MODELAGEM BASEADA EM AGENTES

Gustavo Alexandre ROLDAM

roldam.gustavo@gmail.com

Luiz Carlos BEGOSSO

luiz.begosso@fema.edu.br

RESUMO: Esta pesquisa focou na modelagem baseada em agentes para identificar o algoritmo de ordenação mais otimizado, utilizando o NetLogo para testes. Foram analisados os algoritmos Bubble Sort, Counting Sort, Heap Sort, Insertion Sort, Quick Sort, Selection Sort e Shell Sort. O Counting Sort destacou-se como o mais otimizado após os testes.

PALAVRAS-CHAVE: Modelagem baseada em agentes; Algoritmo de ordenação; Comparação de desempenho; NetLogo;

ABSTRACT: This research focused on agent-based modeling to identify the most optimized sorting algorithm, using NetLogo for testing. The Bubble Sort, Counting Sort, Heap Sort, Insertion Sort, Quick Sort, Selection Sort and Shell Sort algorithms were analyzed. Counting Sort stood out as the most optimized after the tests..

KEYWORDS: Agent-based modeling; Sorting algorithm; Performance comparison; NetLogo;

1. INTRODUÇÃO

Modelos baseados em agentes são modelos de simulação que representam explicitamente agentes individuais, que podem ser humanos, instituições ou organismos com suas características e comportamentos. Estes modelos podem ser representados em plataformas computacionais específicas e consistem em ferramental utilizado em uma ampla gama de campos de pesquisa, incluindo ciências sociais, ciências da saúde, economia, ecologia, educação, ciências básicas e evolução.

A simulação de agentes é um campo de estudo crescente na ciência da computação e em outras áreas tais como a biologia, a economia e a medicina. Na simulação de agentes, o modelo criado é baseado nos comportamentos de indivíduos onde cada um dos quais toma decisões com base no conhecimento local e suas relações com outros agentes. O NetLogo é uma poderosa plataforma que suporta a simulação de modelos que utilizam agentes inteligentes e apresenta uma linguagem de programação que foi projetada para fornecer muitas das primitivas necessárias para simulação multiagente. O NetLogo pode atuar em qualquer domínio de aplicação e oferece uma ampla gama de funcionalidades e operadores genéricos aos seus usuários. Além disso, para Gaudou et al (2017), a plataforma é compatível com as linguagens de programação Python e R. Assim, o objetivo geral desse trabalho consiste em refletir sobre a construção de agente inteligente que perceba o seu ambiente e atue sobre ele. Para atender, especificamente, ao objetivo aqui estabelecido foram exploradas características computacionais do NetLogo que conduziram a: conhecer como o mundo modelizado evolui; identificar os estados do mundo modelizado; analisar como devem ser as percepções do agente; determinar como devem ser as consequências das ações do agente no mundo; caracterizar o sucesso das ações do agente no mundo; verificar como capturar mais conhecimento para o agente sobre o mundo; determinar como o agente deve colaborar ou competir com outros agentes; e, finalmente, aplicar e analisar algoritmos de classificação de dados como Bubble, Counting, Heap, Insertion, Quick, Selection e Shell Sort, para identificar o algoritmo que possui as melhores características de eficiência e robustez.

2. REVISÃO DA LITERATURA

A elaboração de modelos tem a finalidade de explicar e predizer fatos do mundo real. Dessa forma, White e Ingalls (2017) destacam que um modelo consiste numa entidade

que é utilizada para representar alguma outra entidade para um propósito específico. Em geral, os modelos são abstrações simplificadas, que abrangem apenas o escopo e o nível de detalhes necessários para satisfazer estudos em particular.

White e Ingalls (2017) argumentam que os modelos são empregados quando a investigação do sistema real é impraticável ou proibitiva e isso pode ocorrer porque a investigação direta é cara, lenta ou até mesmo pode comprometer a segurança dos pesquisadores ou do objeto em estudo. De fato, modelos podem ser usados para estudar sistemas que existem apenas em conceito e a simulação é uma abordagem particularpara estudar tais modelos, o que configura uma atuação fundamentalmente experimental. Para estes autores, a simulação é muito parecida com a execução de testes de campo, exceto que o sistema de interesse é substituído por um modelo físico ou computacional. Assim, a simulação envolve: criar modelos que tendem a imitar comportamentos de interesse; experimentar o modelo para gerar observações desses comportamentos; por fim, entender e generalizar tais comportamentos.

Para explorar um modelo em particular, utiliza-se de aparato computacional que tenha a condição de gerar estados simulados do cenário ou contexto estudado. Uma plataformade simulação computacional é uma alternativa que tem condições de reproduzir o ambiente real numa proporção simulada e que pode auxiliar o pesquisador a entender e explicar fenômenos. Dessa forma, um simulador pode conter uma linguagem de programação que trata de combinações complexas e apresenta uma interface gráfica para exibir os resultados encontrados.

De acordo com Alyaa e Chasib (2019), fundamentalmente, os simuladores deveriam apresentar as seguintes vantagens ao modelar um sistema real:

Diminuição do custo: o experimento de qualquer projeto executado no mundo real precisa de pessoal especializado, máquinas ou dispositivos que podem elevar os custos de sua execução; maior do que projeto executado virtualmente pela linguagem de simulação;

Diminuição do tempo: qualquer experimento implementado no mundo real leva mais tempo do que o experimento implementado por uma plataforma de simulação;

Melhora da confiabilidade: a implementação do experimento numa plataforma de simulação pode detectar erros no projeto do sistema e os processará antes de enfrentá-los na realidade.

No contexto desse trabalho, foi conduzido um estudo exploratório que investigou o

papel dos modelos baseados em agentes. Um agente é tudo o que pode ser considerado capaz de perceber seu ambiente por meio de sensores e de agir sobre esse ambiente por intermédio de atuadores (Russel e Norvig, 2003).

Em consonância com as ideias de Russel e Norvig, Bezzout e Faylali (2022) afirmam que a técnica de modelagem baseada em agentes é um método confiável e eficiente para simular o comportamento de fenômenos complexos. Na simulação baseada em agentes, cada agente atua como uma entidade autônoma interagindo com seu ambiente e com outros agentes.

O conceito de agente inteligente está relacionado à racionalidade e Russel e Norvig (2003) associaram quatro razões a esta questão:

a medida de desempenho que define critérios de sucesso;

o conhecimento prévio do agente sobre o ambiente;

as ações que o agente é capaz de executar;

a sequência de percepções do agente até o momento.

Em linhas gerais, para estes dois autores, um agente é tudo o que pode ser considerado capaz de perceber seu ambiente por meio de sensores e de agir sobre seu ambiente por intermédio de atuadores.

Thiele, Kurth e Grimm (2012) destacam que os agentes são modelos de simulação que representam explicitamente agentes individuais, que podem ser humanos, instituições ou organismos com suas características e comportamento. Eles são uma ferramenta estabelecida e cada vez mais usada em uma ampla gama de campos de pesquisa, incluindo ciências sociais, ciências da saúde, economia, ecologia e evolução.

Como mencionado anteriormente, no âmbito do presente trabalho será estudado o papel dos modelos baseados em agentes e para atender esta meta a plataforma NetLogo foi a selecionada.

Carbo, Sanchez-Pi e Molina (2018) ressaltam que o NetLogo é um ambiente de modelagem programável para simular fenômenos naturais e sociais. O NetLogo é adequado para modelar sistemas complexos que se desenvolvem ao longo do tempo. Os desenvolvedores de NetLogo podem dar instruções para centenas ou milhares de agentes independentes que operam simultaneamente no ambiente. Assim é possível explorar a conexão entre o comportamento em nível micro de indivíduos e comportamentos coletivos que emergem da interação de muitos indivíduos.

Por fim, autores como Burrows e Borowczak (2017) e Salecker et al (2019), destacam

que a plataforma NetLogo é um ambiente de modelagem baseado na linguagem de programação Java, que apresenta uma sintaxe compreensível, que permite prototipagem rápida de modelos baseados em agentes. Além destas características, o NetLogo oferece recursos para formular e implementar modelos complexos baseados em agentes de forma eficiente.

3. METODOLOGIA

Antes de iniciar a implementação dos algoritmos de ordenação, foi necessário conduzir uma pesquisa aprofundada sobre o NetLogo e a modelagem baseada em agentes. Esse processo envolveu a exploração de diversos artigos científicos que abordam esses tópicos, com um foco especialmente voltado para o conceito de agentes.

Em seguida, deu-se continuidade com a exploração da versão desktop do software NetLogo e a exploração de bibliotecas que permitem o monitoramento do uso da unidade central de processamento (CPU), tempo de execução de funções e consumo de memória. Duas bibliotecas foram utilizadas: a "MGR", responsável por registrar os consumos de memória e CPU, e a "Profiler", utilizada para armazenar os dados gerais de consumo, sendo empregada neste caso para registrar o tempo de duração de um método. Com as preparações do ambiente concluídas, procedeu-se à investigação dos principais métodos de organização. Esses métodos, conhecidos como algoritmos de ordenação, são categorizados em sete códigos distintos, cada um com sua própria lógica de organização. Os algoritmos analisados incluem o Bubble Sort, Counting Sort, Heap Sort, Insertion Sort, Quick Sort, Selection Sort e Shell Sort. Cada um desses algoritmos apresenta abordagens únicas de organização, que serão detalhadamente exploradas neste estudo.

Depois de definir todos os parâmetros, os algoritmos foram implementados no ambiente de desenvolvimento integrado (IDE) "DEV C++". O objetivo foi o de interpretar e ajustar os algoritmos para a linguagem Logo, utilizada na IDE NetLogo. Com os arquivos criados, uma série de testes foi conduzida para verificar o correto funcionamento dos códigos. Após a aprovação dos algoritmos nos testes individuais, um programa central foi criado para reunir todos os algoritmos em um só lugar, simplificando a coleta de dados.

4. ESTUDO DO CASO

A Modelagem Baseada em Agentes tem se destacado como uma abordagem eficiente e flexível para estudar sistemas complexos e adaptativos em diversas áreas do conhecimento. Neste estudo de caso, explorou-se o uso dessa metodologia com foco em três pilares fundamentais: "Agentes", "NetLogo" e "Códigos de Ordenação".

Agentes, representados por entidades autônomas que interagem entre si e com o ambiente, têm sido utilizados como modelos para simular sistemas do mundo real, como populações, ecossistemas, tráfego urbano e redes sociais. O NetLogo, uma linguagem de programação multiagente, é uma ferramenta poderosa e amplamente adotada para a implementação de modelos baseados em agentes. Além disso, este estudo de caso aborda códigos de ordenação, algoritmos utilizados para organizar elementos em uma sequência específica.

Portanto, este estudo de caso busca destacar estes três pilares, apresentando o NetLogo como uma ferramenta-chave para a implementação de modelos, bem como a aplicação de algoritmos de ordenação para investigar dinâmicas e padrões emergentes em sistemas complexos. Com o propósito de encontrar o algoritmo mais otmizado, as escolhas de códigos de ordenação foi feita a partir de sete algoritmos de organização sendo eles: Bubble, Counting, Heap, Insertion, Quick, Selection, e o Shell sort. A seguir, descreve-se os sete algoritmos de ordenação.

4.1. Bubble Sort

O método Bubble Sort (Bolha) é um método de classificação simples frequentemente empregado para ensinar os princípios fundamentais de algoritmos de organização Sua lógica consiste em comparar pares de elementos ao longo de toda a lista, resultando no deslocamento gradual do maior elemento para o final da lista. A Figura 1 ilustra o esquema lógico de trocas entre os elementos a serem ordenados.

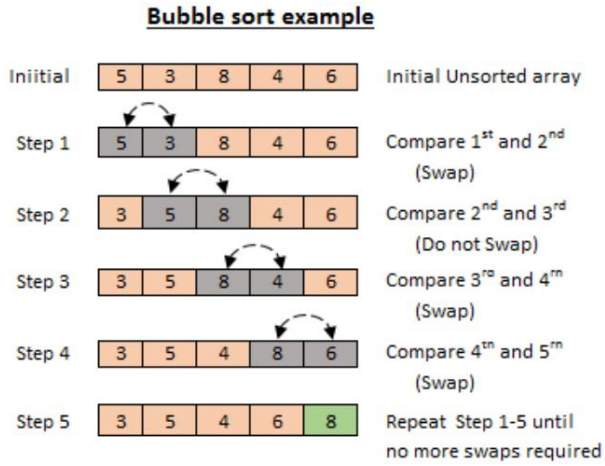

Figura 1 – Bubble SortExemplo de ordenação com o Bubble Sort

A Figura 1 apresenta um exemplo com uma lista de 5 elementos a serem ordenados. O teste de comparação de pares será executado 20 vezes, mesmo que a lista esteja ordenada antes disso. O algoritmo não possui a capacidade de detectar automaticamente que a ordenação foi concluída e, portanto, realiza as comparações até o final. A Figura 2 ilustra o código de ordenação do método Bolha utilizando o NetLogo.

```
to bolha
  update-lista-bolha
  let n1 0
  let aux 0
  let i 0
  let t length my-lista
  while [n1 < t] [
    while [i < t - 1] [
      if item i my-lista > item (i + 1) my-lista [
        set aux item i my-lista
        set my-lista replace-item i my-lista item (i + 1) my-lista
        set my-lista replace-item (i + 1) my-lista aux
      ]
      set i (i + 1)
    ]
    set i 0
    set n1 (n1 + 1)
  ]
  set tempo-bolha timer
end
```

Figura 2– Código NetLogo Bubble Sort

4.2. Counting Sort

O método Counting Sort é um algoritmo de ordenação que classifica elementos de uma lista baseada em seus valores numéricos. A característica distintiva do Counting Sort é que ele explora o intervalo conhecido dos valores a serem ordenados para realizar a

classificação. Ele realiza a ordenação das posições por meio de uma lista auxiliar que conta a quantidade de números em cada posição. Em seguida, um cálculo é realizado nessa lista auxiliar para determinar a posição na qual o valor deve ser organizado. O Counting Sort apresenta uma limitação na questão dos valores que serão ordenados. Por conta de seu método de organização o algoritmo é limitado para listas numéricas sem possibilidade de utilizar caracteres como valor de organização. A Figura 3 apresenta um esquema com a ordenação de valores numéricos com o Counting Sort.

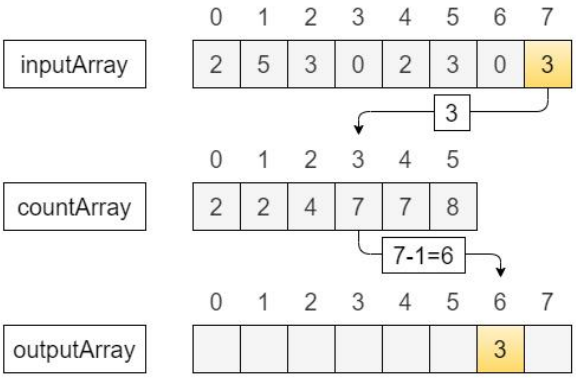

Figura 3 – Exemplo de ordenação de valores com o Counting Sort.

4.3. Heap Sort

O Heap Sort é um algoritmo de ordenação que utiliza árvores binárias para executar sua função. As árvores binárias são uma estrutura de dados composta por nós, sendo que cada nó é denominado pai e pode ter até dois filhos, ou até mesmo nenhum. Além disso, esses filhos, por sua vez, podem também ser pais de outros nós, formando uma hierarquia. Geralmente, os pais que possuem apenas um ou nenhum filho são localizados nas camadas mais baixas da árvore.

O algoritmo Heap gera uma árvore binária a partir da lista de valores, onde os elementos são comparados e reorganizados de tal forma que o maior valor seja posicionado no topo da árvore. A Figura 4 ilustra um exemplo de ordenação de valores com o algoritmo Heap Sort.

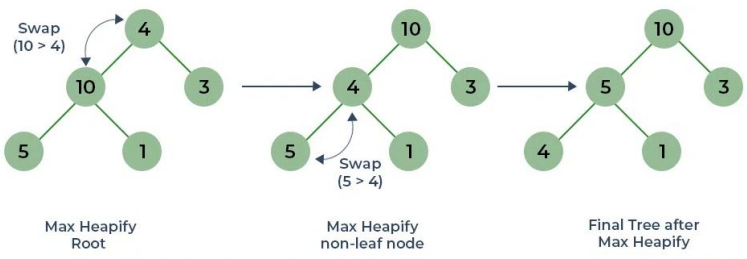

Figura 4– Esquema de ordenação de valores do Heap Sort.

4.4. Insertion Sort

Uma evolução do método de ordenação Bubble, o algoritmo Insertion emprega uma abordagem logicamente semelhante à do Bubble. No entanto, ele apresenta aprimoramentos, como a capacidade de reconhecer quando a lista já está ordenada. Nesse caso, ele movimenta o valor diretamente para a sua posição adequada, evitando trocas desnecessárias. Isso pode ser observado na Figura 5.

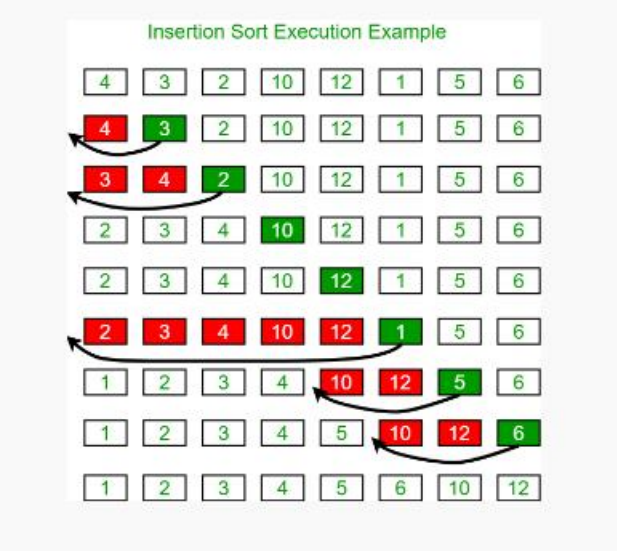

Figura 5 Exemplo de organização de dados com o Insertion Sort.

4.5. Quick Sort

Baseado na ideia de "dividir para conquistar", o Quick Sort é um algoritmo que emprega a recursão de funções, que é uma chamada da função dentro dela mesma. O método de ordenação do Quick Sort envolve a escolha de um elemento pivô, que é posicionado em seu local correto dentro da lista ordenada. Após essa ordenação, o código divide a lista ao meio e continua o processo iterativo até que todas as porções estejam devidamente ordenadas. A Figura 6 apresenta esquema gráfico que ilustra tal situação.

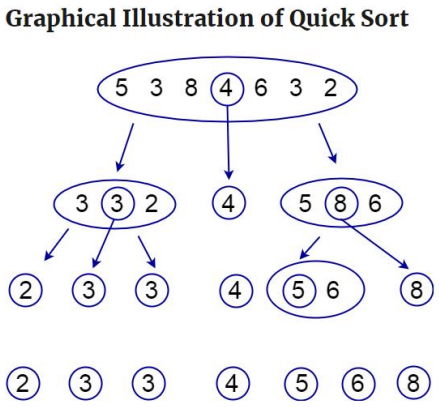

Figura 6 - Exemplo de ordenação com o Quick Sort.

4.6. Selection Sort

O Selection Sort apresenta semelhanças marcantes com o Insertion Sort, embora sua abordagem de verificação seja ligeiramente diferente, sendo verificada toda a lista em vez de um local previamente determinado. Esse algoritmo garante que, após cada verificação, a posição correta de um elemento seja estabelecida, permitindo que essa posição seja excluída da verificação subsequente. Esse método de ignorar posições previamente ordenadas agiliza o processo de classificação em iterações subsequentes. A Figura 7 apresenta exemplo do método.

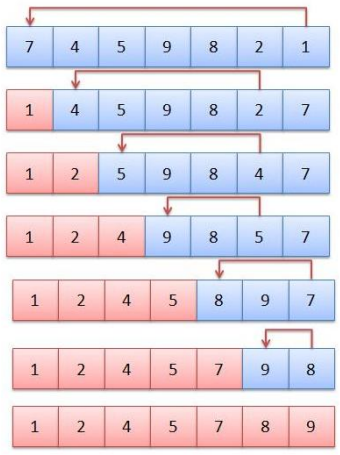

Figura 7– Exemplo do Selection Sort.

4.7. Shell Sort

O Shell Sort, uma evolução do Insertion Sort, destaca-se pela introdução de uma sequência de intervalos definida por "h". Esse valor "h" é calculado utilizando a fórmula

"(h-1) / 3" e é gradualmente reduzido até atingir um valor menor que zero. Conforme "h" diminui e alcança o valor de 1, o Shell Sort passa a se comportar de maneira semelhante ao Insertion Sort, o que evidencia que o Shell Sort é, de fato, uma progressão do mesmo. A Figura 8 representa o algoritmo Shell Sort implementado em NetLogo.

```
to shell
  update-lista-shell
  let i 0
  let j 0
  let h 1
  let aux 0
  let t length my-lista
  reset-timer

  while [h < (t / 3)] [
    set h (3 * h + 1)
  ]

  while [h > 0] [
    set i h

    while [i < t] [
      set aux item i my-lista
      set j i

      while [j >= h and aux < item (j - h) my-lista] [
        set my-lista replace-item j my-lista item (j - h) my-lista
        set j (j - h)
      ]

      set my-lista replace-item j my-lista aux
      set i (i + 1)
    ]

    set h ((h - 1) / 3)
  ]
end
```

Figura 8 - Código NetLogo do algoritmo Shell Sort

A abordagem do Shell Sort, ao permitir a comparação de elementos distantes uns dos outros durante a fase inicial da ordenação, contribui para uma otimização na reorganização dos valores. A utilização de diferentes intervalos "h" durante as iterações proporciona uma melhoria no desempenho, pois permite que elementos distantes sejam comparados e rearranjados em etapas anteriores do processo. Essa característica ajuda a reduzir a quantidade de movimentações de elementos, resultando em um algoritmo mais eficiente que seu antecessor.

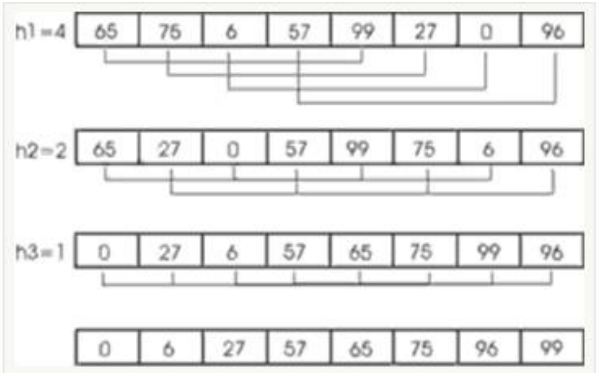

Figura 9 - Esquema gráfico de ordenação utilizando o Shell Sort.

5. RESULTADOS

Com a finalização da implementação lógica, iniciaram-se os procedimentos de teste que abrangeram avaliações de velocidade, consumo de CPU e utilização de memória. Esta fase visava à análise do desempenho e da eficiência dos algoritmos em diversos cenários com números de agentes diferentes, sendo eles 1.000, 10.000 e 100.000 cada valor possui três variações sendo eles com a lista em ordem crescente, decrescente e aleatória. Após realizar os devidos testes chegou-se aos valores apresentados na Figura 10.

QUANTIDADE DE TEMPO (S)									
SORTS	NUMERO DE AGENTES								
-	1000	10000	100000	1000	10000	100000	1000	10000	100000
	ALEATÓRIA			CRESCENTE			DECRESCENTE		
BOLHA	0.513	53.32	29506.08	0.48	55.63	27504.80	0.47	56.53	29167.39
COUNTING	0.01	0.05	0.47	0.01	0.03	0.66	0.01	0.04	0.74
HEAP	0.02	0.26	6.96	0.02	0.20	10.36	0.02	0.18	10.77
INSERTION	0.18	24.09	8058.35	0.17	24.35	12473.47	0.17	23.66	12663.38
QUICK	0.03	0.22	7.62	0.02	0.22	12.80	0.02	0.20	13.10
SELECTION	0.03	2.97	844.88	0.04	3.39	1341.48	0.04	3.32	1413.06
SHELL	0.01	0.20	8.29	0.01	0.20	17.18	0.02	0.20	12.81

TEMPO DA CPU (MS)									
SORTS	NUMERO DE AGENTES								
-	1000	10000	100000	1000	10000	100000	1000	10000	100000
	ALEATÓRIA			CRESCENTE			DECRESCENTE		
BOLHA	0.48	33.19	9073.83	0.48	34.61	4605.77	0.45	34.62	16921.50
COUNTING	0.48	33.20	9074.03	0.48	34.66	4605.92	0.45	34.66	16921.64
HEAP	0.5	33.34	9075.88	0.52	34.78	4607.62	0.48	34.78	16923.45
INSERTION	0.67	47.72	11203.55	0.67	49.19	6645.27	0.64	49.13	19015.66
QUICK	0.69	47.86	11205.60	0.70	49.30	6647.34	0.64	49.28	19017.90
SELECTION	0.72	49.67	11443.38	0.73	51.42	6881.39	0.69	51.31	19272.97
SHELL	0.73	49.81	11445.67	0.75	51.58	6883.59	0.70	51.44	19275.97

QUANTIDADE USADA DA MEMÓRIA (MB)									
SORTS	NUMERO DE AGENTES								
-	1000	10000	100000	1000	10000	100000	1000	10000	100000
	ALEATÓRIA			CRESCENTE			DECRESCENTE		
BOLHA	38.61	52.59	261.93	85.01	86.22	342.69	102.14	50.39	311.62
COUNTING	40.61	70.59	266.42	87.01	41.28	348.15	104.14	67.39	294.10
HEAP	50.61	121.69	268.40	96.01	73.32	178.74	44.14	52.50	347.64
INSERTION	109.61	70.81	205.67	60.05	90.49	139.37	64.07	60.30	218.96
QUICK	118.61	117.86	178.23	69.05	52.54	161.93	73.08	107.68	320.63
SELECTION	37.64	112.92	248.84	84.05	82.66	461.51	87.08	91.75	260.50
SHELL	40.62	78.98	241.35	93.05	71.72	303.07	97.08	100.80	324.02

Figura 10 – Tabela de Dados de Consumo

Após a coleta precisa de todos os dados, nos quais números em vermelho representam os piores desempenhos de suas respectivas colunas, enquanto os elementos em azul indicam os melhores, foi desenvolvida uma quarta tabela. Essa tabela teve como

propósito a atribuição de pontuações aos algoritmos, com o intuito de identificar qual deles apresenta a melhor otimização para a organização dos dados. Cada falha do algoritmo "sort" resultou em uma redução de 1 ponto, a cada sucesso acrescentou-se 2 pontos e caso o ponto seja neutro será acresentado 1 ponto. Após a realização dos cálculos, o resultado obtido está representado na Figura 11.

QUANTIDADE USADA DA MEMÓRIA (MB)										
-	TEMPO			CPU			MEMÓRIA			Total
SORTS	Positivos	Negativos	Neutros	Positivos	Negativos	Neutros	Positivos	Negativos	Neutros	-
BOLHA	0	9	0	9	0	0	3	0	6	21
COUNTING	9	0	0	3	0	6	1	1	7	38
HEAP	0	0	9	0	0	9	1	4	4	20
INSERTION	0	0	9	0	0	9	3	1	5	28
QUICK	0	0	9	0	0	9	1	2	6	24
SELECTION	0	0	9	0	0	9	0	1	8	25
SHELL	2	0	7	0	9	0	0	0	9	11

Figura 11 – Tabela de Resultados com Pontuação

Por fim, chegou-se à conclusão de que o algoritmo Counting Sort foi o mais otimizado com base nos resultados coletados, enquanto o código Shell Sort apresentou a menor otimização em comparação com aos demais.

6. CONCLUSÃO

O presente estudo objetivou realizar uma pesquisa sobre Agentes e sobre o NetLogo, onde foi definido como objetivo analisar diversos códigos de ordenação com a finalidade de descobrir qual era o mais otimizado seguindo os critérios de tempo de execução do algoritmo, tempo de uso da CPU e quantidade de memória consumida, além de adquirir mais conhecimento sobre Modelagem de Agentes.

Os resultados dessa pesquisa indicaram que o algoritmo mais apropriado, com base nos testes e critérios de pontuação estabelecidos para comparação, foi o Counting Sort. Esta descoberta ressalta a eficiência desse algoritmo em relação aos demais avaliados.

Para projetos futuros, é espera-se ampliar a comparação para outros algoritmos de ordenação, como o Merge Sort, a fim de obter uma compreensão mais completa das diferenças de desempenho entre eles. Além disso, poderiam ser incorporados métodos adicionais de avaliação dos códigos, como a inclusão de um contador de trocas na lista, proporcionando uma análise mais detalhada do comportamento de cada algoritmo.

Essas expansões podem oferecer para pesquisas subsequentes uma compreensão maior

sobre os algoritmos de ordenação e sua aplicação em simulações baseadas em agentes.

REFERÊNCIAS BIBLIOGRÁFICAS

BACKES, André. **[ED] Aula 53 - Ordenação – HeapSort.** Site Youtube. Disponível em: https://www.youtube.com/watch?v=zSYOMJ1E52A. Acesso em: 28 mai. 2014-a.

_____. **[ED] Aula 53 - Ordenação - HeapSort.** Site Youtube. Disponível em: https://www.youtube.com/watch?v=zSYOMJ1E52A. Acesso em: 28 mai. 2014-b.

Insertion sort. Khan Academy, 2023. Disponível em: https://pt.khanacademy.org/computing/computer-science/algorithms/insertion-sort/a/insertion-sort. Acesso em 15 jun. 2023.

_____. **[ED] Aula 52 - Ordenação – QuickSort.** Site Youtube. Disponível em: https://www.youtube.com/watch?v=spywQ2ix_Co. Acesso em: 23 mai. 2014.

_____. **[ED] Aula 50 - Ordenação – SelectionSort.** Site Youtube. Disponível em: https://www.youtube.com/watch?v=zjcGGqskf5s. Acesso em: 16 mai. 2014.

_____. **[ED] Aula 140 - Ordenação: ShellSort.** Site Youtube. Disponível em: https://www.youtube.com/watch?v=aiafbYZB7S4. Acesso em: 28 mar. 2022.

BURROWS, Andrea; BOROWCZAK, Mike. Teaching Teachers to Think Like Engineers Using NetLogo. In: ASEE 2017 ANNUAL CONFERENCE, 2017, Columbus – OH, USA. Proceedings of 2017 ASEE Annual Conference & Exposition, jun, 2017, p. 1-11.

CARBO, J.; SANCHES-PI, N.; MOLINA, J. M. Agent-based simulation with NetLogo to evaluate ambient intelligence scenarios. Journal of Simulation, volume 12, number 1, 2018, p. 42 – 52.

Counting Sort. Desenvolvimento Software, 2023. Disponível em: http://desenvolvendosoftware.com.br/algoritmos/ordenacao/counting-sort.html. Acesso em: 02 jun. 2023.

Estruturas de Dados I Algoritmos de Ordenação pt.2*. Caderno Geek, 2023. Disonível em: https://cadernogeek.wordpress.com/tag/shell-sort/. Acesso em: 10 mai. 2023.

FONSECA, Celso. **UM MODELO DE SIMULAÇÃO A PARTIR DO AMBIENTE NETLOGO.** Site Unipampa. Disponivel em: https://eventos.unipampa.edu.br/eremat/files/2014/12/MC_FONSECA_00529108062.pdf. Acesso em 10 fev. 2023.

GAUDOU, B.; LANG, C.; MARILLEAU, N.; SAVIN, G.; COYREHOURCQ, S. R.; NICOD, J. M. Netlogo, an open simulation environment. In: BANOS, Arnaud; LANG, Christophe; MARILLEAU, Nicolas. Agent-based Spatial Simulation with NetLogo, Elsevier, 2017, Volume 2, p. 1-36.

GUPTA, Vikram. **Visualizing, Designing, and Analyzing the Counting Sort Algorithm.** Site Medium. Disponível em: https://levelup.gitconnected.com/visualizing-designing-and-analyzing-the-counting-sort-algorithm-dff56ea01e93. Acesso em 10 fev. 2023.

Heap Sort – Data Structures and Algorithms Tutorials. GeeksforGeeks, 2023. Disponivel em: https://www.geeksforgeeks.org/heap-sort/. Acesso em 25 jul. 2023.

Insertion Sort – Data Structure and Algorithm Tutorials. GeeksforGeeks, 2023. Disponivel em: https://www.geeksforgeeks.org/insertion-sort/. Acesso em 31 mai. 2023.

NETLOGO. Site do NetLogo. Disponível em:

https://ccl.northwestern.edu/netlogo/. Acesso em 09 fev. 2023.
PINTOS, Andrio. **Simulação e avaliação de comportamentos em sistemas multi-agentes baseados em modelos de reputação e interação.** Site Unisinos. Disponível em: http://www.repositorio.jesuita.org.br/handle/UNISINOS/2261. Acesso em 2008.
RAND, Bill. **Introdução à modelagem beseada em agentes (verão de 2016).** Site Complexity Explorer Santa FE Institute. Disponível em: https://www.complexityexplorer.org/courses/23-introduction-to-agent-based-modeling-summer-2016. Acesso em: 09 set. 2016.
SALECKER, J.; SCIAINI, M.; MEYER, K. M.; WIEGAND, K. The nlrx r package: A next-generation framework for reproducible NetLogo model analyses. Methods in Ecology and Evolution, 10(11), 2019, p. 1854-1863.
SEABRA, Giulianna. **Bubble Sort: o que é e como usar? Exemplos práticos!** Site Betrybe. Disponível em: https://blog.betrybe.com/tecnologia/bubble-sort-tudo-sobre/. Acesso em: 20 dez. 2022.
Srikanta. **How to Implement Quick Sort in C?** Site QnA Plus. Disponível em: https://qnaplus.com/implement-quick-sort-c/. Acesso em 01 jun. 2007.
Selection Sort algorithm. Stack overflow, 2023. Disponivel em: https://stackoverflow.com/questions/36700830/selection-sort-algorithm. Acesso em 05 mar. 2016.
THIELE, J. C.; KURTH, W.; GRIMM, V. Agent-based modelling: Tools for linking NetLogo and R. Journal of Artificial Societies and Social Simulation, volume 15, número 3, june, 2012, p. 1 – 8.