MODELOS HÍBRIDOS PARA O PROBLEMA DA MOCHILA BIDIMENSIONAL COM CONFLITOS

Pedro Hokama
Instituto de Computação - Universidade Estadual de Campinas
hokama@ic.unicamp.br

Thiago A. de Queiroz
Departamento de Matemática, Universidade Federal de Goiás - Campus de Catalão
th.al.qz@catalao.ufg.br

Flávio Keidi Miyazawa
Instituto de Computação - Universidade Estadual de Campinas
fkm@ic.unicamp.br

RESUMO

Neste trabalho estamos interessados no Problema da Mochila Bidimensional com Conflitos. Apresentamos uma modelagem em programação linear inteira que resolve o problema através da abordagem *branch-and-cut*, em que separações são encontradas por um algoritmo modelado por programação por restrições. Comparamos nossos resultados com a literatura, através de testes computacionais que comprovaram a eficiência do nosso método.

PALAVRAS-CHAVE: Programação Linear Inteira, Programação por Restrições, Branch-and-Cut.

1. Introdução

No problema da Mochila Bidimensional com Conflitos, recebemos um conjunto de itens e um recipiente bidimensional, em que cada item possui um valor associado. Também recebemos uma lista de conflitos, ou seja, conjuntos de pares de itens que não são compatíveis. O objetivo é encontrar um subconjunto dos itens, que maximize o valor e que possam ser empacotados no recipiente, sendo que dois itens conflitantes entre si não podem estar simultaneamente no recipiente. Esse problema foi investigado na versão unidimensional por Yamada et al. (2002), e a versão bidimensional que denotaremos por 2KPC, foi recentemente investigado por Queiroz e Miyazawa (2013). Na Seção 2 descrevemos formalmente o problema e na Seção 3 descrevemos a formulação de Queiroz e Miyazawa (2013). Depois apresentamos uma nova abordagem branch-and-cut para o problema na Seção 4. Por fim nas seções 5 e 6 apresentamos os resultados computacionais, conclusões e trabalhos futuros.

2. Descrição do problema

A seguir, definimos o problema da mochila bidimensional com conflitos. São dados como entrada: *(i)* Um conjunto I de itens bidimensionais, onde cada item $i \in I$ tem dimensões (w_i, h_i) e um valor associado v_i; *(ii)* Um recipiente B de dimensões (W, H); *(iii)* Um conjunto de conflitos C, onde cada conflito em C é composto por um par de itens $\{i, j\}$. O objetivo é encontrar um subconjunto $J \subseteq I$ e um empacotamento dos itens de J em B, cujo valor total dos itens em J é maximizado e para todo conflito $\{i, j\}$ pertencente a C, temos que i e j não pertencem a J simultaneamente.

3. Formulação por Programação Linear Inteira

A formulação apresentada por Queiroz e Miyazawa (2013) considera uma malha de pontos, obtidos através dos *reduced raster points* (Scheithauer e Terno, 1996), onde os itens podem ser empacotados. Considere S o conjunto de todos os pontos da malha, e S_i o conjunto de pontos onde o item i pode ser empacotado. Além disso, D_{ip} é o conjunto de pontos q, tal que o item i empacotado em q cobre o ponto p. O número de conflitos em que i está presente é representado por α_i. Considere a variável binária $x_{ip} = 1$ se o item i foi empacotado no ponto p, e 0 caso contrário.

$$\max \quad \sum_{i \in I} \sum_{s \in S_i} x_{ip} v_i, \tag{1}$$

$$\text{sujeito a} \quad \sum_{p \in S_i} x_{ip} \leq 1, \qquad i \in I, \tag{2}$$

$$\sum_{i \in I} \sum_{q \in D_{ip}} x_{iq} \leq 1, \qquad \forall p \in S, \tag{3}$$

$$\sum_{\{i,j\} \in C} \sum_{q \in S_j} x_{jq} \leq (1 - \sum_{p \in S_i} x_{ip})|\alpha_i|, \quad \forall i \in I, \tag{4}$$

$$x_{ip} \in \{0, 1\}, \qquad i \in I; \forall p \in S_i. \tag{5}$$

Nessa formulação as restrições (2) garantem que um item da solução só está atribuído a no máximo um ponto. As restrições (3) evitam que dois itens empacotados ocupem um mesmo ponto p. As restrições (4) garantem que se um item i for empacotado em qualquer ponto, nenhum dos itens conflitantes estará na solução. Por fim as restrições (5) garantem que as variáveis são binárias. Denotaremos essa formulação como *Grid*.

4. Algoritmo Branch-and-cut com Programação por Restrições

Nessa formulação o problema da mochila com conflitos é resolvida por Programação Linear Inteira e a cada nó verificamos se a solução parcial pode ser empacotada. Essa verificação é feita por um resolvedor utilizando Programação por Restrições. Para essa formulação, seja q_i uma variável binária que indica a pertinência do item i na solução. Seja $s_i = h_i w_i$ o volume do item i e $S = HW$ o volume no recipiente B. Seja $\mathcal{R}$ o conjunto de todas as combinações de itens que não podem ser empacotadas simultaneamente no recipiente. Por exemplo, se o conjunto de itens $\{3, 5, 8\}$ não pode ser empacotado no recipiente, então $\{3, 5, 8\} \in \mathcal{R}$. Dessa forma podemos formular o problema como segue,

$$\max \quad \sum_{i \in I} q_i v_i, \tag{6}$$

$$\text{sujeito a} \quad \sum_{i \in I} q_i s_i \leq S, \tag{7}$$

$$q_i + q_j \leq 1, \qquad \forall \{i, j\} \in C, \tag{8}$$

$$\sum_{i \in R} q_i \leq |R| - 1, \quad \forall R \in \mathcal{R}, \tag{9}$$

$$q_i \in \{0, 1\} \qquad \forall i \in I. \tag{10}$$

Obviamente, o número de elementos em $\mathcal{R}$ pode ser exponencial, portanto relaxamos a formulação, removendo as restrições (9), e adicionamos as restrições dessa família como cortes. Infelizmente descobrir se uma combinação de itens pertence a $\mathcal{R}$ é um problema NP-difícil, já que é equivalente a resolver um problema de empacotamento bidimensional.

Para resolver o problema de empacotamento bidimensional para uma dada combinação de itens R, utilizamos nessa fase uma modelagem em Programação por Restrições descrita a seguir. Seja X_i a variável que representa a coordenada X em que o item i será empacotado,

e Y_i a variável que representa a coordenada Y em que o item i será empacotado. Iniciamos a formulação atribuindo o domínio as variáveis.

$$Dom(X_i) = [0, \ldots, W - w_i], \tag{11}$$
$$Dom(Y_i) = [0, \ldots, H - h_i]. \tag{12}$$

Como a restrição (8) é respeitada pelos itens de R, não há conflitos e a única restrição necessária é a de que os itens não se sobreponham, dessa forma adicionamos:

$$\begin{aligned} &X[i] + w[i] <= X[j] \text{ ou } Y[i] + h[i] <= Y[j] \text{ ou} \\ &X[j] + w[j] <= X[i] \text{ ou } Y[j] + h[j] <= Y[i]. \end{aligned} \tag{13}$$

Se o resolvedor de programação por restrições provar que o conjunto de itens R não pode ser empacotado, adicionamos um corte da família (9), senão continuamos o algoritmo. Denotaremos essa formulação como *BNC-CP*.

5. Resultados

Os algoritmos propostos foram implementados em c++ e compilados com g++ versão 4.6.4 em um computador com sistema operacional Linux. Os experimentos foram executados em um Processador Intel Xeon 2.93GHz. O resolvedor de programação linear escolhido foi o CPLEX 12.5.1 e o resolvedor de Programação por Restrições foi o CP solver 1.7, ambos da IBM ILOG.

A Tabela 1 apresenta as 48 instâncias geradas por Queiroz e Miyazawa (2013), a quantidade de itens e a quantidade de conflitos em cada instância. Também é apresentado o valor da solução obtida, o tempo de execução e se foi provada a otimalidade da solução por 3 diferentes formulações. A primeira é a formulação *Grid* apresentada em (Queiroz e Miyazawa, 2013) e descrita na Seção 3., a segunda é a formulação *BNC-CP*, a terceira formulação é idêntica a formulação *BNC-CP*, porém substituindo o modelo que encontra a viabilidade do empacotamento pelo descrito por Clautiaux et al. (2008), esse modelo denominamos por *BNC-CP2D*. Para a formulação Grid o tempo de execução foi limitado a 3200 segundos, e nesse tempo conseguiu provar a otimalidade de 31 das 48 instâncias com um tempo médio de 1640,80 segundos. A abordagem branch-and-cut se mostrou muito eficiente. Para os modelos seguintes o tempo de execução foi limitado a 600 segundos. Nesse limite o modelo BNC-CP apenas não provou a otimalidade em 2 instâncias, porém encontrou uma solução ótima em todas as 48 instâncias, com um tempo médio de execução de 51 segundos.

Já a formulação BNC-CP2D se mostrou a mais eficiente, resolvendo todas as instâncias na otimalidade, com um tempo médio de apenas 0.03 segundos, sendo que a instância que levou mais tempo, foi executada em apenas 0.55 segundos. O resultado desse modelo foi bastante satisfatório.

6. Conclusões e/ou perspectivas futuras

Nesse trabalho investigamos um problema da mochila bidimensional com conflitos, um problema pouco investigado na literatura. Apresentamos um modelo que melhorou consideravelmente o tempo para encontrar soluções ótimas. Estamos investigando novos modelos e melhorias e esperamos apresentá-las em trabalhos futuros.

Referências

Clautiaux, F.; Jouglet, A.; Carlier, J.,; Moukrim, A. (2008). A new constraint programming approach for the orthogonal packing problem. *Computers and Operations Research*, 35(3), 944 – 959.

Tabela 1: Instâncias e Resultados anteriores

			Grid			BNC-CP			BNC-CP2D		
Instancia	Itens	Conflitos	Valor	Tempo	Otimo	Valor	Tempo	Otimo	valor	Tempo	Opt
2kdc01	25	21	6203	91.82	SIM	6203	17.54	SIM	6203	0.05	SIM
2kdc02	25	21	5568	504.55	SIM	5568	5.19	SIM	5568	0.19	SIM
2kdc03	25	21	2354	0.13	SIM	2354	0.14	SIM	2354	0.05	SIM
2kdc04	25	21	7487	3200	NAO	7487	0.54	SIM	7487	0.01	SIM
2kdc05	25	30	4245	1173.77	SIM	4245	2.11	SIM	4245	0.2	SIM
2kdc06	25	30	4516	1484.72	SIM	4516	137.25	SIM	4516	0.24	SIM
2kdc07	25	30	2793	4.26	SIM	2793	0.25	SIM	2793	0.31	SIM
2kdc08	25	30	7905	3200	NAO	7905	0.12	SIM	7905	0.01	SIM
2kdc09	25	38	4663	3200	NAO	4663	12.57	SIM	4663	0.08	SIM
2kdc10	25	38	3826	174.77	SIM	3826	0.77	SIM	3826	0.04	SIM
2kdc11	25	38	3764	1.39	SIM	3764	0.02	SIM	3764	0	SIM
2kdc12	25	38	7701	3109.21	SIM	7701	0.26	SIM	7701	0.01	SIM
2kdc13	25	47	4649	391.32	SIM	4649	14.15	SIM	4649	0.29	SIM
2kdc14	25	47	4754	679.28	SIM	4754	4.28	SIM	4754	0.03	SIM
2kdc15	25	47	3350	0.47	SIM	3350	0.07	SIM	3350	0.02	SIM
2kdc16	25	47	5396	3200	NAO	5396	1.45	SIM	5396	0.26	SIM
2kdc17	25	21	8138	1.69	SIM	8138	137.56	SIM	8138	0.11	SIM
2kdc18	25	21	6923	3.93	SIM	6923	0.16	SIM	6923	0.29	SIM
2kdc19	25	21	6560	0.66	SIM	6560	0.11	SIM	6560	0.03	SIM
2kdc20	25	21	12608	3200	NAO	12608	520.39	SIM	12608	0.58	SIM
2kdc21	25	30	11209	139.14	SIM	11209	0.89	SIM	11209	0.09	SIM
2kdc22	25	30	6199	1.11	SIM	6199	9.09	SIM	6199	0.29	SIM
2kdc23	25	30	4207	0.08	SIM	4207	0.11	SIM	4207	0.24	SIM
2kdc24	25	30	12385	3200	NAO	12385	6.88	SIM	12385	0.08	SIM
2kdc25	25	38	9552	3200	NAO	9552	10	SIM	9552	0.08	SIM
2kdc26	25	38	7045	547.99	SIM	7045	2.35	SIM	7045	0.07	SIM
2kdc27	25	38	4555	0.36	SIM	4555	0.22	SIM	4555	0.41	SIM
2kdc28	25	38	11838	3200	NAO	11838	0.05	SIM	11838	0.01	SIM
2kdc29	25	47	5677	688.7	SIM	5677	2.6	SIM	5677	0.53	SIM
2kdc30	25	47	8402	133.77	SIM	8402	0.67	SIM	8402	0.02	SIM
2kdc31	25	47	7158	3131.89	SIM	7158	0.13	SIM	7158	0.02	SIM
2kdc32	25	47	10228	3200	NAO	10228	6.32	SIM	10228	0.02	SIM
2kdc33	25	21	15516	3200	NAO	15516	648.24	NAO	15516	0.09	SIM
2kdc34	25	21	14258	3200	NAO	14258	2.49	SIM	14258	0.03	SIM
2kdc35	25	21	7545	58.21	SIM	7545	0.17	SIM	7545	0.37	SIM
2kdc36	25	21	23253	3200	NAO	23391	220.34	SIM	23391	0.1	SIM
2kdc37	25	30	13512	0.4	SIM	13512	2.65	SIM	13512	0.02	SIM
2kdc38	25	30	16011	3126.36	SIM	16011	31.02	SIM	16011	0.03	SIM
2kdc39	25	30	6647	0.25	SIM	6647	0.25	SIM	6647	0.5	SIM
2kdc40	25	30	18012	3200	NAO	18210	4.4	SIM	18210	0.01	SIM
2kdc41	25	38	9529	31.33	SIM	9529	5.03	SIM	9529	0.36	SIM
2kdc42	25	38	10849	3131.48	SIM	10849	0.88	SIM	10849	0.48	SIM
2kdc43	25	38	9387	2.15	SIM	9387	0.15	SIM	9387	0.04	SIM
2kdc44	25	38	17559	3200	NAO	17559	0.04	SIM	17559	0.01	SIM
2kdc45	25	47	10920	3200	NAO	10920	1.55	SIM	10920	0.32	SIM
2kdc46	25	47	15501	3200	NAO	15501	2.75	SIM	15501	0.03	SIM
2kdc47	25	47	10633	63.09	SIM	10633	0.12	SIM	10633	0.29	SIM
2kdc48	25	47	17913	3200	NAO	17913	632.2	NAO	17913	0.39	SIM

Queiroz, T. A.; Miyazawa, F. K. (2013). Approaches for the 2d 0-1 knapsack problem with conflict graphs. Em *Anais do CLEI 2013, 2013*, 1–8, Naiguata - Venezuela.

Scheithauer, G.; Terno, J. (1996). The g4-heuristic for the pallet loading problem. *Journal of the Operational Research Society*, 47, 511–522.

Yamada, T.; Kataoka, S.,; Watanabe, K. (2002). Heuristic and exact algorithms for the disjunctively constrained knapsack problem. *Information Processing Society of Japan Journal*, 43(9).