$\cup_{m\geq 0}(S(x)\times\Phi)^m$ o conjunto de todas sequencias de pares. Um algoritmo de otimização que não repete avaliações pode ser modelado por uma função $a: d\in D\to\{s\mid s\neq s_i, \text{para } d_i=(s_i,v_i), i\in[|d|]\}$ que mapeia a sequencia atual para a próxima solução a ser avaliada (observe que o algoritmo toma essa decisão em função das soluções anteriormente visitadas e os seus valores). A avaliação de um algoritmo de otimização é através uma função $\Psi(d)$. Ela pode, por exemplo, atribuir a d o valor mínimo encontrado durante a busca.

Teorema 1.2 (Wolpert e Macready (1997))
Para algoritmos a, a', um número de passos m e uma sequencia de valores $v\in\Phi^m$

$$\sum_{f\in\mathcal{F}} P(v\mid f, m, a) = \sum_{f\in\mathcal{F}} P(v\mid f, m, a').$$

O teorema mostra que uma busca genérica não vai ser melhor que uma busca aleatória em média sobre todas funções objetivos. Porém, uma grande fração das funções possíveis não ocorrem na prática (uma função aleatória é incompressível, i.e. podemos especificá-la somente por tabulação, funções práticas muitas vezes exibem localidade). Além disso, algoritmos de busca frequentemente aproveitam a estrutura do problema em questão.

1.2. Representação de soluções

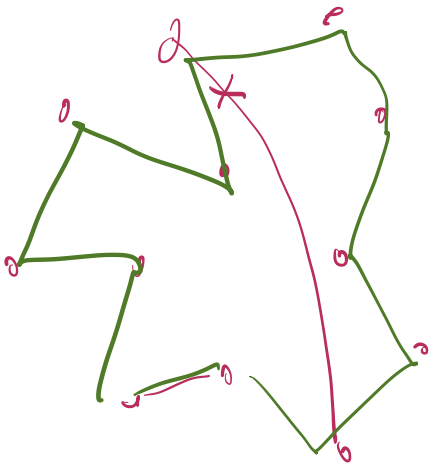
A representação de soluções influencia as operações aplicáveis e a sua complexidade. Por isso a escolha de uma representação é importante para o desempenho de uma heurística. A representação também define o tamanho do espaço de busca, e uma representação compacta (e.g. 8 coordenadas versus permutações no problema das 8-rainhas) é preferível. Para problemas com muitas restrições uma representação implícita que é transformada para uma representação direta por um algoritmo pode ser vantajoso.
Para uma discussão abstrata usaremos frequentemente duas representações elementares. Na *representação por conjuntos* uma solução é um conjunto $S\subseteq U$ de um universo U. Os conjuntos válidos são dados por uma coleção $\mathcal{V}$ de subconjuntos de U. Na *representação por variáveis* uma instância é um subconjunto $I\subseteq U$, e uma solução é uma atribuição de valores de um universo V aos elementos em I.

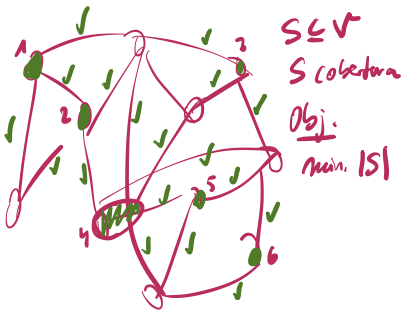
Exemplo 1.1 (Representação do PCV por conjuntos)
Uma representação por conjuntos do PCV sobre um grafo $G=(V,A)$ é o universo de arestas $U=A$, com $\mathcal{V}$ todos subconjuntos que formam ciclos. ◊

Exemplo 1.2 (Representação do PCV por variáveis)
Uma representação por variáveis do PCV sobre um grafo $G=(V,A)$ usa um universo de vértices U. Uma instância $I=V$ atribui a cada cidade a próxima cidade no ciclo. Uma representação alternativa usa $I=[n]$ a atribui a cada variável $i\in I$ a i-ésima cidade no ciclo. ◊

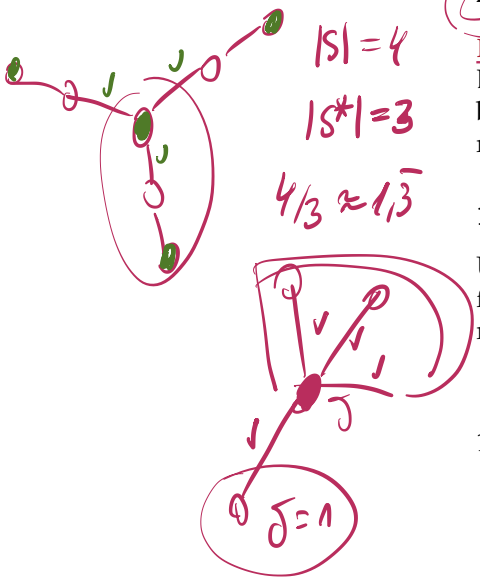
Exemplo 1.3 (Representação da coloração de grafos por variáveis)
Seja U um universo de vértices e C um universo de cores. Uma representação da uma instância $G = (V, A)$ do problema da coloração de grafos usa variáveis $V \subseteq Q$ e atribui cores de C às variáveis. ◊

1.2.1. Reduções de problemas





Cobertura mínima
Min. vertex cover
NP-completo.



Não todos elementos do universo são usados em soluções ótimas: frequentemente eles tem que satisfazer certos critérios para participar numa solução ótima. Isso permite reduzir o problema para um *núcleo*. No problema do PCV, por exemplo, arestas mais longas tem uma baixa probabilidade de fazer parte de uma solução ótima, mas arestas bem curtas aparecem com probabilidade alta na solução ótima. No problema da mochila elementos de alta eficiência (valor por unidade de peso) são mais usados, e de baixa eficiência menos. Se soubéssemos o arco de menor distância não usada numa solução ótima, e de maior distância usado, poderíamos reduzir o problema para um núcleo mais simples. Regras de redução para um núcleo são possíveis em diversos problemas (e.g. o problema da mochila (Kellerer et al. 2004)) e são essenciais para problemas tratáveis por parâmetro fixo (Niedermeier 2002).

Exemplo 1.4 (Núcleo de Buss para cobertura por vértices)
Suponha que estamos interessados numa cobertura pequena de no máximo k vértices num grafo não-direcionado $G = (V, A)$. A cobertura por vértices permite aplicar duas regras de redução:

1. Caso existe um vértice v com $\delta(v) = 0$: ele não faz parte da cobertura, remove o vértice.
2. Caso existe um vértice v com $\delta(v) > k$: ele tem que fazer parte da cobertura, remove o vértice.

Depois de aplicar as regras, temos a seguinte situação: caso $|A| > k^2$ não existe um cobertura de tamanho no máximo k, porque todo vértice cobre no máximo k arestas. Caso contrário $|V| \leq 2k^2$ porque cada aresta possui no máximo dois vértices incidentes diferentes, e logo o problema pode ser resolvido por exaustão em $O(2^{2k^2})$. Junto com as regras de redução temos um algoritmo em tempo $O(n + m + 2^{2k^2})$ ◊

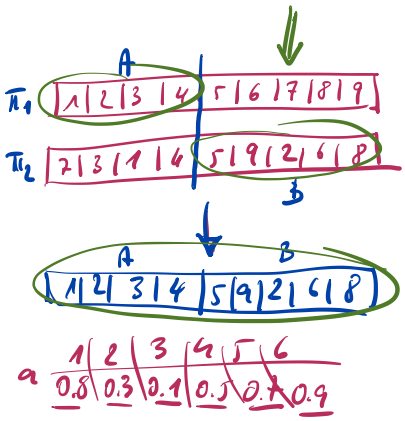
Tratável por parâmetro fixo.
FPT

Princípio de projeto 1.1 (Redução de problemas)
Busca por regras de redução do problema. Procura reduzir o problema para um núcleo. O núcleo pode ser determinado heuristicamente.

1.2.2. Transformações entre representações

Um transformador recebe uma representação de uma solução e transforma ela numa representação diferente. Um algoritmo construtivo randomizado (ver capítulo 3) pode ser visto como um algoritmo que

transforma uma sequencia de números aleatórios em uma solução explicita. Ambas são representações válidas da mesma solução. Essa ideia é aplicada também em algoritmos genéticos, onde a representação fonte se chama *fenótipo* e a representação destino *genótipo*. A ideia de representar uma solução por uma sequência de números aleatórios é usado diretamente em algoritmos genéticos com chaves aleatórias (ver 4.5.6).
Uma transformação é tipicamente sobrejetiva ("many-to-one"), i.e. existem várias representações fonte para uma representação destino. Idealmente, existe o mesmo número de representações fontes para representações destino, para manter a mesma distribuição de soluções nos dois espaços.

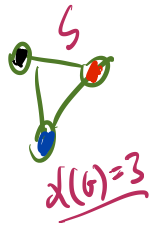


Exemplo 1.5 (Representando permutações por chaves aleatórias)
Uma permutação de n elementos pode ser representada por n números aleatórios reais em $[0, 1]$. Para números aleatórios $a_1, \ldots, a_n$, seja π uma permutação tal que $a_{\pi(1)} \leq \cdots \leq a_{\pi(n)}$. Logo os números a_i representam a permutação π (ou π^{-1}). ◊

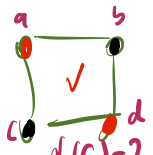
Uma transformação pode ser útil caso o problema possui muitas restrições e o espaço de busca definido por uma representação direta contém muitas soluções inválidas. Em particular buscas locais dependem da geração fácil de soluções. Por isso postulamos o

Princípio de projeto 1.2 (Soluções, Hertz e Widmer (2003))
A geração de soluções deve ser fácil.

Exemplo 1.6 (Coloração de vértices)
Uma representação direta da coloração de vértices pode ser uma atribuição de cores a vértices. Para um limite de no máximo n cores, temos n^n possíveis atribuições, mas várias são infactíveis. Uma representação indireta é uma permutação de vértices. Para uma dada permutação um algoritmo guloso processa os vértices em ordem e atribui o menor cor livre ao vértice atual. A corretude dessa abordagem mostra



Lema 1.1
Para uma dada k-coloração, sejam $C_1 \cup \cdots \cup C_k$ as classes de cores. Ordenando os vértices por classes de cores, o algoritmo guloso produz uma coloração com no máximo k cores.



Prova. Mostraremos por indução que a coloração das primeiras i classes não precisa mais que i cores. Para a primeira classe isso é óbvio. Supõe que na coloração da classe i precisamos usar a cor $i+1$. Logo existe um vizinho com cor i. Mas pela hipótese da indução o vizinho de um vértice da classe $i+1$ não pode ser de uma classe menor. Logo, temos uma aresta entre dois vértices da mesma classe, uma contradição. ■
Com essa representação, todas soluções são válidas. Observe que o tamanho do espaço da busca $n! \approx \sqrt{2\pi n}(n/e)^n$ (por A.5) é similar nas duas representações. ◊