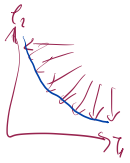
Exemplo 5.17 (MOTS para o problema da mochila bi-objetivo)
O algoritmo determina inicialmente limites $[l, u]$ para o número de itens. Na forma mais simples ele busca soluções eficientes com um número de itens $n = u, u-1, \ldots, l$, numa vizinhança que troca um item selecionado x_i por um item não selecionado x_j. A reinserção do item i fica tabu para 7 iterações e a deleção do item j para 3 iterações. Em cada iteração o algoritmo determina todos vizinhos viáveis não tabu V, que dominam um *ponto de satisfação* σ e não são dominados por uma solução na fronteira eficiente atual $\hat{E}$, e atualiza $\hat{E}$ com estes pontos. O ponto de satisfação σ é 0 para $n = u$ e se aproxima ao nadir η do conjunto eficiente $\hat{E}$ do n anterior de acordo com $\sigma_{n-1} = \sigma_n + (\eta_n - \sigma_n)/\delta$ com um tamanho de passo $\delta \geq 2$. Depois a solução vizinha s' de maior $S(s')$ é selecionada. Caso não existe solução viável em V, algoritmo seleciona o vizinho não-tabu que excede a capacidade da mochila menos possível. Um critério de aspiração permite selecionar uma solução tabu que domina todas soluções V ou que domina um número grande de soluções em $\hat{E}$.
A solução inicial é aleatória (com $n = u$ itens selecionados) e cada direção de busca continua com a solução final anterior. Diminuindo n, o item com o menor valor mínimo dos sobre as dimensões da mochila é removido.
A implementação testa 25 conjuntos de pesos $(\lambda, 1-\lambda)$, com $\lambda = i/24$ para $i = 0, \ldots, 24$, aplica no máximo 500 iterações por busca tabu (para cada conjunto de pesos e cada n), e usa $\delta = 2$ na mesmas instâncias do exemplo anterior. A busca para com $n = l$ ou caso na vizinhança não tem solução que domina o ponto de satisfação. ◊



5.4.2. Busca por recombinação de soluções

A maioria das propostas de heurísticas multi-objetivos recombinando soluções são algoritmos genéticos e evolutivos. Num algoritmo genético somente a seleção de indivíduos para recombinação depende da função objetivo. Portanto, uma das modificações que torna um algoritmo genético multi-objetivo, é uma seleção proporcional com $\omega(s)$ com um vetor de pesos w selecionado aleatoriamente em cada iteração (Murata et al. 1996). Essa abordagem é simples na implementação, mas tem a desvantagem que ela foca em soluções suportadas. Um dos algoritmos pioneiros trabalho com k subpopulações, e seleciona indivíduos em cada subpopulação de acordo com a i-ésima função objetivo (ver algoritmo 5.2).

Algoritmo 5.2 (Seleção VEGA (Vector-evaluated GA))
Entrada A população atual P.
Saída Uma nova população P.

```
para i ∈ k
  seleciona |P|/k indivíduos proporcional com φ_i
aplica recombinação e mutação
```

```
   na união S dos indivíduos selecionados
retorne a nova população
```

Algoritmos recentes determinam o valor de uma solução de acordo com a proximidade com a fronteira eficiente e a densidade na fronteira eficiente, para uma exploração melhor em direção de soluções eficientes e em regiões esparsas. Para um conjunto de soluções S seja $\hat{E}(S) = \hat{E}_1(S)$ a fronteira eficiente (local) e define recursivamente a $k+1$-ésima fronteira eficiente por

$$\hat{E}_{k+1}(S) = \hat{E}(S \setminus \bigcup_{i\in[k]} \hat{E}_k(S)). \tag{5.5}$$

(ver o exemplo da Fig. 5.5).
Seja ainda $B(x,S) = \{s \in S \mid s > x\}$ o conjunto de soluções em S que dominam x e $W(x,S) = \{s \in S \mid x > s\}$ o conjunto de soluções dominadas por x em S. Entre as propostas temos algoritmos que ordenam soluções $s \in P$ da população atual P

- pelo nível k da sua fronteira eficiente $s \in \hat{E}_k(P)$ correspondente (non-dominated sorting GA, NSGA, NSGA-II);
- pelo número $1 + |B(s,P)|$ de soluções que dominam s na população atual P (MOGA);
- pela fração total da cobertura por soluções de um conjunto E eficiente atual $1 + \sum_{t\in B(s,E)} |W(t,P)|/(|P|+1)$ que dominam s (strength Pareto EA, SPEA);
- pelo soma dos postos das soluções que dominam s, $r(s) = 1 + \sum_{t\in B(s,P)} r(t)$.

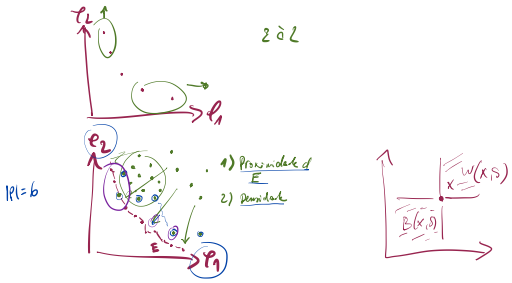
Técnicas para priorizar a exploração de regiões esparsas incluem

- a redução da função objetivo por um fator $|B_\sigma(s) \cap \hat{\varphi}(P)|^{-1}$ (com $B_r(s)$ um esfera de raio r e centro $\hat{\varphi}(s)$ e $\hat{\varphi}(s)$ a função objetivo normalizada para o intervalo $[0,1]$ em cada dimensão) (MOGA);
- a soma das distâncias normalizadas para os predecessores e sucessores na fronteira atual em cada dimensão ("crowding distance") (NSGA-II). Para cada dimensão $i \in [k]$ supõe que as soluções $x^1, \ldots, x^n$ de uma fronteira são ordenadas pela i-ésima coordenada (i.e. $x_i^1 \le x_i^2 \le \cdots \le x_i^n$). Então o crowding distance normalizada da solução x^s na dimensão i é

$$c_i(x^s) = (\varphi_i(x^{s-1}) - \varphi_i(x^{s+1}))/(\varphi_i^{\max} - \varphi_i^{\min})$$

  para $s \in [2, n-1]$, $c_i(x^1) = c_i(x^n) = \infty$ e a crowding distance da solução é $c(x^s) = \sum_{i\in[k]} c_i(x^s)$.

Formas de elitismo incluem manter uma ou mais fronteiras eficiente $\hat{E}_k(P)$ ou $\hat{E}_k(P \cup C)$ com filhos C.



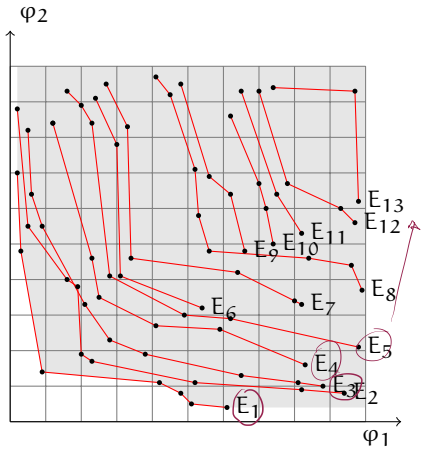
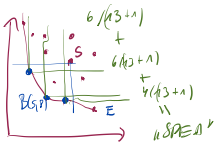
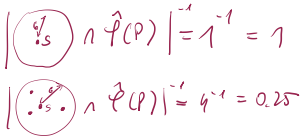
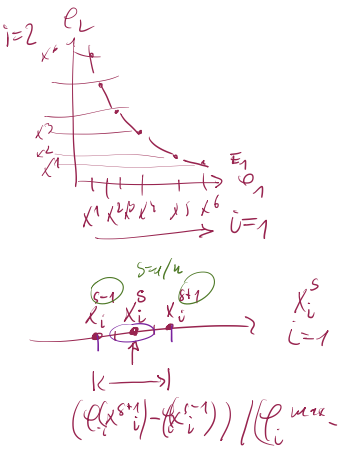


Figura 5.5.: Decomposição de um conjunto de soluções em fronteiras eficientes de acordo com (5.5).







Exemplo 5.18 (NSGA-II)
O algoritmo NSGA-II segue o algoritmo genético 4.3 com uma seleção por um torneio binário de $\mathcal{P}$: entre duas soluções aleatórias a solução de menor nível k ou, no caso de empate, de menor "crowding distance" é selecionada. Ele sempre aplica mutação ($\mathcal{M} = C$). A função `update` que atualiza a população é realizada por

$R := P \cup C$
seja $P := \hat{E}_1(R) \cup \cdots \cup \hat{E}_k(R)$ com k maximal t.q. $|P| \leq n$
if $|P| < n$
  complete P com as $n - |P|$ soluções de $\hat{E}_{k+1}(R)$
    de menor ''crowding distance''
end if

◊

5.5. Heurísticas para problemas contínuos

Uma forma geral de um problema de otimização contínuo é

$$\begin{aligned} \textbf{minimiza} \quad & f(x), \\ \textbf{sujeito a} \quad & g_i(x) \leq 0, && \forall i \in [m], \\ & h_j(x) = 0, && \forall j \in [l], \\ & x \in \mathbb{R}^n, \end{aligned}$$

com uma função objetivo $f : \mathbb{R}^n \to \mathbb{R}$, e restrições $g_i : \mathbb{R}^n \to \mathbb{R}$ e $h_j : \mathbb{R}^n \to \mathbb{R}$. Casos particulares importantes incluem funções lineares e convexas e o caso irrestrito ($m = l = 0$). As definições 2.1 continuam ser válidas com uma vizinhança

$$N_\epsilon(x) = \{x' \in \mathbb{R}^n \mid \|x - x'\| \leq \epsilon\} \tag{5.6}$$

e com a condição adicional que para um mínimo ou máximo local deve existir um $\epsilon > 0$ que satisfaz a definição.
Casos simples de um problema de otimização contínua podem ser resolvidos por *métodos indiretos*. Um método indireto encontra primeiramente todos candidatos para soluções ótimas por critérios necessários para otimalidade local, depois verifica a otimalidade local por critérios suficientes, e finalmente encontra a solução ótima global por comparação das soluções localmente ótimas. Na otimização irrestrita em uma dimensão, por exemplo, temos a condição suficiente $f' = 0$ para otimalidade local, e a condição suficiente $f'' > 0$ para um mínimo local e $f'' < 0$ para um máximo local (dado que as derivadas existem).
Caso resolver $f' = 0$ não é possível técnicas de *busca em linha* (ingl. *line search*) podem ser usadas. Para um domínio restrito $x \in [a, b]$ um método simples é a *busca regular*: escolhe o melhor entre os pontos $x = a + i\Delta x$, para $i = 0, \ldots, \lfloor (b - a)/\Delta x \rfloor$, para um tamanho de passo Δx. Um outro exemplo é uma *busca em linha com backtracking*.