Algoritmo genético para o problema de cobertura única

Gabriel Saraiva de Gouveia[1], Olga Maria dos Santos Gonçalves[1], Rafael Tavares Oliveira[1] e Yasser Farid Pereira[1].

[1]Departamento de Informática – Universidade Estadual de Maringá (UEM-PR)

Maringá – PR – Brasil

{ra129145, ra130002, ra129969, ra129706}@uem.br

Resumo. *Apresentamos a aplicação de uma metaheurística BRKGA (Biased Random-Key Genetic Algorithm) para o Problema da Cobertura Única. Experimentos foram conduzidos em instâncias da OR-Library para avaliar o desempenho do método. Os resultados foram comparados com um algoritmo exato de Branch-and-Cut, Branch-and-Bound com variações e heurística GRASP. A análise demonstra que o BRKGA é uma alternativa competitiva às heurísticas GRASP e superior ao método exato em instâncias complexas onde este não converge no tempo limite.*

1. Fundamentação

Este trabalho apresenta a aplicação de uma metaheurística baseada em um algoritmo genético para encontrar soluções para o problema da cobertura única, usando um modelo de programação linear, o problema é uma variante do problema de cobertura de conjuntos. A cobertura única, conhecida como UCP (Unique Covering Problem), tem como objetivo cobrir o maior número de elementos uma única vez, podendo ter mais de um subconjunto, mas minimizando a quantidade de subconjuntos que cobrem o elemento e podendo assim, deixar elementos fora de cobertura.

O UCP foi apresentado por Demaine et al. (2006) e tem aplicações em telefonia, redes sem fio e rádios broadcast. Ele consiste em um universo U com n elementos e S um conjunto de m subconjuntos de elementos de U, cujo o objetivo é encontrar um subconjunto S′, onde $S' \subseteq S$, que maximize a quantidade de elementos cobertos por um único subconjunto.

O tema, a modelagem e as instâncias foram fundamentados no artigo de Mazaro (2011). O método aplicado utiliza um algoritmo genético para resolver o UCP. Foi conduzido experimentos para avaliar o método proposto a fim de identificar o gap de integralidade e o tempo gasto para resolver diferentes instâncias testes do problema.

A seguir descreveremos a modelagem de PLI e o desenvolvimento do programa com o resolvedor utilizado. Posteriormente, apresentaremos detalhes das instâncias usadas no experimento, bem como, a análise dos resultados e do tempo de execução.

2. Modelo de Programação Linear Inteira

Função Objetivo:

$$\max \sum_{i=1}^{n} y_i$$

Maximizar Σ yi, onde yi = 1 se o elemento i for coberto exatamente uma vez.

Restrições:

$$\begin{aligned} \text{sujeito a} \quad & \sum_{j=1}^{m} a_{ij} x_j = z_i && , \forall i, 1 \leq i \leq n \\ & y_i \leq z_i && , \forall i, 1 \leq i \leq n \\ & (M-1) y_i + z_i \leq M && , \forall i, 1 \leq i \leq n \\ & y \in \{0,1\}^n, z \in \mathbb{Z}^n, x \in \{0,1\}^m. \end{aligned}$$

Onde:

- n: cardinalidade do universo U
- m: cardinalidade do conjunto S
- aij: variável que representa a matriz de incidência
- xj: variável que indica se o subconjunto j foi escolhido.
- zi: número total de vezes que o elemento i foi coberto.
- yi: variável binária que vale 1 se o elemento i foi coberto uma única vez.
- M: constante grande (por exemplo, o número total de subconjuntos).

Considere o exemplo, no conjunto da esquerda, nossa instância inicial, tem o universo U = {1,2,3,4,5,6,7,8,9,10} com n = 10 e um conjunto S = {A, B, C} com m = 3, a solução ótima encontrada pelo método proposto é a cobertura dos subconjuntos {A, C}, que cobrem 9 elementos de 10, com somente uma cobertura não única (elemento 2). Essa é a essência do Problema da Cobertura Única (UCP): não basta cobrir os elementos, é buscado o máximo de coberturas únicas, aceitando não cobrir alguns os elementos ou ter o mínimo de coberturas não únicas.

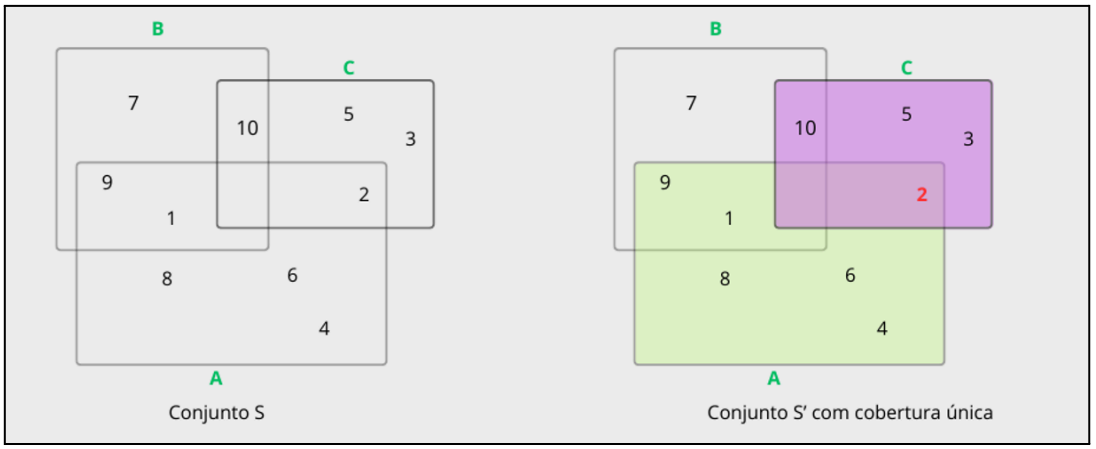

Figura 1. Exemplo de UCP

3. Metodologia

O desenvolvimento do método teve uma abordagem baseada no algoritmo genético BRKGA (Biased Random-Key Genetic Algorithm), aplicado à formulação de PLI descrita por Mazaro (2011). Esta metaheurística é bem conhecida pela sua eficiência em problemas combinatórios, e utiliza vetores de números reais entre 0 e 1 como cromossomos, que são posteriormente decodificados em soluções viáveis do problema. É também uma variação do algoritmo genético clássico (AG), com o objetivo de diminuir a dificuldade do AG em lidar com soluções viáveis.

O método proposto consiste em:

1. **Leitura da instância**: Carregamento dos dados em memória de todas as instâncias, os dados são lidos a partir de arquivos .txt, convertendo para uma matriz binária.
2. **Codificação**: Cada solução candidata é representada por um vetor de chaves aleatórias (valores entre 0 e 1). A posição no vetor corresponde a um subconjunto, e a chave representa a prioridade de inclusão desse subconjunto na solução.
3. **Decodificador**: O decodificador interpreta o vetor de chaves e constrói uma solução viável. Isso é feito por meio de uma heurística gulosa, que considera os subconjuntos com maior peso (definido por sua contribuição à cobertura única). Após a construção, uma busca local é aplicada para refinar a solução.
4. **Configuração do BRKGA**: Parâmetros definidos via arquivo config.conf:
    a. Tamanho da população;
    b. Percentual da elite;
    c. Taxa de mutantes;
5. **Evolução das soluções**: A cada iteração, o BRKGA gera uma nova população com base na recombinação enviesada entre indivíduos da elite e os demais. Também são injetados mutantes aleatórios para manter a diversidade populacional.
6. **Análise de desempenho**: Durante a execução, são registrados o número de iterações, os tempos de atualização da melhor solução, o valor da melhor cobertura única encontrada e os gaps de integralidade. Os resultados são escritos em um arquivo de saída.

A implementação foi realizada em Python 3.11, com uso do framework BRKGA-MP-IPR e foram mantidas as mesmas 8 instâncias do conjunto Set Covering Problem de tamanhos variados, obtidas na *OR-Library* de Beasley (2010), utilizadas no trabalho anterior com o solver CBC, permitindo uma comparação direta entre as abordagens. A execução do algoritmo foi através de um computador pessoal, disposto das seguintes especificações: sistema operacional Windows 11 Home Single Language, versão 24H2, com processador 13th Gen Intel(R) Core(TM) i5-13450HX 2.40 GHz e 16GB de memória RAM.

4. Resultados

O resultado dos experimentos foi limitado a 3600 segundos de execução, o objetivo é avaliar o desempenho do método proposto em encontrar soluções ótimas em um tempo de execução viável. Para compreendermos a qualidade da solução, foi calculado o GAP

de integralidade em %, o GAP é usado para indicar o quão próximo a solução encontrada está da solução ótima conhecida ou de um limite.

Na Tabela 1 é exposto o resultados experimentais, mostrando as instâncias da OR Library usadas e sua quantidade de elementos (n) e quantidade de conjuntos (m), seguido pelo tempo de execução do solver até encontrar a solução ótima ou até o limite de tempo, depois temos o Gap e o valor da solução factível encontrada e a média de cobertura dos elementos (C).

Tabela 1 - Resultados experimentais

Instância	n	m	Tempo	Valor obtido	Gap %	|C|
scp49.txt	200	1000	1h 0min 1s 434ms	195	2.56	1.22
scp57.txt	200	2000	1h 0min 0s 705ms	197	1.52	1.07
scp63.txt	200	1000	1h 0min 4s 199ms	161	24.22	1.65
scpa5.txt	300	3000	1h 0min 27s 545ms	279	7.53	1.19
scpb3.txt	300	3000	1h 0min 5s 432ms	221	35.75	1.79
scpc3.txt	400	4000	1h 0min 22s 946ms	340	17.65	1.47
scpd1.txt	400	4000	1h 0min 8s 100ms	266	50.38	2.59
scpe4.txt	50	500	1h 0min 0s 515ms	45	11.11	1.06

Para avaliar o desempenho da metaheurística BRKGA, vamos comparar os resultados obtidos com os apresentados no artigo de referência de Mazaro (2011). É importante ressaltar que os resultados do artigo de Mazaro são referentes a grupos de instâncias e o deste experimento é de uma única instância do grupo.

No trabalho de Mazaro, as instâncias foram submetidas a nove estratégias experimentais (nomeadas de E1 a E9), divididas em dois grupos, de Branch-and-Bound com variações e heurísticas GRASP, com uma variação de 1% e 1,5% nos resultados, respectivamente. Queremos medir a qualidade do método proposto usando algoritmo genético, sendo assim, vamos comparar os resultados somente com os melhores resultados do artigo de referência, afirmados pela autora, que são o E4 (Branch-and-Bound com heurística GRASP-FirstFit) e E9 (Heurística GRASP-Disjunto). Além disso, comparamos com o gap do trabalho anterior que foi usado um algoritmo de Branch-and-Cut.

Tabela 2 - Comparação de Gaps

Instâncias	BRKGA	Branch-and-Cut	BB + GRASP FF	GRASP Disjunto
scp4	2,56	0.01	0	4,5
scp5	1,52	0.01	0	1,4
scp6	24,22	38.89	24,2	21,5

scpa	7,53	16.09	4	7,4
scpb	35,75	368.75	31,4	30,2
scpc	17,65	52.87	17,2	13,9
scpd	50,38	875.61	40,2	40
scpe	11,11	13.64	13,6	12,1

O método usando BRKGA demonstrou um bom desempenho, alcançando soluções quase ótimas com GAPs inferiores a 3% nas instâncias mais simples (scp4 e scp5) e se mostrando uma alternativa competitiva à melhor heurística de Mazaro, o GRASP Disjunto. Os resultados reforçam que a dificuldade do problema não está somente no tamanho da instância, mas sim na média de cobertura dos elementos (C). Isso é visto pelo fato de que instâncias com baixa sobreposição de cobertura, como scp57 e scpe4, apresentaram GAPs menores, enquanto instâncias com alta sobreposição, como scpd1, resultaram no maior GAP (50,38%), confirmando que a complexidade aumenta com a sobreposição dos conjuntos.

5. Considerações Finais

O presente trabalho implementou e avaliou uma metaheurística BRKGA para o problema da cobertura única, confirmando seu desempenho na obtenção de soluções factíveis de qualidade em um tempo de execução limitado. A abordagem se mostrou vantajosa em relação aos resultados de um algoritmo exato de Branch-and-Cut que não convergiu a tempo, e apresentou-se como uma alternativa competitiva às heurísticas GRASP propostas por Mazaro (2011). Como trabalhos futuros, sugere-se a avaliação de outras formulações ou o uso de técnicas como a geração de colunas para obter limitantes duais mais justos, uma vez que os atuais se mostraram fracos em alguns casos.

6. Referências

Beasley, J. E., (2010). Or-library. http://people.brunel.ac.uk/˜mastjjb/jeb/orlib/scpinfo.html.

MAZARO, Regina; HOSHINO, Ayako, (2011). O problema de cobertura única. Anais do XLIII Simpósio Brasileiro de Pesquisa Operacional, Ubatuba, São Paulo. Recuperado de http://www.din.uem.br/sbpo/sbpo2011/pdf/88121.pdf

ANDRADE, C. E. brkga_mp_ipr_python: Python implementation of the BRKGA-MP-IPR. Disponível em: https://github.com/ceandrade/brkga_mp_ipr_python. Acesso em: 20 jul. 2025.

RESENDE, M. G. C. (2013) Introdução aos algoritmos genéticos de chaves aleatórias viciadas. In: XSLVSBPO, p. 3680–3691.