Avaliação da Eficiência da Paralelização com `Sections` OpenMP em um Algoritmo Genético através de Rastros de Execução

Gabriella Lopes Andrade[1], **Márcia Cristina Cera**[1]

[1]Ciência da Computação – Universidade Federal do Pampa (UNIPAMPA)
Campus Alegrete – Av. Tiarajú, 810, Bairro Ibirapuitã, CEP: 97546-550 – Alegrete – RS

gabie.lop.s@gmail.com, marciacera@unipampa.edu.br

Resumo. *O rastreamento de execução permite identificar o impacto do uso de diretivas no desenvolvimento de aplicações paralelas. O foco deste trabalho é a diretiva sections do OpenMP, usada na paralelização de um Algoritmo Genético onde há carga de trabalho para apenas 2 threads. Nossos resultados mostram que, embora o uso de sections aumente o tempo de sincronização, ele melhora o desempenho em até 7% quando comparado com a ausência desta primitiva.*

1. Introdução

Esse trabalho realiza a análise de um Algoritmo Genético (AG) aplicado ao Problema de Roteamento de Veículos (PRV) desenvolvido por [Gressler and Cera 2014], através de seus rastros de execução [Schnorr 2014]. O PRV é um problema de otimização combinatória que consiste em rotear veículos com uma certa capacidade para atender as requisições de um grupo de cidades. A solução do PRV compreende um conjunto de rotas capazes de satisfazer a demanda de todas as cidades e cujo somatório seja mínimo [Rego and Alidaee 2006]. Entre os métodos propostos para solucionar o PRV, o AG foi o escolhido por [Gressler and Cera 2014] por encontrar boas soluções quando aplicado a problemas que não possuem uma técnica especializada para resolvê-los.

Buscando reduzir o tempo de computação do AG, [Gressler and Cera 2014] realizaram a paralelização do mesmo utilizando a interface de programação de aplicações (*Application Program Interface* - API) *Open Multi Processing* (OpenMP) [Chapman et al. 2008], com `sections` e `parallel for`. A região paralelizada com `sections` possui dois blocos de código independentes, onde cada bloco fica dentro de uma `section`. Logo, essa região será sempre executada por 2 *threads*. O objetivo deste trabalho é avaliar a eficiência do uso de `sections` em arquiteturas que suportem mais de 2 *threads*.

2. Algoritmo Genético

O AG é um algoritmo baseado na teoria da evolução natural de Charles Darwin [Linden 2008]. No AG cada possível solução do PRV é codificada em uma estrutura de dados, chamada de cromossomo ou indivíduo, que compõem a população inicial. Essa população é avaliada e aplicada ao processo evolutivo, que envolve a seleção de quais indivíduos passarão pelo processo reprodutório, aplicação dos operadores genéticos de cruzamento e mutação, e avaliação dos indivíduos gerados. A cada ciclo de evolução uma nova população é gerada, a qual substitui a antiga. Após realizar vários ciclos de evolução

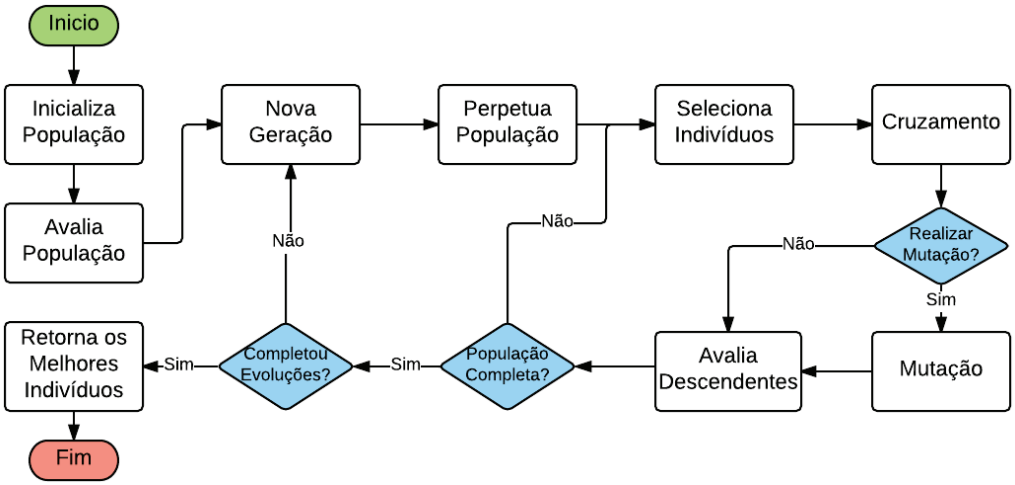


Figura 1. Fluxograma do Algoritmo Genético.

a população final deverá conter os indivíduos mais aptos [Linden 2008]. A Figura 1 apresenta um fluxograma com o funcionamento do AG.

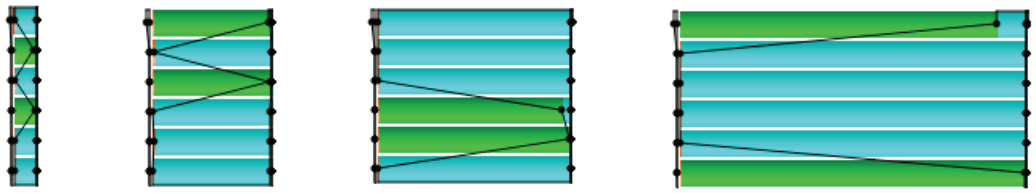
A partir do perfil da execução gerado pela ferramenta `gprof` [Gressler and Cera 2014] identificaram duas regiões de maior processamento no AG. A primeira região possui dois blocos de código responsáveis por copiar uma porcentagem de indivíduos para a nova geração, representados na Figura 1 por "**Perpetua População**". Para paralelizar essa região foi utilizada a diretiva `#pragma omp sections`, pois esses blocos podem ser executados em paralelo e não possuem tarefas iterativas. A segunda região possui um laço que controla a geração de uma nova população, representado na Figura 1 por "**População Completa?**". Logo, foi utilizada a diretiva `#pragma omp for`, dessa forma as iterações do laço serão distribuídas entre as *threads*. Neste trabalho, nossa análise estará focada no desempenho da paralelização de primeira região (não iterativa), permanecendo a análise da segunda região para trabalhos futuros.

3. Ambiente de Execução

As instâncias do PRV utilizadas pertencem ao *Benchmark* de [Christofides et al. 1979] e possuem restrição quanto à capacidade de transporte do veículo. Foram utilizadas instâncias com 50, 100, 120 e 150 cidades. Nossos testes foram executados na *Scherm Workstation* do Laboratório de Estudos Avançados em Computação (LEA) da UNIPAMPA Campus Alegrete, que possui um processador Intel® Xeon® E5-2603 v3, com frequência de 1.9 GHZ e 6 núcleos físicos. O sistema operacional utilizado foi o Ubuntu 16.04 LTS com compilador GCC em sua versão 5.31. Executamos com 2, 3, 4, 5 e 6 *threads*, além da versão sequencial. Para coletar os rastros de execução usou-se a ferramenta Score-P[1] na versão 2.0.2 e para visualizar os rastros usou-se a ferramenta Vampir[2] na versão de demonstração 9.

Baseando-se em [Gressler and Cera 2014], identificou-se os parâmetros do AG que levaram a uma melhora na qualidade das soluções, sendo N o número de cidades: **Tamanho da População:** $N \times 10$; **Número de Evoluções:** $N \times N \times 10$; **Técnica de Cruzamento:** 1 ponto, onde escolhe-se um ponto em dois cromossomos e intercala-se as

[1]Disponível em `http://www.vi-hps.org/projects/score-p/`
[2]Disponível em `https://www.vampir.eu/`



(a) c50 (b) c100 (c) c120 (d) c150

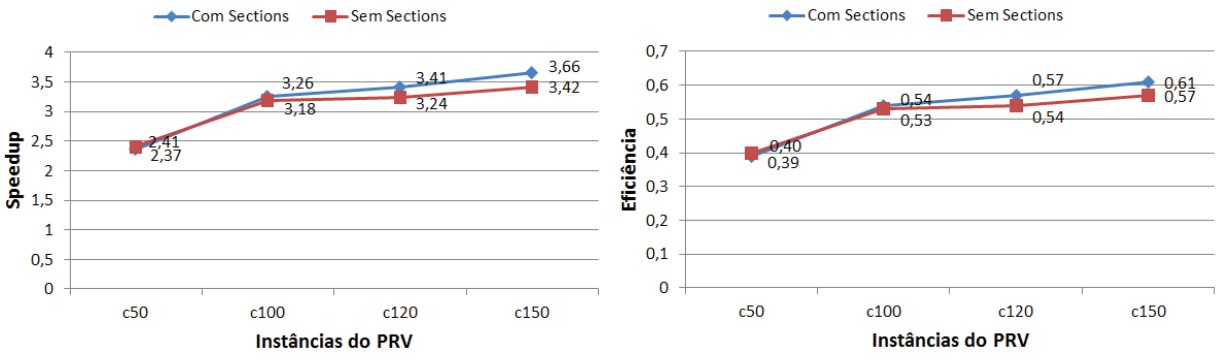
Figura 2. *Speedup* e Eficiência Obtidos pela Execução do AG Paralelo Com o Uso de `sections` Quando se Executa com 6 *Threads*.

seções de gene resultantes gerando dois novos indivíduos; **Probabilidade de Mutação:** 20%; **Taxa de Mutação:** variável entre 4 a 10%; **Técnica de Mutação:** Randômica, a qual sorteia qual técnica de mutação utilizar entre: troca (troca um gene de lugar), troca bloco (troca um bloco de genes), inversão (seleciona e inverte uma seção de genes) e insere (remove uma seção de genes e a insere em outro lugar do mesmo cromossomo).

4. Resultados

No OpenMP, ao término da execução das regiões paralelas ocorre uma sincronização implícita, a partir da qual o programa continua sua execução apenas com a *thread* mestre. O tempo de espera em sincronização representa o tempo que uma *thread* que já terminou sua computação espera até que todas as demais também. Os resultados de trabalhos anteriores [Andrade and Cera 2016b] mostram que esse tempo aumenta conforme se aumenta o número de *threads*. Embora a execução com 6 *threads* forneça o melhor desempenho [Andrade and Cera 2016a], ela acumula um tempo de sincronização até 3 vezes maior quando comparado com a execução com apenas 2 *threads* [Andrade and Cera 2016b]. Isto é claramente ilustrado na Figura 2, onde as `section` estão representadas em verde e o tempo de espera em azul, vemos que com 6 *threads* teremos 4 delas ociosas.

O tempo de espera em sincronização quando se utiliza `section` é, em geral, até 2 vezes maior do que quando não se utiliza [Andrade and Cera 2016b]. A Figura 3 apresenta os gráficos do *Speedup* (Figura 3(a)) e Eficiência (Figura 3(b)) obtidos pelo AG paralelo ao executar com e sem o uso de `sections` com 6 *threads*. Esses dados representam a média de tempo de 30 execuções, sendo que o desvio padrão ficou sempre abaixo de 1% dessa média. Observando a Figura 3(a) vemos que a instância c50 é a única



(a) *Speedup*. (b) Eficiência.

Figura 3. *Speedup* e Eficiência Máximos Obtidos Com e Sem o Uso de `sections` para as Instâncias Alvo Quando se Executa com 6 *Threads*.

em que o desempenho não melhora com o uso de `sections` (a diferença de *Speedup* está dentro da variação das coletas). Isso ocorre devido a sua pequena carga computacional (tempo de execução sequencial de 6,78 segundos). Para as demais instâncias o uso de `sections` melhora o desempenho do AG, sendo 2,52% maior para a instância c100, 5,25% maior para a c120 e 7,00% para a c150. A carga computacional destas instâncias é aproximadamente 61 vezes maior para a instância c100, 177 vezes para a c120 e 715 vezes para a c150, do que a carga da instância c50.

Observando a Figura 3(b) nota-se que a eficiência obtida também apresenta o mesmo comportamento do *Speedup*. Sendo que para a instância c50 a eficiência é 2,56% maior sem o uso de `sections` do que quando não se utiliza. Para as outras instâncias o uso de `sections` aumenta a eficiência da aplicação em 1,89% para a instância c100, 5,55% para a c120 e 7,02% para a c150. Logo, embora o uso de `sections` melhora o tempo de sincronização, seu uso aumenta o desempenho do AG paralelo conforme se aumenta a carga computacional das instâncias do PRV utilizadas.

5. Conclusão

Nossa pesquisa indicou na implementação avaliada do AG, quando se executa em uma arquitetura com mais de 2 núcleos, a paralelização com `sections` leva a um aumento do tempo de espera em sincronização de até 7%. Mesmo com o aumento do tempo de espera, o uso de `sections` consegue melhorar o desempenho da aplicação em até 7%. Logo, embora haja um aumento no tempo de sincronização, este aumento tem um impacto menor no desempenho final da aplicação do que a execução sequencial das regiões paralelizadas com `sections`.

Referências

Andrade, G. L. and Cera, M. C. (2016a). Avaliando a paralelização de um algoritmo genético com OpenMP. In *Anais do WSCAD-WIC*, pages 68–73. SBC, Aracaju, SE.

Andrade, G. L. and Cera, M. C. (2016b). Avaliando o uso de sections OpenMP na paralelização de um algoritmo genético. In *Anais do SIEPE*, number 2. UNIPAMPA, Uruguaiana, RS.

Chapman, B., Jost, G., and Van DeR Pas, R. (2008). *Using OpenMP: portable shared memory parallel programming*. MIT Press, Cambridge, MA, USA.

Christofides, N., Mingozzi, A., Toth, P., and Sandi, C. (1979). *Combinatorial Optimization*. John Wiley & Sons, Chichester, UK.

Gressler, H. d. O. and Cera, M. C. (2014). O impacto da paralelização com OpenMP no desempenho e na qualidade das soluções de um algoritmo genético. *Revista Brasileira de Computação Aplicada*, pages 35–47.

Linden, R. (2008). *Algoritmos genéticos: uma importante ferramenta da inteligência computacional*. Brasport, Rio de Janeiro, RJ, 2a edition.

Rego, C. and Alidaee, B. (2006). *Metaheuristic Optimization via Memory and Evolution: Tabu Search and Scatter Search*. Operations Research/Computer Science Interfaces Series. Springer, USA.

Schnorr, L. M. (2014). Análise de desempenho de programas paralelos. In *Anais da ERAD/RS 2014*, pages 57–81. SBC, Alegrete, RS.