Distribuindo Usuários em uma Célula de Rede Móvel utilizando Simulação de Monte Carlo

Diogo Maciel da Cunha
Instituto de Computação
Universidade Estadual de Campinas
Campinas, São Paulo
Email: diogo@lrc.ic.unicamp.br

***Abstract*—A simulação de Monte Carlo é uma técnica estatística usada para entender o comportamento de sistemas complexos por meio da geração de amostras aleatórias. Este trabalho tem como objetivo utilizar a geração de pontos para determinar a distância média de um usuário para a estação base em uma célula de redes móveis. O simulador apresentou desempenho satisfatórtio, convergindo para os valores esperados no modelo analítico conforme o número de execuções crescia em todos os cenários utilizados no trabalho mas apresentou uma maior dificuldade com o segundo cenário e aparentou necessitar demais rodadas para alcançar a convergência completa.**

I. Introdução

A simulação de Monte Carlo é uma técnica estatística usada para entender o comportamento de sistemas complexos por meio da geração de amostras aleatórias [1]. Quando aplicada à distribuição de pontos em um círculo, essa técnica pode ser utilizada para estimar a densidade de pontos em diferentes regiões do círculo ou para simular fenômenos que envolvem distribuições circulares, como a distribuição de usuários dentro de uma célula de uma estação base em uma rede móvel.

Este trabalho tem como objetivo utilizar a geração de pontos para determinar a distância média de um usuário para a estação base em uma célula de redes móveis. Para isso, serão utilizados dois tipos de distribuições de pontos no circulo: 1. pontos uniformemente distribuídos segundo a localização no círculo e 2. uniformemente distribuídos segundo a distância até o centro do círculo.

No primeiro cenário, onde os pontos são uniformemente distribuídos segundo a localização no círculo, podemos pensar na distribuição dos usuários na célula como sendo uniforme em termos de sua localização angular. Isso significa que, independentemente da direção em que você olhe dentro da célula da estação base, a probabilidade de encontrar um usuário é constante. Por exemplo, se você estiver em um ponto aleatório dentro da célula e olhar em qualquer direção, a probabilidade de encontrar um usuário será a mesma em qualquer direção.

No segundo cenário, onde os pontos são uniformemente distribuídos segundo a distância até o centro do círculo, podemos relacionar isso com a distribuição de usuários em termos de sua distância até a estação base. Nesse caso, os usuários mais próximos do centro da célula têm uma densidade maior em comparação com os usuários mais distantes. Isso pode ser útil para entender a cobertura da célula, pois áreas mais próximas ao centro podem ter uma densidade maior de usuários, enquanto áreas mais distantes podem ter uma densidade menor.

Após isso, ambos os resultados foram comparados com seus respectivos modelos analíticos para validar o simulador. Foi constando um bom grau de precisão do simulador no primeiro cenário mas o segundo cenário não convergiu totalmente para seu modelo análitico por falta de rodadas de simulação. A seguir, encontrasse a descrição do sistema, os resultados e discussão e as conclusões sobre o trabalho.

II. Metodologia

A. Descrição do Sistema

A distribuição de pontos em uma célula pertencentes à uma estação base foi assumido como um fenômeno estático, que não varia com a alteração do tempo da simulação. Ademais, a célula foi representada por um círculo de raio *R* com centro na origem do plano cartesiano de duas dimensões, $O(0,0)$. Cada ponto, dos *k* pontos gerados, é um par ordenado composto por duas variáveis aleatórias continuas, $A(x_1, y_1)$, que determinam sua posição e está a uma distância *d* do centro da circunferência que é computada como o produto escalar dos valores anteriores.

$$d = \sqrt{x_1^2 + y_1^2}$$

Para o primeiro cenário a ser simulado, foi assumido que a probabilidade de um ponto aparecer em qualquer posição dentro do círculo segue uma distribuição uniforme. Para garantir essa propriedade, não foram impostas restrições de posição nos pontos gerados além do valor máximo do raio. A função de probabilidade, s_i , desse cenário é definida como o dobro do valor da distância, d divido pelo quadrado do raio, *R*.

$$f_{d_k}^{s_i}(d) = \begin{cases} \frac{2d}{R^2}, & \text{se } 0 < d < R \\ 0, & \text{do contrário} \end{cases}$$

Para garantir a uniformidade das distâncias no segundo cenário, foi utilizado coordenadas polares na geração das coordenadas de *A* mas foram assumidas como expressões lineares das distribuições uniformes que geram as coordenadas do primeiro cenário. Neste caso, a função de probabilidade das distâncias geradas é determinada apenas como uma distribuição uniforme definida pelo raio *R*.

$$f_{d_k}^{s_{ii}}(x) = \begin{cases} \frac{1}{R}, & \text{se } 0 < x < R \\ 0, & \text{do contrário} \end{cases}$$

B. Simulador

Para construir o simulador, foram utilizados duas linguagens de programação *Golang* e *Python*. A primeira linguagem foi utilizada para escrever o código responsável pelas simulações. Utilizando as ferramentas de paralelismo da linguagem *Go*, a execução dos experimentos foi separado em multiplas *threads* chamadas *goroutines*. Elas são divididas em coletores, agregadores e executores.

Os coletores são responsáveis por gerenciar os canais de comunicação por onde os executores enviaram seus resultados para os agregadores. Os executores executam as simulações gerando os pontos, calculando a distância para o centro e as métricas de dispersão (média, variância e desvio padrão). Por fim, os agregadores colhem os dados enviados através dos canais de comunicação e escrevem em um *buffer*, que periodicamente é escrito em um arquivo *CSV* (Do inglês *Comma-Separeted Values*) e esvasiado.

Após a execução do simulador, os dados armazenados em arquivos *CSVs* são lidos por um *Jupyter Notebook* que realização a visualização dos dados e a comparação com os valores esperados nos modelos analíticos. Para a leitura e tratamentos dos dados em csv, foi utilizado a biblioteca *pandas*. Para comparação com os valores analíticos, foram utilizadas as bibliotecas *mathplotlib* e *numpy*.

C. Experimentos

Para a execução dos experimentos, foi escolhido a geração de 40 pontos em um circulo de raio 0.75 durante um total de

$$\sum_{i=1}^{6} 10^i$$

execuções. Para cada execução, foram armazenados os pontos gerados e suas medidas de dispersão. Cada um desses valores, foi escrito em um arquivo *CSV* que foi lido pelo *Jupyter Notebook*, onde foram executadas as comparações com o modelo analítico.

A comparação é realizada em três etapas, comparação dos valores gerados com as *PDFs* e *CDFs* de cada modelo, *plot* de amostras de pontos gerados em cada cenário e comparação dos valores da média, devio padrão e variância. Na primeira etapa, são plotados os valores esperados da *CDF* e *PDF* de cada cenário para mil pontos, depois são plotadas as distâncias geradas para experimentos de $10^1, 10^2, 10^3, 10^4, 10^5$ e 10^6 execuções e observação a convergência entre ambas as curvas.

Após a comparação das distâncias geradas por monte carlo com as do modelo analítico, é escolhida uma amostra de *k* pontos gerados durante a execução da simulação e plotados em um circulo com as mesmas propriedades das simulações, com o objetivo de mostrar visualmente que os pontos estão sendo gerados corretamente. E por fim, é comparado as medidas de dispersão do analítico com as medidas de dispersões médias geradas durante as diferentes execuções.

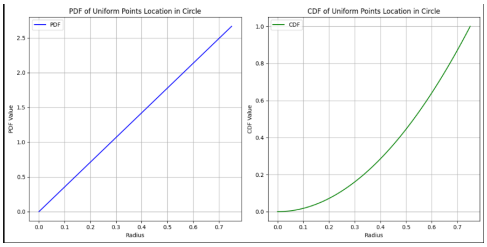

Fig. 1. PDF e CDF do cenário 1

$$F_{s_i}(x) = \begin{cases} \frac{x^2}{R^2}, & \text{se } 0 < x < R \\ 0, & \text{do contrário} \end{cases}$$

$$F_{s_{ii}}(x) = \begin{cases} \frac{x}{R}, & \text{se } 0 < x < R \\ 0, & \text{do contrário} \end{cases}$$

$$E_{s_i}(X) = \frac{2R}{3}$$

$$E_{s_{ii}}(X) = \frac{R}{2}$$

$$\sigma_{s_i}(X) = \frac{R^2}{18}$$

$$\sigma_{s_{ii}}(X) = \frac{R^2}{12}$$

III. Resultados e Discurssão

Após as execuções, foram gerados um total de 44444400 pontos e calculadas 1.111.110 métricas para cada cenário. Esses dados foram comparados com os valores do modelo analítico para 1000 distâncias através de um *fit* visual entre os gráficos gerados pelos modelos e os valores da simulação de monte carlo.

Como pode ser observado nos gráficos do primeiro cenário (Figuras 1, 2, 3), os modelos convergem para o mesmo ponto conforme a quantidade de simulações aumenta. Entretanto, esse comportamento não é observado na função de probabilidade do segundo cenário (Figuras 5, 6, 7). Isso pode ser um indicativo de que o segundo cenário necessita de mais simulações para convergir, pois a CDF pareceu convergir quase perfeitamente com os valores de encontrados por monte carlo.

Na segunda etapa da comparação, é possível observar que os pontos gerados na simulação (Figuras 4, 8), imitam com precisão o comportamento esperado para ambos os cenários. Por fim, na comparação das medidas de dispersão (Tabelas III e III), também é observado uma convergência para os valores analíticos conforme o número de rodadas de simulação aumenta, mas também é observado um certo ruido nos valores nos valores, principalmente no segundo cenário, os quais se afastam do valor ideal quando o número de execuções aumenta.

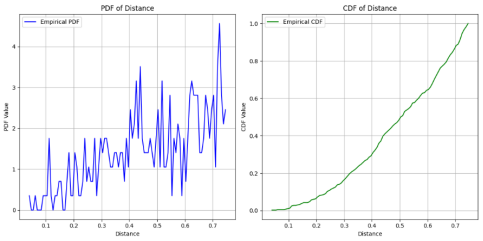

Fig. 2. Distâncias geradas após 10 execuções no cenário 1

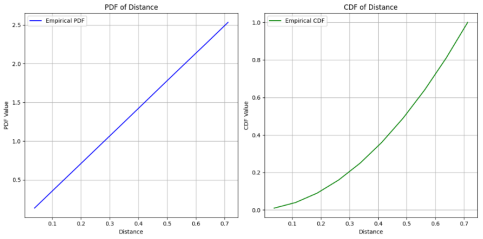

Fig. 3. Distâncias geradas após 10^6 execuções no cenário 1

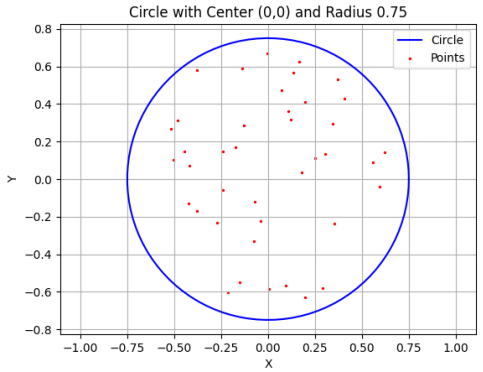

Fig. 4. Amostra de pontos gerados no cenário 1

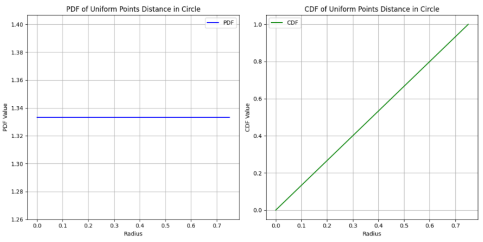

Fig. 5. PDF e CDF do cenário 2

IV. Conclusão

A simulação de Monte Carlo é uma técnica estatística usada para entender o comportamento de sistemas complexos por meio da geração de amostras aleatórias. Neste trabalho, foi implementado um simulador com duas linguagens, *Golang* e *Python*, baseado em morte carlo para a distribuição de k pontos em um circulo de raio R. O simulador apresentou desempenho satisfatórtio, convergindo para os valores esperados no modelo analítico conforme o número de execuções crescia em todos

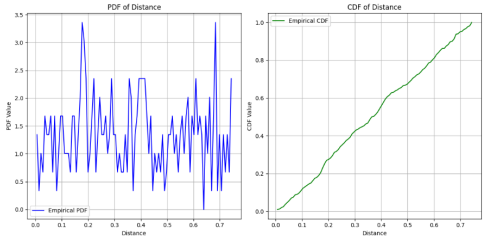

Fig. 6. Distâncias geradas após 10 execuções no cenário 2

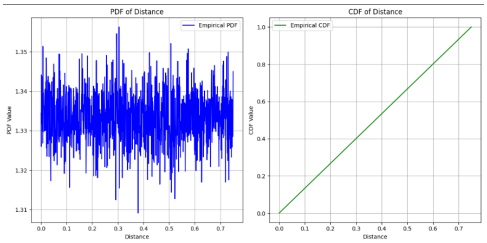

Fig. 7. Distâncias geradas após 10^6 execuções no cenário 2

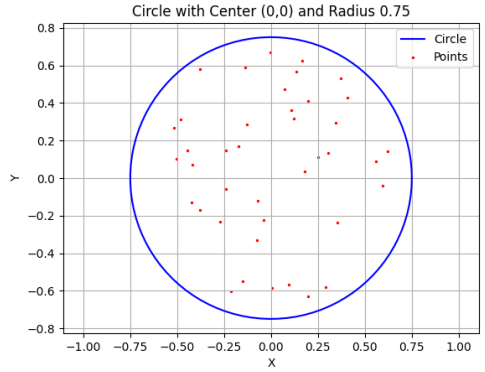

Fig. 8. Amostra de pontos gerados no cenário 2

os cenários utilizados no trabalho mas apresentou uma maior dificuldade com o segundo cenário e aparentou necessitar demais rodadas para alcançar a convergência completa.

References

[1] R. Jain, *The art of computer systems performance analysis: techniques for experimental design, measurement, simulation, and modeling*. Wiley New York, 1991, vol. 1.

Num. Exec.	Média	Variância	Dv. Padrão
10^1	0.0062018	0.0000065	0.0005494
10^2	0.0029307	0.0013139	0.0028627
10^3	0.0009386	0.000199	0.0002352
10^4	0.0003674	0.0000207	0.0008619
10^5	0.0000018	0.0000039	0.0008289
10^6	0.0000113	0.0000035	0.0008248

TABLE I
Diferença entre os valores das medidas de dispersão geradas por Monte Carlo e pelo modelo analítico no cenário 1

Num. Exec.	Média	Variância	Dv. Padrão
10^1	0.0021531	0.0073867	0.0029199
10^2	0.0031699	0.0013798	0.0004154
10^3	0.0003869	0.0000548	0.0002627
10^4	0.0003061	0.0004146	0.0000695
10^5	0.0000699	0.0005142	0.0000302
10^6	0.0000119	0.0005483	0.0000155

TABLE II

DIFERENÇA ENTRE OS VALORES DAS MEDIDAS DE DISPERSÃO GERADAS POR MONTE CARLO E PELO MODELO ANALÍTICO NO CENÁRIO 2