

Otimização de cache e paralelização de um algoritmo genético para o problema da mochila multidimensional

Henrique de Oliveira Marques¹, Janderson Souza Barros¹
Bianca de Almeida Dantas¹

¹Faculdade de Computação – Universidade Federal de Mato Grosso do Sul (UFMS)
Campo Grande – MS – Brasil

{henrique.marques, janderson.barros, bianca.dantas}@ufms.br

Resumo. *O problema da mochila multidimensional é uma extensão do problema da mochila binária com ampla aplicação prática, como na alocação de recursos limitados. Por pertencer à classe $\mathcal{NP}$ -difícil, para a qual não se conhece algoritmo exato de tempo polinomial, a busca por métodos de resolução eficientes é recorrente na literatura. Este trabalho apresenta uma reformulação do algoritmo genético de Chu e Beasley orientada à localidade de cache, complementada por paralelização em OpenMP com modelo de ilhas. A proposta foi avaliada contra o algoritmo genético sequencial de referência sobre 335 instâncias clássicas da literatura, com validação por testes estatísticos. Os resultados mostram speedup médio de $5,6\times$ (de $1,5\times$ a $13,8\times$), em sua maior parte decorrente da reformulação sequencial, sem comprometer a qualidade das soluções obtidas.*

1. Introdução

O problema da mochila binária consiste em selecionar, de um conjunto de n objetos com peso w_i e valor v_i , o subconjunto de maior valor total cujo peso não exceda a capacidade W , sem fracionar itens.

No problema da mochila multidimensional (*Multidimensional Knapsack Problem* – MKP), múltiplas restrições (como volume, custo ou tempo) devem ser respeitadas simultaneamente [Fréville 2004, Fréville and Hanafi 2005]. À medida que o número de restrições aumenta, a aplicação de métodos exatos torna-se frequentemente inviável, o que motiva a utilização de heurísticas para encontrar soluções de alta qualidade em tempo de processamento aceitável.

Heurísticas são métodos que buscam soluções satisfatórias para problemas complexos em tempo computacional viável, sem garantia de otimalidade. Meta-heurísticas, foco deste trabalho, são heurísticas genéricas que, com poucas adaptações, podem ser aplicadas a diferentes problemas de otimização. Dentre elas, os Algoritmos Genéticos (AGs) destacam-se no MKP, problema $\mathcal{NP}$ -difícil; adota-se como referência o AG de Chu e Beasley [Chu and Beasley 1998].

Este trabalho propõe e avalia, primeiramente, uma **reformulação do algoritmo orientada à localidade de cache**, sobre a qual se aplica **paralelização em CPU multinúcleo com OpenMP** (*Open Multi-Processing*) [Dagum and Menon 1998], com o objetivo de reduzir o tempo de obtenção das soluções sem comprometer sua qualidade.

As principais contribuições deste trabalho são: (i) **otimização orientada à cache** — representação do cromossomo em blocos de 64 bits, *layout* de pesos orientado a item e avaliação incremental do *fitness* —; (ii) **paralelização com OpenMP** em modelo de ilhas; e (iii) **validação estatística** dos resultados sobre 335 instâncias clássicas.

2. Metodologia

O ponto de partida de todas as comparações é a implementação sequencial do AG de Chu e Beasley [Chu and Beasley 1998] desenvolvida na tese de referência [Dantas 2016], executada sem modificações. Trata-se de um AG de estado estacionário (*steady-state*) com seleção por torneio, cruzamento uniforme, mutação de dois bits e reparo de viabilidade `dropAdd` por pseudoutilidades.

Na versão proposta, a população é dividida em subpopulações independentes (*ilhas*), cada uma evoluindo o AG localmente; periodicamente, uma *migração* cruza o melhor indivíduo global com o melhor de uma ilha aleatória.

Para comparar as duas versões aplicou-se o teste estatístico U de Mann-Whitney [Mann and Whitney 1947], com nível de significância $\alpha = 0,05$ e correção de **Holm-Bonferroni** [Holm 1979] sobre as 335 comparações.

As etapas mais executadas do AG — a avaliação do *fitness* e o reparo de viabilidade — percorrem repetidamente a matriz de pesos e o cromossomo; em processadores modernos, o custo desses laços é ditado pelo volume de instruções executadas e pelo comportamento da hierarquia de memória [Hennessy and Patterson 2017]. As seguintes reformulações foram aplicadas:

- **Layout orientado a item**
- **Cromossomo em blocos de 64 bits**
- **Uso de gerador de 64 bits (`mt19937_64`)**
- **Buffers `thread_local`**

Sobre essa base eficiente aplicou-se a paralelização em memória compartilhada com OpenMP na CPU multinúcleo, mapeando cada ilha a uma *thread*, sem transferência explícita de dados exigida por *clusters* (MPI) ou aceleradores (GPU).

2.1. Ambiente experimental

Todos os experimentos foram executados em uma única máquina dedicada e ociosa, com processador AMD Ryzen 7 5700G (8 núcleos, 16 *threads*, 16 MiB de *cache* L3), em Linux via Windows Subsystem for Linux (WSL), compilando com `g++ -O3 -fopenmp`.

3. Resultados e discussão

3.1. Comparação geral

A comparação foi realizada sobre as 335 instâncias, executando 30 vezes cada versão na mesma máquina (população 100, 4 ilhas, 10^6 gerações). A Tabela 1 resume os resultados por classe; *speedup* é a razão entre tempos médios, *gap* mede o afastamento da melhor solução conhecida, “Tempo sig.” indica ganho de tempo significativo (Mann-Whitney+Holm) e “OMP $\geq$ orig.” indica qualidade não inferior à versão original.

Tabela 1. Comparação do AG original com a proposta por classe (30 execuções).

Classe ($m \times n$)	Inst.	Speedup	Gap médio (%)		Tempo sig.	OMP $\geq$ orig.
			Orig.	OMP		
OR 5×100	30	5,06	0,009	0,000	30/30	30/30
OR 5×250	30	7,55	0,060	0,010	30/30	30/30
OR 5×500	30	9,28	0,077	0,014	30/30	30/30
OR 10×100	30	4,23	0,059	0,015	30/30	30/30
OR 10×250	30	6,17	0,124	0,044	30/30	30/30
OR 10×500	30	7,18	0,153	0,065	30/30	30/30
OR 30×100	30	3,79	0,082	0,047	30/30	27/30
OR 30×250	30	5,04	0,221	0,130	30/30	30/30
OR 30×500	30	5,84	0,266	0,160	30/30	30/30
GK	11	6,19	0,274	0,222	11/11	11/11
Menores (SAC-94)	54	3,19	0,009	0,000	54/54	54/54
Total / Média	335	5,57	0,105	0,051	335/335	332/335

4. Conclusão

Este trabalho apresentou uma reformulação do AG de Chu e Beasley para o MKP orientada à **localidade de cache**, complementada por paralelização OpenMP em modelo de ilhas. A avaliação sobre as 335 instâncias clássicas, validada pelo teste de Mann-Whitney com correção de Holm, mostrou reduções de tempo significativas em todas as instâncias, sem comprometer a qualidade das soluções. Como trabalhos futuros, pretende-se comparar a versão em OpenMP a uma implementação em CUDA, avaliar instâncias em escala cujas matrizes excedam as *caches* e investigar a aplicação das mesmas otimizações a outras metaheurísticas.

Referências

- Chu, P. C. and Beasley, J. E. (1998). A genetic algorithm for the multidimensional knapsack problem. *Journal of Heuristics*, 4(1):63–86.
- Dagum, L. and Menon, R. (1998). OpenMP: An industry-standard API for shared-memory programming. *IEEE Computational Science and Engineering*, 5(1):46–55.
- Dantas, B. (2016). *Metaheurísticas para o Problema da Mochila Multidimensional*. PhD thesis, Faculdade de Computação, Universidade Federal de Mato Grosso do Sul, Campo Grande, MS.
- Fréville, A. (2004). The multidimensional 0–1 knapsack problem: An overview. *European Journal of Operational Research*, 155(1):1–21.
- Fréville, A. and Hanafi, S. (2005). The multidimensional 0–1 knapsack problem—bounds and computational aspects. *Annals of Operations Research*, 139(1):195–227.
- Hennessy, J. and Patterson, D. (2017). *Computer Architecture: A Quantitative Approach*. Morgan Kaufmann, 6th edition.
- Holm, S. (1979). A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6(2):65–70.
- Mann, H. and Whitney, D. (1947). On a test of whether one of two random variables is stochastically larger than the other. *Annals of Mathematical Statistics*, 18(1):50–60.