Alocação Eficiente de Mercados Financeiros em Computadores Usando o Multiway Partitioning Problem

Gabriel C. M. Fernandes[1], **Cícero C. Maciel** [1]

[1] Departamento de Informatica – Universidade Federal de Viçosa (UFV)
Caixa Postal 36570-900 – Viçosa, MG – Brasil

{gabriel.fernandes1,cicero.maciel}@ufv.br, @ufv.br

Resumo. *Este artigo apresenta o uso de metaheurísticas para resolver um problema de distribuição de mercados de corretoras entre computadores que monitoram eventos do mercado financeiro, visando equilibrar a carga de trabalho. O problema se enquadra no Multiway Partitioning Problem, mas com uma formulação inédita que combina restrições específicas: uma restrição hard, que limita o número de mercados de uma mesma corretora por computador, e uma restrição soft, que busca concentrar, quando possível, mais mercados de uma mesma corretora em um único computador. Para isso, foram aplicadas as metaheurísticas Iterated Greedy (IG) e Algoritmo Genético (GA), obtendo resultados relativamente promissores em qualidade de solução e tempo de execução.*

1. Introdução

No mercado financeiro, o monitoramento em tempo real de múltiplos mercados é uma atividade essencial para corretoras, plataformas de negociação e provedores de dados. Para garantir eficiência, desempenho e confiabilidade, é necessário distribuir de forma inteligente os mercados que cada computador deve acompanhar, evitando sobrecarga e falhas de processamento de eventos.

Uma solução ingênua para esse cenário seria concentrar todos os mercados em um único computador de grande capacidade. No entanto, essa abordagem pode ser financeiramente inviável e pouco prática, já que exige infraestrutura cara, maior risco de pontos únicos de falha e limitações físicas de processamento. Em muitos casos, é mais eficiente adotar uma estrutura distribuída, em que vários computadores — muitas vezes instalados em data centers ou colocation, localizados próximos às corretoras — dividem a carga de trabalho de forma balanceada. Essa proximidade reduz a latência no recebimento e processamento dos dados, garantindo maior agilidade e precisão no monitoramento dos eventos de mercado.

A ideia de distribuir os mercados de forma equilibrada entre os computadores também visa minimizar o número total de máquinas utilizadas e sua capacidade individual, promovendo economia de recursos e redução dos custos operacionais. Assim, a alocação inteligente dos mercados contribui para a eficiência financeira e operacional do sistema de monitoramento.

Nesse contexto, surge um problema de particionamento que se enquadra no Multiway Partitioning Problem, em que é preciso dividir um conjunto de mercados em subconjuntos, respeitando restrições operacionais. No caso abordado neste artigo, cada computador deve respeitar um limite máximo de mercados de uma mesma corretora (restrição

hard) e, sempre que possível, concentrar mercados de uma mesma corretora em um único computador (restrição soft). Esta configuração reflete a necessidade de equilibrar o uso dos recursos computacionais e garantir um monitoramento eficiente, robusto e economicamente viável para as operações de mercado em tempo real.

2. Revisão bibliográfica

Analisando a literatura sobre o Multiway Number Partitioning (MNP), não foram encontrados estudos que abordem a variação proposta neste artigo ou algo semelhante. Existem, porém, variações diversity-aware [1], que buscam maximizar a diversidade dos subconjuntos, objetivo oposto à restrição soft deste trabalho, que visa minimizar a diversidade de mercados em cada computador, agrupando mercados da mesma corretora. Dessa forma, as técnicas diversity-aware são usadas apenas como referência do que deve ser evitado neste estudo.

3. Métodos

Neste trabalho, todo o código foi desenvolvido em C++. As metaheurísticas utilizadas foram a Iterated Greedy (IG) e o Algoritmo Genético (GA). Ambas operam a partir de uma classe Solução, que representa um conjunto de computadores e armazena informações relacionadas a cada um deles. Essas informações são derivadas das cargas suportadas pelos computadores, que por sua vez são calculadas a partir das cargas dos mercados atribuídos a cada computador. O desvio padrão dessas cargas é a principal métrica considerada para avaliação.

3.1. Função de Avaliação

Considere um conjunto de m computadores, onde cada computador i possui uma carga total C_i que deve ser processada por ele. Essa carga é dada pelo somatório das cargas dos mercados atribuídos a esse computador:

$$C_i = \sum_{j \in M_i} w_j,$$

onde M_i é o conjunto de mercados alocados ao computador i e w_j é a carga do mercado j.

A função objetivo utilizada para avaliar a qualidade de uma solução é o desvio padrão das cargas dos computadores, que mede o equilíbrio da distribuição das cargas entre eles. O valor da solução f é dado por:

$$f = \sqrt{\frac{1}{m} \sum_{i=1}^{m} (C_i - \bar{C})^2},$$

onde $\bar{C}$ é a média das cargas dos computadores:

$$\bar{C} = \frac{1}{m} \sum_{i=1}^{m} C_i.$$

O objetivo é minimizar f, buscando uma distribuição equilibrada das cargas entre os computadores.

3.1.1. Função da Restrição Soft

Para cada computador i e corretora k, definimos o valor normalizado $v_{i,k}$ como:

$$v_{i,k} = 100 \times \frac{n_{i,k}}{L_k},$$

onde $n_{i,k}$ é o número de mercados da corretora k alocados ao computador i e L_k é o limite máximo de mercados dessa corretora por computador. Quando o computador i não possui mercados da corretora k, então $n_{i,k} = 0$ e, consequentemente, $v_{i,k} = 0$.

Em seguida, calculamos o desvio padrão dos valores $v_{i,k}$ para cada computador i:

$$d_i = \sqrt{\frac{1}{K} \sum_{k=1}^{K} (v_{i,k} - \bar{v}_i)^2},$$

onde K é o número total de corretoras e $\bar{v}_i$ é a média dos valores $v_{i,k}$ para o computador i.

Por fim, calculamos o desvio padrão dos valores d_i entre todos os computadores:

$$D = \sqrt{\frac{1}{m} \sum_{i=1}^{m} (d_i - \bar{d})^2},$$

onde m é o número de computadores e $\bar{d}$ é a média dos valores d_i.

Valores maiores de D indicam maior concentração dos mercados de poucas corretoras em alguns computadores, o que satisfaz a restrição soft.Esta função é utilizada como critério de desempate entre soluções que apresentam o mesmo valor de avaliação principal.

3.2. Iterated Greedy

O IG é uma metaheurística do tipo construtivo-destrutiva que parte de uma solução inicial, remove parte de sua estrutura de forma controlada e a reconstrói repetidamente. Essa abordagem permite escapar de mínimos locais e explorar o espaço de soluções de forma mais ampla.

3.2.1. Método guloso inicial

A seguir, apresenta-se o pseudocódigo da método guloso utilizada para construir a solução inicial:

1. Ordenar os mercados por carga média (decrescente).
2. Inicializar uma fila de prioridade de computadores, ordenada pela carga.
3. Para cada computador:
    - Criar vazio e inserir na fila.
4. Enquanto houver mercados não alocados:

- Retirar o computador com menor carga.
- Procurar o primeiro mercado que não viole a restrição hard.
- Se encontrado:
  - Alocar o mercado ao computador.
  - Atualizar carga e restrições.
  - Recolocar o computador na fila.
- Caso contrário, encerrar (solução inviável).

5. Formar a solução final.
6. Avaliar a solução.

3.2.2. Operações de Vizinhança

A vizinhança de uma solução é explorada por meio de duas operações principais:

- **Move**: transfere um único mercado de um computador para outro.
- k**-Swap**: realiza uma troca cíclica entre k mercados de k computadores distintos, onde cada mercado é deslocado para o computador seguinte no ciclo.

Ambas as operações podem gerar vizinhos que violam as restrições hard, ou seja, soluções temporariamente inválidas. Essa possibilidade permite explorar regiões do espaço de soluções mais amplas, ajudando o algoritmo a escapar de mínimos locais e potencialmente encontrar soluções de melhor qualidade.

3.2.3. Busca Local

As operações de vizinhança *Move* e k*-Swap* geram muitos vizinhos, o que pode tornar a busca custosa. Para lidar com isso, utiliza-se uma busca local do tipo *first improvement*, que avalia vizinhos gerados aleatoriamente até encontrar uma melhoria.

O pseudocódigo é o seguinte:

1. Para um número fixo de iterações:
   (a) Gerar uma mudança (move ou k-swap).
   (b) Avaliar o impacto parcial da mudança na solução.
   (c) Se a solução melhorar:
      i. Aplicar a mudança.
      ii. Parar a busca local.

3.2.4. Destruição e Reconstrução

A fase de destruição consiste em remover aleatoriamente um conjunto de mercados da solução atual, promovendo diversificação na busca.

Em seguida, a fase de reconstrução utiliza a função greedy inicial para realocar os mercados removidos, garantindo a viabilidade da solução e aproveitando a heurística construída previamente.

3.2.5. Iterated Greedy

O pseudocódigo da aplicação da metaheurística usada nesse trabalho é a seguinte:

1. Inicializar a solução usando o método greedy inicial.
2. Calcular o valor da restrição soft da solução.
3. Para $j = 1$ até o número máximo de iterações:
    (a) Repetir:
        i. Para $i = 1$ até o número de iterações internas:
            A. Aplicar busca local na solução.
            B. Reconstruir a solução usando o método greedy.
        ii. Aplicar busca local na solução.
    (b) Enquanto a solução não for válida em relação às restrições hard.
    (c) Recalcular o valor da restrição soft da solução.
    (d) Armazenar a solução atual em uma lista de soluções.
4. Selecionar a melhor solução da lista, considerando:
    (a) Menor valor da função objetivo principal.
    (b) Em caso de empate, maior valor da função de restrição soft.
5. Atualizar a solução final com a melhor encontrada.

3.3. Algoritmo Genético

No Algoritmo genético, a procura pela solução ótima começa pela geração da população inicial do GA, a qual tem um tamanho total definido,e é constituída metade por soluções aleatoriamente geradas e metade por soluções geradas usando a função greedy para criação da solução inicial, todas soluções são armazenadas em um min heap, e são validadas em relação ao hard constraint.

O pseudocódigo da geração de uma solução aleatória é a seguinte:

1. Crie uma solução vazia `sol`.
2. Para cada mercado m em `i.markets`:
    (a) Escolha aleatoriamente um computador `rc` da solução `sol`.
    (b) Adicione o mercado m à lista de mercados do computador `rc`.
3. Retorne a solução `sol`.

Após a geração da população inicial, começa-se o processo de reprodução e seleção das soluções. Da população total, são selecionados as soluções com os 25% melhores valores, essas soluções selecionadas são agrupadas em pares e para cada par são gerados 4 soluções filhas, construídas de uma das metades do conjunto de computadores de cada pai, algumas destas soluções sofrem mutações, mudando aleatoriamente parte de seus computadores. Além disso, as 3% melhores soluções das soluções escolhidas para se reproduzir são clonadas e inseridas na nova geração, isso é feito para reduzir a probabilidade das novas gerações produzirem resultados piores que as prévias, melhorando assim a performance desta metaheurística. Após cada geração, a melhor solução encontrada é armazenada no min heap,e, após todas as gerações serem criadas, a melhor solução é encontrada no topo deste heap.

O pseudocódigo do algoritmo genético é o seguinte:

1. Inicialize as listas: `populacao`, `good_sols` e `elite_sols`.

2. Para a de 0 até $\frac{\texttt{i.markets}}{4} - 1$:
   (a) Adicione $\texttt{i\_markets}[a]$ em `good_sols`.
   (b) Se $a < \frac{3}{100} \times \texttt{i.markets}$, adicione $\texttt{i\_markets}[a]$ em `elite_sols`.
3. Embaralhe a lista `good_sols`.
4. Limpe a lista `populacao`.
5. Para a de 0 até `good_sols.size()` $- 1$:
   (a) Defina $next = 0$.
   (b) Se $a + 1 <$ `good_sols.size()`, então $next = a + 1$.
   (c) Gere filhos a partir dos índices a e $next$ com `gerar_filhos(a, next)`.
   (d) Se ocorrer `chance_mut`, aplique `mutate(filho)`.
   (e) Adicione os filhos gerados à lista `populacao`.
6. Para cada elemento e em `elite_sols`, adicione e à lista `populacao`.

4. Experimentos

Para os experimentos deste artigo, as instâncias foram criadas a partir de dados gerados especificamente. Foram elaborados 8 arquivos contendo as cargas de mercados, um para cada corretora: Binance, B3, Mercado Bitcoin, NYSE, Bitmex, Bybit, Deribit e OKX.

As instâncias são composições aleatórias desses arquivos, formando diferentes combinações de corretoras. O nome de cada instância segue o formato `xeyc`, onde x representa o número de corretoras incluídas e y o número de computadores necessários para monitorar todos os mercados. Por exemplo, a instância `2e3c` representa um cenário com 2 corretoras e 3 computadores.

Após a preparação das instâncias, elas são fornecidas ao executável como um argumento da linha de comando, junto ao número de computadores a serem usados no experimento, com esses parâmetros, as metaheurísticas são capazes de procurar a melhor solução, o que é feito por um número pré-determinado de iterações definido pela metaheurística usada, após a conclusão destas iterações, a melhor solução encontrada é retornada.

5. Resultados

Após a execução das instâncias, os seguintes resultados foram obtidos:

Tabela 1. Valores do desvio padrão

	Random	Guloso	IG	GA
2e3c	7992.40293	55.11806	2.94392	7.07107
3e4c	11365.5419	79.40521	207.93073	31
3e9c	1120683367.26291	0.4969	0.68493	82061251.06896
4e5c	229315899.31334	196373704.18246	196373704.18246	1005662.0189
5e9c	791681319.99021	0.41574	0.41574	38719567.45386
6e7c	449293730.08801	98643338.67584	98643338.67584	12386864.89306
7e9c	1052701884.23197	0.4969	0.4969	0.4969
8e9c	1023671038.46764	196373704.18246	0.4969	54168277.87687

Tabela 2. Valores do soft constraint

	Random	Guloso	IG	GA
2e3c	3.68179	8.73053	4.10961	6.59966
3e4c	3.37819	3.03848	6.17291	5.22074
3e9c	2.8903	2.0699	4.73558	1.98138
4e5c	4.84253	12.5471	12.83988	2.41449
5e9c	1.79565	1.3783	5.41025	0.68387
6e7c	0.92241	7.16197	7.65101	1.95114
7e9c	1.57676	1.46236	3.70623	1.17559
8e9c	1.67296	0.89588	3.01430	0.97992

5.1. Conclusão

Os resultados indicam que o desempenho das metaheurísticas varia conforme a instância considerada. O Iterated Greedy (IG) apresenta frequentemente os menores valores de desvio padrão, sugerindo uma distribuição mais equilibrada das cargas entre os computadores. Além disso, o IG também alcança valores elevados no critério da restrição soft, indicando uma concentração eficaz dos mercados de poucas corretoras em determinados computadores.

O Algoritmo Genético (GA) apresenta desempenho intermediário, com resultados competitivos na restrição soft, enquanto o método guloso, embora melhore algumas soluções iniciais, geralmente não atinge a qualidade das outras técnicas.

Esses padrões mostram que, apesar de algumas metaheurísticas se destacarem em determinados aspectos, a performance é sensível à instância usada. Por isso, recomenda-se a aplicação de múltiplas metaheurísticas para buscar a melhor solução possível, já que nenhuma delas, isoladamente, garante o resultado ótimo para todas as situações analisadas.

Atribuição

Cícero C. Maciel: definição do problema e abstrações, *Iterated Greedy* e simulações; Gabriel C. M. Fernandes: depuração, algoritmo genético e função de avaliação.

Referências

[1] Riley McDole, Patrick Steele, Siyuan Zeng. *Multiway Number Partitioning Problem: An Analysis of Heuristics and Machine Learning Approaches*. Stanford University, CS230 Project Report, 2021. Disponível em: `https://cs230.stanford.edu/projects_fall_2021/reports/103066753.pdf`