X Congresso Brasileiro de Inteligência Computacional (CBIC'2011), 8 a 11 de Novembro de 2011, Fortaleza, Ceará

ANÁLISE DE FATORES DETERMINANTES NO DESEMPENHO DOS ALGORITMOS GENÉTICOS PARALELOS EM PROCESSADORES MULTINÚCLEOS

M. S. Pais[1,2], I. S. Peretta[2], G. F. M. Lima[2], J.Tavares [2], H. Rocha[2], K. Yamanaka[2]

[1]Instituto Federal Goiano - Campus Urutaí, GO, Brazil

[2]Faculdade de Engenharia Elétrica de Uberlândia - Uberlândia, MG, Brazil

monicaspais@gmail.com, {iperetta,gersonlima}@ieee.org, josycbelo@gmail.com, hxr@ifg.edu.br, keiji@ufu.br

Resumo – Processadores multinúcleos estão presentes e disponíveis em grande variedade de arquiteturas computacionais (hardware). A sua capacidade de processamento baseia-se no trabalho paralelo executado pelos seus vários núcleos. Aplicações sequenciais não tem mais como garantir a melhoria do desempenho através da simples atualização do processador, pois os novos processadores são multinúcleos e a sua maior capacidade de processamento está fundamentada no trabalho em paralelo dos seus núcleos. Desta forma, as aplicações necessitam maximizar a paralelização do seu código. Os Algoritmos Genéticos (AGs) canônicos possuem um paralelismo inerente, uma vez que é possível identificar facilmente partes do algoritmo que são independentes. Os Algoritmos Genéticos Paralelos (AGPs) estavam restritos à execução em supercomputadores, máquinas caras e pouco acessíveis. Com os processadores multinúcleos largamente disponíveis, podemos fazer uso dos AGPs com maior facilidade. Porém, existem vários parâmetros configuráveis que podem influenciar no seu desempenho. O planejamento experimental envolve investigações preliminares para a determinação dos fatores que influenciam ou não o resultado desejado. No caso deste trabalho, o resultado almejado é o melhor desempenho de um AGP em um processador multinúcleo. São apresentados os resultados da análise dos fatores que influenciam o desempenho da implementação de um AGP. Os testes foram realizados em um sistema com processador multinúcleo.

Palavras-chave – Algoritmos Evolucionários Paralelos, Algoritmos Genéticos Paralelos, Topologias de Migração, Programação Paralela, Processadores Multinúcleos, Planejamento Experimental, Fatorial Completo.

Abstract – Multicore processors are available in all kinds of architecture types. Their processing capacity is on the parallel work of their multicores. Sequential applications can not get for grant an improve on their performance by just upgrading to a new processor. Software needs to change to parallel programming. Genetic algorithms are embarrassing parallel applications. It is easy to find independent parts of code that can be executed in parallel, e.g. fitness avaliation. If once the parallel programming were restrict to supercomputers and distributed computing, today it can be executed in commercial off the shelf personal computers. Parallel Genetic Algorithms (PGA) can improve the execution performance and the quality of solution of conventional Genetic Algorithms (GAs), and multicore processors can execute PGAs well. This work presents results of an experimental factorial design for tunning parameters of PGAs. The factorial design assists in finding which parameters have significant effect on algorithms performance in a multicore processor.

Keywords – Parallel Evolutionary Algorithms, Parallel Genetic Algorithms, Migration Toplogies, Multicore Systems, Parallel Programming, Multicore processors, Experimental Design, Factorial Design.

1. INTRODUÇÃO

A indústria de microprocessadores manteve o ritmo de crescimento exponencial no aumento da densidade de integração de transistores em áreas cada vez menores, possibilitando a redução no tamanho físico e a melhoria do desempenho dos computadores por vários anos. Para manter essa evolução, a indústria dos microprocessadores além de aumentar a densidade, aumentou a frequência do *clock*. Essa estratégia implica em dois grandes problemas: o aumento do consumo de energia e a necessidade de dissipação do calor gerado. Portanto, esse tipo de estratégia já não é mais viável. A indústria de microprocessadores superou esses problemas com a criação de processadores multinúcleos (*multicore*). Este tipo de processador possui várias vantagens dentre as quais podemos destacar: frequências menores, consumo de energia reduzido e consequentemente um menor calor dissipado [1].

Os processadores multinúcleos estão presentes e disponíveis em grande variedade de arquiteturas, incluindo vários recursos de otimização de desempenho [2]. A sua capacidade de processamento baseia-se no processamento em paralelo, mantendo a garantia de melhor desempenho geral do sistema para aplicativos de *desktop* e para programas paralelos.

Sabendo disso, como os Algoritmos Genéticos (AGs) podem se beneficiar das tecnologias dos processadores multinúcleos? O primeiro passo é a paralelização do seu código. Os processadores atuais tem maior capacidade de processamento quando os seus núcleos trabalham em paralelo.

Os AGs, mesmos em suas implementações sequenciais clássicas, possuem um paralelismo inerente: é possível identificar facilmente partes do algoritmo que são independentes, como por exemplo, o cálculo da aptidão dos indivíduos, o que torna fácil a sua paralelização. Desde a década de 80, surgiram várias pesquisas de métodos para melhorar o tempo de execução dos AGs fazendo uso dos supercomputadores e da computação distribuída. A disponibilidades desses equipamentos era restrita devido ao seu alto custo econômico. No final da década de 90 surgiram os *clusters*, construídos com computadores pessoais [3], tornando a computação de alto desempenho mais acessível a centros de pesquisa com orçamentos mais modestos. Ainda assim, é prática muito comum implementar AGs de modo convencional [4]. Esse tipo de abordagem foi mantido com a ajuda da indústria de microprocessadores que garantiram o aumento do desempenho das aplicações lançando periodicamente processadores mais potentes [5].

Os processadores multinúcleos tem muito em comum com os multiprocessadores simétricos mas com diferenças importantes em relação aos tempos de latência do acesso a memória (menores nos multinúcleos), taxa de transferência de dados - *throughput* (maior nos multinúcleos) e ao compartilhamento de recursos. Os multinúcleos possuem diferentes arquiteturas, com particularidades em relação ao compartilhamento de recursos dentro do processador, como memória cache, barramento e memória principal [6]. Essas particularidades podem influenciar no desempenho de programas paralelos e devem ser levadas em consideração no projeto de paralelização dos AGs.

Uma das dificuldades em experimentos é determinar a influência de uma ou mais varáveis sobre outra variável de interesse. No caso da avaliação de um AGP, existem vários parâmetros configuráveis que podem influenciar no seu desempenho. O planejamento do experimento envolve investigações preliminares para a determinação dos fatores que influenciam ou não o resultado desejado. O resultado almejado por este trabalho é o melhor desempenho de um AGP em um processador multinúcleo. São apresentados os resultados da análise dos fatores que influenciam o desempenho da implementação de um AGP. A implementação aqui chamada de PGA baseia-se no modelo de múltiplas populações (ou modelo de ilhas), e testes foram realizados em um sistema com processador multinúcleo.

2. ALGORITMOS GENÉTICOS PARALELOS (AGPs)

A Computação Evolutiva (CE) por muito tempo focou essencialmente na evolução de uma única população, também conhecida como população *panmictic*, caracterizada por seus indivíduos que podem cruzar entre si. Entretanto, Darwin percebeu que existe uma estrutura espacial na população e que essa estrutura pode influenciar na sua dinâmica. Por exemplo, em populações isoladas (*demes*) como as que vivem em ilhas, algumas espécies evoluem de maneira diferente das que viviam em ambiente não isolado. Embora a diversidade nos *demes* seja baixa, a diversidade da população geral é grande [7].

Grosso (1985) *apud* [7] realizou um dos primeiros estudos sistemáticos sobre AGPs com múltiplas populações e migração. Constatou que estes exploram melhor o espaço de busca de um problema e evitam a convergência para um mínimo local graças à sua capacidade de manter a diversidade da população geral.

Os métodos de paralelização dos AGs seguem basicamente duas abordagens: manter uma única população ou dividir a população em várias populações locais relativamente isoladas. Em [8] os AGPs são classificados em 3 tipos básicos: o modelo Mestre-Escravo; o modelo *Fine-Grained* (granularidade fina), também conhecido como modelo celular; e o modelo *Coarse-Grained* (granularidade grossa), ou modelo de ilhas. Existem muitos modelos híbridos que são combinações desses modelos.

O modelo Mestre-Escravo possui uma população única e global: a população é controlada pelo processo Mestre que a distribui entre os Escravos para que estes realizem em paralelo e de forma síncrona a avaliação da aptidão dos indivíduos e/ou a aplicação dos operadores genéticos (mutação e cruzamento). A comunicação ocorre somente nos momentos em que a população é enviada aos escravos e no retorno dos valores das aptidões calculadas. O algoritmo é síncrono: espera até receber a avaliação de toda população antes de ir para a próxima geração. O AG mestre-escravo síncrono possui as mesmas propriedade de um AG sequencial com o diferencial da diminuição do tempo de resposta.

O modelo *Fine-Grained* também possui de uma única população com indivíduos distribuídos em vários processadores. Numa situação ideal tem-se um indivíduo por processador. Os operadores de seleção e o cruzamento estão restritos à área definida como sua vizinhança.

O modelo *Coarse-Grained*, também conhecido como modelo de ilhas, divide a população em populações locais e executam em paralelo o algoritmo de um AG convencional com a adição opcional de rotinas que implementam a migração de indivíduos entre as populações locais. São os modelos mais utilizados.

Em [7], é apresentada uma outra classificação baseada na maneira como encontra-se estruturada a população: modelo de ilhas e modelo celular. No modelo de ilhas ou multipopulações, cada ilha é um vértice do grafo e as arestas são os caminhos por onde ocorrem os movimentos de migração. Em um nível mais baixo, a subpopulação de cada ilha pode ser vista como um grafo completo onde os vértices são os indivíduos e as arestas são as possibilidades de cruzamentos. Já no Modelo Celular, ou Modelo Difuso, os indivíduos que constituem a população estão dispostos na forma de um reticulado.

Os AGPs possuem maior complexidade que os AGs sequenciais [4]. A migração é controlada por vários parâmetros: número de populações, topologia das conexões entre as populações, frequência da migração, número de indivíduos, seleção dos indivíduos, reposição dos indivíduos, síncrona ou assíncrona. Desta forma, o ajuste dos AGPs é uma tarefa desafiadora.

2.1 TRABALHOS RELACIONADOS

Um levantamento do estado da arte em AGPs é apresentado por [9] e um novo modelo de AGPs é proposto. Dentre os problemas ainda não resolvidos listados pelos autores está o custo da conexão entre as populações processadas em paralelo, ou

seja, quais os recursos necessários para se prover esta conexão.

Alba (1997) [10] mostra que os AGPs podem alcançar um aumento de desempenho que cresce linearmente com o aumento de processadores.

O trabalho realizado por Rucinski (2010) [4] analisa o impacto das diferentes topologias de migração no modelo de ilhas para paralelização de algoritmos da Evolução Diferencial e do *Simulated Annealing*, implementando e testando 14 topologias. Apresenta os resultados do desempenho das implementações sem considerar custos adicionais do barramento/CPU.

Em [11], Alba (2006) aborda a avaliação de implementações de metaheurísticas paralelas e como estas devem ser relatadas de maneira que as comparações sejam justas. Assim como Tomassini (2005) em [7], enfatiza que o cálculo da medida do desempenho de algoritmos paralelos (*speedup*), quando aplicado a algoritmos não-determinísticos deve ser calculado com valores médios dos tempos de execução. textitSpeedup é a razão entre o tempo gasto na execução de um algoritmo em um processador e o tempo gasto quando executado em *m* processadores,

Barreto et al.(2010) [12] preocupam-se em como interpretar, analisar e relatar os resultados obtidos nos experimentos. Mostram as dificuldades de se avaliar métodos estocásticos onde existe um nível de imprevisibilidade no seu comportamento. A partir da técnica de perfis de desempenho (*performance profiles*), propõem uma extensão chamada de perfis de desempenho probabilísticos capazes de incorporar e representar a variância associada ao funcionamento de um algoritmo estocástico.

Uma metodologia para ajuste de parâmetros de algoritmos da Programação Genética (PG) é apresentado por Lima (2010) [13], onde são utilizadas técnicas de planejamento experimental. Dentre elas o planejamento fatorial completo é utilizado para determinação do impacto de parâmetros no desempenho do algoritmo.

3. ALGORITMO PROPOSTO

O algoritmo PGA trabalha com múltiplas populações. A população inicial é dividida entre os vários processos que trabalham em paralelo na evolução de sua população local, denominados de Subpopulações. Um dos processos é chamado de Processo Central. O Processo Central, além de evoluir a sua população, sincroniza o início e o final do trabalho das outras Subpopulações. É também responsabilidade do Processo Central enviar às Subpopulações os parâmetros iniciais e indicar o início do trabalho, como mostra o algoritmo na Figura 1.

```
Leitura do parâmetros iniciais;
Envio dos parâmetros para Subpopulações;
Enquanto (não ocorre a condição de parada) faça
    Gera nova população aplicando operadores de cruzamento e mutação;
    Avalia a população;
    Se (geração atual MOD taxa de migração) = 0)
        Seleciona indivíduos;
        Envia indivíduos;
        Recebe indivíduos;
        Substitui indivíduos locais pelos indivíduos recebidos;
        Avalia a população;
    Recebe melhor indivíduo de cada Subpopulação;
    Verifica condição de parada;
    Envia comando continuar/parar para Subpopulações;
fim
```

Figura 1: Pseudo código do algoritmo do Processo Central.

As Subpopulações trabalham em paralelo. Cada uma gera aleatoriamente a sua população local e inicia o ciclo de evolução. A cada intervalo de gerações previamente configurado (parâmetro taxa de migração) envia aos vizinhos, de acordo com a topologia de migração escolhida, um determinado número de indivíduos (parâmetro número de migrantes) selecionados aleatoriamente ou pelo valor da sua aptidão (parâmetro seleção dos migrantes). Em seguida, recebe os indivíduos que migram de outras Subpopulações vizinhas. O esquema de vizinhança é definido pelo parâmetro topologia de migração, conforme descrito na seção 3.1. Os indivíduos que chegam são colocados na população local, substituindo indivíduos escolhidos aleatoriamente ou os com menor aptidão (parâmetro reposição dos migrantes), com exceção do melhor indivíduo local que tem seu lugar garantido. O próximo passo é enviar ao Processo Central o seu melhor indivíduo e aguardar o retorno com o comando para continuar ou finalizar, como mostra a Figura 2.

3.1 Topologia de migração

A topologia de migração define como estão conectadas as Subpopulações e desta forma determina a vizinhança e direciona a migração. As topologias de migração implementadas são: anel e totalmente ligada. A Figura 3 mostra os grafos representando as topologias. Na Figura 3, a topologia em anel (a) mostra que, por exemplo, a Subpopulação 1 envia para a Subpopulação 2 e recebe da Subpopulação 4. Na topologia totalmente ligada (b), a Subpopulação 1 envia para todas as Subpopulações e recebe de todas as Subpopulações.

```
Recebe do Processo Central os parâmetros iniciais;
Inicia a população local;
Enquanto (comando = continuar) faça
    Gera nova população aplicando operadores de cruzamento e mutação;
    Avalia a população;
    Se (geração atual MOD taxa de migração) = 0)
        Seleciona indivíduos;
        Envia indivíduos;
        Recebe indivíduos;
        Substitui indivíduos locais pelos indivíduos recebidos;
        Avalia a população;
    Envia melhor indivíduo local para Processo Central;
    Recebe comando do Processo Central;
fim
```

Figura 2: Pseudo código do algoritmo das Subpopulações.

(a) (b)

Figura 3: Esquema das Topologias de Migração entre 4 Subpopulações: (a) em anel e (b) totalmente ligada.

4. EXPERIMENTOS E RESULTADOS

O algoritmo PGA foi implementado em C/C++, utilizando a biblioteca MPICH2 [14] (ver seção 4.2) e a biblioteca de geração de números aleatórios [15] (ver seção 4.1). Os parâmetros de configuração do PGA que foram utilizados nos testes e cujos valores não foram alterados nas execuções são mostrados na Tabela 1. Os parâmetros de configuração que foram variados para os testes são mostrados na Tabela 2.

Tabela 1: Valores dos parâmetros fixos.

Parâmetro	Valor
Seleção	Torneio (n = 2)
Taxa de cruzamento	0.9
Taxa de mutação	0.001
Operador de cruzamento	*Simulated Binary Crossover Operator* (SBX)
Condição de parada	solução ótima ou exata

4.1 Gerador de números aleatórios

AGs são processos estocásticos que usam intensamente geradores de números aleatórios [7]. Gerar números aleatórios não é uma tarefa fácil para um computador, uma vez que se trata de uma máquina determinística. É impossível criar números aleatórios verdadeiros sem algum hardware adicional. O termo pseudo-aleatoriedade refere-se aos números aleatórios gerados por computador. O prefixo pseudo- é usado para distinguir esse tipo de número de um "verdadeiro" número aleatório gerado por um processo físico (ex. deterioração radioativa). Geradores de números (pseudo-) aleatórios (PRNG, em inglês) são algoritmos que, dado um valor inicial (chamado semente), geram uma sequência de números aparentemente aleatória. A ideia é que a sequência gerada de números pseudo-aleatórios se comportem como uma sequência de números verdadeiramente aleatórios, mas que são repetidos após um período fixo (desejado o mais longo possível).

A diferença entre PRNGs de alta e baixa qualidade é que os de alta qualidade possuem uma melhor estrutura oculta mais difícil de ser detectada por testes estatísticos de aleatoriedade e gera menos interferência em aplicações específicas. Os mais utilizados são: *Mersenne Twister* (MT), *The Mother-of-All Random Generators* (MoA), *SIMD-oriented Fast Mersenne Twister* (SFMT), *Well Equidistributed Long-period Linear* (WELL).

Na implementação do PGA foi utilizado o PRNG SMFT1, uma combinação do SFMT [16] com o MoA [17], da biblioteca disponibilizada por Agner Fog via GNU General Public License [15].

4.2 MPI - Message Passing Interface

MPI (*Message Passing Interface Standard*) é uma especificação para desenvolvedores e usuários para a normatização das bibliotecas de passagem de mensagens. A MPI possui duas versões, as normas MPI-1 e MPI-2. Essas especificações foram

definidas para programas em C/C++ ou Fortran. MPI é a única biblioteca de passagem de mensagens que possui o status de padrão (até mesmo industrial) e é suportada em virtualmente todas as plataformas computacionais. Diversas implementações estão disponíveis, livres ou proprietárias: MPICH, Open MPI, MVAPICH, MS MPI, Intel MPI.

A implementação MPICH2 [14] é livre e possui *benchmarks* onde foram testadas e comprovadas a sua escalabilidade nos processadores multinúcleos [18] e o seu desempenho na troca de mensagens em memória compartilhada [19].

4.3 Problema de Otimização Global

Para o levantamento de dados, foi escolhida uma função contínua de otimização global: função Rastrigin. É uma função multimodal e separável, e é uma das funções utilizadas no *benchmark* CEC 2010 [20].

A função Rastrigin é dada pela seguinte equação:

$$f_{Ras}(X) = 10n + \sum_{i=1}^{n}(x_i^2 - 10cos(2\pi x_i))$$

onde n é o número de dimensões e o mínimo global é $f_{Ras}(X^*) = 0$, e $X^* = [0, 0, ..., 0]$. Foram utilizadas 30 dimensões ($n = 30$) nos testes realizados nesse trabalho.

4.4 Plataforma Computacional

Para a execução dos testes foi utilizado um computador com processador Intel® Xeon® E5504, com duas CPUs com quatro núcleos cada, totalizando 8 núcleos a 2.00 GHz, cache L2 de compartilhada, FSB a 1333 MHz, 4 GB, sistema operacional Ubuntu 10.04 LTS (kernel 2.6.32).

4.5 Planejamento Experimental

Os métodos de planejamento experimental são baseados em princípios estatísticos e são ferramentas importantes que auxiliam pesquisadores a extrair do sistema em estudo o máximo de informação útil fazendo um mínimo de experimentos de forma racional e econômica. Um desses métodos é o método do planejamento fatorial completo que possibilita a determinação da influência de uma ou mais variáveis sobre uma outra variável de interesse no resultado do experimento [21].

Nesse trabalho, estamos interessados em descobrir como (e quanto) o aumento de desempenho do PGA em processadores multinúcleos depende dos parâmetros de configuração e como esses parâmetros podem influenciar-se mutuamente. De acordo com a nomenclatura do método [21], a propriedade de interesse do nosso experimento, chamada de **resposta**, é o aumento no desempenho do PGA. As variáveis que em princípio influenciam a resposta, chamados de **fatores**, são os parâmetros de configuração, e a função que descreve essa influência é chamada de **superfície de resposta**. O nosso objetivo é descobrir quais os valores desses fatores, chamados de **níveis**, que produzem a melhor resposta possível.

4.6 Descrição do experimento

O planejamento fatorial inicia-se com a especificação dos fatores e dos seus níveis. Os fatores escolhidos correspondem a 7 parâmetros de configuração com dois níveis cada, constituindo em um planejamento experimental do tipo fatorial 2^k completo, com k igual a 7. Os fatores e os seus níveis[1] estão listados na Tabela 2.

Tabela 2: Fatores analisados pelo planejamento experimental.

Fator	Nível -	Nível +
1. Número de Subpopulações	16	4
2. Topologia da migração	anel	totalmente ligada
3. Taxa da migração	10 gerações	100 gerações
4. População total	1600 indivíduos	3200 indivíduos
5. Número de migrantes	1 indivíduo	5 indivíduos
6. Seleção dos migrantes	Aleatória	Melhor(es) individuo(s)
7. Reposição dos migrantes	Aleatória	Substituição dos piores indivíduos

Os níveis para o fator **1** (número de subpopulações) foram definidos em função do número de núcleos existentes no processador: dobro e metade do número de núcleos. O fator **2** (topologia de migração) pode assumir o valor anel ou totalmente ligada. O fator **3** (taxa de migração) representa a frequência com que ocorrem as migrações: a cada 10 ou 100 gerações. O fator **4** (população total) corresponde ao tamanho do total da população do PGA que é divida entre as subpopulações. O fator **5** (número de migrantes) representa o número de indivíduos que migram de uma subpopulação para outra. O fator **6** (forma de seleção dos migrantes) representa a forma de seleção dos indivíduos locais que participarão da migração; aleatória ou escolha dos melhores. Finalmente, o fator **7** (tipo de reposição dos migrantes) determina de que forma os indivíduos que chegam a uma subpopulação são posicionados: de forma aleatória ou nas posições dos piores indivíduos.

[1] A escolha dos níveis + e - associado aos valores é arbitrária, ou seja, não afeta a conclusão da análise.

Para fazer um planejamento fatorial completo, devemos realizar ensaios com todas as possíveis combinações dos níveis dos fatores. São necessários 128 ensaios, sendo que cada ensaio corresponde a uma dada configuração do PGA a ser executada. Foram 30 réplicas de cada ensaio e a sequência de execução de cada configuração foi definida aleatoriamente.

A cada execução do PGA foi medido o tempo total de processamento (*wall clock*). Com o tempo de execução de cada combinação foi medida a melhora no desempenho do PGA através da medida *speedup* conhecida no contexto da computação paralela. O *speedup* é calculado pela divisão do tempo de execução do programa sequencial pelo tempo de execução do programa paralelo. Segundo Tomassini [7], no caso dos algoritmos evolucionários paralelos, para se ter uma medida de desempenho mais justa e significativa, deve-se usar como tempo de execução sequencial aquele medido na execução do mesmo algoritmo paralelo configurado com apenas uma subpopulação. Foi seguindo essa orientação que este trabalho considerou o tempo sequencial para o cálculo do *speedup* como o tempo médio de execução do PGA configurado com apenas uma subpopulação. Para obter o tempo médio de execução, o PGA foi executado 30 vezes para o tamanho da população total com valor igual a 1600, e outras 30 vezes para uma população de 3200 indivíduos. Quando o PGA é executado com apenas uma subpopulação não exite a migração.

Com os valores de *speedup* obtidos foi realizada uma análise de variância, também conhecida como ANOVA (*Analysis of Variance*) [21]. Os resultados desta análise são mostrados na Tabela 3.

Tabela 3: Tabela de Análise da Variância - ANOVA.

Fator	Soma Quadrática	Graus de Liberdade	Média Quadrática	F	Valor-P
1. Número de Subpopulações	86,316	1	86,3158	149,69	0
2. Topologia da migração	9,192	1	9,1921	15,94	0,0001
3. Taxa da migração	10,464	1	10,4638	18,15	0
4. População total	38,581	1	38,5812	66,91	0
5. Número de migrantes	9,591	1	9,5905	16,63	0,0001
6. Seleção dos migrantes	1,049	1	1,049	1,82	0,18
7. Reposição dos migrantes	0,001	1	0,0008	0	0,9702
Erro	69,198	120	0,5766		
Totalmente	224,391	127			

Os valores da coluna Valor-P da Tabela 3, nos fornecem uma medida qualitativa da influência de uma determinada variável no resultado de um processo, como mostra a Tabela 4 definida por Arsham (1988) [22]. No caso presente, a interpretação do Valor-P nos possibilita identificar com relativa certeza estatística quais dos 7 fatores utilizados em nosso experimento são mais relevantes para melhoria no *speedup* do PGA.

Tabela 4: Interpretação do Valor-P.

Valor-P	Interpretação do Valor-P
Valor-P $< 0,01$	Evidência muito forte
$0,01 \leq$Valor-P$< 0,05$	Evidência moderada
$0,05 \leq$Valor-P$< 0,10$	Evidência sugestiva
$0,10 \leq$Valor-P	Nenhuma evidência

De acordo com a Tabela 4, e analisando a ANOVA (Tabela 3), identificamos uma forte evidência estatística de que os fatores **1**, **2**, **3**, **4** e **5** são relevantes para *speedup* do PGA. Esses fatores correspondem ao número de subpopulações (ou ilhas), topologia de migração, taxa de migração, população total e número de migrantes.

Em seguida, baseados nos resultados obtidos nos ensaios do fatorial completo, foram calculados os efeitos de cada fator. O efeito de um determinado fator corresponde ao valor quantitativo da sua influência na resposta. O valor da influência de cada fator na resposta é conhecido como efeito principal. É chamado de efeito de interação o valor da influência conjunta de dois ou mais fatores. O efeito é, por definição, a média dos efeitos do fator nos níveis dos outros fatores. Quando interpretamos os efeitos, os valores positivos mostram o quanto a resposta melhora quando mudamos o fator do nível menos para o nível mais. E de forma análoga, valores negativos representam o quanto a resposta melhora quando mudamos o nível do fator de mais para menos [21].

O cálculo do erro padrão dos efeitos calculados [21] foi de 0,02. Para o intervalo de confiança de 99% podemos considerar que o valor de um efeito tenha a tolerância de $\pm$ 0,06,

Analisando os efeitos principais mostrados na Tabela 5, identificamos como fatores mais significativos na melhoria do desempenho do PGA os fatores **1**, **4**, **3**, **5** e **2**, em ordem decrescente de valor absoluto do efeito. O efeito principal do fator **1** é positivo, ou seja, quando mudamos o número de subpopulações de 16 para 4, o *speedup* aumenta em 1,64. Da mesma forma, o efeito principal positivo do fator **4** indica que quando trocamos o tamanho total da população de 1600 para 3200, o *speedup* aumenta em 1,10. Já o efeito principal do fator **3** é negativo, ou seja, quando alteramos a taxa de migração de 100 para 10 gerações (do nível mais para o nível menos), o *speedup* aumenta em 0,57. O efeito principal positivo do fator **5** sinaliza que a mudança do número de migrantes de 1 indivíduo para 5 aumenta em 0,55 o *speedup*. O fator **2** também tem seu efeito principal positivo e, portanto, a alteração da topologia de migração de anel para totalmente ligada traz o aumento de 0,54 no *speedup* .

É importante ressaltar que o efeito principal do fator **1** não pode ser entendido simplesmente como aumento ou redução do número de subpopulações. Os níveis definidos para esse fator refletem duas situações extremas: uma situação em que o número de processos é o dobro do número de núcleos do processador, e outra em que o número de processos é a metade do número de núcleos do processador.

Tabela 5: Efeitos calculados para o planejamento fatorial 2^7.

Média Global:													
2,22													
Efeitos Principais:													
1		**2**		**3**		**4**		**5**		**6**		**7**	
1,64		0,54		-0,57		1,10		0,55		0,18		0,01	
Efeitos de interação:													
12	**13**	**14**	**15**	**16**	**17**	**23**	**24**	**25**	**26**	**27**	**34**	**35**	**36**
-0,39	0,06	0,07	-0,28	0,07	-0,02	0,46	0,05	0,04	-0,35	0	0,04	0,1	0,24
37	**45**	**46**	**47**	**56**	**57**	**67**							
0,02	0,06	-0,02	0,02	0,01	0,02	0,01							

Os fatores de interação de segunda ordem apresentam vários valores que não são significativos quando comparados com os valores dos efeitos principais. Entretanto, temos o efeito da interação **23** (topologia e taxa de migração), cujo valor é 0,46. Para entender o que representa o valor do efeito de interação de dois fatores, podemos usar uma interpretação geométrica representando o planejamento experimental num sistema cartesiano, com um eixo para cada fator. O efeito de interação é o contraste entre as duas diagonais, considerando-se positiva a diagonal que liga o ensaio (- -) ao ensaio (++). Assim, ele pode ser interpretado geometricamente como uma diferença média [21]. No caso da interação **23**, quando mudamos da diagonal negativa para a positiva o *speedup* aumenta em 0,46. Ou seja, uma mudança da configuração anel e migração a cada 100 gerações (e/ou da totalmente ligada com migração a cada 10 gerações) para anel e migração a cada 10 gerações (e/ou totalmente ligada e migração a cada 100 gerações) aumenta o *speedup*.

Dentre os efeitos de interação de ordem maior que dois encontramos valores significativos para o efeito de interação **236** (0,51) e interação **1236** (-0,46). A interpretação dessas interações necessita de uma análise geométrica mais complexa, e por isso é objeto da continuidade desse trabalho.

5. CONCLUSÕES E TRABALHOS FUTUROS

Os processadores multinúcleos estão presentes e disponíveis em grande variedade de arquiteturas. A sua capacidade de processamento baseia-se no processamento em paralelo, mantendo a garantia de melhor desempenho geral do sistema para aplicativos de *desktop* e para programas paralelos.

Os AGs possuem um paralelismo inerente: é possível identificar com facilidade partes do algoritmo que são independentes, como por exemplo, o cálculo da aptidão dos indivíduos, o que torna fácil a sua paralelização.

Uma das dificuldades em se avaliar um AGP é a quantidade de parâmetros configuráveis que podem influenciar o seu desempenho. O planejamento do experimento de avaliação envolve investigações preliminares de determinação dos fatores que influenciam ou não no desempenho de um AGP em um processador multinúcleo. Este trabalho apresentou os resultados obtidos com um planejamento experimental do tipo fatorial 2^k completo para análise dos fatores que influenciam no desempenho da implementação PGA. Testes foram realizados em processador multinúcleo e os resultados mostram que a escolha do número de subpopulações (mediada pelo número de núcleos do processador), bem como a escolha do tamanho da população total tem forte impacto no desempenho do PGA. A taxa de migração, o número de migrantes e a topologia de migração também são fatores que tem forte influência no desempenho, embora em um grau menor.

A partir desses resultados, novos testes podem ser planejados e executados com o objetivo de construir uma orientação para a configuração dos AGPs executados em processadores multinúcleos.

REFERÊNCIAS

[1] S. K. Moore. "Multicore CPUs: Processor Proliferation". *IEEE Spectrum*, 2011.

[2] S. Akhter and J. Roberts. *Multi-Core Programming*, p. 336. Intel Corporation, first edition edition, 2006.

[3] T. L. Sterling, D. Savarese, D. J. Becker, J. E. Dorband, U. A. Ranawake and C. V. Packer. "BEOWULF: A Parallel Workstation for Scientific Computation". In *International Conference on Parallel Processing*, pp. 11–14, 1995.

[4] M. Ruciński, D. Izzo and F. Biscani. "On the impact of the migration topology on the Island Model". *Parallel Comput.*, vol. 36, pp. 555–571, October 2010.

[5] A. Munawar, M. Wahib, M. Munetomo and K. Akama. "A Survey: Genetic Algorithms and the Fast Evolving World of Parallel Computing". In *High Performance Computing and Communications, 10th IEEE International Conference on*, volume 0, pp. 897–902, Los Alamitos, CA, USA, 2008. IEEE Computer Society.

[6] J. Dongarra, D. Gannon, G. Fox and K. Kennedy. "The Impact of Multicore on Computational Science Software". *CTWatch Quarterly*, vol. 3, no. 1, February 2007.

[7] M. Tomassini. *Spatially Structured Evolutionary Algorithms: Artificial Evolution in Space and Time (Natural Computing Series)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005.

[8] E. Cantú-Paz. "A Survey of Parallel Genetic Algorithms". *CALCULATEURS PARALLELES*, vol. 10, 1998.

[9] D. S. Knysh and V. M. Kureichik. "Parallel genetic algorithms: a survey and problem state of the art". *Journal of Computer and Systems Sciences International*, vol. 49, no. 4, pp. 579–589, 2010.

[10] E. Alba. "Parallel evolutionary algorithms can achieve super-linear performance". *Information Processing Letters*, vol. 82, pp. 7–13, 2002.

[11] E. Alba and G. Luque. "Evaluation of Parallel Metaheuristics". In *Workshop on Empirical Methods for the Analysis of Algorithms*, PPSN-EMAA'06, pp. 9–14, Reykjavik, Iceland, September 2006.

[12] A. M. Barreto, H. S. Bernardino and H. J. Barbosa. "Probabilistic performance profiles for the experimental evaluation of stochastic algorithms". In *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, GECCO '10, pp. 751–758, New York, NY, USA, 2010. ACM.

[13] E. de Lima, G. Pappa, J. de Almeida, M. Gondalves and W. Meira. "Tuning Genetic Programming parameters with factorial designs". In *IEEE Congress on Evolutionary Computation (CEC'10)*, pp. 1 –8, july 2010.

[14] Argonne National Laboratory: MPICH2. http://www.mcs.anl.gov/research/projects/mpich2/.

[15] A. Fog. "Random Number Generator Libraries". 2008-2010, Instructions for the random number generator libraries on www.agner.org. GNU General Public License. Version 2.01. 2010-08-03.

[16] M. Saito and M. Matsumoto. "SIMD-Oriented Fast Mersenne Twister: a 128-bit Pseudorandom Number Generator". In *Monte Carlo and Quasi-Monte Carlo Methods 2006*, edited by A. Keller, S. Heinrich and H. Niederreiter, pp. 607–622. Springer Berlin Heidelberg, 2008.

[17] G. Marsaglia. "The Mother of All Random Generators". Published by Bob Wheeler in sci.stat.consult and sci.math.num-analysis in the name of George Marsaglia, 28th October 1994. Code and comments in: http://www.stat.berkeley.edu/classes/s243/mother.c.

[18] D. Buntinas, G. Mercier and W. Gropp. "Implementation and evaluation of shared-memory communication and synchronization operations in MPICH2 using the Nemesis communication subsystem". *Parallel Computing*, vol. 33, no. 9, pp. 634 – 644, 2007. Selected Papers from EuroPVM/MPI 2006.

[19] P. Balaji, D. Buntinas, D. Goodell, W. Gropp, T. Hoefler, S. Kumar, E. Lusk, R. Thakur and J. L. Traeff. "MPI on Millions of Cores". *Parallel Processing Letters (PPL)*, vol. 21, no. 1, pp. 45–60, Mar. 2011.

[20] K. Tang, X. Li, P. N. Suganthan, Z. Yang and T. Weise. "Benchmark Functions for the CEC'2010 Special Session and Competition on Large-Scale Global Optimization". 2009.

[21] B. de Barros Neto, I. S. Scarminio and R. E. Bruns. *Como Fazer Experimentos - Pesquisa e Desenvolvimento na Ciência e na Indústria*. Editora UNICAMP, fourth edition, 2010.

[22] H. Arsham. "Kuiper's P-value as a Measuring Tool and Decision Procedure for the Goodness-of-fit Test". *Journal of Applied Statistics*, vol. 15, pp. 131–135, 1988.