Algoritmo Genéticos Paralelo: uma abordagem hierárquica

Derik Evangelista Rodrigues da Silva[1], **Raphael Henrique Ferreira de Andrade**[1], **Eduardo Spinosa**[1]

[1]Departamento de Informática – Universidade Federal do Paraná (UFPR)
Caixa Postal 19081 – 81531-980 – Curitiba – PR – Brasil

{dersilva, rhfandrade, spinosa}@inf.ufpr.br

Abstract. *@TODO Abstract*

Resumo. *@TODO Resumo*

1. Introdução

Algoritmo Genetico (*Genetic Algorithm* – GA) são algoritmos de busca inspirados no processo de evolução e seleção natural [Goldberg 1989] e tem tido grande sucesso em problemas de busca e de otimização, principalmente quando o espaços de busca é grande, complexo ou pouco conhecido, onde métodos de buscas convencionais (enumerativos, heurísticos, ...) não são apropriados [Herrera et al. 1998].

Um GA sequencial inicia-se gerando um conjunto de indivíduos para formar uma população inicial. Cada indivíduo representa uma possível solução do problema. Usando uma função de avaliação (chamada de função *fitness*), mede-se a qualidade de cada indivíduo desta população. O cálculo do *fitness* é, geralmente, o processo mais custoso de um GA [Nowostawski and Poli 1999]. Seleciona-se aleatoriamente, então, um subconjunto de indivíduos desta população e neste é aplicado operadores estocásticos de seleção, mutação e cruzamento. Por fim, os indivíduos menos adaptados (ou seja, com pior *fitness*) são descartados, para dar lugar a indivíduos mais bem adaptados.

Apesar do sucesso em muitas aplicações em diferentes domínios, existem, de acordo com [Nowostawski and Poli 1999], algums problemas que podem ser resolvidos com o uso de um Algoritmo Genético Paralelo (*Parallel GA* – PGA):

- Para alguns tipos de problemas, o tamanho da população precisa ser muito grande, requerendo, consequentemente, uma grande quantidade de memória, podendo impossibilitar a execução eficiente em uma única máquina.
- O cálculo do *fitness* consome muito tempo. Há registros na literatura de uma única execução consumindo mais de 1 ano de CPU.
- GA's sequencias podem ficar presos em regiões sub-ótimas, ficando impossibilitados de encontrar uma melhor solução. PGA's podem buscar em multiplos subespaços de busca em paralelo, e tem menos chance de ficar preso em regiões sub-ótimas.

O motivo mais importante para se estudar PGAs, ainda segundo [Nowostawski and Poli 1999], é que em muitos casos eles tem uma melhor performance do que os sequenciais, mesmo quando o paralelismo é simulado em uma máquina convencional. Segundo [Alba and Troya 1999], eles atingem o objetivo ideal de se construir um algoritmo paralelo, onde o todo é melhor do que a soma das partes.

Este trabalho tem como objetivo comparar três tipos de arquiteturas de PGAs: múltiplas populações, arquitetura mestre-escravo e um híbrido de ambas, ou seja, uma combinação de múltiplas populaçõoes com mestre-escravo, aplicadas a otimização de funções. Além disso, compararemos os resultados com um GA sequencial convencional.

2. Revisão de literatura

O Algoritmo Genético foi desenvolvido por John Holland na Universidade de Michigan, em 1970 [Holland 1975], inspirado no processo de seleção natural e evolução, e apresenta uma alternativa as técnicas clássicas de otimização, usando buscas aleatórias dirigidas para localizar soluções ótimas em espaços de buscas complexos [Srinivas and Patnaik June]. O objetivo original de Holland não era construir um algoritmo que resolvesse um problema específico, mas formalizar o estudo do fenômeno de adaptação da mesma forma que este acontece na natureza e desenvolver mecanismos de importar este comportamento em sistemas computacionais [Michell 1998].

Tendo sua inspiração na biologia, alguns termos desta área são usados para descrever o GA [Luke 2009]:

Indivíduo Solução candidata;
População Conjunto de indivíduos;
Filhos e Pais Um filho é uma cópia perturbada de seu pai (ambos são indivíduos);
***Fitness* (Adaptabilidade)** Medida de qualidade de dada solução;
Função de *Fitness* Função de qualidade.
Seleção Escolha de indivíduos, baseado em seu *fitness*;
Mutação Pequena perturbação na solução;
Recombinação / Cruzamento Grande perturbação na estrutura do indivíduo. Geralmente gera dois filhos recombinando a estrutura de seus pais.
Genoma / Genótipo Estrutura do indivíduo;
Geração População gerada em cada ciclo do algoritmo, que envolve as funções e transformações previamente definidas.

O algoritmo apresentado em [Holland 1975] é usualmente chamado de canônico [Yang 2002] ou Algoritmo Genético Simples (SGA)[Srinivas and Patnaik June] e trabalha, essencialmente, com indivíduos sendo um vetor de bits, ou seja, a solução é codificada em termos de 0 e 1. Como método de seleção, o SGA usa o esquema de *roleta*, onde um determinado indivíduo tem mais chance de ser escolhido para procriar dependendo de seu *fitness* calculado.

Algumas variações foram apresentadas, como a inclusão de elitismo [De Jong 1975], que consiste em manter um número de indivíduos com melhor *fitness* de uma geração para outra, o *Steady-State Genetic Algorithm* [Whitley et al. 1988], que atualiza a população assim que os filhos são gerados, descartando-os ou inserindo-os no lugar de alguns de indivíduos piores da população e o *Tree-Style Genetic Programming Pipeline*, que utiliza uma forma diferente de procriação: com 90% de probabilidade, dois pais serão selecionados e será efetuado o cruzamento convencional e, por outro lado, com 10% de probabilidade, será selecionado apenas um pai, que será copiado para a nova população. Existem versões, também, que se preocupam em adaptar as taxas de cruzamento e mutação em tempo de execução e abordagens híbridas, como efetuar

uma busca local em cada indivíduo, usando outro algoritmo [Bersini and Renders 1994] [Katare et al. 2004].

@TODO REVISÃO AG PARALELO: HIERARQUIAS

3. Avaliação Experimental

Esboço: Comparação entre ag tradicional, ag paralelo com multiplas populações, ag paralelo master slave e ag hibrido das duas anteriores.

Problema: funções de de jong.

Modelagem: Vetor de números reais. Cruzamento convencional. Mutação: pequena soma/subtração de número entre 0.01 e 0.05. Seleção torneio.

Análise: tabelas: tempo gasto, resultado médio em 10 execuções para cada um dos ag's, ótimo conhecido. Gráfico: fitness x geração, testes: kruskal-wallis e/ou analise de covariancia (ancova).

Referências

Alba, E. and Troya, J. M. (1999). A survey of parallel distributed genetic algorithms. *Complex.*, 4(4):31–52.

Bersini, H. and Renders, J.-M. (1994). Hybridizing genetic algorithms with hill-climbing methods for global optimization: Two possible ways. In *International Conference on Evolutionary Computation*, pages 312–317.

De Jong, K. A. (1975). *An analysis of the behavior of a class of genetic adaptive systems.* PhD thesis, Ann Arbor, MI, USA. AAI7609381.

Goldberg, D. (1989). *Genetic algorithms in search, optimization, and machine learning*. Artificial Intelligence. Addison-Wesley.

Herrera, F., Lozano, M., and Verdegay, J. L. (1998). Tackling real-coded genetic algorithms: Operators and tools for behavioural analysis. *Artif. Intell. Rev.*, 12(4):265–319.

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

Katare, S., Bhan, A., Caruthers, J. M., Delgass, W. N., and Venkatasubramanian, V. (2004). A hybrid genetic algorithm for efficient parameter estimation of large kinetic models. *Computers & Chemical Engineering*, 28(12):2569 – 2581.

Luke, S. (2009). *Essentials of Metaheuristics*. Lulu. Disponível em: <http://cs.gmu.edu/∼sean/book/metaheuristics/>. Acesso em: 27/02/2013.

Michell, M. (1998). *An Introduction to Genetic Algorithms*. Complex Adaptive Systems Series. Mit Press.

Nowostawski, M. and Poli, R. (1999). Parallel genetic algorithm taxonomy. In *Proceedings of the Third International*, pages 88–92. IEEE.

Srinivas, M. and Patnaik, L. (June). Genetic algorithms: a survey. *Computer*, 27(6):17–26.

Whitley, D., Kauth, J., and of Computer Science, C. S. U. D. (1988). *GENITOR: A Different Genetic Algorithm*. Technical report (Colorado State University. Dept. of Computer Science). Colorado State University, Department of Computer Science.

Yang, S. (2002). Genetic algorithms based on primal-dual chromosomes for royal road functions.