Escalonando *Workflows* com Redes Convolucionais de Grafos

Lucas Oliveira Macedo[1], **Guilherme Piêgas Koslosvki**[1]

[1]Universidade do Estado de Santa Catarina (UDESC) – Joinville, SC – Brasil

Resumo. *Este trabalho propõe o uso de uma Residual Graph Convolutional Network (ResGCN) para selecionar dinamicamente uma política para escalonamento de workflows. O modelo avalia indicadores de makespan, footprint, slowdown e balanceamento de carga. A ResGCN obteve mais de 56,67% de acurácia, indicando um caminho promissor na identificação de gargalos e adaptação ao estado da infraestrutura.*

1. Introdução

Ambientes de computação de alto desempenho (HPC - *High Performance Computing*) são essenciais para a execução de aplicações científicas complexas em áreas como sismologia, astronomia e simulações climáticas. Frequentemente, essas aplicações são modeladas como *workflows* estruturados na forma de Grafos Acíclicos Direcionados (DAGs - *Directed Acyclic Graphs*). O escalonamento eficiente dessas tarefas nos recursos computacionais disponíveis é um problema clássico, classificado como NP-Difícil [Brucker 2007]. Para lidar com essa complexidade em tempo hábil, diversas heurísticas de escalonamento, como o HEFT (*Heterogeneous Earliest Finish Time*) [Topcuoglu et al. 2002] e o EFT (*Earliest Finish Time*) foram propostas. No entanto, essas estratégias tradicionais tendem a ser estáticas em relação a fatores dinâmicos como a degradação e o congestionamento da rede, o que pode comprometer o desempenho global.

Com o amadurecimento das técnicas de Aprendizado de Máquina, há a possibilidade de criar escalonadores adaptáveis. Neste cenário, as Redes Neurais em Grafos (GNNs - *Graph Neural Networks*) se destacam por conseguirem processar diretamente a estrutura topológica dos *workflows*, extraindo padrões complexos sobre o caminho crítico e os potenciais gargalos antes mesmo da execução [Scarselli et al. 2009]. Neste contexto, o presente trabalho propõe a utilização de uma rede neural profunda baseada em grafos, especificamente uma *Residual Graph Convolutional Network* (ResGCN), para atuar como um classificador que decide o algoritmo de escalonamento adequado para um dado cenário.

O restante deste artigo está organizado da seguinte forma: os trabalhos relacionados são discutidos na Seção 2. A Seção 3 detalha a metodologia proposta. A Seção 4 apresenta discute os resultados. A Seção 5 traz as considerações finais e perspetivas.

2. Trabalhos relacionados

A aplicação de *Machine Learning* (ML) para otimizar o escalonamento em HPC tem se mostrado promissora em comparação as heurísticas estáticas. Trabalhos recentes demonstraram a viabilidade do uso de ML clássico para prever a eficiência de ações de escalonamento [Kraus et al. 2024] e para compor ativamente políticas baseadas no estado do sistema [Diel et al. 2024]. Paralelamente, a necessidade de processar a topologia estrutural de *workflows* impulsionou a adoção de GNNs. O sistema *Decima* [Mao et al. 2019],

por exemplo, emprega GNNs utilizando *Reinforcement Learning* (RL) para otimizar o tempo de conclusão (*makespan*) de *jobs* em *clusters* de dados. Enquanto as abordagens baseadas em ML tradicional [Diel et al. 2024, Kraus et al. 2024] dependem da extração manual de *features* e soluções como o *Decima* focam no *makespan*, este artigo propõe uma interseção dessas áreas. Esta proposta reside no uso de uma *ResGCN* para a seleção dinâmica de políticas sob uma ótica multiobjetivo.

3. Método e materiais

Este artigo emprega uma rede convolucional de grafo como classificador binário com o objetivo de determinar a política adequada para um dado conjunto de tarefas.

3.1. Dados

Um *workflow* científico é uma representação gráfica de tarefas computacionais de múltiplas etapas [Deelman et al. 2019]. Neste estudo, esta representação descreve as dependências entre tarefas na forma de um DAG. No presente estudo, vale ressaltar a importância da heterogeneidade dos *workflows* para garantir a simulação de diferentes estados de rede. Alguns *workflows* têm alto grau de paralelismo, o que gera maior congestionamento de rede. Em contrapartida, outros *workflows* costumam ter cadeias sequenciais mais longas, favorecendo algoritmos que priorizam o caminho crítico. Outro fator importante é a quantidade de tarefas por *workflow*. A amostra final consiste em 300 *workflows*, variando entre 50 e 200 tarefas cada, baseados no projeto Pegasus [Deelman et al. 2019].

3.2. Algoritmos tutores

Para este estudo foram selecionados três algoritmos de escalonamento: *(i)* HEFT, que se destaca na minimização do *makespan* de um *workflow*; *(ii)* EFT, um algoritmo simples com uma abordagem gulosa; *(iii) Round-Robin*, um algoritmo que escalona tarefas em máquinas de forma circular. Cada *workflow* é processado pelos três escalonadores. Ao final, é gerada uma lista de informações que contêm os seguintes dados: *(i)* o *makespan* do *workflow*; *(ii) bounded slowdown*; *(iii) footprint*, que é o custo total de transferência de dados de uma tarefa; *(iv)* o balanceamento de carga, que é a distribuição das tarefas entre as máquinas.

3.3. Seleção do modelo de GNN

Modelos rasos de GNN, como a *Graph Convolutional Network* (GCN), sofrem com o problema do gradiente evanescente e do *oversmoothing*, quando a profundidade da rede é extensa. Para mitigar esses efeitos, o modelo ResGCN [Li et al. 2019] foi proposto. Sua arquitetura incorpora: *(i)* conexões residuais, evitando a degradação do sinal; *(ii) batch normalization*, que re-centraliza a distribuição dos dados a cada camada, prevenindo a saturação; e *(iii)* ativação ReLU.

3.4. Ambiente de simulação e topologia de rede

Para avaliar as políticas de escalonamento, foi desenvolvido um simulador em *Python* que implementa a topologia de rede *Dragonfly* [Kim et al. 2008] e a degradação de largura de banda. Uma característica importante da topologia é que a comunicação intra-grupo apresenta baixa latência e alta largura de banda. Por outro lado, a comunicação inter-grupos

utiliza conexões que podem sofrer rapidamente com gargalos quando a rede está congestionada. Como ResGCN recebe como entrada o fator de degradação da rede e o tamanho estimado das comunicações, ele se torna capaz de inferir quando as transferências globais de dados serão muito custosas.

3.5. Detalhes de implementação

Para cada *workflow* i, calculou-se uma pontuação $S_{i,j}$ para cada escalonador $j \in \{HEFT, EFT, RR\}$ baseada em uma soma ponderada de métricas normalizadas: $S_{i,j} = \alpha \cdot \hat{M}_{ij} + \beta \cdot \hat{F}_{ij} + \gamma \cdot \hat{S}_{ij} + \delta \cdot \hat{L}_{ij}$, sendo que $\hat{M}$, $\hat{F}$, $\hat{S}$ e $\hat{L}$ representam, respectivamente, os valores normalizados (escala 0-1) de *makespan*, *footprint*, *bounded slowdown* e desvio padrão de carga. Para diminuir o desbalanceamento de classes adotou-se a seguinte configuração de pesos: $(\alpha = 0.1, \beta = 0.4, \gamma = 0.3, \delta = 0.2)$. Esta configuração penaliza o uso excessivo da rede. O algoritmo *Round-Robin* não obteve desempenho superior em nenhum cenário, reduzindo o problema a uma classificação binária.

4. Resultados e discussões

Os experimentos foram conduzidos utilizando a arquitetura ResGCN proposta, treinada ao longo de 1.000 épocas com um *dataset* balanceado contendo 300 *workflows*. A validação cruzada foi realizada com uma divisão de 80% para treino e 20% para teste.

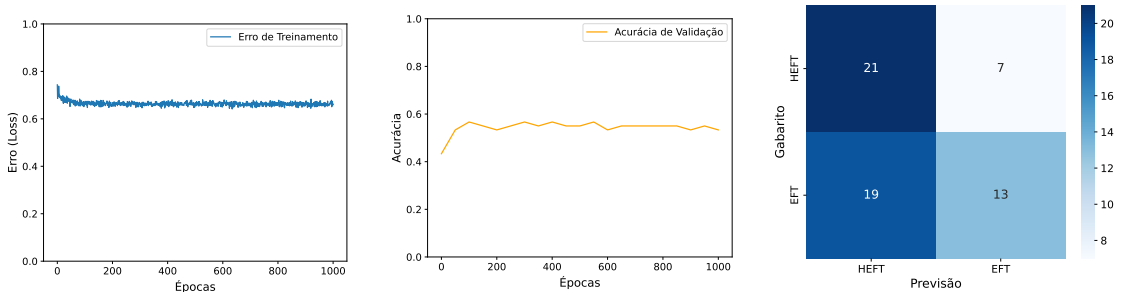
A Figura 1(a) ilustra a evolução da função de perda durante o treinamento. Observa-se que o modelo apresentou uma queda consistente e gradual do erro, estabilizando-se em torno de 0,66. Paralelamente, a Figura 1(b) demonstra a evolução da acurácia de validação. O modelo superou consistentemente a barreira dos 50% (acerto aleatório), atingindo picos de validação de 56,67%.

Para avaliar a capacidade de decisão do modelo, analisou-se a matriz de confusão no conjunto de teste, conforme apresentado na Figura 1(c). A matriz revela que a ResGCN não apresenta viés significativo para uma única classe.

Para avaliar a robustez da *ResGCN*, o desempenho do modelo preditivo (56,67% de acurácia) foi comparado com três linhas de base (*baselines*). A primeira, uma escolha aleatória, que atinge cerca de 50,33% de acurácia. As outras duas consistem em: uma abordagem 'Sempre HEFT', que obteve 56% de acerto; uma política 'Sempre EFT', que atingiu 44,4% de acerto. Embora o HEFT estático apresente uma taxa de sucesso razoável, ele o faz de forma inflexível. A GNN supera essa *baseline* ao atuar de forma preditiva e adaptável, prevendo o escalonador antes da execução. Isso evita os cenários de pior caso das heurísticas estáticas e contorna o *overhead* computacional que as tradicionais estratégias de escalonamento dinâmico costumam inserir no sistema.

5. Conclusão

Os resultados indicam que a utilização de Redes Neurais em Grafos é uma abordagem viável para o problema de seleção dinâmica de escalonadores. O modelo foi capaz de correlacionar a degradação da rede com a estrutura do grafo, superando métodos de escolha estática. Trabalhos futuros incluem a expansão do *dataset*, a implementação de múltiplos algoritmos tutores e a aplicação do modelo em ambientes reais de grade computacional.



(a) Evolução da função de perda. (b) Evolução da acurácia. (c) Matriz de confusão.

Figura 1. Resultados preliminares da avaliação do modelo ResGCN.

Agradecimentos: Este trabalho recebeu apoio do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), da Fundação de Amparo à Pesquisa e Inovação (FAPESC), desenvolvido no LabP2D.

Referências

Brucker, P. (2007). *Scheduling Algorithms*. Springer Berlin Heidelberg, Berlin, Heidelberg, 5th edition.

Deelman, E., Vahi, K., Juve, G., Rynge, M., Callaghan, S., Maechling, P. J., et al. (2019). The evolution of the Pegasus workflow management software. *Computing in Science & Engineering*, 21(4):22–36.

Diel, G., Kraus, A. E. N., and Koslovski, G. P. (2024). Usando aprendizado supervisionado para composição de políticas de escalonamento. In *Anais da XXIV Escola Regional de Alto Desempenho da Região Sul (ERAD-RS)*, pages 25–28. SBC.

Kim, J., Dally, W. J., Scott, S., and Abts, D. (2008). Technology-driven, highly-scalable dragonfly topology. In *2008 International Symposium on Computer Architecture*, pages 77–88. IEEE.

Kraus, A. E. N., Diel, G., and Koslovski, G. P. (2024). Predicting the efficiency of job scheduler actions. In *Anais da XXIV Escola Regional de Alto Desempenho da Região Sul (ERAD-RS)*, pages 33–36. SBC.

Li, G., Muller, M., Thabet, A., and Ghanem, B. (2019). DeepGCNs: Can GCNs go as deep as CNNs? In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9267–9276.

Mao, H., Schwarzkopf, M., Venkatakrishnan, S. B., Meng, Z., and Alizadeh, M. (2019). Learning scheduling algorithms for data processing clusters. In *Proceedings of the ACM Special Interest Group on Data Communication (SIGCOMM)*, pages 270–288.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. (2009). The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80.

Topcuoglu, H., Hariri, S., and Wu, M.-Y. (2002). Performance-effective and low-complexity task scheduling for heterogeneous computing. *IEEE Transactions on parallel and distributed systems*, 13(3):260–274.