6
Exemplos e Aplicações

Neste capítulo, ilustramos o uso do nosso modelo na especificação de heurísticas e na versão de um conhecido resultado que relaciona heurísticas gulosas com problemas cujo domínio de respostas possui uma determinada estrutura. Apresentamos, ainda, uma generalização da definição de estratégia de busca para permitir a modelagem de meta-heurísticas com caráter estocástico (e.g., *simulated annealing*, algoritmos genéticos etc.).

6.1
Especificando Heurísticas

6.1.1 Usando a linguagem interna de ECF para especificar heurísticas. Usando os termos e predicados definidos no capítulo anterior, podemos especificar propriedades sobre os morfismos *init* e *step* de uma estratégia de busca. Nesta seção, faremos isto para duas estratégias relativamente simples, para fins de ilustração.

6.1.2 Busca em profundidade. Este primeiro exemplo envolve uma estratégia de busca onde as notas dos nós não são importantes:

Uma heurística $\langle\ G, \lambda, \mathit{init}, \mathit{step}\ \rangle$ pode ser classificada como uma estratégia de busca em profundidade quando, a cada passo depois do inicial, a nova corda visitada, caso exista, é

1. uma corda sucessora da corda visitada no passo anterior, ou, caso a corda visitada no passo anterior seja uma folha,
2. uma sucessora não visitada da corda ancestral não explorada mais jovem da corda visitada no passo anterior.

Usaremos o predicado

$$\mathbf{isClosestUnexploredAncestor}(H, \kappa, H', \kappa', T, \theta, G, \lambda)$$

para dizer que a corda $\langle\ H, \kappa\ \rangle$ é a ancestral não-explorada mais jovem da corda $\langle\ H', \kappa'\ \rangle$ (i.e., tal que $\langle\ H, \kappa\ \rangle$ tem alguma sucessora que pertence a $\langle\ G, \lambda\ \rangle$ mas que não pertence à árvore $\langle\ T, \theta\ \rangle$, com $\langle\ T, \theta\ \rangle$ o sub-objeto de $\langle\ G, \lambda\ \rangle$ que armazena as cordas visitadas até então). Este predicado pode ser definido como

$$\begin{array}{l}
\mathbf{isClosestUnexploredAncestor}(H, \kappa, H', \kappa', T, \theta, G, \lambda) \Leftrightarrow \\
\quad \langle\ T, \theta\ \rangle\ \in \mathit{trees}(\langle\ G, \lambda\ \rangle) \wedge \\
\quad \langle\ H', \kappa'\ \rangle\ \in \mathit{strings}(T, \theta) \wedge \\
\quad \mathbf{isAncestor}(H, \kappa, H', \kappa', T, \theta) \wedge \\
\quad \exists \langle\ X, \xi\ \rangle\ \in \mathit{strings}(G, \lambda) : (\\
\qquad \mathbf{isChild}(X, \xi, H, \kappa) \wedge \\
\qquad \langle\ X, \xi\ \rangle\ \notin \mathbf{strings}(T, \theta) \\
\quad) \wedge \\
\quad \forall \langle\ Y, \iota\ \rangle\ \in \mathit{strings}(T, \theta) : (\\
\qquad (\mathbf{isAncestor}(Y, \iota, H', \kappa', T, \theta) \wedge \\
\qquad\quad \mathbf{isDescendant}(Y, \iota, H, \kappa, T, \theta))\ \Rightarrow \\
\qquad \forall \langle\ Z, \zeta\ \rangle\ \in \mathit{strings}(G, \lambda) : (\\
\qquad\quad \mathbf{isChild}(Z, \zeta, Y, \iota, G, \lambda)\ \Rightarrow \\
\qquad\quad \langle\ Z, \zeta\ \rangle\ \in \mathit{strings}(T, \theta) \\
\qquad) \\
\quad)
\end{array}$$

Usando o predicado acima, escrevemos uma fórmula

$$\mathbf{isDepthFirst}(\mathit{init}, \mathit{step}, G, \lambda)$$

que é satisfeita exatamente por estratégias de busca que percorrem $\langle\ G, \lambda\ \rangle$ em profundidade.

Na fórmula, a abreviatura *stage* denota o morfismo definido por recursão simples pela estratégia $\langle\ \mathit{init}, \mathit{step}\ \rangle$, como descrito em 5.5.4.

$$
\begin{array}{l}
\textbf{isDepthFirst}(\langle\ init, step\ \rangle, G, \lambda) \Leftrightarrow \\
\quad \textbf{isSearchStrategy}(init, step, G, \lambda) \wedge \\
\quad \forall n_1, n_2 \in N : (\\
\qquad \pi_1(stage(n_1)) = \pi_1(stage(n_2)) \Rightarrow stage(n_1) = stage(n_2) \\
\quad) \\
\quad \wedge \\
\quad \forall \langle\ H_1, \kappa_1\ \rangle, \langle\ H_2, \kappa_2\ \rangle\ \in strings(F) : \\
\quad \forall n_1, n_2, n_3 \in N : (\\
\qquad (n_3 = s(n_2)\ \wedge\ n_2 = s(n_1)\ \wedge\ \pi_1(stage(n_3)) \neq \pi_1(stage(n_2)) \wedge \\
\qquad\quad \textbf{wasAdded}(H_2, \kappa_2, \pi_1(stage(n_3)), \pi_1(stage(n_2))) \wedge \\
\qquad\quad \textbf{wasAdded}(H_1, \kappa_1, \pi_1(stage(n_2)), \pi_1(stage(n_1))) \\
\qquad) \\
\qquad \Rightarrow \\
\qquad (\textbf{isChild}(H_2, \kappa_2, H_1, \kappa_1, G, \lambda) \vee \\
\qquad\quad (\textbf{isLeaf}(H_1, \kappa_1, G, \lambda) \wedge \\
\qquad\qquad \exists \langle\ H, \kappa\ \rangle\ \in strings(G, \lambda) : (\\
\qquad\qquad\quad \textbf{isChild}(H_2, \kappa_2, H, \kappa, G, \lambda) \wedge \\
\qquad\qquad\quad \textbf{isClosestUnexploredAncestor} \\
\qquad\qquad\qquad (H, \kappa, H_1, \kappa_1, \pi_1(stage(n_2)), G, \lambda) \\
\qquad\qquad) \\
\qquad\quad) \\
\qquad) \\
\quad)
\end{array}
$$

As seguintes observações se aplicam:

- Lembre-se de que $stage(n)$ representa um par $(\langle\ T, \theta\ \rangle, \langle\ H, \kappa\ \rangle)$ composto pela árvore de cordas visitadas até o passo n e pela corda visitada no passo n. Assim, $\pi_1(stage(n))$ representa a árvore de cordas visitadas até o passo n;
- Uma corda nunca é visitada mais do que uma vez; logo, para todo n menor que o número da última iteração, $\pi_1(stage(n)) \neq \pi_1(stage(sn))$. Aliás, o predicado **isIrrevocable**, abaixo, diz exatamente que a estratégia $\langle\ init, step\ \rangle$ nunca

visita a mesma corda duas vezes durante a busca:

$$\begin{array}{l}\textbf{isIrrevocable}(\langle\ init, step\ \rangle, G, \lambda) \Leftrightarrow \\ \quad \forall n_1, n_2 \in N : (\\ \quad\quad \pi_1(stage(n_1)) = \pi_1(stage(n_2)) \Rightarrow stage(n_1) = stage(n_2) \\ \quad)\end{array}$$

- Para todo $n \geq 2$, $\langle\ H_2, \kappa_2\ \rangle$ é a corda acrescentada no passo $n-1$, e $\langle\ H_1, \kappa_1\ \rangle$ é a corda acrescentada no passo $n-2$;
- Não é especificada qualquer condição de término. A busca pode continuar visitando novas cordas enquanto isto for possível.

6.1.3 Busca gulosa. Em uma heurística gulosa, a corda visitada a cada passo n ($n > 0$) é a melhor sucessora da corda visitada no passo $n-1$, desde que esta sucessora seja melhor ou de mesma qualidade que a corda visitada no passo $n-1$. Como na busca em profundidade, uma corda nunca é visitada mais do que uma vez durante a busca.

O seguinte predicado é satisfeito exatamente por heurísticas gulosas. Observe que, no caso de empate entre duas ou mais sucessoras de uma corda, o predicado não especifica qual das sucessoras deve ser escolhida.

$$
\begin{array}{l}
\textbf{isGreedy}(\langle\ init, step\ \rangle, G, \lambda) \Leftrightarrow \\
\quad \textbf{isSearchStrategy}(init, step, G, \lambda) \wedge \\
\quad \textbf{isIrrevocable}(\langle\ init, step\ \rangle, G, \lambda) \wedge \\
\quad \forall \langle\ H, \kappa\ \rangle\ \in \textbf{strings}(G, \lambda) : (\\
\qquad \textbf{isRoot}(H, \kappa, G, \lambda)\ \Rightarrow \\
\qquad\quad grade(\pi_2(init(\star)), G, \lambda)\ \leq\ grade(H, \kappa, G, \lambda) \\
\quad) \\
\quad \wedge \\
\quad \forall n \in N : (\\
\qquad stage(n) = stage(sn)\ \Leftrightarrow \\
\qquad\quad \forall \langle\ H, \kappa\ \rangle\ \in strings(G, \lambda) : (\\
\qquad\qquad \textbf{isChild}(H, \kappa, \pi_2(stage(n)), G, \lambda)\ \Rightarrow \\
\qquad\qquad\quad grade(H, \kappa, G, \lambda) > grade(\pi_2(stage(n))) \\
\qquad\quad) \\
\quad) \\
\quad \wedge \\
\quad \forall \langle\ H_1, \kappa_1\ \rangle, \langle\ H_2, \kappa_2\ \rangle\ \in strings(G, \lambda) : \\
\quad \forall n_1, n_2, n_3 \in N : (\\
\qquad (n_3 = s(n_2)\ \wedge\ n_2 = s(n_1)\ \wedge\ \pi_1(stage(n_3)) \neq \pi_1(stage(n_2)) \wedge \\
\qquad\quad \textbf{wasAdded}(H_2, \kappa_2, \pi_1(stage(n_3)), \pi_1(stage(n_2))) \wedge \\
\qquad\quad \textbf{wasAdded}(H_1, \kappa_1, \pi_1(stage(n_2)), \pi_1(stage(n_1))) \\
\qquad) \\
\qquad \Rightarrow \\
\qquad (\textbf{isChild}(H_2, \kappa_2, H_1, \kappa_1, G, \lambda) \wedge \\
\qquad\quad grade(H_2, \kappa_2, G, \lambda)\ \leq\ grade(H_1, \kappa_1, G, \lambda) \wedge \\
\qquad\quad \forall \langle\ H, \kappa\ \rangle\ \in strings(G, \lambda) : (\\
\qquad\qquad \textbf{isChild}(H, \kappa, H_1, \kappa_1) \\
\qquad\qquad\quad \Rightarrow\ grade(H_2, \kappa_2, G, \lambda)\ \leq\ grade(H, \kappa, G, \lambda) \\
\qquad\quad) \\
\qquad) \\
\quad)
\end{array}
$$

As seguintes observações podem ser feitas:

– A estratégia escolhe como nó inicial a raiz de melhor qualidade dentre todas as raízes possíveis. Se isto não for realista, a terceira fórmula da conjunção deve ser eliminada;

– A busca termina quando a corda atual não possui sucessora de melhor ou igual qualidade.

6.2
Matróides e Heurísticas Gulosas

6.2.1 Apresentação. Nesta seção, apresentaremos a versão, no nosso modelo, de um conhecido resultado relacionando a estratégia de busca gulosa e problemas de otimização cujas respostas são conjuntos satisfazendo certas condições. Não se trata de provar o mesmo resultado no nosso modelo, mas sim de verificar que o nosso modelo traduz adequadamente este relacionamento entre problemas e heurísticas, provado em outro contexto.

6.2.2 Definição: sistema de conjuntos com função-objetivo. Um sistema de conjuntos é um par $(S, \mathcal{F})$, com S um conjunto finito e $\mathcal{F}$ uma coleção (não-vazia) hereditária de subconjuntos de S, i.e., tal que, para todo A e para todo B

$$A \in \mathcal{F} \ \wedge \ B \subseteq A \ \Rightarrow \ B \in \mathcal{F}$$

Uma função-objetivo para $\mathcal{F}$ é uma função $w : \mathcal{F} \to \mathbb{N}$ finitamente aditiva, i.e., satisfazendo

$$w(A \cup B) = w(A) \cup w(B)$$

sempre que A e B são conjuntos disjuntos, elementos de $\mathcal{F}$, com $A \cup B$ também elemento de $\mathcal{F}$.

Um sistema de conjuntos com função-objetivo é uma tripla $(S, \mathcal{F}, w)$ com $(S, \mathcal{F})$ um sistema de conjuntos e w uma função-objetivo para $\mathcal{F}$.

6.2.3 O significado de um sistema de conjuntos com função-objetivo. Observe que, como $\mathcal{F}$ é hereditário e w é finitamente aditiva, cada função-objetivo $w : \mathcal{F} \to \mathbb{N}$ corresponde a uma função $w' : \bigcup \mathcal{F} \to \mathbb{N}$, e vice-versa. Em outras palavras, para todo conjunto $A \in \mathcal{F}$, o valor de $w(A)$ é determinado pelos elementos de A.

Um sistema de conjuntos com função-objetivo $(S, \mathcal{F}, w)$ pode ser visto como uma instância de um problema de maximização: resolver $(S, \mathcal{F}, w)$ é achar um conjunto $M \in \mathcal{F}$ tal que $w(M) \geq w(M')$ para todo $M' \in \mathcal{F}$.

6.2.4 Definição: matróide com função-objetivo. Um matróide é um sistema de conjuntos $(S, \mathcal{F})$ satisfazendo a condição adicional

$$\begin{array}{l} \forall A, B \in \mathcal{F} : (\\ \quad |A| = |B| + 1 \ \Rightarrow \\ \qquad \exists a \in A \setminus B : B \cup \{a\} \in \mathcal{F} \\ ) \end{array}$$

Um matróide com função-objetivo é um sistema de conjuntos com função-objetivo $(S, \mathcal{F}, w)$ com $(S, \mathcal{F})$ um matróide.

6.2.5 Definição: algoritmo guloso. O algoritmo guloso tenta resolver a instância do problema de maximização associado ao sistema de conjuntos com função-objetivo $(S, \mathcal{F}, w)$ retornando um conjunto $A \in \mathcal{F}$ construído elemento a elemento como se segue:

1. $A \leftarrow \varnothing$
2. Se existir $a \in S$ tal que
   $a \notin A$,
   $A \cup \{a\} \in \mathcal{F}$, e
   $w(A \cup \{a\}) = \max\{w(A \cup \{x\}) \mid A \cup \{x\} \in \mathcal{F}\}$
   então
   $A \leftarrow A \cup \{a\}$
   ir para 2.
   senão
   retornar A

6.2.6 Exemplo. Considere o exemplo onde

- S é o conjunto de arestas de um grafo finito conexo g, tal que a cada aresta a está associado um valor natural $w(a)$, o peso de a;
- $\mathcal{F}$ contém exatamente os conjuntos de arestas que formam subflorestas de g;

– w associa a cada subfloresta $A \in \mathcal{F}$ a soma dos pesos das arestas de A.

A instância do problema de otimização associado a $(S, \mathcal{F}, w)$ consiste em encontrar a árvore geradora de peso máximo de g.[1] Verifica-se que $(S, \mathcal{F}, w)$ é um matróide com função-objetivo, e que o algoritmo guloso sempre retorna a resposta ótima para este problema. Este fato é consequência do seguinte resultado geral, inicialmente provado em [34]:

6.2.7 Teorema: Seja $(S, \mathcal{F}, w)$ um sistema de conjuntos com função-objetivo. Então, o algoritmo guloso retorna uma resposta ótima para a instância do problema de otimização associado se e somente se $(S, \mathcal{F}, w)$ é um matróide com função-objetivo.

6.2.8 Versão categórica. As definições e resultados a seguir traduzem o teorema acima para o nosso modelo baseado em categorias e *topoi*:

6.2.9 Definição: a categoria SC de sistemas de conjuntos com função-objetivo. Os objetos de **SC** são todos os sistemas de conjuntos com função-objetivo, e um morfismo $(S, \mathcal{F}, w) \xrightarrow{f} (S', \mathcal{F}', w')$ é uma função $f : S \to S'$ que, quando estendida de maneira óbvia para $f : \wp(S) \to \wp(S')$, preserva a pertinência em $\mathcal{F}$, i.e., para todo $A \subseteq S$,

$$A \in \mathcal{F} \;\Rightarrow\; f(A) \in \mathcal{F}'$$

e que preserva a otimalidade segundo w, i.e., para todo $A \in \mathcal{F}$,

$$w(A) = \max\{w(X) \mid X \in \mathcal{F}\} \;\Rightarrow\; w'(f(A)) = \max\{w'(X') \mid X' \in \mathcal{F}'\}$$

6.2.10 Definição: a categoria ProbOt de problemas de otimização com uma instância. **ProbOt** é a subcategoria plena de **Prob** cujos objetos são problemas

[1] A subfloresta geradora de peso máximo de g deve ser uma árvore porque g é um grafo conexo. Para achar a árvore geradora de peso mínimo de g, basta maximizar a função-objetivo w' com

$$w'(A) = T - w(A)$$

onde T é a soma dos pesos de todas as arestas de g.

da forma

$$\langle\ \{w\}, \wp(S), p\ \rangle$$

onde

- S é um conjunto finito;
- $w : \wp(S) \to \mathbb{N}$ é uma função parcial, finitamente aditiva, definida para no mínimo um elemento;
- $p = \{(w, A) \mid w(A) = \max\{w(A') \mid A' \in \wp(S), w(A')\downarrow\}\}$

A intuição de que cada sistema de conjuntos com função-objetivo corresponde a um problema de otimização com uma instância, e vice-versa, é formalizada a seguir:

6.2.11 Proposição: SC e $\mathbf{ProbOt}^{op}$ são isomorfas. É um isomorfismo o funtor $I : \mathbf{SC} \to \mathbf{ProbOt}^{op}$ tal que

$$I(S, \mathcal{F}, w) = \langle\ \{\overline{w}\}, \wp(S), p\ \rangle$$

onde, para todo $A \in \wp(S)$,

$$\overline{w}(A) = \begin{cases} w(A) & \text{se } A \in \mathcal{F} \\ \uparrow & \text{senão} \end{cases}$$

e

$$I((S, \mathcal{F}, w) \xrightarrow{f} (S', \mathcal{F}', w')) = \\ \langle\ \{\overline{w}\}, \wp(S), p\ \rangle \xrightarrow{(\tau,\sigma)^{op}} \langle\ \{\overline{w'}\}, \wp(S'), p'\ \rangle$$

com

$$\tau(\overline{w'}) = \overline{w}$$

e, para todo $A \in \wp(S)$,

$$\sigma(A) = f(A)$$

O funtor inverso de I é $I^{-1} : \mathbf{ProbOt}^{op} \to \mathbf{SC}$ tal que

$$I^{-1}(\langle\ \{w\}, \wp(S), p\ \rangle) = (S, \{A \mid w(A)\downarrow\}, \underline{w})$$

onde

$$\underline{w}(A) = w(A)$$

e

$$I^{-1}(\langle\ \{w\}, \wp(S), p\ \rangle \xrightarrow{(\tau,\sigma)^{op}} \langle\ \{w'\}, \wp(S'), p'\ \rangle) =$$
$$(S, \{A \mid w(A)\downarrow\}, \underline{w}) \xrightarrow{f} (S', \{A' \mid w'(A')\downarrow\}, \underline{w'})$$

com

$$f(A) = \sigma(A)$$

para todo $A \in \{A \mid w(A)\downarrow\}$.

6.2.12 Definição: a categoria Matr de matróides com função-objetivo. **Matr** é a subcategoria plena de **SC** cujos objetos são todos os matróides com função-objetivo.

6.2.13 Espaços e estratégias de busca para ProbOt. Desejamos agora restringir nosso modelo à categoria **ProbOt** de problemas de otimização com uma instância. Onde antes considerávamos $\mathbf{Prob}_0$ uma subcategoria de **Prob**, agora consideramos $\mathbf{Prob}_0$ uma subcategoria de **ProbOt**:

- $\mathbf{Prob}_0$, como em 3.3.7, é uma subcategoria rarefeita e esquelética de **ProbOt** tal que o problema
$$P_1 = \langle\ \{\star\}, \{\star\}, \{(\star, \star)\}\ \rangle$$
é o objeto inicial de $\mathbf{Prob}_0$. Observe que P_1 é (isomorfo a) um problema de otimização com uma instância; i.e., $P_1 \cong \langle\ \{w\}, \{\varnothing\}, \{w, \varnothing\}\ \rangle$;
- Seja P um objeto de $\mathbf{Prob}_0$; uma estratégia de construção de respostas $\langle\ G, \lambda\ \rangle$ escolhe uma floresta de respostas para P, com nós rotulados por pares (C, k), com $C \subseteq \wp(S)$ um conjunto de subconjuntos de S, e $k \in \mathbb{N}$;
- Uma estratégia de buscas $\langle\ init, step\ \rangle$ determina como a floresta de respostas de P será percorrida em busca de uma resposta.

6.2.14 Definição: heurística gulosa para problemas em ProbOt. Considerando o contexto acima, definiremos uma heurística gulosa

$$\langle\ G, \lambda, init, step\ \rangle$$

para o problema de otimização com uma instância $P = \langle\ \{w\}, \wp(S), p\ \rangle$:

- $\langle G, \lambda \rangle$ é a estratégia de construção de floresta de respostas que associa a P uma floresta adequada para ser percorrida pela busca gulosa. Esta floresta é, na verdade, uma árvore onde
  - A raiz é um nó rotulado pelo par

$$\left(\{\varnothing\}, \sum_{\substack{X \in \wp(S) \\ w(X)\downarrow}} w(X) \right)$$

  - Cada nó rotulado pelo par $(\{A\}, k)$, com A um elemento de $\wp(S)$, possui o seguinte conjunto de filhos:

$$\{(\{A \cup \{x\}\}, k - w(x)) \mid w(A \cup \{x\})\downarrow\}$$

  Observe que, nesta árvore, qualquer busca se inicia com o conjunto vazio (que possui a nota máxima). A passagem de um nó para um filho corresponde à adição de um elemento x ao conjunto corrente A tal que $A \cup \{x\}$ é uma resposta viável. A nota do filho é sempre menor ou igual à nota do pai, pois o acréscimo de x corresponde a uma diminuição de $w(x)$ unidades na nota. (Lembre-se de que notas menores significam melhor qualidade.)
- A estratégia de busca $\langle init, step \rangle$ deve satisfazer a especificação em 6.1.3.

6.2.15 Teorema: **Matr** é isomorfa à subcategoria plena de **ProbOt** cujos objetos são problemas de otimização para os quais a heurística gulosa retorna respostas ótimas.

Mais precisamente, para todo problema de otimização com uma instância P, temos que $I^{-1}(P)$ é um matróide[2] se e somente se toda heurística

$$\langle G, \lambda, init, step \rangle$$

(com $\langle G, \lambda \rangle$ a estratégia de construção de florestas de respostas descrita acima, e $\langle init, step \rangle$ uma estratégia de busca gulosa satisfazendo a especificação em 6.1.3) retornar uma resposta exata para P.

[2] I^{-1} é o isomorfismo descrito em 6.2.11.

Esboço de prova. Basta mostrar que a sequência de nós visitados para a heurística gulosa especificada em 6.1.3 é igual à sequência de nós visitados pelo algoritmo guloso definido em 6.2.5, o que pode ser provado por indução sobre os comprimentos dos prefixos da sequência.

6.3
Meta-heurísticas

6.3.1 Meta-heurísticas com caráter estocástico. A maior parte das meta-heurísticas encontradas na literatura, como *simulated annealing*, busca tabu, algoritmos genéticos e outras, possui algum elemento probabilístico. Nosso modelo, conforme definido até agora, não é capaz de representar este tipo de estratégia de busca. Esta seção apresenta uma generalização da nossa definição de estratégia de busca para permitir a modelagem destas meta-heurísticas.

6.3.2 Estratégias de busca estocásticas. Embora um espaço de busca seja definido da mesma maneira para estratégias tradicionais que para meta-heurísticas, o comportamento de uma estratégia de busca não pode mais ser representado por uma sequência de pares da forma $(\langle\, T, \theta \,\rangle,\ \langle\, H, \kappa \,\rangle)$, com $\langle\, T, \theta \,\rangle$ a árvore de cordas já visitadas e $\langle\, H, \kappa \,\rangle$ a corda atualmente visitada pela busca. Em vez disso, em um dado instante, a busca pode estar visitando qualquer uma de um conjunto de cordas com probabilidade não-nula de serem visitadas. Isto nos motiva a definir uma estratégia de busca estocástica como uma sequência de conjuntos:

6.3.3 Definição: estratégia de busca estocástica. Como na definição anterior de estratégia de busca (5.5.2), dada uma estratégia de construção de florestas de respostas $\langle\, G, \lambda \,\rangle$, seja C o objeto

$$\coprod_{\langle\, T,\theta \,\rangle \,\in trees(\langle\, G,\lambda \,\rangle)} strings(\langle\, T, \theta \,\rangle)$$

de **ECF**, cujos elementos são todos os pares da forma

$$(\langle\, T, \theta \,\rangle, \langle\, H, \kappa \,\rangle)$$

com $\langle\ T, \theta\ \rangle$ uma árvore de $\langle\ G, \lambda\ \rangle$, e $\langle\ H, \kappa\ \rangle$ uma corda de $\langle\ T, \theta\ \rangle$. Seja o objeto C'

$$C \times \{q \in Q \mid 0 < q \leq 1\}$$

com Q o objeto de números racionais descrito em 4.6. Os elementos de C' são triplas da forma

$$(\langle\ T, \theta\ \rangle, \langle\ H, \kappa\ \rangle, q)$$

com q um racional maior que 0 e menor ou igual a 1. Estamos interessados em conjuntos não-vazios e finitos de triplas em que a soma de todos os q seja 1. Assim, definimos o objeto S como

$$\left\{ A \in PC' \;\middle|\; A \neq \varnothing, A \text{ finito}, \sum_{(x,y,q)\in A} q = 1 \right\}$$

Então, dado um objeto $\langle\ G, \lambda\ \rangle$ de **ECFR**, com G não- colapsante (ver 5.2.3), uma estratégia de busca estocástica para $\langle\ G, \lambda\ \rangle$ é um par de morfismos em **ECFR**

$$\langle\ init, step\ \rangle$$

onde

1. O morfismo

   $$init : 1 \to S$$

   escolhe um elemento de S contendo apenas triplas da forma

   $$(\langle\ H, \kappa\ \rangle, \langle\ H, \kappa\ \rangle, q)$$

   com $\langle\ H, \kappa\ \rangle$ uma corda raiz de $\langle\ G, \lambda\ \rangle$;

2. O morfismo

   $$step : S \to S$$

   leva um conjunto $A \in S$ em um conjunto $step(A)$ tal que, para toda tripla em $step(A)$

   $$(\langle\ T', \theta'\ \rangle, \langle\ H', \kappa'\ \rangle, q')$$

   existe alguma tripla em A

   $$(\langle\ T, \theta\ \rangle, \langle\ H, \kappa\ \rangle, q)$$

   tal que $\langle\ T, \theta\ \rangle\ =\ \langle\ T', \theta'\ \rangle$ ou

– $\langle\, T, \theta \,\rangle \subseteq \langle\, T', \theta' \,\rangle$ e
– $\langle\, T', \theta' \,\rangle$ tem, no máximo, uma corda a mais que $\langle\, T, \theta \,\rangle$, e esta corda a mais, caso exista, é $\langle\, H', \kappa' \,\rangle$.

6.3.4 Estratégias de busca estocástica na linguagem interna. O seguinte predicado especifica quando um par $\langle\, init, step \,\rangle$ define uma estratégia de busca estocástica:

$$
\begin{array}{l}
\textbf{isStochasticSearchStrategy}(init, step, G, \lambda) \Leftrightarrow \\
\quad \textbf{isNonCollapsing}(G) \wedge \\
\quad \forall(\langle\, T, \theta \,\rangle, \langle\, H, \kappa \,\rangle, q) \in init(\star) : (\\
\qquad \langle\, T, \theta \,\rangle \;=\; \langle\, H, \kappa \,\rangle \;\wedge\; \textbf{isRoot}(H, \kappa, G, \lambda) \\
\quad) \\
\quad \wedge \\
\quad \forall A, B \in S : (\\
\qquad (A, B) \in step \;\Rightarrow \\
\qquad\quad \forall(\langle\, T, \theta \,\rangle, \langle\, H, \kappa \,\rangle, q) \in B : \\
\qquad\quad \exists(\langle\, T', \theta' \,\rangle, \langle\, H', \kappa' \,\rangle, q') \in A : (\\
\qquad\qquad \langle\, T, \theta \,\rangle \;=\; \langle\, T', \theta' \,\rangle \vee \\
\qquad\qquad\quad (\textbf{extendsByOne}(T, \theta, T', \theta') \wedge \\
\qquad\qquad\qquad \textbf{wasAdded}(H, \kappa, T, \theta, T', \theta') \\
\qquad\qquad\quad) \\
\qquad\quad) \\
\quad)
\end{array}
$$

6.3.5 Exemplo: *simulated annealing*. O algoritmo a seguir descreve a estratégia de busca local conhecida como *simulated annealing* ([2]), frequentemente usada para resolver problemas de otimização combinatória. A cada iteração, um sucessor do nó atual é escolhido aleatoriamente; o sucessor escolhido é visitado ou não, com uma probabilidade que depende da diferença entre as notas do nó atual e do sucessor escolhido (ΔE), e de um parâmetro T, que, por analogia com um processo físico, é chamado de "temperatura". A cada iteração n, o valor da temperatura T é dado por uma função $sched(n)$. A idéia é definir a função $sched$ de forma a fazer com que ocasionalmente ocorra que um sucessor com qualidade pior do que o nó atual seja visitado, para evitar que a busca caia em um ótimo local.

No algoritmo abaixo, *atual*, *inicial* e *prox* são nós do espaço de busca:

1. $atual \leftarrow inicial$
2. $n \leftarrow 0$
3. $T \leftarrow sched(n)$
4. Se $T = 0$ então retornar $atual$
5. $prox \leftarrow$ sucessor de $atual$ escolhido aleatoriamente
6. $\Delta E \leftarrow nota(prox) - nota(atual)$
7. Se $\Delta E \leq 0$ então $atual \leftarrow prox$
   senão $atual \leftarrow prox$ com probabilidade $\exp(-\Delta E/T)$
8. $n \leftarrow n + 1$
9. Ir para 3

Os seguintes termos e predicados são usados para especificar o comportamento de uma estratégia de busca estocástica do tipo *simulated annealing* no nosso modelo (onde cordas são análogas a nós).

O termo $probVisitedIfChosen(t, H, \kappa, H', \kappa', G, \lambda)$ representa a probabilidade de a corda sucessora $\langle\ H', \kappa'\ \rangle$ da corda atual $\langle\ H, \kappa\ \rangle$ ser visitada (supondo que $\langle\ H', \kappa'\ \rangle$ foi escolhida) na iteração t.

Dada uma fórmula φ, o termo $\iota q \in Q : \varphi$ significa "o único elemento q de Q tal que $\varphi(q)$ é verdadeira".

O termo exp' representa uma aproximação racional da função real $\exp : x \mapsto e^x$.

$$
\begin{array}{l}
probVisitedIfChosen(t, H, \kappa, H', \kappa', G, \lambda) = \\
\quad \iota q \in Q : (\\
\qquad (grade(H, \kappa, G, \lambda) \geq grade(H', \kappa', G, \lambda)\ \wedge\ q = 1) \\
\qquad \vee \\
\qquad (grade(H, \kappa, G, \lambda) < grade(H', \kappa', G, \lambda) \wedge \\
\qquad\quad q = exp'((grade(H, \kappa, G, \lambda) - grade(H', \kappa', G, \lambda))/t) \\
\qquad) \\
\quad)
\end{array}
$$

O termo $probNoAdvance(t, H, \kappa, G, \lambda)$ representa a probabilidade de que nenhuma sucessora da corda $\langle\ H, \kappa\ \rangle$ seja visitada. Isto pode ocorrer quando alguma sucessora é escolhida mas não é visitada:

$$
\begin{array}{l}
probNoAdvance(t, H, \kappa, G, \lambda) = \\
\quad 1 - (1/card(children(H, \kappa, G, \lambda))) \cdot \\
\quad \cdot \sum_{\langle\, H', \kappa' \,\rangle \in children(H, \kappa, G, \lambda)} probVisitedIfChosen(t, H, \kappa, H', \kappa', G, \lambda)
\end{array}
$$

O termo $prob(t, H, \kappa, H', \kappa', G, \lambda)$ representa a probabilidade de uma sucessora $\langle\, H', \kappa' \,\rangle$ da corda $\langle\, H, \kappa \,\rangle$ ser escolhida, com probabilidade

$$1/card(successors(H, \kappa, G, \lambda))$$

onde *card* significa "cardinalidade" – ver 5.4.26 – e visitada, com probabilidade

$$probVisitedIfChosen(t, H, \kappa, H', \kappa', G, \lambda)$$

No caso em que $\langle\, H, \kappa \,\rangle = \langle\, H', \kappa' \,\rangle$, o termo representa a probabilidade de nenhuma sucessora de $\langle\, H, \kappa \,\rangle$ ser visitada.

$$
\begin{array}{l}
prob(t, H, \kappa, H', \kappa', G, \lambda) = \\
\quad \iota q \in Q : (\\
\qquad (\langle\, H, \kappa \,\rangle \neq \langle\, H', \kappa' \,\rangle \wedge \\
\qquad\quad q = (1/card(children(H, \kappa, G, \lambda))) \cdot \\
\qquad\qquad probVisitedIfChosen(t, H, \kappa, H', \kappa', G, \lambda) \\
\qquad) \\
\qquad \vee \\
\qquad (\langle\, H, \kappa \,\rangle = \langle\, H', \kappa' \,\rangle \wedge \\
\qquad\quad q = probNoAdvance(t, H, \kappa, G, \lambda) \\
\qquad) \\
\quad)
\end{array}
$$

Finalmente, usando os termos acima, o predicado que afirma que

$$(init, step, G, \lambda)$$

é uma heurística do tipo *simulated annealing* é:

$$
\begin{array}{l}
\textbf{isSimAnn}(\langle\, init, step \,\rangle, G, \lambda) \Leftrightarrow \\
\quad \textbf{isStochasticSearchStrategy}(init, step, G, \lambda) \wedge \\
\quad \textbf{isIrrevocable}(\langle\, init, step \,\rangle, G, \lambda) \wedge \\
\quad init(\star) = \\
\qquad \{(\langle\, H, \kappa \,\rangle, \langle\, H, \kappa \,\rangle, 1/card(trees(G, \lambda))) \mid isRoot(H, \kappa, G, \lambda)\} \\
\quad \wedge \\
\quad \exists sched \in (Q^+)^N : \\
\quad \forall n \in N : (\\
\qquad sched(n) \geq sched(sn) \\
\qquad \wedge \\
\qquad (sched(n) = 0 \;\Rightarrow\; stage(n) = stage(sn)) \\
\qquad \wedge \\
\qquad (sched(n) \neq 0 \;\Rightarrow \\
\qquad\quad \forall(\langle\, T, \theta \,\rangle, \langle\, H, \kappa \,\rangle, q) \in stage(n) : \\
\qquad\quad \forall\langle\, H', \kappa' \,\rangle \in strings(G, \lambda) : \\
\qquad\quad \forall\langle\, T', \theta' \,\rangle \in trees(G, \lambda) : (\\
\qquad\qquad (\\
\qquad\qquad\quad (\textbf{isChild}(H', \kappa', H, \kappa, G, \lambda) \;\wedge\; \textbf{wasAdded}(H', \kappa', T', \theta', T, \theta)) \\
\qquad\qquad\quad \vee \\
\qquad\qquad\quad (\\
\qquad\qquad\qquad \langle\, T, \theta \,\rangle = \langle\, T', \theta' \,\rangle \wedge \langle\, H, \kappa \,\rangle = \langle\, H', \kappa' \,\rangle \wedge \\
\qquad\qquad\qquad prob(sched(n), H, \kappa, H, \kappa, G, \lambda) > 0 \\
\qquad\qquad\quad) \\
\qquad\qquad) \\
\qquad\qquad \Leftrightarrow \\
\qquad\qquad (\langle\, T', \theta' \,\rangle, \langle\, H', \kappa' \,\rangle, prob(sched(n), H, \kappa, H', \kappa', G, \lambda)) \\
\qquad\qquad\quad \in stage(sn) \\
\qquad\quad) \\
\qquad) \\
\quad)
\end{array}
$$

As seguintes observações se aplicam:

- A estratégia nunca volta a cordas em níveis superiores ao da corda atualmente visitada;
- Cada raiz tem a mesma probabilidade de ser a corda inicial da busca;

- Existe um morfismo $sched : N \to Q^+$, decrescente, que dá a temperatura de cada iteração;
- Quando a temperatura chega a 0, a busca termina – i.e., $stage(n) = stage(sn)$;
- Caso contrário, toda corda $\langle\ H, \kappa\ \rangle$ com probabilidade não-nula de ser a corda atual tem cada sucessora sua $\langle\ H', \kappa'\ \rangle$ visitada com probabilidade

$$prob(sched(n), H, \kappa, H', \kappa', G, \lambda)$$

- Além disso, pode ocorrer, com probabilidade

$$prob(sched(n), H, \kappa, H, \kappa, G, \lambda)$$

que nenhuma sucessora de $\langle\ H, \kappa\ \rangle$ seja visitada; note, porém, que se todas as sucessoras de $\langle\ H, \kappa\ \rangle$ tiverem notas melhores que $\langle\ H, \kappa\ \rangle$, alguma delas certamente será visitada.

6.3.6 Exemplo: algoritmos genéticos. O paradigma de algoritmos genéticos constitui uma das mais antigas meta-heurísticas, tendo sido inicialmente desenvolvido por Holland [35] nos anos 60 e 70. Desde então, inúmeras variações têm aparecido na literatura, como relatado em [36]. No que se segue, usaremos o nosso modelo para desenvolver uma caracterização de algoritmos genéticos geral o bastante para incluir a maior parte dessas variações.

Algoritmos genéticos são uma técnica de busca baseada em populações. Uma população é uma coleção finita de respostas viáveis para a instância do problema que está sendo resolvida. As respostas que formam uma população são chamadas de *indivíduos*. A busca começa com a definição de uma população inicial; em seguida, a cada passo da execução, a população atual é modificada, de maneira controlada, para gerar uma nova população. Espera-se que, ao final de diversas iterações deste processo, a população resultante contenha um indivíduo que seja uma resposta correta para a instância em questão.

A maneira como uma população é alterada depende da forma como seus indivíduos são codificados. Na definição original de algoritmo genético, por exemplo, uma resposta viável é representada por uma cadeia de *bits*. A população evolui através da combinação das representações de dois ou mais indivíduos, em um processo chamado *crossover*, e através da alteração isolada de um indivíduo, em um processo chamado *mutação*. Estes dois processos geram novos indivíduos através da combinação e/ou alteração de suas codificações binárias; os indivíduos gerados

passam a fazer parte da nova população, substituindo indivíduos da população original.

Na definição de um algoritmo genético específico, algumas das questões que devem ser decididas são as seguintes:

- Qual o tamanho da população? Este tamanho permanece constante ao longo da busca?
- Como a população inicial é gerada?
- De que forma uma resposta é codificada?
- Como os indivíduos são selecionados para a aplicação da operação de *crossover*?
- Como as representações dos indivíduos selecionados são manipuladas e combinadas pela operação de *crossover*?
- Como um indivíduo é selecionado para a aplicação da operação de mutação?
- Como a representação do indivíduo selecionado é manipulada e alterada pela operação de mutação?
- Como os novos indivíduos gerados por *crossover* e mutação são adicionados à população? Mais especificamente, que indivíduos devem ser retirados da população para dar lugar aos novos indivíduos?

Para operacionalizar algumas destas decisões, define-se uma *função de aptidão* (*fitness function*) para associar uma "nota" a cada indivíduo. A idéia é que indivíduos com valores melhores desta função (i.e., os mais aptos) tenham probabilidade maior de serem escolhidos para *crossover* (em algumas estratégias, porém, alguns ou todos os indivíduos escolhidos para *crossover* são selecionados ao acaso). O valor da aptidão de um indivíduo também pode ser usado para decidir se o indivíduo deve ser retirado da população para dar lugar a novos indivíduos. São estes usos da função de aptidão que justificam a analogia entre o funcionamento dos algoritmos genéticos e a teoria de evolução das espécies por seleção natural: indivíduos mais aptos têm maior probabilidade de se reproduzir e transmitir características desejáveis para seus descendentes.

Como no cenário biológico, as mutações servem para gerar maior diversidade, criando indivíduos que não poderiam se originar da combinação de quaisquer outros indivíduos da população.

A seguir, apresentaremos uma breve discussão sobre como a lógica interna de **ECF** poderia ser utilizada na caracterização de estratégias de busca guiadas pelo

paradigma de algoritmos genéticos. Como os operadores de *crossover* e mutação costumam ser definidos probabilisticamente, lidaremos com estratégias de busca estocásticas, conforme descrito em 6.3.2.

População. Dada uma ECFR $\langle\ G, \lambda\ \rangle$ e um problema $P = \langle\ D, R, p\ \rangle$, lembre-se de que o rótulo $\lambda(v)$ de um nó v de GP é um par (A, k), com A um subconjunto de R e k um natural.

No contexto de algoritmos genéticos, A representa a população associada ao nó v. O valor k não é utilizado na especificação de algoritmos genéticos, uma vez que não é a população que recebe uma nota por sua qualidade; em vez disso, cada indivíduo da população é avaliado através de uma função de aptidão f:

Função de aptidão. A qualidade de um indivíduo da população é dada por um termo

$$f : G_R \to N$$

do tipo $\mathbf{N}^{\mathbf{G_R}}$, onde G_R é o termo que representa os domínios de respostas (ver 5.4.28) e N é o termo que representa o objeto de números naturais de **ECF**, descrito na seção 4.5.

Existem variações de algoritmos genéticos onde a função de aptidão não é estática; nestas variações, um mesmo indivíduo pode receber notas diferentes em instantes diferentes, dependendo da população a que pertence e do estágio da execução da busca. Para modelar estas variações, associaremos uma função de aptidão f a cada corda de $\langle\ G, \lambda\ \rangle$. Isto é realizado através de um termo

$$F : \mathit{strings}(G, \lambda) \to N^{G_R}$$

do tipo $(\mathbf{N}^{\mathbf{G_R}})^{\mathit{strings}(\mathbf{G},\lambda)}$. Assim, a função de aptidão f associada a uma corda $\langle\ H, \kappa\ \rangle$ é dada pelo termo $F(\langle\ H, \kappa\ \rangle)$.

Operadores. Conforme descrito mais acima, um passo na execução de um algoritmo genético consiste, basicamente, em três operações:

1. Seleção de indivíduo(s) da população atual para reprodução;

2. Aplicação de um operador (*crossover* ou mutação) sobre o(s) indivíduo(s) selecionado(s), gerando um ou mais indivíduo(s);

3. Construção de uma nova população, onde cada indivíduo ou pertencia à população original ou pertence ao conjunto de indivíduos gerados no passo anterior.

Cada uma destas ações pode ser de caráter estocástico, de forma que seus resultados são conjuntos de alternativas, cada alternativa acompanhada de uma probabilidade não-nula. Além disso, as operações podem depender dos valores da função de aptidão dos indivíduos envolvidos. Mais precisamente, as três operações acima são representadas na linguagem de **ECF** pelos seguintes termos:

1. A seleção de indivíduos da população atual para reprodução corresponde a um termo

$$s : PG_R \times N^{G_R} \to P((G_R)^n \times Q)$$

para algum natural positivo n.

Aqui, Q é o objeto de números racionais descrito em 4.6. Basicamente, este termo s corresponde a uma maneira de mapear um par

$$(\Pi, f)$$

onde Π é uma população e f é a função de aptidão que atribui notas aos indivíduos de Π, em um conjunto finito de pares

$$s(\Pi, f) = \{\ (\Sigma_1, q_1), \ldots, (\Sigma_i, q_i), \ldots\ \}$$

onde cada par é composto de

- Um conjunto de n-uplas $\Sigma_i = \{t_{i1}, \ldots, t_{im}\}$. Cada n-upla corresponde a um grupo de indivíduos escolhidos para reprodução (tal grupo será chamado de "casal", embora n possa ser diferente de 2). A utilização de tuplas em vez de conjuntos se deve ao fato de que, em muitas estratégias, os indivíduos são escolhidos para reprodução de tal forma que a ordem da escolha é relevante; assim, fala-se do primeiro pai, do segundo pai, e assim por diante.
- Uma probabilidade q_i.

Um operador de seleção s escolhe, então, com a probabilidade q_i, um conjunto Σ_i de "casais". A cada um destes "casais" será aplicado um operador de reprodução, conforme descrito a seguir.[3]

Obviamente, devem ser satisfeitas as condições de que cada tupla de Σ_i seja composta apenas de elementos da população Π e de que a soma das probabilidades q_i seja igual a 1.

2. A reprodução que se dá em um "casal" é representada por um termo

$$c : (G_R)^n \times N^{G_R} \to P((G_R)^{n'} \times Q)$$

para algum natural positivo n'.

O termo c corresponde a uma maneira de mapear um par

$$(t, f)$$

onde t é uma n-upla (i.e., um "casal") e f é a função de aptidão, a um conjunto finito de pares

$$c(t, f) = \{ (t'_1, q'_1), \ldots, (t'_j, q'_j), \ldots \}$$

onde cada par é composto de

- Uma (n')-upla t'_j cujas componentes formam a "prole" do "casal" representado pela n-upla t. Como antes, usa-se uma tupla em vez de um conjunto para que seja possível falar do primeiro filho, do segundo filho, e assim por diante. Neste caso, o operador c, quando aplicado a um "casal" de n "pais", produz uma "prole" de n' indivíduos.
- Uma probabilidade q'_j.

A soma de todas as probabilidades q'_j deve ser igual a 1.

3. A construção da nova população se dá através de aplicações dos termos s e c descritos acima. Usaremos o termo i (significando "inserção") para denotar a operação de construir a nova população Π' a partir da população original Π.

O termo i é do tipo

$$PG_R \times N^{G_R} \times \mathit{tipo}(s) \times \mathit{tipo}(c) \to P(PG_R \times Q)$$

onde $\mathit{tipo}(s)$ e $\mathit{tipo}(c)$ são os tipos dos operadores de seleção e reprodução descritos acima.

[3]Na definição de um operador de mutação, que é usualmente aplicado a um único indivíduo, n é igual a 1; i.e., o "casal" consiste de um único indivíduo.

O termo i corresponde a uma maneira de mapear uma quádrupla

$$(\Pi, f, s, c)$$

onde Π é a população original, f é a função de aptidão, s é o operador de seleção de "casais", e c é o operador de reprodução, a um conjunto finito de pares

$$i(\Pi, f, s, c) = \{\ (\Pi'_{11}, q'_{11}), \ldots, (\Pi'_{ij}, q'_{ij}), \ldots\ \}$$

onde cada par é composto de uma nova população Π'_{ij} e uma probabilidade q'_{ij}. O índice i significa que a nova população baseia-se na alternativa i do operador de seleção s (i.e., que seleciona o conjunto Σ_i de "casais"). O índice j significa que a nova população baseia-se na alternativa j do operador de reprodução (i.e., que produz a "prole" t'_j para cada "casal" t).

As seguintes condições devem ser satisfeitas:

- Para todos i, j, deve-se ter que $q'_{ij} = q_i \cdot q'_j$, onde q_i é a probabilidade da i-ésima alternativa do operador s descrito acima, e q'_j é a probabilidade da j-ésima alternativa do operador c;
- Para todos i, j, deve-se ter que

  $$\Pi'_{ij} \subseteq \Pi \cup prole_{ij}$$

  onde $prole_{ij}$ é um termo que representa o conjunto de indivíduos gerado a partir da i-ésima alternativa do operador s e da j-ésima alternativa do operador c, definido da seguinte forma:

  - Dados a função de aptidão f e o conjunto Σ_i de "casais" que compõe a i-ésima alternativa de $s(\Pi, f)$:

    $$\Sigma_i = \{\ t_1, \ldots, t_m\ \}$$

    aplicar o operador c a cada "casal"; o resultado de tal aplicação será denotado por $map(c(\_, f))(\Sigma_i)$, que consiste no seguinte conjunto de pares (onde cada "linha" corresponde ao resultado da aplicação de c a um "casal" de Σ_i):

    $$\left\{ \begin{array}{cccc} (t'_{11}, q'_{11}), & \ldots, & (t'_{1j}, q'_{1j}), & \ldots \\ \vdots & & \vdots & \\ (t'_{m1}, q'_{m1}), & \ldots, & (t'_{mj}, q'_{mj}), & \ldots \end{array} \right\}$$

– Definimos $prole_{ij}$ como o conjunto de todas as componentes das tuplas $t'_{1j}, \ldots, t'_{mj}$ do conjunto acima. Em outras palavras, a j-ésima "coluna" do conjunto $map(c(\_, f))(\Sigma_i)$, como representado acima, contém exatamente as proles dos "casais" selecionados pela i-ésima alternativa do operador s, proles estas produzidas pela j-ésima alternativa do operador c.

A condição $\Pi'_{ij} \subseteq \Pi \cup prole_{ij}$ significa, então, que o operador i de inserção se limita a substituir indivíduos da população original por indivíduos gerados pelos operadores de seleção e de reprodução, levando em conta o caráter estocástico dos operadores.

6.4
Conclusões do Capítulo

6.4.1 Resumo. Este capítulo apresentou alguns exemplos de especificações de heurísticas e meta-heurísticas (incluindo estratégias de busca estocásticas); além disso, mostrou que nosso modelo traduz adequadamente um resultado já estabelecido sobre heurísticas gulosas e matróides.

6.4.2 O que significa implementar uma heurística? O exemplo de especificação de *simulated annealing* (6.3.5) é bem instrutivo a respeito dos possíveis parâmetros de uma heurística. Ali, a fórmula inclui uma variável *sched* quantificada existencialmente. Informalmente falando, uma estratégia de busca pode ser considerada como *simulated annealing* se existir uma função que retorna a temperatura para cada iteração e que rege o comportamento da busca (i.e., as cordas visitadas e as probabilidades). Nada mais é dito sobre *sched*.

Neste contexto, implementar *simulated annealing* significa "instanciar" esta variável *sched*. Para usar a terminologia mais exata, implementar uma heurística, em termos gerais, é *skolemizar* a fórmula que a especifica, definindo valores específicos para os parâmetros da busca.