Computação em grade pode ser compreendida como um modo de estruturar a cooperação entre diferentes máquinas a fim de realizar diferentes tarefas. Em um cenário onde máquinas, com diferentes recursos computacionais (poder de processamento, disponibilidade de armazenamento, memória RAM, entre outros), podem estar em diferentes locais e receber diferentes tipos de tarefas (computação distribuída), se faz necessário pensar em algoritmos para escalonar essas tarefas em paralelo a fim de otimizar o tempo e os recursos disponíveis. Portanto, para a computação em grade, distribuir o poder computacional e otimizar sua utilização é mais relevante do que adquirir mais recursos[1].

Por se tratar de um problema NP-Completo (detalhar mais), é fácil conferir a exatidão das soluções encontradas, mas não existe uma maneira de encontrar a melhor solução possível sem verificar todas as possibilidades. Consequentemente, para que se encontre soluções boas, ou quase ótimas, em um tempo menor, é necessário aplicar algoritmos de escalonamento que utilizem funções heurísticas [3] (detalhar mais). Todavia, essas heurísticas precisam se adaptar a heterogeneidade dos recursos disponíveis e das tarefas as serem realizas em pouco tempo e, se conhecer as aplicações e máquinas disponíveis, aplicar heurísticas que consumam menos recurso para escalonar as tarefas e achem boas soluções para cenários com características em comum. Assim sendo, é vantajoso conhecer quais problemas ou tarefas o sistema será encarregado, quais recursos estão disponibilizados pelas máquinas e quão heterogêneo as máquinas e as tarefas são, ou poderão vir a ser.

**Objetivos**

O objetivo principal desse trabalho é analisar e propor alterações do simulador implementado por Verma[7] e modificado por Barra[8], sendo elas:

* Parametrizar o simulador para que seja mais prático ao usuário inserir dados sobre as máquinas disponíveis e as tarefas a serem realizadas;
* Permitir que o usuário possa escolher entre gerar novos dados para realizar a simulação ou utilizar dados existentes em um arquivo;
* Permitir que o usuário possa salvar os resultados obtidos em um arquivo separado, permitindo resultados de outras pesquisas possam ser comparadas com mais facilidade;
* Implementar outras heurísticas, fim de poder analisa-las em comparação com as já implementadas por Verna[7] e Barra[8] (MET, MCT, MaxMin, MinMin, MinMean, MinVar, OLB, XSufferage).

Ideias:

* + Maximum Standard Deviation
  + Duplex: The Duplex heuristic is literally a combination of the Minmin and Maxmin heuristics. The Duplex heuristic performs both of the Minmin and Maxmin heuristics and then uses the better solution [3, 17]. Duplex can be performed to exploit the conditions in which either Minmin or Maxmin performs well, with negligible overhead.

**Referências**

[1] – SINGH, S.; BAWA, R. **An efficient job scheduling algorithm for grid computing**. Proceedings of the International Conference on Advances in Computing and Artificial Intelligence. **Anais**... Em: ACAI ’11: INTERNATIONAL CONFERENCE ON ADVANCES IN COMPUTING AND ARTIFICIAL INTELLIGENCE. Rajpura/Punjab India: ACM, 21 jul. 2011. Disponível em: <<https://dl.acm.org/doi/10.1145/2007052.2007097>>. Acesso em: 9 fev. 2023

[2] – BHATIA, M. **RR based grid scheduling algorithm**. Proceedings of the International Conference on Advances in Computing and Artificial Intelligence. **Anais**... Em: ACAI ’11: INTERNATIONAL CONFERENCE ON ADVANCES IN COMPUTING AND ARTIFICIAL INTELLIGENCE. Rajpura/Punjab India: ACM, 21 jul. 2011. Disponível em: <<https://dl.acm.org/doi/10.1145/2007052.2007076>>. Acesso em: 10 fev. 2023

[3] – ZHANG, Y.; INOGUCHI, Y.; SHEN, H. A Dynamic Task Scheduling Algorithm forGrid Computing System. Em: CAO, J. et al. (Eds.). **Parallel and Distributed Processing and Applications**. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004. v. 3358p. 578–583.

<https://sci-hub.hkvisa.net/10.1007/978-3-540-30566-8_69>

[4] – BRAUN, T. D. et al. A Comparison of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing Systems. **Journal of Parallel and Distributed Computing**, v. 61, n. 6, p. 810–837, jun. 2001.

<https://sci-hub.ru/https:/www.sciencedirect.com/science/article/abs/pii/S0743731500917143>

[5] – DONG, Fangpeng; AKL, Selim G. Scheduling Algorithms for Grid Computing: State of the Art and Open Problems. *[S. l.]*, [s.d.].

<https://research.cs.queensu.ca/home/akl/techreports/GridComputing.pdf>

[6] – HEMAMALINI, M.; SRINATH, M. V. STATE OF THE ART: TASK SCHEDULING ALGORITHMS IN A HETEROGENEOUS GRID COMPUTING ENVIRONMENT. *[S. l.]*, v. 1, n. 1, 2014.

[State of the art: Task scheduling Algorithms in Grid Computing Environment (ejerm.com)](http://ejerm.com/vol1_aug2014/img/cs004%20%20CRC%20STATE%20OF%20THE%20ART%20TASK%20SCHEDULING%20ALGORITHMS%20IN%20A%20HETEROGENEOUS%20GRID%20COMPUTING%20ENVIRONMENT.pdf)

[7] –VERMA, Apurv. **dapurv5/distributedscheduling**. 2022. Disponível em: <https://github.com/dapurv5/distributedscheduling>. Acesso em: 10 fev. 2023.

[8] - BARRA, Marcos Victor de Aquino. **Comparação de algoritmos para o problema de escalonamento de tarefas em grades computacionais**. Orientador: Paulo Henrique Ribeiro Gabriel. 2022. 45 p. Trabalho de conclusão de curso (Bacharelado em Ciência da Computação) - Universidade Federal de Uberlândia, Uberlândia, 2022.