Aplicação de Algoritmos Genéticos Paralelos para Formação de Grupos em Ambientes E-learning

Nelson Luiz Silva Ferreira[1], Rafael H. Vincence[1], Avanilde Kemczinski[2], Rafael Stubs Parpinelli[2]

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

[2]Programa de Pós Graduação em Computação Aplicada
Universidade do Estado de Santa Catarina (UDESC) – Joinville, SC – Brasil

{nersosfs,rafaelvincence}@gmail.com[1]

{avanilde.kemczinski,rafael.parpinelli}@udesc.br[2]

Resumo. *Este trabalho descreve a paralelização de um Algoritmo Genético, utilizado para a composição de grupos heterogêneos de estudantes. Para analisar o speed-up obtido em relação ao algoritmo sequencial, técnicas de processamento paralelo em um ambiente com memória compartilhada foram aplicadas em uma arquitetura Master-Slave. A análise experimental discute e compara o speed-up obtido em diferentes cenários de execução.*

1. Introdução

A formação de grupos de alunos é uma tarefa complexa que deve considerar uma efetiva interação entre os participantes de um grupo para se chegar a um bem comum [Citadin et al. 2014]. Para tentar resolver esse problema NP-Hard, [Citadin 2015] aplicou um Algoritmo Genético (AG) para formação de grupos heterogêneos, levando em conta duas características dos alunos (conhecimento(notas) e interação), a fim de aproximar uma solução para o problema em questão. Os AGs foram um dos primeiros algoritmos com inspiração natural, tendo como base a teoria da evolução das espécies proposta por Darwin, onde os indivíduos mais bem adaptados ao ambiente tem maiores chances de sobreviver e repassar seu material genético para gerações futuras [Holland 1975]. Como uma das contribuições do trabalho de [Citadin 2015] é utilizar o AG em ambientes *e-learning*, foi verificado que o algoritmo desenvolvido possui alguns gargalos que poderiam ser otimizados. Sendo assim, este trabalho tem como objetivo paralelizar o AG desenvolvido por [Citadin 2015], utilizando algumas abordagens de paralelismo. Para realizar a paralelização do AG foi feito um perfil da implementação sequencial (através da ferramenta *gprof*). Nesse processo, identificaram-se os gargalos de processamento existentes na implementação e constatou-se que a avaliação do *fitness* dos indivíduos da população é a responsável por grande parte do tempo de processamento.

2. Descrição do Problema

O problema abordado considera um grupo de alunos, onde o tutor deseja dividi-los em grupos de mesmo número de componentes. O objetivo é gerar grupos que sejam heterogêneos (alunos que possuem características diferentes) e com o maior número de

interações entre os estudantes do mesmo grupo. Além disso, cada estudante deve pertencer a apenas um único grupo. Cada estudante possui uma média de suas avaliações (conhecimento) e também um número de interações efetuadas com os demais colegas de grupo. Desta forma, objetivando maximizar a heterogeneidade dos grupos, a Equação 1 foi desenvolvida por [Citadin 2015] e considera as interações e conhecimentos dos alunos. Essa equação é chamada de função objetivo e o valor encontrado em sua aplicação é conhecido como *fitness*.

$$Max f(x) = \omega 1 \sum_{i=1}^{g} f(het\_conhect_i) + \omega 2 \sum_{i=1}^{g} f(int_i) + (1 - \sum_{i=1}^{g} f(inter\_homo_i)) \quad (1)$$

Nesta Equação1, $\omega 1$ e $\omega 2$ são os pesos para os conhecimentos e interações, respectivamente; *i* indica o índice dos grupos; *g* representa a quantidade de grupos; f(het_conhect$_i$) é a função de cálculo da heterogeneidade dos conhecimentos; f(int$_i$) é a função de cálculo das interações; f(inter-homo$_i$) é a função de cálculo da inter-homogeneidade.

3. Algoritmo Genético e Paralelismo

O AG é um algoritmo de busca heurística, populacional, que se aplica a problemas complexos. A cada geração do algoritmo é realizada a avaliação de *fitness* da população para quantificar o quão adaptados estão os indivíduos em relação a função objetivo. No contexto de aplicações paralelas, os AGs são altamente indicados por serem algoritmos populacionais onde cada indivíduo da população representa uma possível solução para o problema e sua avaliação é feita de maneira independente [Mole et al. 2002]. Vale ressaltar que neste trabalho, a avaliação de *fitness* é onde se encontra o gargalo de processamento.

Alguns modelos de AGs paralelos podem ser identificados na literatura: *Master-Slave*, Ilha e Hierárquico, são alguns deles. Neste trabalho, o modelo *Master-Slave* foi escolhido porque se apresenta como um método eficiente de paralelização quando o gargalo de processamento é a avaliação da função de *fitness*. Logo, foram desenvolvidas três implementações do modelo *Master-Slave* a fim de verificar o *speed-up* do AG em relação a aplicação sequencial. Todas implementações fazem uso de *threads* porém com estratégias de programação (ferramental) diferentes. As ferramentas de implementação utilizadas foram: *threads* puro (T-MS) (biblioteca *pthreads*), *Workpool* (W-MS) (biblioteca *pthreads*) e API *OpenMP* (O-MS).

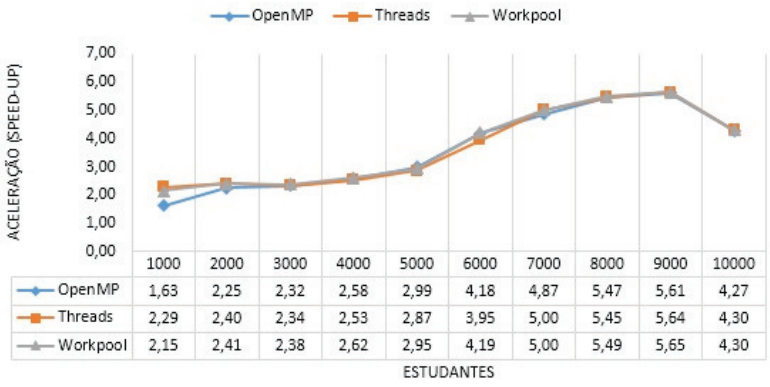
A codificação dos indivíduos se dá por um vetor de inteiros de tamanho *n*, com *n* sendo o número de estudantes. Nesta codificação, permutações destes inteiros indicam quais alunos irão pertencer a quais grupos. Sendo assim, cada indivíduo da população do AG é uma possível solução para o problema. Neste trabalho, os objetos a serem paralelizados são os indivíduos que constituem a população que, por sua vez, será avaliada pela Equação 1.

4. Experimentos e Resultados

Os experimentos foram realizados em máquinas virtuais criadas na nuvem da UDESC com processador Intel Xeon E312 *family* 6, 4 GB de RAM e sistema operacional Ubuntu

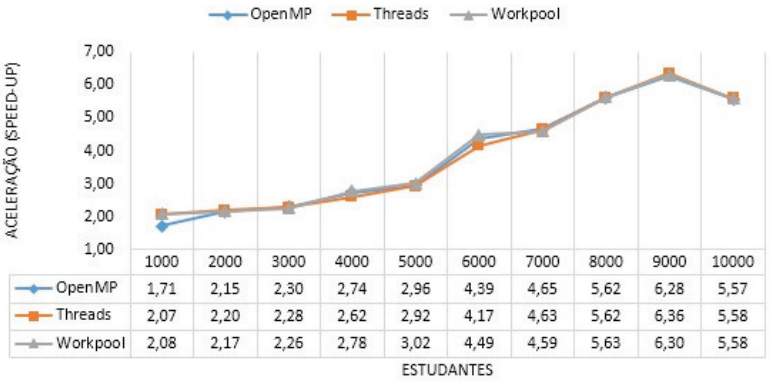
14.04 LTS. A linguagem de programação utilizada foi C ANSI. Como o hardware utilizado possui apenas 4 núcleos, os testes foram restritos a 3 *threads* a fim de não saturar o processador. Para comparação dos resultados é calculado o *speed-up* que é obtido através da divisão do tempo do algoritmo sequencial pelo tempo do algoritmo paralelo. Foram realizados testes para cada uma das abordagens aplicadas (T-MS, W-MS e O-MS) com n variando de 1.000 à 10.000 alunos. Em todos os experimentos a população é composta por 100 indivíduos.

Três casos foram levados em conta ao realizar os testes: (i) pesos iguais (conhecimentos e interações); (ii) peso do conhecimento maior; (iii) peso da interação maior. Para cada experimento, foram realizadas 30 execuções, sendo que os gráficos apresentam o *speed-up* obtido através da média dessas execuções, para cada abordagem implementada. Os dados utilizados para representar tanto o conhecimento quanto a interação entre os alunos foram gerados aleatoriamente, tornando o cenário mais próximo a um caso real. As Figuras 1, 2 e 3 apresentam os *speed-ups* médios obtidos para pesos iguais, peso do conhecimento maior e peso da interação maior, respectivamente.



	1000	2000	3000	4000	5000	6000	7000	8000	9000	10000
OpenMP	1,63	2,25	2,32	2,58	2,99	4,18	4,87	5,47	5,61	4,27
Threads	2,29	2,40	2,34	2,53	2,87	3,95	5,00	5,45	5,64	4,30
Workpool	2,15	2,41	2,38	2,62	2,95	4,19	5,00	5,49	5,65	4,30

Figura 1. Caso(i): *Speed-up* obtido para os pesos iguais.

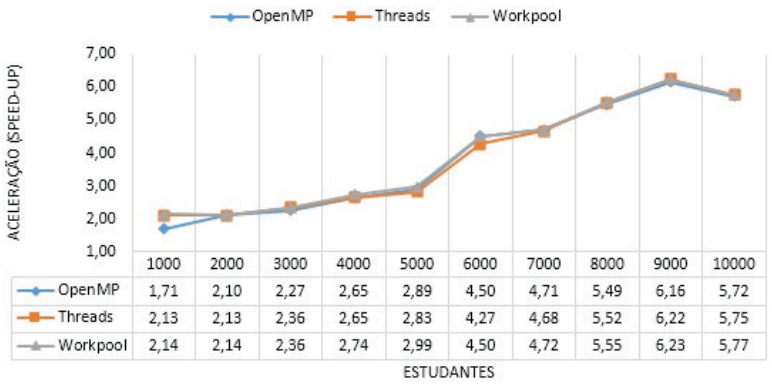


	1000	2000	3000	4000	5000	6000	7000	8000	9000	10000
OpenMP	1,71	2,15	2,30	2,74	2,96	4,39	4,65	5,62	6,28	5,57
Threads	2,07	2,20	2,28	2,62	2,92	4,17	4,63	5,62	6,36	5,58
Workpool	2,08	2,17	2,26	2,78	3,02	4,49	4,59	5,63	6,30	5,58

Figura 2. Caso(ii): *Speed-up* obtido para o peso do conhecimento maior.

5. Análises e Conclusão

No contexto do presente trabalho, o AG foi aplicado no agrupamento de alunos. Após ser estudado, o AG foi paralelizado utilizando as seguintes abordagens do modelo *Master-Slave*: API *OpenMP*, *Workpool* e *Threads*. O modelo *Master-Slave* tem como proposta



	1000	2000	3000	4000	5000	6000	7000	8000	9000	10000
OpenMP	1,71	2,10	2,27	2,65	2,89	4,50	4,71	5,49	6,16	5,72
Threads	2,13	2,13	2,36	2,65	2,83	4,27	4,68	5,52	6,22	5,75
Workpool	2,14	2,14	2,36	2,74	2,99	4,50	4,72	5,55	6,23	5,77

Figura 3. Caso(iii): *Speed-up* obtido para o peso da interação maior.

dividir um processo principal em vários sub-processos escravos, cada um executando em um processador. Cada abordagem de desenvolvimento possui suas características a fim de otimizar o tempo de processamento do algoritmo.

A partir dos resultados alcançados, identificou-se que a paralelização utilizando o modelo *Master-Slave* é uma forma eficiente de alcançar *speed-up*, possibilitando um *speed-up* máximo médio para o caso(i) de 5.65 vezes, para o caso (ii) de 6.36 vezes e para o caso (iii) de 6.23 vezes em comparação ao algoritmo sequencial. Em vários momentos pôde-se observar um *speed-up* superlinear, onde o *speed-up* com P núcleos é maior que o número de núcleos. Isto se deve a dois fatores onde o primeiro refere-se a várias buscas sendo realizadas ao mesmo tempo (processos *slaves*) e o segundo fator sendo a utilização mais eficiente dos recursos computacionais com multi-processadores [P. Agrawal and Sienicki 1995]. Das abordagens utilizadas (*OpenMP*, *Threads* e *Workpool*), apesar das acelerações obtidas serem semelhantes, a abordagem que provou ser a mais eficiente neste caso, foi a abordagem *OpenMP*. Isto se deve ao fato desta abordagem não requerer modificações drásticas no código, sendo de simples implementação em comparação às outras abordagens utilizadas neste trabalho. A API *OpenMP* utiliza um conjunto de diretivas que facilita a criação e destruição das *threads*, possuindo assim, um alto nível de abstração do modelo paralelo.

Referências

Citadin, J. R. (2015). Um algoritmo genético para formação de grupos heterogêneos na aprendizagem colaborativa. Dissertação (Mestrado em Computação Aplicada), UDESC.

Citadin, J. R., Kemczinski, A., and de Matos, A. V. (2014). Colaboração em massive open online courses (moocs). *Anais do Computer on the Beach*, pages 233–242.

Holland, J. H. (1975). *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence.* U Michigan Press.

Mole, V. L. D. et al. (2002). Algoritmos genéticos: uma abordagem paralela baseada em populações cooperativas. Dissertação (Mestrado em Ciência da Computação), UFSC.

P. Agrawal, V.D. Agrawal, M. B. and Sienicki, J. (1995). Superlinear speedup in multiprocessing environment. In *First International Workship on Parallel Processing*, pages 261–265.