

Joanna JÓZEFOWSKA, Wojciech SURDYKA, Natalia SHAKHLEVICH
Politechnika Poznańska

INSTANCJE TESTOWE DLA JEDNOWYMIAROWEGO PROBLEMU PAKOWANIA Z KONFLIKTAMI

Streszczenie. W pracy przedstawiono analizę algorytmów generowania instancji testowych dla problemu pakowania z konfliktami. Porównano jakość rozwiązań uzyskiwanych dla różnych typów grafu konfliktów.

BENCHMARKS FOR ONE DIMENSIONAL BIN PACKING PROBLEM WITH CONFLICTS

Summary. Algorithms for generating benchmarks for the bin packing problem with conflicts are analyzed. Solutions obtained by simple greedy heuristics are compared with those obtained by the Hexaly solver for various conflict graphs.

1. Sformułowanie problemu

Problem pakowania (ang. *Bin Packing*) jest jednym z najbardziej znanych problemów optymalizacji kombinatorycznej, rozważanym w literaturze od lat siedemdziesiątych dwudziestego wieku [13].

Problem pakowania formułuje się jako problem przyporządkowania elementów ze skończonego zbioru $\mathcal{I} = \{1, 2, \dots, n\}$, o rozmiarach, odpowiednio $s_1, \dots, s_n$, do jak najmniejszej liczby pojemników o ograniczonej pojemności c . Bez utraty ogólności zakłada się, że $0 < s_i < c, i \in \mathcal{I}$. Podzbiór elementów $\mathcal{I}' \subseteq \mathcal{I}$ można przydzielić do tego samego pojemnika, gdy $\sum_{i \in \mathcal{I}'} s_i \leq c$.

Problem ten jest silnie NP-trudny [13]. W literaturze przedstawiono wiele propozycji algorytmów przybliżonych, jak i dokładnych rozwiązywania tego problemu. Obszerną monografię problemu stanowi praca [10].

Problem pakowania doczekał się różnorodnych rozszerzeń, między innymi rozważa się problem wielowymiarowy [9], problem z różnymi rozmiarami pojemników o różnych kosztach w celu zapakowanie elementów przy najniższym całkowitym koszcie [6], czy problem z konfliktami wprowadzony przez Jansena i Oehringa w roku 1997 [8], którego znaczenie wzrosło ostatnio w kontekście alokacji zadań w systemach rozproszonych [11].

W problemie pakowania z konfliktami uwzględnia się dodatkowo listę par elementów, które nie mogą się znaleźć w tym samym pojemniku. Wygodnie jest przedstawić tę listę w postaci grafu $G(E, V)$, gdzie E jest zbiorem wierzchołków odpowiadających pakowanym elementom, a V jest zbiorem łuków odpowiadających konfliktom, tj., jeżeli łuk $(i, j) \in V, i, j = 1, \dots, n, i \neq j$, to elementy i oraz j nie mogą być przy-

dzielone do tego samego pojemnika. Rozwiązanie dopuszczalne musi zatem dodatkowo spełniać ograniczenie $(i, j) \in V \wedge i \in \mathcal{I}' \implies j \notin \mathcal{I}'$.

Najprostsze heurystyki stosowane do rozwiązywania problemu pakowania bazują na określonej kolejności zadań i pojemników. Kolejne zadanie z listy jest przydzielane do pojemnika z najwyższym priorytetem, dla którego spełnione są warunki dopuszczalności. Gdy element nie może być przydzielony do żadnego z "otwartych" pojemników, otwiera się kolejny pojemnik (zwiększając tym samym wartość funkcji celu o jeden). Przez r_k oznaczono dostępną pojemność pojemnika k obliczaną jako $r_k = c - \sum_{i \in \mathcal{I}_k} s_i$. W algorytmie FF (ang. *First Fit*) pojemniki są rozważane w dowolnej kolejności, BF (ang. *Best Fit* wg niemalejących wartości r_k , a w WF (ang. *Worst Fit*) wg nierosnących wartości r_k . W wariantach FFD, BFD, WFD pakowane elementy są rozważane w kolejności nierosnących rozmiarów s_i . Heurystyki powyższe łatwo można rozszerzyć do problemu pakowania z konfliktami, sprawdzając w każdym kroku dodatkowo występowanie konfliktów. W niniejszej pracy nie rozważa się bardziej zaawansowanych heurystyk.

2. Instancje testowe

Istnieje kilka zbiorów instancji testowych, tzw. *benchmarks* dla problemu pakowania, można je znaleźć na portalu Uniwersytetu w Bolonii [1]. Najciekawszym zbiorem jest zaproponowany przez Falkenauera [5] zbiór zawierający instancje skonstruowane w ten sposób, że w rozwiązaniu optymalnym w każdym pojemniku muszą się znaleźć dokładnie trzy elementy. Biblioteka zawiera jednak znacznie więcej propozycji. Naturalne było skonstruowanie instancji testowych dla problemu pakowania z konfliktami poprzez dodanie do instancji dla problemu BPP grafów konfliktów. Podejście takie zrealizowali Genreau, Laporte i Semet [7]. Użyte przez nich instancje są również dostępne na portalu Uniwersytetu w Bolonii [1] i przez wiele lat były wykorzystywane jako benchmarki dla problemu pakowania z konfliktami. W roku 2017 ukazała się praca Bacci i Nicoloso [4], w której wskazano, że algorytm użyty w [7] generuje grafy o specyficznej strukturze, tzw. grafy progowe *threshold graphs*, mimo, że autorzy powoływali się na pracę, w której zaproponowano algorytm generujący całkowicie dowolne grafy o zadanej gęstości [12]. Ponieważ nadal instancje zaproponowane w [7] są wykorzystywane w eksperymentach badających efektywność algorytmów dla problemu pakowania z konfliktami w niniejszej pracy zaproponowano nowe instancje, bazujące na zbiorach Falkenauera, ale z grafami wygenerowanymi wg algorytmu przedstawionego w pracy [12]. Instancje te udostępniono na portalu Vectorpack [3].

3. Eksperymenty obliczeniowe

Przeprowadzono wstępne eksperymenty obliczeniowe w celu zbadania, jak sposób generowania grafu konfliktów wpływa na jakość rozwiązań uzyskiwanych przez najprostsze heurystyki. Algorytmy heurystyczne zaimplementowano w języku C++ w środowisku Vectorpack [3]. Obliczenia przeprowadzono na komputerze z procesorem Intel Core i5-4310U z dwoma rdzeniami fizycznymi i czterema wątkami sprzętowymi oraz 3,4 GB RAM. Rozwiązania referencyjne zostały obliczone przez solver Hexaly [2] na komputerze z systemem operacyjnym Windows 11 Pro i procesorem Intel Core

i5-7500 z częstotliwością 3,4 GHz i 24 GB RAM.

Jako podstawę do wygenerowania instancji testowych przyjęto zbiory instancji zaproponowanych w [5]: 4 zbiory po 10 instancji z dowolnymi losowo generowanymi rozmiarami elementów (120, 250, 500 i 1000 elementów) oraz 4 zbiory po 10 instancji z elementami tworzącymi tzw. "trójki" *triplets* (60, 120, 249 i 501 elementów). Dla każdego zbioru wygenerowano grafy konfliktów o gęstości w przybliżeniu 20%, 40%, 60% i 80% w dwóch różnych wariantach zróżnicowania stopni wierzchołków w grafie [12]. Ponadto każdy z algorytmów losowych został wykonany dla dwóch różnych wartości startowych. Dla każdego zbioru elementów mamy zatem 5 grafów konfliktów o tej samej gęstości: 4 dowolne i 1 progowy. Ponieważ wyniki dla grafów dowolnych były bardzo podobne w tabeli 1 pokazano średnią dla wszystkich 40 instancji. Dla przykładu w tabeli 1 przedstawiono średnie względne odchylenie rozwiązań dla grafów dowolnych (kolumna A) oraz progowych (kolumna T) od dolnego ograniczenia znalezione przez solver Hexaly uzyskane dla instancji b_3 ($n = 500$) oraz b_7 ($n = 249$, tzw. "trójki"). OPT oznacza względną liczbę rozwiązań optymalnych znalezionych przez solver Hexaly w danym zbiorze instancji.

Tabela 1

Względne odchylenie od dolnego ograniczenia

Zbiór b_3								
Gęstość grafu	20%		40%		60%		80%	
	A	T	A	T	A	T	A	T
FFD	1,95	11,42	3,06	25,94	5,77	16,88	11,51	7,45
FF	6,11	6,77	6,61	11,14	7,85	8,85	11,66	3,36
BFD	1,90	11,86	3,02	26,34	5,61	16,98	11,46	7,56
BF	5,42	5,98	5,86	13,71	7,18	10,61	11,03	4,42
WFD	1,95	10,13	3,33	19,22	6,61	11,67	13,26	4,32
WF	14,82	15,79	15,39	21,63	15,94	12,87	18,71	5,65
Hexaly	0,05	0	0,16	4,81	0,62	3,28	3,24	1,63
OPT	90	100	67,5	0	7,50	0	0	60
Zbiór b_7								
Gęstość grafu	20%		40%		60%		80%	
	A	T	A	T	A	T	A	T
FFD	15,30	14,82	15,66	15,75	17,51	5,01	22,41	1,96
FF	12,68	12,17	13,34	8,93	15,20	3,49	20,27	1,66
BFD	15,30	14,82	15,60	16,56	17,39	5,63	22,29	2,11
BF	12,53	11,93	13,10	12,75	14,78	5,82	20,12	2,26
WFD	15,21	14,70	15,99	16,36	17,82	5,89	23,87	2,46
WF	12,98	12,77	13,80	12,22	16,88	4,02	23,81	2,02
Hexaly	1,05	0,12	1,20	0	1,57	0	2,83	0
OPT	10	100	0	100	7,50	100	37,5	100

4. Wnioski

Przeprowadzone eksperymenty nie w pełni potwierdziły wyniki przedstawione w pracy [4]. W zbiorze b_7 solver znajduje rozwiązania optymalne dla wszystkich instancji z grafami progowymi, również heurystyki zasadniczo znajdują rozwiązania bliższe optimum dla grafów progowych. Różnica rośnie wraz z gęstością grafu. Regularność ta nie występuje jednak w zbiorze b_3. W tym przypadku jakość rozwiązań znajdowanych przez solver jak i heurystyki zachłanne jest lepsza dla grafów progowych jedynie w przypadku grafów o gęstości 80% .

LITERATURA

1. BPPLIB – A Bin Packing Problem Library. <https://site.unibo.it/operations-research/en/research/bpplib-a-bin-packing-problem-library>.
2. Hexaly, Gurobi, OR-Tools on the Bin Packing Problem with Conflicts (BPPC). <https://www.hexaly.com/benchmarks/hexaly-gurobi-or-tools-bin-packing-problem-with-conflicts-bppc>.
3. Vectorpack. <https://github.com/Vectorpack/Vectorpack>.
4. Bacci T., Nicoloso S.: On the benchmark instances for the bin packing problem with conflicts, In: Graphs and Combinatorial Optimization: from Theory to Applications: CTW2020 Proceedings, Springer, 2020, p. 171–179.
5. Falkenauer E.: A hybrid grouping genetic algorithm for bin packing, Journal of heuristics, 1996, vol. 2, p. 5–30.
6. Friesen D., Langston M.: Variable sized bin packing, SIAM Journal on Computing, 02 1986, vol. 15, p. 222–230.
7. Gendreau M., Laporte G., Semet F., Semet F.: Heuristics and lower bounds for the bin packing problem with conflicts, Computers & Operations Research, 2004, vol. 31, p. 347–358.
8. Jansen K., Oehring S.: Approximation algorithms for time constrained scheduling, Information and Computation, 1997, vol. 132, p. 85 – 108.
9. Kou L., Markowsky G.: Multidimensional bin packing algorithms, IBM Journal of Research and Development, 10 1977, vol. 21, p. 443 – 448.
10. Martello S., Toth P.: Knapsack problems: Algorithms and computer implementations. John Wiley & Sons, 1990.
11. Mommessin C., Yang R., Shakhlevich N. V., Sun X., Kumar S., Xiao J., Xu J., Xu J.: Affinity-aware resource provisioning for long-running applications in shared clusters, Journal of Parallel and Distributed Computing, 2023, vol. 177, p. 1–16.
12. Soriano P., Gendreau M.: Tabu search algorithms for the maximum clique problem, In: Johnson D., Trick M., editors DIMACS - Series in Discrete Mathematics and Theoretical Computer Science, AMS, 1996, p. 221–242.
13. Ullman J.: The performance of a memory allocation algorithm. No. 1000 Princeton University, Princeton, N.J. 1971.