Representação de soluções

Alexandre Checoli Choueiri

19/08/2023

Representações

Representações

Toda metaheuristica precisa de uma **codificação** (representação) para uma solução. A escolha da codificação da solução desempenha um papel crucial no desempenho dos algoritmos. Podem existir diversos tipos de representação para um mesmo problema. Uma boa representação deve possuir as seguintes características:

1. **Completude**: todas as soluções de um problema (pelo menos as factíveis) devem ser passíveis de representação.

Representações

Toda metaheuristica precisa de uma **codificação** (representação) para uma solução. A escolha da codificação da solução desempenha um papel crucial no desempenho dos algoritmos. Podem existir diversos tipos de representação para um mesmo problema. Uma boa representação deve possuir as seguintes características:

1. **Completude**: todas as soluções de um problema (pelo menos as factíveis) devem ser passíveis de representação.
2. **Conectividade**:é necessário que exista um caminho entre quaisquer duas soluções.

Representações

Toda metaheuristica precisa de uma **codificação** (representação) para uma solução. A escolha da codificação da solução desempenha um papel crucial no desempenho dos algoritmos. Podem existir diversos tipos de representação para um mesmo problema. Uma boa representação deve possuir as seguintes características:

1. **Completude**: todas as soluções de um problema (pelo menos as factíveis) devem ser passíveis de representação.
2. **Conectividade**:é necessário que exista um caminho entre quaisquer duas soluções.
3. **Eficiência**: a representação precisa ser de fácil manipulação pelos operadores de movimento. Também deve facilitar o cálculo da função objetivo e verificação de violação de restrições.

Representações

Existem muitas codificações diretas para famílias de problemas de otimização tradicionais:

- Problema da mochila
- Problema de satisfabilidade booleana
- Programação inteira binária

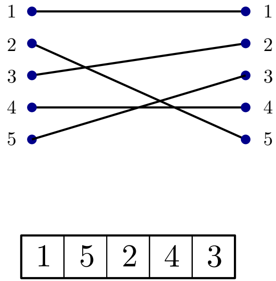
1	0	0	0	1	0	0	1	0	1

Vetor de valores binários

Representações

Existem muitas codificações diretas para famílias de problemas de otimização tradicionais:

- Problema de localização
- Problema de designação



Vetor de valores discretos

Representações

Existem muitas codificações diretas para famílias de problemas de otimização tradicionais:

- Otimização global

$$f(x, y, z, h, j) = 2x + xy - 3z^3 + \frac{h}{i}$$

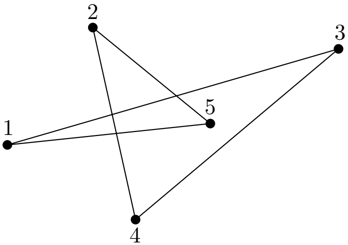
1.0	2.8	5.5	4.1	9.9

Vetor de valores reais

Representações

Existem muitas codificações diretas para famílias de problemas de otimização tradicionais:

- Problema do caixeiro viajante
- Problemas de sequenciamento



1	5	2	4	5

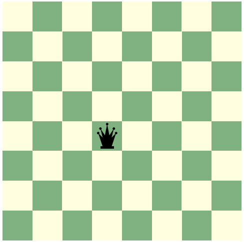
Permutação

Representações

Reduzindo o espaço de busca

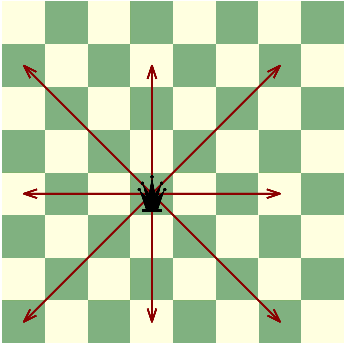
Uma escolha inteligente de codificação para uma solução, pode até mesmo reduzir o espaço de busca do problema. Considere o **problema das 8 rainhas**:

> O problema (ou quebra cabeça) das 8 rainhas consiste em posicionar 8 rainhas em um tabuleiro de xadrez, de forma que nenhuma delas seja capaz de capturar nenhuma outra.



Representações

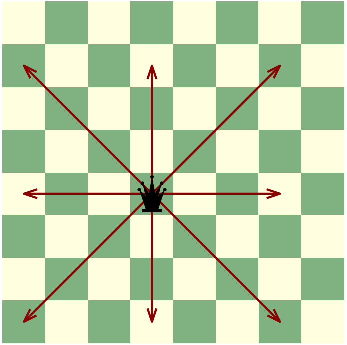
Reduzindo o espaço de busca



Lembrando dos **movimentos** da rainha em um jogo de xadrez.

Representações

Reduzindo o espaço de busca



Como poderia ser a **codificação** para a resposta deste problema?

Representações

Reduzindo o espaço de busca

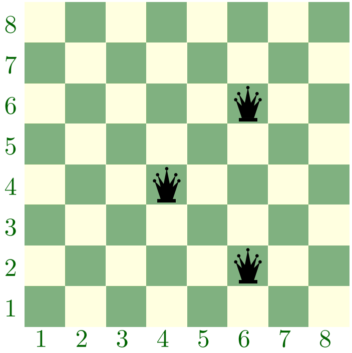
A solução para esse problema representa a **designação de 8 rainhas no tabuleiro**. Inicialmente, poderíamos codificar a solução como um vetor de coordenadas:

$$x = (p_1, p_2, ..., p_8)$$

Em que $p_i = (x_i, y_i)$ representa a posição cartesiana da rainha i, com $x_i, x_j \in 1, ..., 8$.

Representações

Reduzindo o espaço de busca

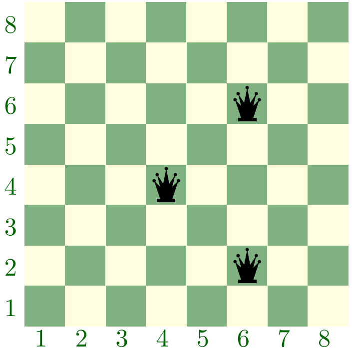


Por exemplo, as rainhas acima seriam representadas como:

$$x = ((4, 4), (6, 2), (6, 6)) \tag{1}$$

Representações

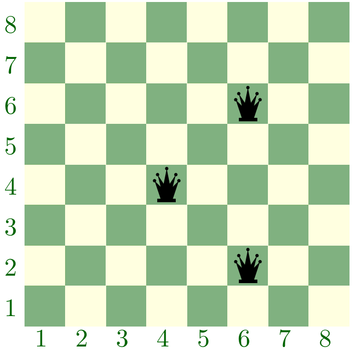
Reduzindo o espaço de busca



Com essa representação, o número de possíveis soluções (espaço de busca) é de 64^8, mais de **4 bilhões de possibilidades**.

Representações

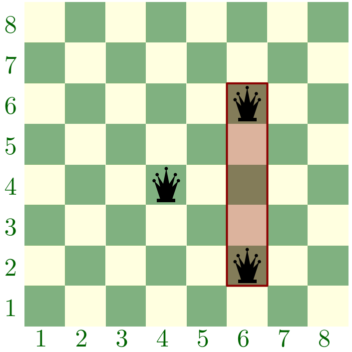
Reduzindo o espaço de busca



Ainda, para verificar a **factibilidade** de cada solução, temos de computar se nenhuma rainha se encontra na **horizontal**, na **vertical** e na **diagonal** entre si.

Representações

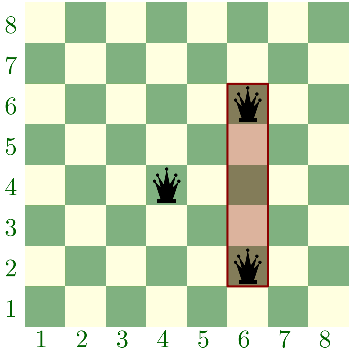
Reduzindo o espaço de busca



Note que **nunca** uma solução em que uma rainha está na mesma coluna de outra será factível.

Representações

Reduzindo o espaço de busca



Note que **nunca** uma solução em que uma rainha está na mesma coluna de outra será factível. Será que conseguimos "absorver"essa característica diretamente na codificação da solução?

Representações

Reduzindo o espaço de busca

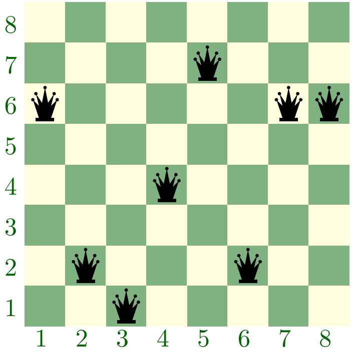
Seja a nova solução representada pela codificação:

$$x = (p_1, p_2, ..., p_8)$$

Em que cada índice i do vetor indica **uma coluna**. E o elemento p_i indica, para aquela coluna, em qual linha existe uma rainha posicionada.

Representações

Reduzindo o espaço de busca

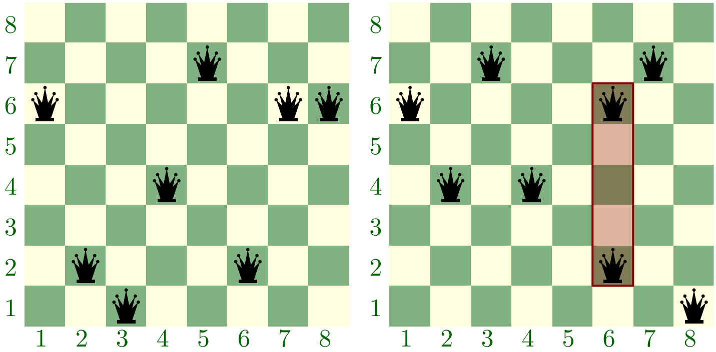


x = (6,2,1,4,7,2,6,6)

Um exemplo de solução com essa codificação é mostrada acima.

Representações

Reduzindo o espaço de busca

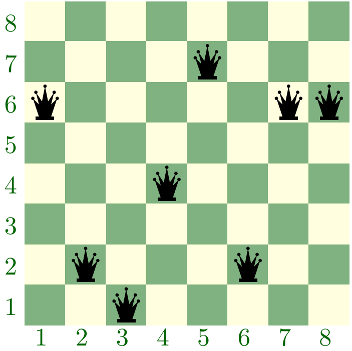


x = (6,2,1,4,7,2,6,6) x = (6,4,7,4,$\boxed{??,??}$,7,1)

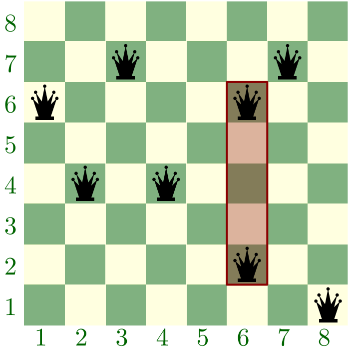
Note, porém, que algumas soluções naturalmente infactíveis, **nem podem ser representadas pela nova codificação**!

Representações

Reduzindo o espaço de busca



x = (6,2,1,4,7,2,6,6)



x = (6,4,7,4,??,??,7,1)

Nenhuma solução com rainhas na mesma coluna pode ser representadas com essa codificação.

Representações

Reduzindo o espaço de busca

Isso reduz o espaço de busca de **4 bilhões** de soluções para 8^8, que é aproximadamente **16 milhões de possibilidades**.

Representações

Reduzindo o espaço de busca

Isso reduz o espaço de busca de **4 bilhões** de soluções para 8^8, que é aproximadamente **16 milhões de possibilidades**.

Será que é possível pensar em uma codificação que reduz ainda mais o espaço de busca?

Representações

Reduzindo o espaço de busca

Isso reduz o espaço de busca de **4 bilhões** de soluções para 8^8, que é aproximadamente **16 milhões de possibilidades**.

> Será que é possível pensar em uma codificação que reduz ainda mais o espaço de busca?

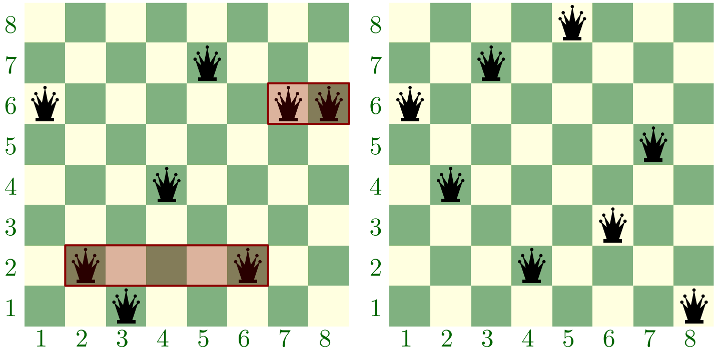
Pela mesma premissa passada, sabemos que nenhuma solução que mantenha **duas rainhas na mesma linha** é uma solução **factível**.
Seja a nova codificação uma permutação de 8 elementos:

$$x = (p_1, p_2, ..., p_8)$$

Em que o significado permanece o mesmo da codificação passada, porém, em se tratando de uma permutação, garantimos que **nenhuma linha será igual**.

Representações

Reduzindo o espaço de busca



$x = (6,\boxed{2},1,4,7,\boxed{2},\boxed{6},\boxed{6})$ $\quad$ $x = (6,4,7,2,8,3,5,1)$

Não conseguimos mais representar a primeira solução nessa nova codificação: ela não é uma permutação de elementos!

Representações

Reduzindo o espaço de busca

Agora, temos que nenhuma solução com duas rainhas na mesma linha e na mesma coluna podem ser representadas. Isso reduz o espaço de busca novamente, de **16 milhões** para $\mathbf{8! = 40.320}$ possibilidades. A única verificação que deve ser feita é **em relação à diagonal das rainhas**.

Representações

Reduzindo o espaço de busca

Agora, temos que nenhuma solução com duas rainhas na mesma linha e na mesma coluna podem ser representadas. Isso reduz o espaço de busca novamente, de **16 milhões** para **8! = 40.320** possibilidades. A única verificação que deve ser feita é **em relação à diagonal das rainhas**.

$$\begin{cases} \textbf{Cod. 1}: \text{4 bilhões de possibilidades} \\ \textbf{Cod. 2}: \text{16 milhões de possibilidades} \\ \textbf{Cod. 3}: \text{40.320 possibilidades} \end{cases}$$

Representações

Reduzindo o espaço de busca

Agora, temos que nenhuma solução com duas rainhas na mesma linha e na mesma coluna podem ser representadas. Isso reduz o espaço de busca novamente, de **16 milhões** para **8! = 40.320** possibilidades. A única verificação que deve ser feita é **em relação à diagonal das rainhas**.

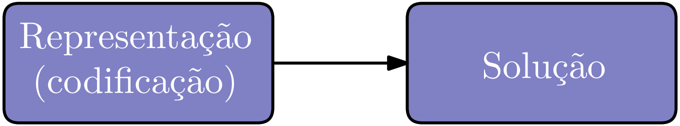
$$\begin{cases} \textbf{Cod. 1}: \text{4 bilhões de possibilidades} \\ \textbf{Cod. 2}: \text{16 milhões de possibilidades} \\ \textbf{Cod. 3}: \text{40.320 possibilidades} \end{cases}$$

Conclusão

A escolha adequada de codificação da solução impacta fortemente no desempenho dos algoritmos. Pelo exemplo acima, conseguimos reduzir substancialmente o espaço de busca do problema, simplesmente alterando a forma de representar uma solução.

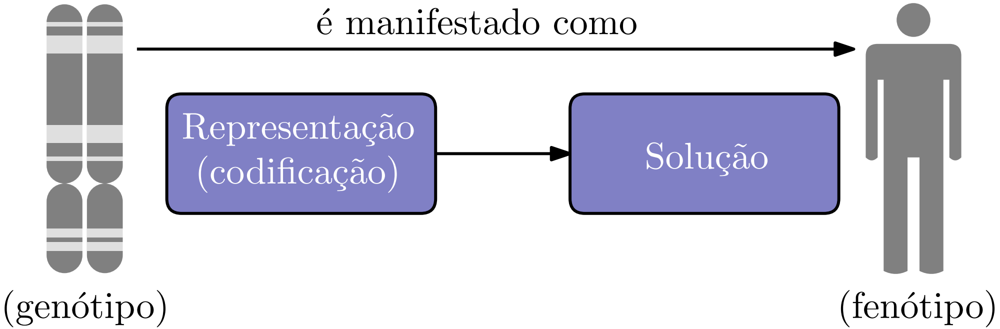
Classificações

Classificações



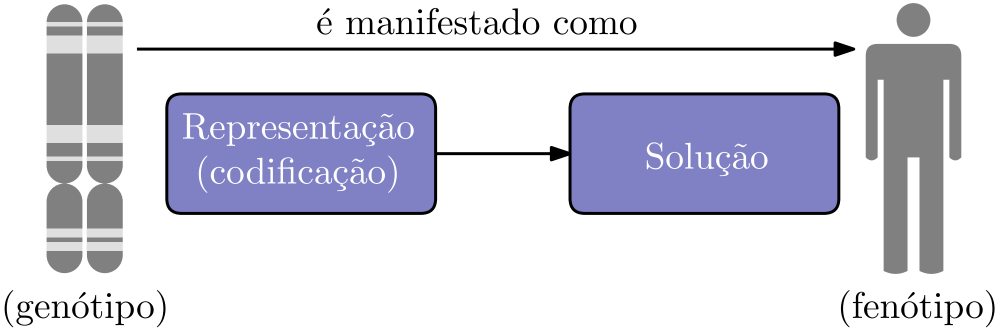
De forma geral, temos uma codificação, que é uma representação computacional de uma solução.

Classificações



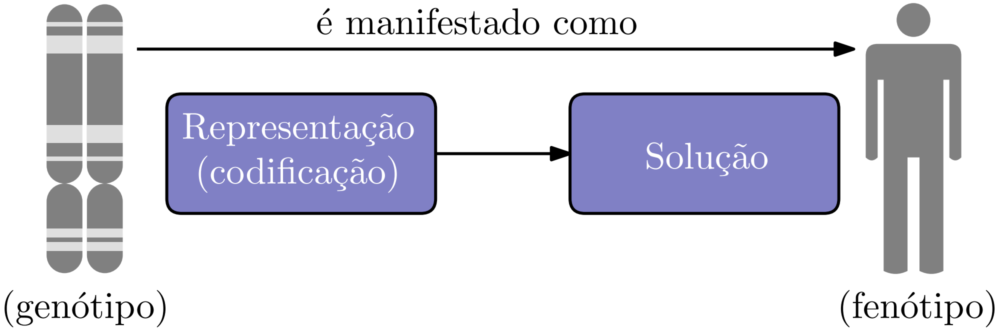
Uma terminologia muito usada para esses conceitos, provenientes da biologia (e dos **algoritmos genéticos**), se refere a representação como **genótipo** e a solução como **fenótipo**.

Classificações



Lembrando que o genótipo diz respeito aos genes de um individuo, e o fenótipo às características do indivíduo que são manifestadas pelos genes. O **fenótipo** é influenciado pelo meio, ou seja, um mesmo **genótipo** pode gerar diferentes **fenótipos**.

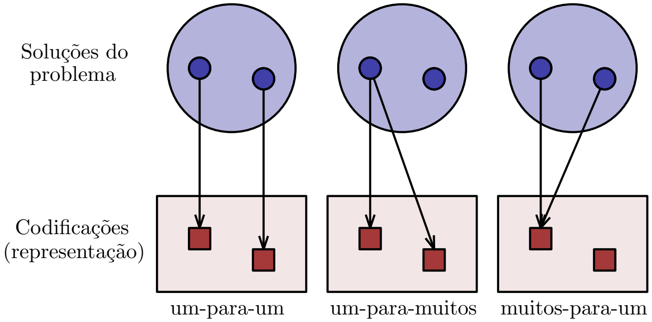
Classificações



Isso nos remete às soluções: pode ocorrer de uma mesma codificação representar diferentes soluções. Todos os casos são apresentados a seguir, no chamado **mapeamento representação-solução**

Classificações

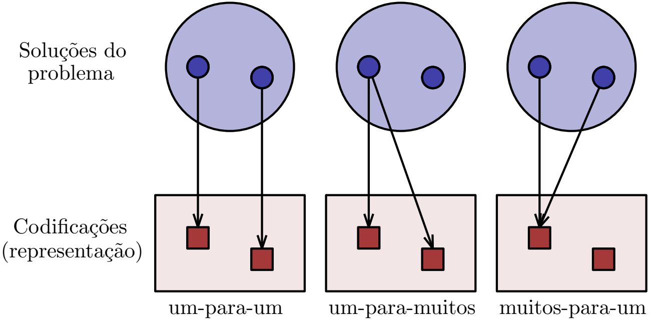
Mapeamento representação-solução



O **mapeamento representação-solução** pode ser de um dos tipos mostrados na imagem acima

Classificações

Mapeamento representação-solução



(Um-para-um): esse é o mapeamento clássico. Uma solução é representada por uma única codificação, e toda codificação representa somente uma solução.

EXEMPLO: A representação permutacional para os problemas do caixeiro viajante se enquadra no mapeamento **um-para-um**. Considerando a codificação:

$$x = [1, 4, 3, 2, 5, 6, 7]$$

Só existe uma solução representada por essa codificação: partir do ponto 1, seguir para o 4, e assim sucessivamente até retornar ao 1 (após o 7).

EXEMPLO: A representação permutacional para os problemas do caixeiro viajante se enquadra no mapeamento **um-para-um**. Considerando a codificação:

$$x = [1, 4, 3, 2, 5, 6, 7]$$

Só existe uma solução representada por essa codificação: partir do ponto 1, seguir para o 4, e assim sucessivamente até retornar ao 1 (após o 7).

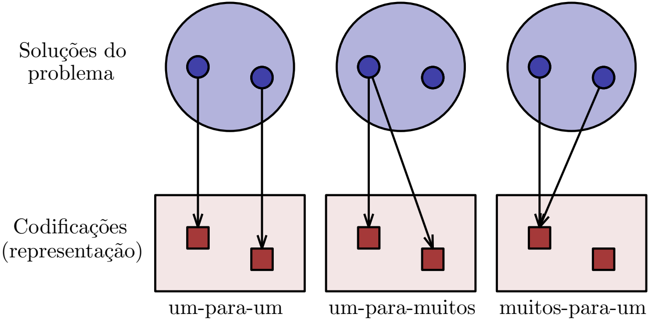
EXEMPLO: A representação binária para os problemas da mochila também são mapeamentos **um-para-um**. Considerando a codificação:

$$x = [0, 1, 1, 0, 1]$$

Só existe uma solução para essa codificação: carregar os itens 1,2 e 3, e nenhum outro na mochila.

Classificações

Mapeamento representação-solução



(Um-para-muitos): nesse mapeamento, uma solução pode ser representada por mais de uma codificação. A redundância na codificação vai aumentar o espaço de busca, e consequentemente pode afetar a eficiência da metaheuristica.

EXEMPLO: Em problemas de particionamento de conjunto, precisamos particionar um conjunto S de elementos em subconjuntos disjuntos s_i, onde $\cup s_i = S$ e $s_i \cap s_j = \emptyset$. O **bin-packing** é um problema de particionamento de conjuntos. Seja os itens:

$$L = [1, 2, 3, 4]$$

Classificações

Mapeamento representação-solução

EXEMPLO: Em problemas de particionamento de conjunto, precisamos particionar um conjunto S de elementos em subconjuntos disjuntos s_i, onde $\cup s_i = S$ e $s_i \cap s_j = \emptyset$. O **bin-packing** é um problema de particionamento de conjuntos. Seja os itens:

$$L = [1, 2, 3, 4]$$

Podemos usar a seguinte codificação para indicar que os itens 1 e 4 fazem parte de um conjunto, e os itens 2 e 3 de outro:

$$ABBA$$

EXEMPLO: Em problemas de particionamento de conjunto, precisamos particionar um conjunto S de elementos em subconjuntos disjuntos s_i, onde $\cup s_i = S$ e $s_i \cap s_j = \emptyset$. O **bin-packing** é um problema de particionamento de conjuntos. Seja os itens:

$$L = [1, 2, 3, 4]$$

Podemos usar a seguinte codificação para indicar que os itens 1 e 4 fazem parte de um conjunto, e os itens 2 e 3 de outro:

$$ABBA$$

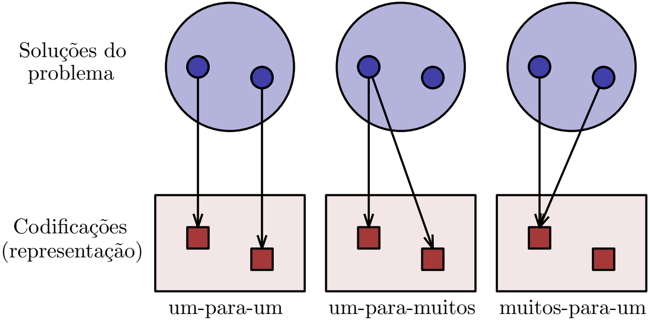
No entanto, a codificação:

$$BAAB$$

Indica a mesma solução. Essa codificação é então um exemplo de **um-para-muitos**.

Classificações

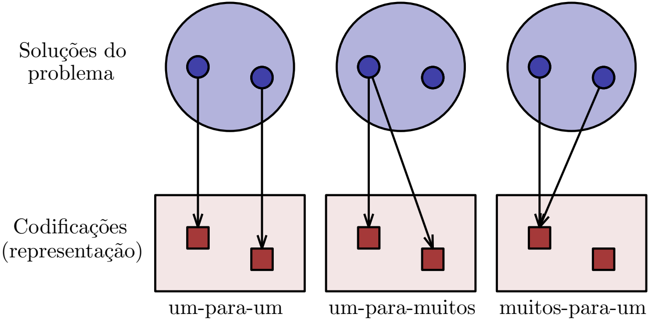
Mapeamento representação-solução



(Muitos-para-um): nesta classe, várias soluções podem ser representadas por uma mesma codificação. Essas codificações são caracterizadas por uma falta de detalhes em relação ao problema original, algumas informações não são explicitamente representadas.

Classificações

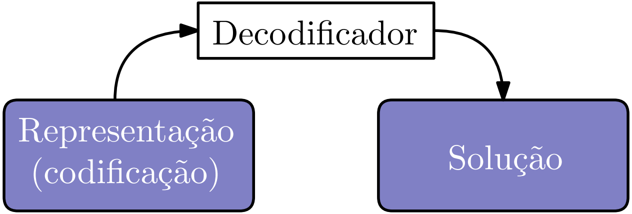
Mapeamento representação-solução



(Muitos-para-um): Em alguns casos isso reduz o espaço de busca original, aumentando a eficiência da metaheuristica. Essa classe de representações também é chamada de **representações indiretas**.

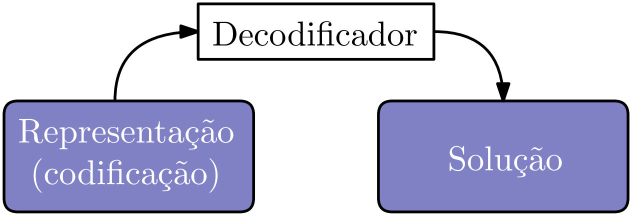
Representações indiretas

Representações indiretas



Ao se usar uma **codificação indireta**, a representação não é uma solução completa do problema. Um **decodificador** é necessário para expressar a solução dada pela codificação. De acordo com as informações na representação, o decodificador pode ter mais ou menos trabalho para derivar uma solução completa.

Representações indiretas



O decodificador pode ser **não-determinístico**, o que implica que a cada decodificação uma solução diferente pode ser derivada. Representações indiretas são muito usadas em problemas com *muitas restrições*, como **scheduling**.

Representações indiretas

EXEMPLO: Considere o problema de **job-shop scheduling (JSS)**:

Definição

Job-shop scheduling: No JSS, um conjunto de **n** jobs devem ser processados em **m** máquinas. Cada job é composto de um número n_i de operações, sendo que cada operação deve ser processada em uma máquina específica. A ordem de execução das operações deve ser respeitada. Um objetivo é minimizar o *makespan*, ou seja, o tempo de termino da última tarefa.

Representações indiretas

	1	**2**	**3**
J1	M1(2)	M2(3)	
J2	M1(2)	M2(1)	M3(1)
J3	M2(4)	M1(1)	M3(2)

Por exemplo, considere a instância para o problema com 3 *jobs* e 3 **máquinas**, dada pela Tabela acima. O job 1 deve ser processado nas máquinas $M1 \rightarrow M2$, levando um tempo de 2 unidades na máquina 1 e 3 na máquina 2.

Representações indiretas

	1	**2**	**3**
J1	M1(2)	M2(3)	
J2	M1(2)	M2(1)	M3(1)
J3	M2(4)	M1(1)	M3(2)

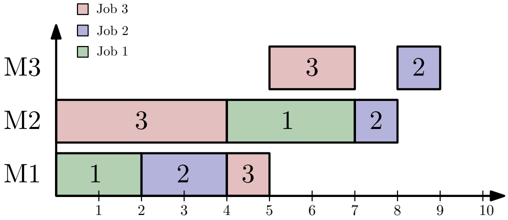
Por exemplo, considere a instância para o problema com 3 *jobs* e 3 **máquinas**, dada pela Tabela acima. O job 1 deve ser processado nas máquinas $M1 \rightarrow M2$, levando um tempo de 2 unidades na máquina 1 e 3 na máquina 2.

Qual poderia ser uma codificação para a solução deste tipo de problema?

Representações indiretas

	1	2	3
J1	M1(2)	M2(3)	
J2	M1(2)	M2(1)	M3(1)
J3	M2(4)	M1(1)	M3(2)

Primeiramente vamos tentar visualizar uma possível solução. Para problemas de scheduling, fica fácil representar as soluções por meio de **gráficos de Gannt**, em que os recursos são as máquinas. Uma possível solução é mostrada na Figura (com custo de 9):



Representações indiretas

Uma representação (direta) para o problema, pode ser uma lista de máquinas. Cada máquina, por sua vez, mantém uma lista em que cada elemento mantém os 3 valores:

$$(j, o, b, e)$$

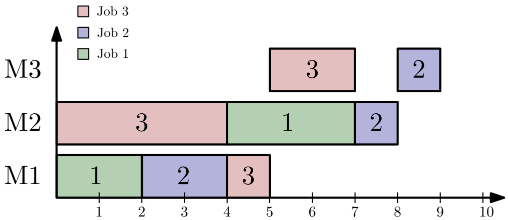
Em que:

$$\begin{cases} \text{j:} & \text{Job que está sendo processado} \\ \text{o:} & \text{Tarefa do job que está sendo processada} \\ \text{b:} & \text{Tempo de inicio do processamento} \\ \text{e:} & \text{Tempo de término do processamento} \end{cases}$$

Representações indiretas

Uma representação (direta) para o problema, pode ser uma lista de máquinas. Cada máquina, por sua vez, mantém uma lista em que cada elemento mantém os 3 valores:

$$\begin{array}{c} \left[\ (3,3,6,7) \quad (2,3,9,9)\ \right] \\ \left[\ (3,1,1,4) \quad (1,2,5,7) \quad (2,2,8,8)\ \right] \\ \left[\ (1,1,1,2) \quad (2,1,3,4) \quad (3,2,5,5)\ \right] \end{array}$$



Representações indiretas

Uma possível representação indireta para soluções, consiste em um vetor de **prioridades de execução dos jobs**, por exemplo:

$$x = \begin{bmatrix} J2 & J1 & J3 \end{bmatrix} \tag{2}$$

Representações indiretas

Uma possível representação indireta para soluções, consiste em um vetor de **prioridades de execução dos jobs**, por exemplo:

$$x = \begin{bmatrix} J2 & J1 & J3 \end{bmatrix} \tag{2}$$

Um decodificador possível poderia funcionar da seguinte forma:

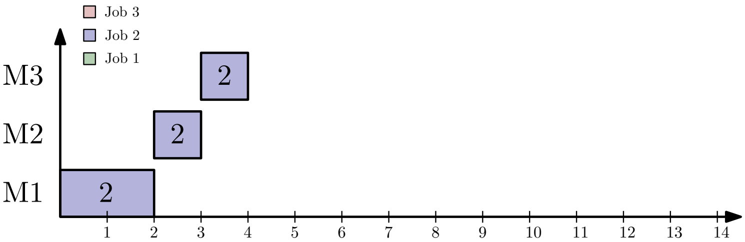
Seguindo a ordem de jobs dada pela representação, atribua cada operação do job **ao instante de tempo mais cedo possível** na máquina.

Representações indiretas

	1	**2**	**3**
J1	M1(2)	M2(3)	
J2	M1(2)	M2(1)	M3(1)
J3	M2(4)	M1(1)	M3(2)

$$x = \left[\begin{array}{ccc} J2 & J1 & J3 \end{array} \right]$$

A solução decodificada para a representação acima fica (com custo de 14):

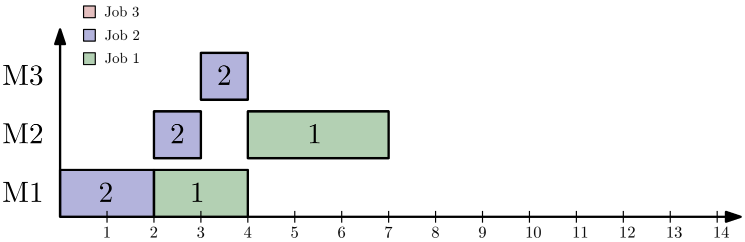


Representações indiretas

	1	**2**	**3**
J1	M1(2)	M2(3)	
J2	M1(2)	M2(1)	M3(1)
J3	M2(4)	M1(1)	M3(2)

$$x = \begin{bmatrix} J2 & J1 & J3 \end{bmatrix}$$

A solução decodificada para a representação acima fica (com custo de 14):

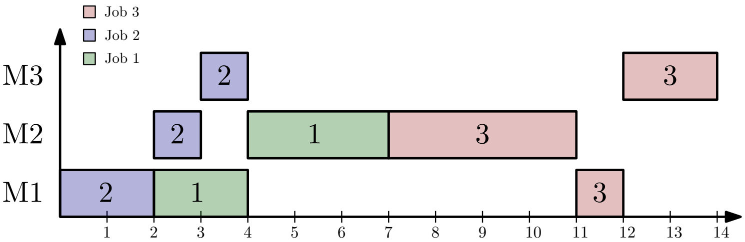


Representações indiretas

	1	2	3
J1	M1(2)	M2(3)	
J2	M1(2)	M2(1)	M3(1)
J3	M2(4)	M1(1)	M3(2)

$$x = \left[\begin{array}{ccc} J2 & J1 & J3 \end{array}\right]$$

A solução decodificada para a representação acima fica (com custo de 14):



Representações indiretas

EXERCÍCIO Algumas considerações em relação a esta representação indireta:

1. Em qual categoria ela se enquadra? Um-para-um, um-para-muitos ou muitos-para-um?
2. Lembrando que uma boa codificação deve possuir as 3 características:
   2.1 Completude
   2.2 Conectividade
   2.3 Eficiência

   Ela satisfaz a primeira característica? Se não, dê um exemplo do porque.
3. Escreva a decodificação de mais 2 representações (quaisquer) para o exemplo acima, qual foi a melhor?

Atividades

Atividade 1

1. Considerando o artigo 2010 - Application of Genetic Algorithm for the Bin Packing Problem with a new representation scheme - Mohamadi, leia e entenda as 3 formas de representar uma solução para o problema do **bin-packing**.
2. Uma codificação indireta muito utilizada é chamada de **random keys** (proposta por Beam). Considerando o artigo 1994 - Genetic Algorithms and Random Keys for Sequencing and Optimization - Beam, entenda como representar uma solução para o *single machine scheduling problem*, e para o *multiple machine scheduling problem* usando as random-keys.
3. Considerando o artigo 2012 - A parallel multi-population biased random-key genetic algorithm for a container loading problem, leia o parágrafo indicado, e relacione com os conceitos explicados em aula.
4. Considerando o artigo 2007 - A hybrid genetic and variable neighborhood descent algorithm for flexible job shop scheduling problems - Gao, entenda a codificação dos autores para o flexible-job-shop scheduling (é a mesma coisa que o JSS, porém cada operação pode ser executada por mais de uma máquina).