Universidade Estadual de Maringá

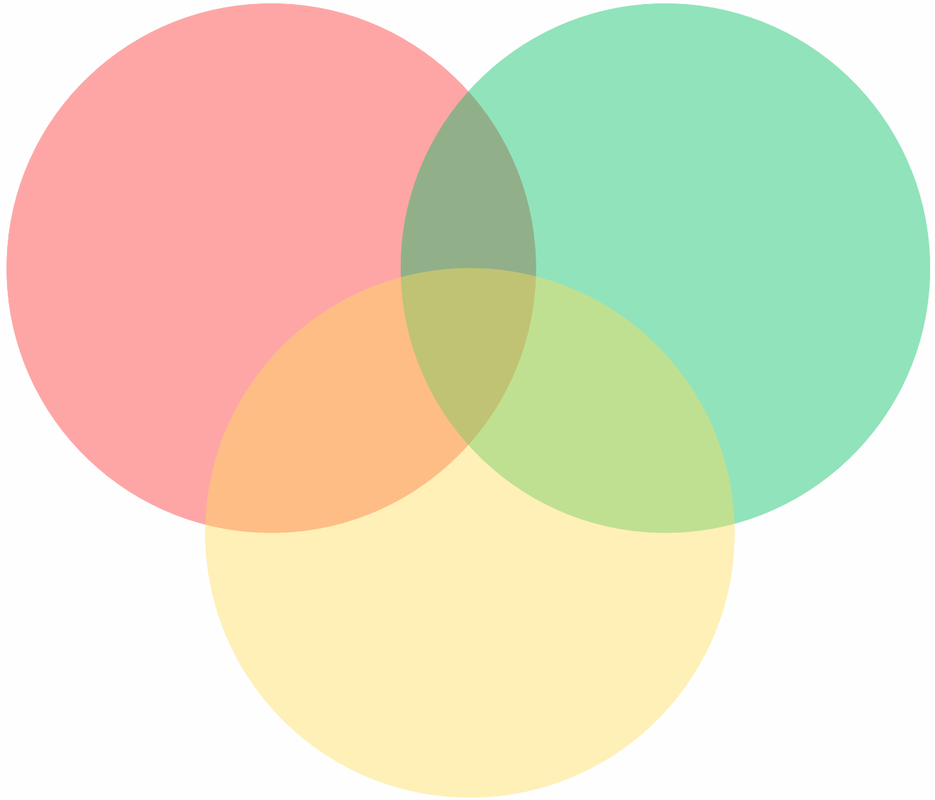
Algoritmo genético para o problema de cobertura única

Gabriel, Yasser, Rafael e Olga - G.Y.R.O.

Modelagem e Otimização Algoritmica - Graduação Informática

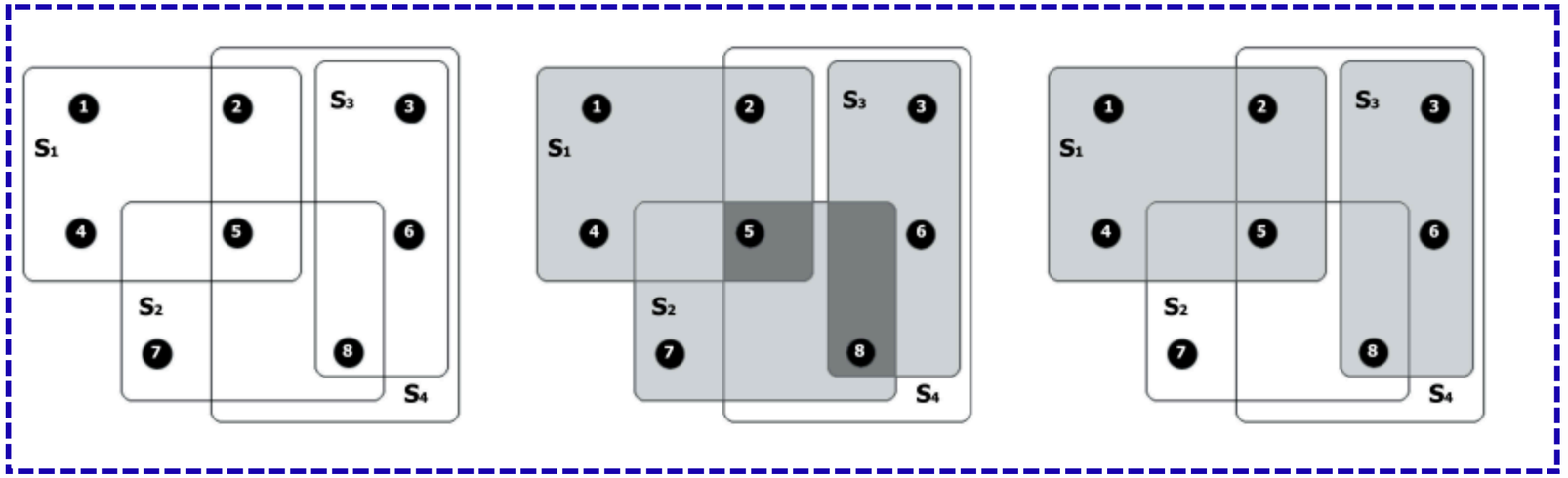
Introdução

Vamos apresentar um método para o problemas de cobertura única, uma variação do clássico problema de cobertura máxima de conjunto.
Com experimentos para avaliar o desempenho do método e a qualidade das soluções fornecidas pelo framework BRKGA



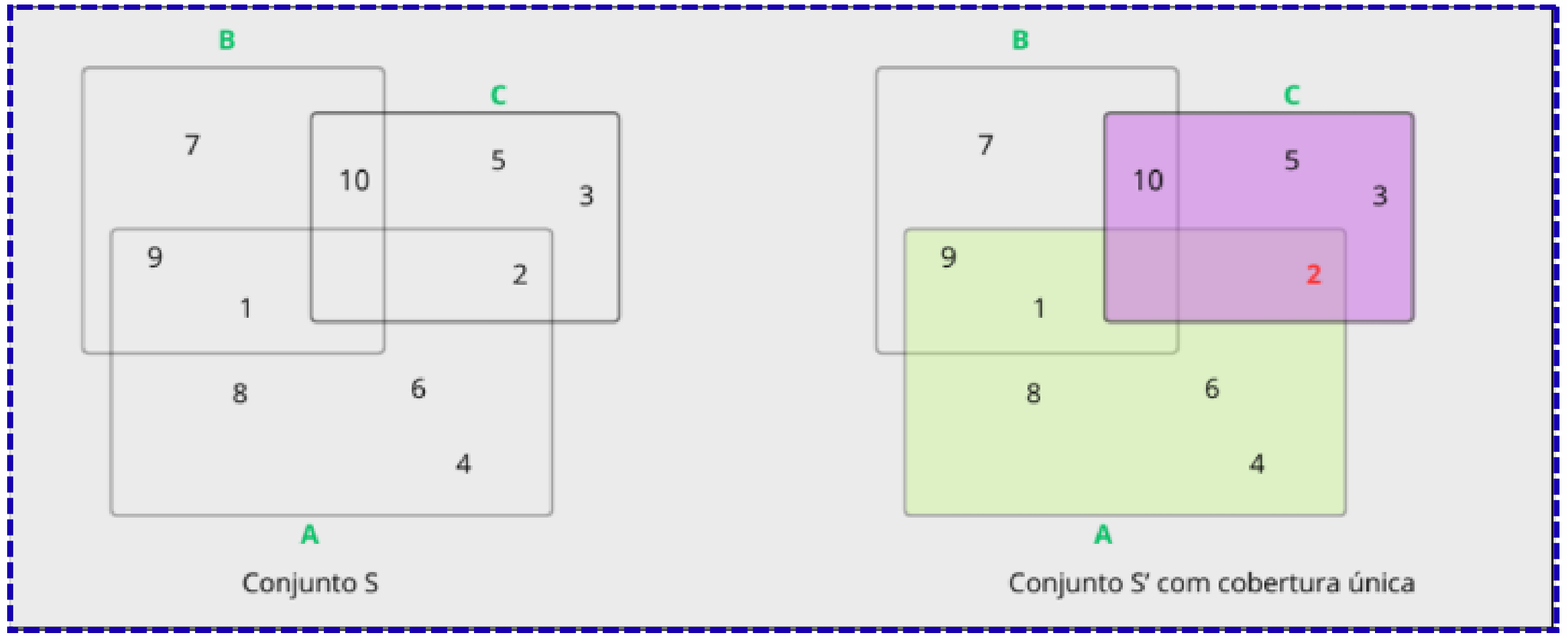
Problema da cobertura única de conjuntos

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.



Exemplo

- 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 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.



Modelagem

Função Objetivo:

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

Sujeito a:

$$\sum_{j=1}^{m} a_{ij} x_j = z_i \quad , \forall i, 1 \leq i \leq n$$

$$y_i \leq z_i \quad , \forall i, 1 \leq i \leq n$$

$$(M-1)y_i + z_i \leq M \quad , \forall i, 1 \leq i \leq n$$

$$y \in \{0,1\}^n, z \in \mathbb{Z}^n, x \in \{0,1\}^m.$$

- **n:** quantidade de elementos
- **m**: quantidade de subconjuntos
- **Aij:** 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 (número total de subconjuntos).

Método usando algoritmo genético

BRKGA

O BRKGA (Biased Random-Key Genetic Algorithm) é uma variação de algoritmo genético que utiliza vetores de chaves aleatórias (números reais entre 0 e 1) para representar soluções. Essas chaves são decodificadas em soluções viáveis por um decodificador específico do problema.

Caracteristicas:

- Decodificador é um componente específico para cada problema que interpreta o vetor de chaves aleatórias e gera a solução real.
- Utilizar um cruzamento viciado, onde os genes dos melhores indivíduos têm maior chance de serem herdados.
- Mantem a diversidade genética por meio da introdução de mutantes aleatórios.

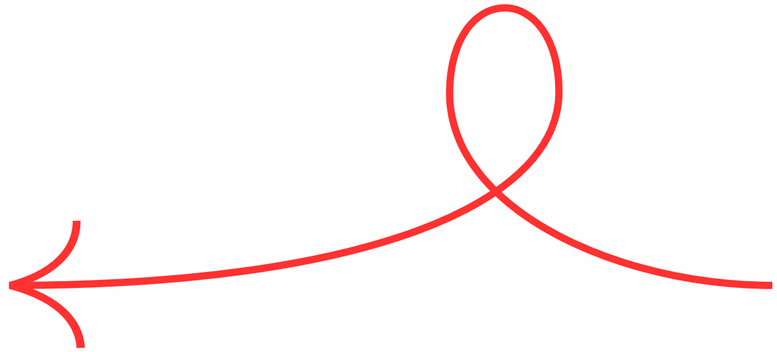
```
Algoritmo UniqueCoverSolverBRKGA(instâncias)
1. Início
2. Para cada instância em 'instâncias':
3.     Ler matriz de cobertura (elementos × subconjuntos)
4.     Inicializar subconjuntos e elementos
5.     Criar decodificador SCPDecoder com parâmetro α
6.     Gerar população inicial com vetores de chaves aleatórias
7.     Avaliar população usando o decodificador (função objetivo)
8.     Definir parâmetros do BRKGA (população, elite, mutantes, etc.)
9.     Enquanto critério de parada não for satisfeito:
10.        Selecionar elite da população atual
11.        Gerar nova população:
12.            - Cruzar elite com não-elite (viés para elite)
13.            - Injetar indivíduos mutantes aleatórios
14.        Decodificar e avaliar nova população
15.        Se nova melhor solução for encontrada:
16.            Atualizar melhor solução global
17.     Armazenar resultados: tempo, valor, GAP, cobertura média
18.     Salvar resultados em arquivo de saída
19. Fim para
20. Retornar resultados de todas as instâncias
21. Fim do algoritmo
```

Execução da instância exemplo

Instâncias

Foram utilizadas 8 instâncias do conjunto Set Covering Problem de tamanhos variados, obtidas na OR-Library de Beasley (2010).

	1	2	3
1	1	1	0
2	1	0	1
3	0	0	1
4	1	0	0
5	0	0	1
6	1	0	0
7	0	1	0
8	1	0	0
9	1	1	0
10	0	1	1



```
10 3
1 1 1
2
1 2
2
1 3
1
3
1
1
1
3
1
1
1
1
2
1
1
2
1 2
2
2 3
```

Funcao objetivo:

$\text{Max} \sum y_i$ — para i=1 até Linhas

Sujeito a:

$\sum a_{ij} \cdot x_j = z_i$ — $\forall i,\ 1 \leq i \leq \text{Linhas}$, para j=1 até Colunas

$y_i \leq z_i$ — $\forall i,\ 1 \leq i \leq \text{Linhas}$

$(M - 1)y_i + z_i \leq M$ — $\forall i,\ 1 \leq i \leq \text{Linhas}$

$y \in \{0,1\}^n,\ z \in \mathbb{Z}^n,\ x \in \{0,1\}^m$

Arquivo de configuração: config.conf

Parâmetros do algoritmo:

```
>  -population_size 50
>  -elite_percentage 0.3
>  -mutants_percentage 0.15
>  -num_elite_parents 2
>  -total_parents 3
>  -bias_type LOGINVERSE
>  -num_independent_populations 3
>  -pr_number_pairs 0
>  -pr_minimum_distance 0.15
>  -pr_type PERMUTATION
>  -pr_selection BESTSOLUTION
>  -alpha_block_size 1.0
>  -pr_percentage 1.0
>  -exchange_interval 200
>  -num_exchange_indivuduals 2
>  -reset_interval 0
```

```
Instância: instancia.txt
Elementos = 10
Subconjutos = 3

Seed: 1753659493.4814298
Tempo máximo de execução (s): 120.00057053565979


Construindo solver BRKGA...

Generando sequência inicial...

Custo inicial: 8

Iter | Resp | Temp
* 1 | 8 | 1.00

Uma solucao factivel foi encontrada.
Melhor resultado:                       8
GAP integralidade:                      20.00%
GAP proporcional à solução encontrada: 25.00%
Peso total:                            -8
Quantidad de nós:                       2
Total de iterações:                     28767
Cobertura media dos elementos:          1.20
Seed:                                   1753659493.4814298
Tempo em segundos:                      120.00s
Tempo em horas:                         0h 2min 0s 1ms
Maior número de iterações sem melhora: 0
Última iteração de melhora:             0
Último momento de melhora:              0.00s
```

```
Conjuntos selecionados:
- Conjunto S 1
    Elementos cobertos: [1, 2, 4, 6, 8, 9]
- Conjunto S 3
    Elementos cobertos: [2, 3, 5, 10]

Detalhes da cobertura por elemento:
Elemento 1: coberto 1 vez(es). Cobertura unica: Sim
Elemento 2: coberto 2 vez(es). Cobertura unica: Não
Elemento 3: coberto 1 vez(es). Cobertura unica: Sim
Elemento 4: coberto 1 vez(es). Cobertura unica: Sim
Elemento 5: coberto 1 vez(es). Cobertura unica: Sim
Elemento 6: coberto 1 vez(es). Cobertura unica: Sim
Elemento 7: coberto 0 vez(es). Cobertura unica: Não
Elemento 8: coberto 1 vez(es). Cobertura unica: Sim
Elemento 9: coberto 1 vez(es). Cobertura unica: Sim
Elemento 10: coberto 1 vez(es). Cobertura unica: Sim
```

Resultados

O resultado dos experimentos com as 8 instâncias 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.

Na Tabela temos a 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).

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

Comparação de resultados

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

Análise sobre o método

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).

Considerações finais

A implementação de uma metaheurística BRKGA para o problema da cobertura única, confirma 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 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.