Marcos Daniel V. Baroni

A Hybrid Heuristic for the Multi-objective Knapsack Problem

Vitória - Espírito Santo - Brasil
November 27, 2017

Marcos Daniel V. Baroni

A Hybrid Heuristic for the Multi-objective Knapsack Problem

Tese de Doutorado apresentada de acordo com o regimento do Programa de Pós-graduação em Informática da Universidade Federal do Espírito Santo.

Universidade Feredal do Espírito Santo – UFES
Departamento de Informática
Programa de Pós-Graduação em Informática

Advisor: Dr. Flávio Miguel Varejão

Vitória - Espírito Santo - Brasil
November 27, 2017

Marcos Daniel V. Baroni

A Hybrid Heuristic for the Multi-objective Knapsack Problem

Tese de Doutorado apresentada de acordo com o regimento do Programa de Pós-graduação em Informática da Universidade Federal do Espírito Santo.

Work approved. Vitória - Espírito Santo - Brasil, November 27, 2017:

Dr. Flávio Miguel Varejão
Advisor

Dr.ª Maria Claudia Silva Boeres
Member

Dr. Arlindo Gomes de Alvarenga
Guest

Dr.ª Simone de Lima Martins
Guest

Vitória - Espírito Santo - Brasil
November 27, 2017

Resumo

Many real applications like project selection, capital budgeting and cutting stock involves optimizing multiple objectives that are usually conflicting and can be modelled as a multi-objective knapsack problem (MOKP). Unlike the single-objective case, the MOKP is considered a NP-Hard problem with considerable intractability. This work propose a hybrid heuristic for the MOKP based on the shuffled complex evolution algorithm. A multi-dimensional indexing strategy for handling large amount of intermediate solutions are proposed as an optimization, which yields considerable efficiency, especially on cases with more than two objectives. A series of computational experiments show the applicability of the proposal to several types of instances.

Keywords: Multi-objective Knapsack Problem, Metaheuristic, Shuffled Complex Evolution, Multi-dimensional indexing

Contents

1 Introdução

Intro...

2 O Problema da Mochila Multi-objetivo

Em problemas reais é comum a existência de situações em que deseja-se otimizar mais de um objetivo os quais, geralmente, são conflitantes. Estes problemas são chamados multi-objetivos e tipicamente não possuem uma solução sendo a melhor em todos os objetivos, mas as possuem várias soluções de interesse chamadas *soluções eficientes*.

Um problema de otimização multi-objetivo com m objetivos pode ser descrito como uma função vetorial $f(x) = \Big(f_1(x), \ldots, f_p(x)\Big)$ para a qual deseja-se encontrar um vetor $x \in X$ que maximize simultaneamente as m funções objetivo. Formalmente:

$$\begin{aligned} &\max\ f(x) = \Big(f_1(x), f_2(x), \ldots, f_m(x)\Big) \\ &\text{sujeito a } x \in X \end{aligned}$$

Definição 1 (Dominância, Eficiência e conjunto Pareto)**.** *Considere um problema de otimização multi-objetivo. Diz-se que uma solução $x \in X$* domina *uma solução $y \in X$, denotado por $dom(x, y)$ se, e somente se, x é ao menos tão boa quanto y em todos os objetivos e melhor que y em ao menos um dos objetivos. Formalmente:*

$$dom(x, y) = \begin{cases} \forall i \in \{1, 2, \ldots, m\} : f_i(x) \geqslant f_i(y)\ e \\ \exists j \in \{1, 2, \ldots, m\} : f_j(x) > f_j(y) \end{cases}$$

Uma solução $x \in X$ é dita eficiente*, denotado por $\mathit{eff}(x)$, se, e somente se, x não é dominada por nenhuma outra solução pertencente a X. Formalmente:*

$$eff(x) \iff \nexists\Big(y \in X \wedge dom(y, x)\Big)$$

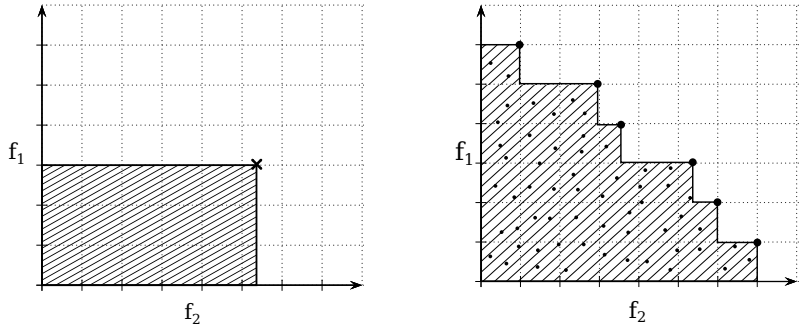
O conjunto de todas as soluções eficientes de um problema multi-objetivo, denotado por $Par(X)$, é chamado de conjunto Pareto *ou* conjunto Pareto-ótimo*. Formalmente:*

$$Par(X) = \{x \in X \mid \mathit{eff}(x)\}$$

Resolver um problema multi-objetivo consiste em determinar seu conjunto Pareto. Este conceito foi primeiramente elaborado por Vilfredo Pareto em 1896, que enunciou a relação Pareto-Ótima que diz: "não é possível melhorar uma característica do problema sem piorar outra", o que caracteriza a relação conflitante entre os objetivos na otimização multi-objetivo.

Na Figura 1a ilustra o conceito de dominância. A solução marcada domina todas as soluções existentes na área hachurada. As soluções em destacadas na Figura 1b formam um conjunto Paretopor dominarem sobre todas as outras soluções.

Um dos problemas multiobjetivos mais importantes da literatura é o problema da mochila multiobjetivo (MOKP). Muitas problemas reais podem ser modelados como uma



(a) Região de dominância de uma solução.

(b) Exemplo de conjunto Pareto.

Figura 1: Exemplos de solução dominante e conjunto Pareto.

instância do MOKP como seleção de projetos (TENG; TZENG, 1996), orçamento de capital (ROSENBLATT; SINUANY-STERN, 1989), carregamento de carga (TENG; TZENG, 1996) e planejamento de estoque (ISHIBUCHI; AKEDO; NOJIMA, 2015).

Comentar sobre a dificuldade de problemas MObj. Explosão do pareto com o aumento da quantidade de objectivos. Poucos métodos extados eficientes, geralmente utiliza-se métodos heurísticos.

O problema da mochila multi-objetivo pode ser descrito como uma função vetorial f que mapeia uma variável de decisão (solução) a uma tupla de m valores (objetivos). Formalmente:

$$\begin{aligned} \max\ & y = f(x) = \Big(f_1(x), f_2(x), \ldots, f_m(x)\Big) \\ \text{sujeito a}\ & x \in X \end{aligned}$$

onde x é a *variável de decisão*, X denota o conjunto de soluções viáveis e y representa o *vetor de objetivos* para os quais deseja-se maximizar.

Vale resaltar que o tamanho do conjunto Paretopara o problema em questão tende a crescer rapidamente com o tamanho do problema, especialmente com o número de objetivos.

Uma instância de um problema da mochila multi-objetivo (MOKP) com m objetivos consiste em uma capacidade inteira $W > 0$ e n itens. Cada item i possui um peso inteiro positivo w_i e m lucros inteiros $p_i^1, p_i^2, \ldots, p_i^m$ não negativos. O lucro p_i^k representa a contribuição do i-ésimo item para com o k-ésimo objetivo. Uma solução é representada por um conjunto $x \subseteq \{1, \ldots, n\}$ contendo os índices dos itens incluídos na mochila. Uma solução é viável se o peso total incluído na mochila não ultrapassa a capacidade da

mochila. Formalmente a definição do problema é a seguinte:

$$\begin{aligned} \max\ & f(x) = \Big(f_1(x), f_2(x), \ldots, f_m(x)\Big) \\ \text{subject to}\ & w(x) \leqslant W \\ & x \in \{0,1\}^n \end{aligned}$$

where

$$f_j(x) = \sum_{i=1}^{n} p_i^j x_i \quad j = 1, \ldots, m$$

$$w(x) = \sum_{i=1}^{n} w_i x_i$$

O MOKP é considerado um problema $\mathcal{NP}$-Hard visto set uma generalização do bem conhecido problema da mochila $0-1$, para o qual $m = 1$. É consideravelmente difícil determinar o conjunto Paretopara um MOKP, especialmente para vários objetivos. Até mesmo para casos bi-objetivos, problemas pequenos podem se apresentar intratáveis. Por este motivo interessa-se no desenvolvimento de métodos eficientes para manipular uma grande quantidade de soluções, o que pode eventualmente trazer tratabilidade a instâncias antes intratáveis.

A literatura contém várias propostas para resolver o MOKP de forma exata. Porém, nenhum método tem provado ser eficiente para grande instâncias com mais de dois objetivos. Mesmo para problemas bi-objetivo, algumas instâncias de tamanho considerado médio têm aprestando difculdades na determinação da solução exata, o que tem motivado o desenvolvimento de métodos heurísticas que buscam determinar um conjunto Paretoaproximado em tempo computacional razoável.

2.1 Métodos Exatos

Comentar sobre background de propostas de métodos exatos.

Comentar sobre o método da Bazgan como sendo considerado o melhor, citar melhorias propostas.

Explicar que o algoritmo Bazgan considera um conceito generalizado de dominância aplicado a cada iteração.

Dizer que a explicação do desenvolvimento do algoritmo será reproduzido segundo o artigo original.

Algorithm 1 O algoritmo de Nemhauser e Ullmann para o MOKP.

```
1: function DP(p, w, W)
2:     S^0 = {∅}
3:     for k ← 1, n do
4:         S*^k = S^{k-1} ∪ {x ∪ k | x ∈ S^{k-1}}
5:         TODO...
6:     end for
7:     P = {x | ∄a ∈ S^n : △a x | w(x) ⩽ W}
8:     return P
9: end function
```

O processo sequencial executado pelo algoritmo de programação dinâmica consiste de n iterações. A cada k-ésima iteração é gerado o conjunto de estados S^k, que representa todas as soluções viáveis compostas de itens exclusivamente pertencentes aos k primeiros itens ($k = 1, \ldots, n$). Um estado $s_k = (s_k^1, \ldots, s_m^k, s_{m-1}^k) \in S_k$ represena uma solução viável que tem valor s_k^i como i-ésimo objetivo ($i = 1, \ldots, m$) e s_k^{m-1} de peso. Portanto, temos $S_k = S_{k-1} \cup \{(s_{k-1}^1 + p_k^1)\}$

> Comentar sobre as estratégias de redução dos conjuntos de estados, motivando as definições a seguir.

> Definir conjunto cobertura, conjunto independente, conjunto eficiente reduzido, etc., definir parte simétrica/assimétrica de $\underline{\triangle}$?

Definição 2 (Extensão, Restrição e Complemento)**.** *Considere o Algoritmo 1 e qualquer estado $s_k \in S_K (k < n)$. Um* complemento *de s_k é qualquer subconjunto $J \subseteq \{k+1, \ldots, n\}$ tal que $s_k^{m+1} + \sum_{j \in J} w_j \leqslant W$. Assumiremos que qualquer estado $s_n \in S_n$ admite o conjunto vazio como único complemento. Um estado $s_n \in S_n$ é uma* extensão *de $s_k \in s_k (k \leqslant n)$ se, e somente se, existe um complemento J de s_k tal que $s_n^i = s_k^i + \sum_{j \in J} p_j^i) para\ i = 1, \ldots, m$ e $s_n^{p+1} = s_k^{p+1} + \sum_{j \in J} w_j$. O conjunto de extenções de s_k é denotado por $Ext(s_k)(k \leqslant n)$. Um estado $s_k \in S_k (k \leqslant n)$ é uma* restrição *do estado $s_n \in S_n$ se, e somente se, s_n é uma extenão de s_k.*

2.1.1 As relações de dominância

> Introdução às relações de dominância?

Definição 3 (Relação de dominância entre soluções)**.** *Uma relação D_k sobre $S_k, k = i,\ ldots, n$, é uma relação de dominância se, e somente se, para todo $s_k, s_{k'} \in S_k$,*

$$s_k D_k s_{k'} \Rightarrow \forall s_{n'} \in Ext(s_{n'}), \exists s_n \in Ext(s_k), s_n \underline{\triangle} s_{n'} \tag{2.1}$$

Apesar das relações de dominância não serem transitivas por definição, costumam ser transitivas por construção, como é o caso das três relações de dominância da Seção 2.1.1. Vale notar que se $D_k^i,\ i = 1, \ldots, p$ são relações de dominância então $D_k = \bigcup_{i=1}^{p} D_k^i$ é também uma relação de dominância, geralmente não transitiva mesmo se $D_k^i,\ i = 1, \ldots, p$ forem transitivas.

Para se ter uma implementação eficiência do algoritmo de programação dinâmica é recomendável utilizar múltiplas relações de dominância $D_k^1, \ldots, D_k^p (p \leqslant 1)$ a cada execução da k-ésima iteração $k = 1, \ldots, n$ uma vez que cada relação D_k^i explora características específicas.

Algorithm 2 Algoritmo de programação dinâmica utilizando múltiplas relações de dominância.

1: **function** DP
2: $C^0 \leftarrow \{(0, \ldots, 0)\}$
3: **for** $k \leftarrow 1, n$ **do**
4: $C_k^0 \leftarrow C_{k-1} \cup \left\{(s_{k-1}^1 + p_k^1, \ldots, s_{k-1}^m + p_k^m, s_{k-1}^{p+1} + w_k) \mid s_{k-1}^{p+1} + w_k \leqslant W, s_{k-1} \in C_{k-1}\right\}$
5: **for** $i \leftarrow 1, p$ **do**
6: Determinar um conjunto C_k^i cobertura do conjunto C_{k-1}^i com respeito a D_k^i
7: **end for**
8: $C_k \leftarrow C_k^p$
9: **end for**
10: **return** C_n
11: **end function**

Proposição 1. *Para quaisquer relações de dominância $D_k^1, \ldots, D_k^p (p \geqslant 1)$ sobre S_k, o conjunto C_k^p obtido pelo Algoritmo 2 em cada iteração é um conjunto cobertura de C_k^0 com respeito a $D_k = \bigcup_{i=1}^{p} D_k^i$ $(k = 1, \ldots, n)$.*

Demonstração. Considere $s_k \in C_k^0 \backslash C_k^p$, este foi removido quando selecionado um conjunto cobertura na iteração da linha 5. Seja $i_1 \in \{1, \ldots, p\}$ a iteração da linha 5, tal que $s_k \in C_k^{i_1 - 1} \backslash C_k^{i_1}$. Uma vez que $C_k^{i_1}$ é um conjunto cobertura de $C_k^{i_1 - 1}$ com respeito a $D_{k'}^{i_1}$, existe $s_{k'}^{(1)} \in C_k^{i_1}$ tal que $s_{k'}^{(1)} D_k^{i_1} s_k$. Se $s_{k'}^{(1)} \in C_k^m$ então as propriedades de cobertura são válidas, uma vez que $D_k^{i_1} \subseteq D_k$. Caso contrário, existe uma iteração $i_2 > i_1$, correspondente à iteração da linha 5 tal que $s_{k'}^{(1)} \in C_k^{i_2 - 1} \backslash C_k^{i_2}$. Como anteriormente, estabelecemos que existe $s_{k'}^{(2)} \in C_k^{i_2}$ tal que $s_{k'}^{(2)} D_k^{i_2} s_{k'}^{(1)}$. Uma vez que $D_k^{i_2} \subseteq D_k$ temos que $s_{k'}^{(2)} D_k s_{k'}^{(1)} D_k s_k$ e por transitividade de D_k garantimos que $s_{k'}^{(2)} D_k s_k$. Pela repetição desde processo podemos garantir que a existência de um estado $s_{k'} \in C_k^m$, tal que $s_{k'} D_k s_k$. □

Temos agora condições para garantir que o Algoritmo 2 gera o conjunto de vetores objetivos não dominados.

Teorema 1. *Para qualquer família de relações de dominância $D_k^i (i = 1, \ldots, p; k = 1, \ldots, n)$, o Algoritmo 2 retorna C_n que é um conjunto cobertura de S_n com respeito a $\underline{\triangle}$. Além disso, se na iteração n utilizarmos ao menos uma relação $D_n^i = \underline{\triangle}$ e garantirmos que o conjunto cobertura C_n^i é também independente com respeito a D_n^i então C_n representa o conjunto ND de vetores objetivos não dominados.*

Demonstração. Considere $s_{n'} \in S_n \backslash C_n$, todas as restrições foram removidas ao reter-se o conjunto cobertura com respeito a $D_k = \cup_{i=1}^m D_k^i$ durante as iterações $k \leqslant n$. Seja k_1 a iteração mais alta em que $C_{k_1}^0$ ainda contém restrições de $s_{n'}$, as quais serão removidas aplicando uma das relações $D_{k_1}^i (i = 1, \ldots, p)$. Considere alguma destas restrições, denotada por $s_{k_1'}^{(n)}$. Uma vez que $s_{k_1'}^n \in C_{k_1}^0 \backslash C_{k_1}$ sabemos, pela Proposição 1, que existe $s_{k_1} \in C_{k_1}$ tak que $s_{k_1} D_{k_1} s_{k_1'}^{(n)}$. Pela Equação 2.1, uma vez que D_k é uma relação de dominância, temos que todas as extensões de $s_{k_1}^{(n)}$, e particularmente para $s_{n'}$, existe $s_{n_1} \in Ext(s_{k_1})$ tal que $s_{n_1} \underline{\triangle} s_{n'}$. Se $s_{n_1} \in C_n$, então a propriedade de cobertura é válida. Caso contrário, existe uma iteração $k_2 > k_1$, correspondente a iteração mais alta em que $C_0^{k_2}$ ainda contém restrições de s_{n_1}, que serão removidas aplicando-se uma das relações $D_{k_2}^i (i = 1, \ldots, p)$. Considere uma destas restrições, denotada por $s_{k_2}^{(n_1)}$. Como anteriormente, estabelecemos a existência de um estado $s_{k_2} \in C_{k_2}$ tal que existe $s_{n_2} \in Ext(s_{k_2})$ tal que $s_{n_2} \underline{\triangle} s_{n_1}$. A transitividade de $\underline{\triangle}$ garante que $s_{n_2} \underline{\triangle} s_{n'}$. Pela repetição deste processo, estabelecemos a existência de um estado $s_{k_2} \in C_{k_2}$ tal que existe $s_{n_2} \in Ext(s_{k_2})$ tal que $s_n \underline{\triangle} s_{n'}$.

Além disso, selecionando-se um conjunto C_n^i que é conjunto independente com relação a $D_n^i = \underline{\triangle}$, esta propriedade se mantém válida para C_n^p o qual é subconjunto de C_n^i. Dessa forma C_n, que corresponde ao conjunto eficiente reduzido, representa o conjunto de vetores objetivos não dominados. □

O teorema anterior apenas querer que um dos $n.p$ conjuntos coberturas seja independente com respeito a sua relação de dominância. Mesmo se todos os conjunto C_k^i....

2.1.2 Geração de conjuntos cobertura e independente

Apresentamos no Algoritmo 3 uma maneira de produzir C_k^i um conjunto cobertura e independente de C_k^{i-1} com respeito a relação transitiva D_k^i (linha ??? do Algoritmo 2).

Proposição 2. *Para qualquer relação de dominância D_k^i sobre S_k, o Algoritmo 3 retorna C_k^i um conjunto cobertura e independente de C_k^{i-1} com respeito a $D_k^i (j = 1, \ldots, n; i = 1, \ldots, p)$.*

Demonstração. Claramente C_k^i é independente com relação a D_k^i, uma vez que inserimos o estado s_k ao conjunto C_k^i na linha ???? apenas se não for dominado por nenhum outro estado em C_k^i (linha ???) e todos os estados dominados por s_k tenham sido removidos de C_k^i (linha ??? e ???).

Algorithm 3 Computação do conjunto C_k^i cobertura e independente do conjunto C_k^{i-1} com respeito a relação transitiva D_k^i.

1: **function** $(C_k^{i-1} = \{s_{k(1)}, \ldots, s_{k(r)}\})$
2: $C_k^i \leftarrow \{s_{k(1)}\}$
3: **for** $h \leftarrow 2, r$ **do**
4: Seja $C_k^i = \{s_{k'(1)}, \ldots, s_{k'(l_h)}\}$
5: $dominated \leftarrow false$;
6: $dominates \leftarrow false$;
7: $j \leftarrow 1$;
8: **while** $j \leqslant l_h$ and $\neg dominated$ and $\neg dominates$ **do**
9: **if** $s_{k'(j)} D_k^i s_{k(h)}$ **then**
10: $dominated \leftarrow true$;
11: **else if** $s_{k(h)} D_k^i s_{k'(j)}$ **then**
12: $C_k^i \leftarrow C_k^i \backslash \{s_{k'(j)}\}$;
13: $dominates \leftarrow true$;
14: **end if**
15: $j \leftarrow j + 1$;
16: **end while**
17: **if** $\neg dominated$ **then**
18: **while** $j \leqslant l_h$ **do**
19: **if** $s_{k(h)} D_k^i s_{k'(j)}$ **then**
20: $C_k^i \leftarrow C_k^i \backslash \{s_{k'(j)}\}$;
21: **end if**
22: $j \leftarrow j + 1$;
23: **end while**
24: $C_k^i \leftarrow C_k^i \cup \{s_{k(h)}\}$;
25: **end if**
26: **end for**
27: **return** C_k
28: **end function**

Mostramos agora que C_k^i é um conjunto covertura de C_k^{i-1} com respeito a D_k^i. Considere $s_{k'} \in C_k^{i-1} \backslash C_k^i$. Isto ocorre tanto porque não passa no teste da linha ??? quanto porque foi removido na linha ??? ou ???. Isto é devido respectivamente a um estado s_{-k} já existente em C_k^i ou a ser incluído em C_k^i (na linha ???) tal que $s_{-k} D_k^i s_{k'}$. Pode ser que s_{-k} seja removido de C_k^i em uma iteração posterior da linha ??? se existir um novo estado $s^{\hat{k}} \in C_k^{i-1}$ a ser incluído em C_k^i, tal que $s_{\hat{k}} D_k^i s_{-k}$. Entretanto, a transitividade de D_k^i garante a existência, ao fim da iteração k, de um estado $s_k \in C_k^i$ tal que $s_k D_k^i s_{k'}$. □

O Algoritmo 3 pode ser aprimorado uma vez que geralmente é possível gerar estados de $C_k^{i-1} = \{s_{k(1)}, \ldots, s_{k(r)}\}$ conforme uma *ordem de preservação de dominância* para D_k^i de forma que todo $l < j (1 \leqslant l, j \leqslant r)$ temos tanto $s_{k(l)} D_k^i s_{k(j)}$ ou $\neg(s_{k(j)} D_k^i s_{k(l)})$. A proposição seguinte provê a condição necessária e suficiente para estabelecer a existência da ordem de preservação de dominância para uma relação de dominância.

Proposição 3. *Seja D_k uma relação de dominância sobre S_k. Existe uma ordem de*

preservação de dominância para D_k se, e somente se, D_k não admite ciclos em sua parte assimétrica.

Demonstração.
$\Rightarrow$ A existência de um ciclo na parte assimétrica de D^k implicaria na existência de dois estados consecutivos $s_{k(j)}$ e $s_{k(l)}$ neste ciclo sendo $j > l$, o que é uma contradição.
$\Leftarrow$ Qualquer ordem topológica baseada na parte assimétrica de D_k é uma ordem de preservação de dominância para D_k. □

Na Seção seguinte é apresentado um exemplo de ordem de preservação de dominância. Se os estados em C_k^{i-1} são gerados conforme uma ordem de preservação de dominância para D_k^i, a linha ??? e o laço ???-??? do Algoritmo 3 podem ser omitidos.

2.2 Detalhes de implementação

Primeiramente é apresentada a ordem na qual serão considerados os itens no processo sequencial do algoritmo. Posteriormente são apresentados a três relações de dominância utilizadas no algoritmo ??? e a maneira com que serão utilizadas.

2.2.1 Ordem dos itens

A ordem na qual os itens são considerados é uma questão crucial na implementação do algoritmo. Sabe-se que no caso do problema da mochila unidimensional, para se obter uma boa solução, os itens devem ser geralmente considerados em ordem decrescente segundo a proporção p_i/w_i (EHRGOTT, 2013; MARTELLO; TOTH, 1990). Porém, para o caso multi-objetivo não existe uma ordem natural.

São introduzidas duas ordenações $\mathcal{O}^{sum}$ e $\mathcal{O}^{max}$ derivadas da agregação das ordens $\mathcal{O}^j$ inferidas pelas proporções p_i^j/w_i para cada objetivo ($j = 1, \ldots, m$). Considere r_i^l o rank (ou posição) do item l na ordenação $\mathcal{O}^j$. $\mathcal{O}^{sum}$ denota uma ordenação segundo valores crescentes da soma dos ranks dos itens nas m ordenações ($i = 1, \ldots, m$). $\mathcal{O}^{max}$ denota uma ordenação segundo valores crescentes de máximo ou piores ranks de itens nas m ordenações $\mathcal{O}^j (j = 1, \ldots, m)$, onde o pior rank do item l nas m ordenações $\mathcal{O}^j (j = 1, \ldots, m)$ é calculado como $max_{i=1}^m \{r_l^i\} + \frac{1}{mn} \sum_{i=1}^m r_l^i$ para distinguir itens com o mesmo rank máximo.

> Dizer que o artigo original conclui atraves de testes qual é a melhor ordenação para ser utilizada nas iterações do algoritmo.

2.2.2 Relações de dominância

Cada relação de dominância exploca uma consideração específica. Por isto recomenda-se utilizar relações de dominância que sejam complementares. Além disso, para se escolher

uma relação de dominância é necessário considerar sua potencial capacidade de descarte de estados diante do custo computacional requerido.

A seguir serão apresentas as três relações de dominância utilizadas no algoritmo. As primeiras duas relações são reazoavelmente triviais de se estabelecer sendo a última ainda considerada, mesmo sendo um tanto mais complexa, devido à sua complementaridade diantes das duas primeiras.

A primeira relação de dominância se estabelece segundo a seguinte observação. Quando a capacidade residual de uma solução associada a um estado s_k da iteração k é maior ou igual a soma dos pesos dos itens restantes, i.e. itens $k+1, \ldots, n$, o único complemento de s_k que pode resultar em uma solução eficiente é o complemento máximo $J = \{k+1, \ldots, n\}$. Portanto, neste contexto, não se faz necessário gerar as extensões de s_k que não contenham todos os itens restants. Define-se então a relação de dominância D_k^r sobre S^k para $k = 1, \ldots, n$ como:

$$\forall s_k, s_{k'} \in S_k, \;\; s_k D_k^r s_{k'} \Leftrightarrow \begin{cases} s_{k'} \in S_{k-1}, \\ s_k = (s_{k'}^1 + p_k^1, \ldots, s_{k'}^m + p_k^m, s_{k'}^{m+1} + w_k), \text{ e} \\ s_{k'}^{m+1} \leqslant W - \sum_{i=k}^{n} w_i \end{cases}$$

A seguinte proposição mostra que D_k^r é de fato uma relação de dominância e apresenta propriedades adicionais de D_k^r.

Proposição 4 (Relação D_k^r)**.**

(a) D_k^r *é uma relação de dominância*

(b) D_k^r *é transitiva*

(c) D_k^r *admite ordem de preservação de dominância*

Demonstração.

(a) Considere dois estados s_k e $s_{k'}$ tal que $s_k D_k^r s_{k'}$. Isto implica que $s_k \underline{\triangle} s_{k'}$. Além disso, uma vez que $s_k^{m+1} = s_{k'^{m+1}} + w_k \leqslant W - \sum_{i=k+1}^{n} w_i$, qualquer subconjunto $J \subseteq \{k+1, \ldots, n\}$ é um complemento de $s_{k'}$ e s_k. Portanto, para todo estado $s_{n'} \in Ext(s_{k'})$, existe $s_n \in Ext(s_k)$, baseado no mesmo complemento que $s_{n'}$, tal que $s_n \underline{\triangle} s_{n'}$. Isto estabelece que D_k^r satisfaz a Equação 2.1 da Definição 3.

(b) Trivial.

(c) Pela Proposição 3, uma vez que D_k^r é transitiva.

□

2.3 Métodos Heurísticos

3 A k-d tree

4 Experimentos

5 Conclusão

References

EHRGOTT, M. *Multicriteria optimization.* [S.l.]: Springer Science & Business Media, 2013.

ISHIBUCHI, H.; AKEDO, N.; NOJIMA, Y. Behavior of multiobjective evolutionary algorithms on many-objective knapsack problems. *IEEE Transactions on Evolutionary Computation*, IEEE, v. 19, n. 2, p. 264–283, 2015.

MARTELLO, S.; TOTH, P. *Knapsack Problems, J.* [S.l.]: Wiley & Sons, Chichester, 1990.

ROSENBLATT, M. J.; SINUANY-STERN, Z. Generating the discrete efficient frontier to the capital budgeting problem. *Operations Research*, INFORMS, v. 37, n. 3, p. 384–394, 1989.

TENG, J.-Y.; TZENG, G.-H. A multiobjective programming approach for selecting non-independent transportation investment alternatives. *Transportation Research Part B: Methodological*, Elsevier, v. 30, n. 4, p. 291–307, 1996.