ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 2002
Seria: AUTOMATYKA z. 136 Nr kol. 1556

Krzysztof FLESZAR
Politechnika Warszawska

NOWE HEURYSTYCZNE METODY ROZWIĄZYWANIA JEDNOWYMIAROWEGO PROBLEMU BIN-PACKING

Streszczenie. Zadanie upakowania danego zestawu elementów w możliwie małej liczbie jednakowych pojemników, zwane bin-packing, występuje często w problemach dystrybucji i produkcji. W referacie opisuję kilka nowych metod heurystycznych rozwiązywania jednowymiarowej wersji tego problemu. Część metod opiera się na heurystyce MBS (minimum bin slack) autorstwa Gupty i Ho, która znajduje rozwiązanie wyznaczając najlepsze możliwe wypełnienie kolejnych pojemników. Inna metoda bazuje na metaheurystyce VNS (variable neighbourhood search) Mladenovićia i Hansena, stosowanej do rozwiązywania wielu problemów optymalizacji dyskretnej. Eksperymenty wykonane z wykorzystaniem standardowych testów porównawczych pokazują, że prezentowane metody mogą konkurować nie tylko z innymi heurystykami, ale również z metodami dokładnymi.

NEW HEURISTICS FOR ONE-DIMENSIONAL BIN-PACKING

Summary. Several new heuristics for solving the one-dimensional bin packing problem are presented. Some of these are based on the minimum bin slack (MBS) heuristic of Gupta and Ho. A different algorithm is one based on the variable neighbourhood search metaheuristic of Mladenović and Hansen. When tested on standard benchmark problem instances, algorithms proved capable of achieving very good results in comparison to other methods, both heuristic and optimum seeking.

1. Wprowadzenie

Problem *bin-packing* można sformułować w następujący sposób. Dany jest zestaw n elementów o wielkościach t_i, gdzie $i = 1,...,n$, oraz nieograniczona liczba pojemników o jednakowej pojemności c (wszystkie wielkości są liczbami naturalnymi). Należy znaleźć takie przyporządkowanie wszystkich elementów do pojemników, aby nie przekroczyć pojemności pojemników oraz żeby liczba użytych pojemników była minimalna.

Bin-packing należy do klasy problemów NP-trudnych [3]. Istnieje wiele heurystycznych i dokładnych procedur jego rozwiązywania. Najprostsze i najbardziej znane są heurystyki FFD (*first-fit descending*) i BFD (*best-fit descending*). Sortują one elementy według malejących wielkości, potem umieszczają kolejne elementy w pojemnikach w następujący sposób: FFD umieszcza element w pierwszym pojemniku, w którym jest jeszcze wystarczająco miejsca, a BFD umieszcza element w pojemniku, w którym jest najmniej, ale wciąż wystarczająco wolnego miejsca.

Scholl, Klein i Jürgens podają przegląd istniejących metod rozwiązywania problemu *bin-packing* [7]. Poza podanymi przez nich metodami należy wspomnieć o dwóch nowych algorytmach. Jeden z nich to algorytm dokładny autorstwa Carvalho [8], będący połączeniem metody podziału i oszacowań z metodą generacji kolumn, drugi to heurystyka MBS (*minimum bin slack*), której autorami są Gupta i Ho [4].

W tym referacie prezentowanych jest kilka nowych heurystyk. Większość z nich bazuje na wspomnianej wcześniej heurystyce MBS, a jedna jest zastosowaniem metaheurystyki VNS (*variable neighbourhood search*), której twórcami są Mladenović i Hansen [5, 6]. Szczegółowe wyniki prezentowanej tutaj pracy znajdują się w [2].

2. Heurystyka MBS

Minimum bin slack (MBS) [4] buduje rozwiązanie wypełniając po jednym pojemniku na raz. Dla pierwszego pojemnika wyznaczany jest zbiór elementów lokalnie optymalny, tzn. taki, który nie przekracza pojemności pojemnika i pozostawia w pojemniku najmniejszą możliwą wolną przestrzeń. Po znalezieniu takiego zbioru jego elementy są umieszczane w pojemniku. Dla kolejnego pojemnika zbiór elementów lokalnie optymalny wyznaczany jest z wyłączeniem wcześniej użytych elementów. Wypełnianie pojemników kończy się, gdy użyte zostaną wszystkie elementy.

Najlepszy zbiór elementów wypełniający jeden pojemnik wyznaczany jest przez przegląd wszystkich możliwych podzbiorów zbioru dostępnych elementów w kolejności leksykograficznej zgodnej z malejącymi wielkościami elementów. Algorytm wyznacza takie podzbiory przez posortowanie elementów w kolejności nierosnących wielkości, a następnie próbne przypisywanie kolejnych elementów do podzbioru testowego. Procedura kończy się,

gdy znaleziony zostaje podzbiór wypełniający pojemnik całkowicie, lub gdy przetestowane zostaną wszystkie możliwe podzbiory.

Implementacje opisywanej procedury można znaleźć w [2, 4]. Jej pesymistyczna złożoność obliczeniowa wynosi $O(2^n)$, jednak w praktycznym zastosowaniu czas obliczeń może nie być wygórowany z następujących powodów:

- Liczba elementów mieszczących się w jednym pojemniku jest zazwyczaj nieduża, co znacznie redukuje liczbę przeglądanych podzbiorów. Jeżeli w jednym pojemniku mieści się nie więcej niż m elementów, to pesymistyczna złożoność procedury wynosi $O(n^m)$.
- Podzbiory całkowicie wypełniające początkowe pojemniki są zwykle wyznaczane po bardzo niewielu próbach.
- MBS przegląda najpierw podzbiory zawierające duże elementy o niewielkiej liczności, co przyśpiesza ich wyznaczanie.

Ostatnia własność nie tylko przyśpiesza poszukiwanie rozwiązania, ale również poprawia średnią jakość rozwiązań. Gdyby poszukiwania zużywały najpierw najmniejsze elementy, to pod koniec zostałoby dużo elementów dużych, trudnych do upakowania.

Gupta i Ho pokazują, że MBS daje znacznie lepsze wyniki, niż proste heurystyki FFD lub BFD [4]. Wadą algorytmu jest to, że w specyficznych przypadkach czas przetwarzania może być bardzo długi. Na przykład wystarczy, aby wielkości elementów były liczbami parzystymi, a wielkość pojemnika – liczbą nieparzystą, aby procedura przeglądała wszystkie możliwe zestawy (żaden zestaw nie wypełni pojemnika całkowicie). W praktyce z takimi przypadkami można sobie poradzić ograniczając maksymalną liczbę przeglądanych zbiorów.

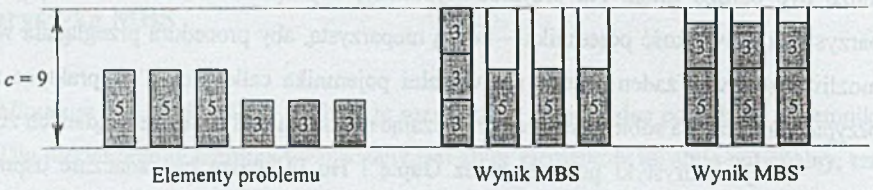
Wersja heurystyki podanej przez Guptę i Ho [4] może być znacznie usprawniona. Możliwe jest znaczne ograniczenie zakresu przeglądanych podzbiorów bez utraty rozwiązania lokalnie optymalnego. Dokładnie temat ten jest opisany w [2].

3. Nowa heurystyka MBS'

Do heurystyki MBS wprowadzamy następującą modyfikację: przed rozpoczęciem wyznaczania lokalnie optymalnego podzbioru wypełniającego pojemnik największy dostępny element jest umieszczany w pojemniku na stałe, a dopiero pozostała przestrzeń jest wypełniana poprzez przegląd mieszczących się w niej podzbiorów. Dzięki temu w pojemniku

pozostaje mniej miejsca do wypełnienia, co skraca czas przeszukiwania. Jednocześnie wymusza to używanie większych elementów w pierwszej kolejności.

Zmodyfikowany algorytm nazwany został MBS'. Jest oczywiste, że wyniki MBS' mogą się różnić od wyników MBS. Nowy algorytm może znaleźć gorsze rozwiązanie w przypadku, gdy zbuduje niepełne zestawy dla pierwszych pojemników zużywając elementy, które mogłyby należeć do pełnych zestawów, gdyby zastosowano MBS. Z drugiej strony oryginalny MBS ma tendencje do wczesnego zużywania małych elementów, jeśli większe elementy sprawiają kłopoty (trudno z ich użyciem znaleźć zestaw dobrze wypełniający pojemnik). Wtedy na koniec przetwarzania pozostają tylko duże elementy, przez co w ostatnich pojemnikach zostaje dużo wolnego miejsca. Przypadek ten ilustruje następujący przykład (rys. 1). 6 elementów o wielkościach $t_1 = t_2 = t_3 = 5$ i $t_4 = t_5 = t_6 = 3$ należy upakować w pojemnikach o wielkości $c = 9$. MBS umieści w pierwszym pojemniku wszystkie elementy o wielkości 3, gdyż jest to jedyny podzbiór, który całkowicie wypełnia pojemnik. W rezultacie pozostałe trzy elementy zajmą trzy kolejne pojemniki. Ponieważ MBS' wymusza użycie aktualnie największego elementu w każdym pojemniku, wszystkie pojemniki zostaną wypełnione parami elementów o wielkościach 5 i 3, dając rozwiązanie zajmujące trzy pojemniki, a więc o jeden mniej, niż w rozwiązaniu z heurystyki MBS.



Rys. 1. Porównanie działania heurystyk MBS i MBS' dla przykładowego problemu

Fig. 1. Comparison of MBS and MBS' heuristics on an example instance

Generalnie żadna z dwóch rozważanych heurystyk nie jest dominująca. W jednym przypadku lepsze rozwiązania może dać MBS, w innym – MBS'. Jednak statystycznie MBS' powinna osiągać lepsze wyniki w krótszym czasie niż MBS.

4. Heurystyki oparte na MBS'

Na podstawie heurystyki MBS' powstały kolejne heurystyki. Każda z nich wyznacza wiele rozwiązań, a jako wynik zwraca najlepsze uzyskane rozwiązanie.

- **Relaxed MBS'**. Modyfikacja wprowadzona w tej heurystyce polega na przerywaniu wyszukiwania lokalnie optymalnego podzbioru wypełniającego pojemnik w przypadku, gdy najlepszy znaleziony dotąd podzbiór pozostawia nie więcej niż v miejsca w pojemniku. Tak zmodyfikowana heurystyka MBS' jest uruchamiana wielokrotnie dla v zmieniającego się od c do $\lceil 0.8c \rceil$ co $\lceil 0.005c \rceil$ (parametry dobrane w oparciu o testy).
- **Perturbation MBS'**. Rozpoczynając od rozwiązania uzyskanego przez MBS', Perturbation MBS' wielokrotnie wykonuje następujące czynności: dodaje nowy pojemnik, wyznacza podzbiór elementów najlepiej wypełniający ten pojemnik, przenosi elementy do nowego pojemnika i usuwa puste pojemniki. Ten krok wykonywany jest 1000 razy. Przy wyznaczaniu podzbioru elementów najlepiej wypełniających nowy pojemnik przeglądane są wszystkie możliwe podzbiory podobnie jak w MBS'. Jednak kolejność przeglądania jest w tym przypadku zależna nie od malejących wielkości elementów, lecz od wielkości miejsc w pojemnikach, w których w aktualnym rozwiązaniu znajdują się elementy. Najpierw rozważane są podzbiory zawierające elementy z mniej, a później z bardziej wypełnionych pojemników. Jeśli elementy znajdują się w pojemnikach o jednakowym stopniu wypełnienia, to kolejność ich rozważania jest losowa.
- **Sampling MBS'**. Sampling MBS' wyznacza wiele rozwiązań wywołując wielokrotnie heurystykę MBS'. W kolejnych przebiegach modyfikowana jest kolejność przeglądania podzbiorów elementów. Kolejność rozważania elementów jest ustalana przed każdym uruchomieniem heurystyki MBS' losowo, z prawdopodobieństwem wybrania elementu proporcjonalnym do jego wielkości. Heurystyka MBS' uruchamiana jest 50 razy.

5. Metaheurystyka VNS

Variable neighbourhood search (VNS) jest metodą optymalizacji globalnej [5, 6]. Dla konkretnego problemu określane są sąsiedztwa $N_k(x)$ rozwiązania x, przy czym $N_{k+1}(x)$ jest szersze niż $N_k(x)$. Poszukiwanie zaczyna się od rozwiązania początkowego $x := x_0$ oraz $k := 1$. Dopóki k nie przekroczy pewnej ustalonej stałej k^{max}, VNS wykonuje następujące czynności:

1. **Wstrząsanie** (*shaking*) - generuje losowo punkt x' należący do sąsiedztwa $N_k(x)$.
2. **Optymalizacja lokalna** (*local optimisation*) - startując z x' znajduje minimum lokalne x''.
3. **Przesunięcie lub nie** (*move or not*) - jeśli x'' jest lepsze niż x, to podstawia $x := x''$ i $k := 1$, w przeciwnym przypadku przechodzi do szerszego sąsiedztwa podstawiając $k := k + 1$.

VNS dla problemu *bin-packing* bazuje na dwóch **przekształceniach** rozwiązania x:

- **przeniesienie** - jeden element w x zostaje przeniesiony do innego pojemnika,
- **zamiana** - dwa elementy z różnych pojemników w x zamieniają się miejscami.

Rozważane są jedynie przekształcenia, które nie prowadzą do przekroczenia pojemności pojemników i nie dodają nowych pojemników. Z jednej strony jest to korzystne, gdyż nie pogarsza rozwiązania w sensie liczby użytych pojemników, z drugiej strony można uznać to za wadę, gdyż znacznie ogranicza swobodę przeszukiwania sąsiedztwa.

Korzystając z definicji przekształcenia definiowane są podstawowe elementy algorytmu. Losowe rozwiązanie x' z sąsiedztwa $N_k(x)$ wyznaczane jest przez wykonanie k losowych przekształceń na x. Minimum lokalne wyznaczane jest metodą najszybszego spadku. Przeglądane są wszystkie możliwe przekształcenia poprawiające dane rozwiązane (ich wyznaczanie można znacznie zoptymalizować, patrz [2]) i wykonywane jest to, które prowadzi do największej poprawy wartości funkcji celu. Ta operacja jest powtarzana do chwili, gdy nie ma już przekształcenia, które poprawiłoby rozwiązanie.

Korzystanie w algorytmie z trywialnej funkcji celu, tzn. liczby użytych pojemników, nie prowadzi do dobrych rezultatów. W typowych przypadkach potrzeba wiele przekształceń, aby zmniejszyć liczbę pojemników o jeden. Z tego powodu używana jest funkcja:

$$\max f(x) = \sum_{\alpha=1}^{m} [l(\alpha)]^2 , \qquad (1)$$

gdzie m jest liczbą pojemników użytych w rozwiązaniu x, a $l(\alpha)$ oznacza zajętość pojemnika α w x. Dzięki tej funkcji procedura minimalizacji lokalnej dąży do maksymalnego wypełnienia pojemników, a więc pośrednio do opróżniania pojemników mało wypełnionych.

Przy wyznaczaniu wartości przekształcenia (przeniesienia lub zamiany) nie jest konieczne wyliczanie nowej wartości funkcji celu. Wystarczy policzyć zmianę jej wartości Δf, żeby ocenić, które przekształcenie jest najkorzystniejsze. Zarówno w przypadku przeniesienia elementu z pojemnika α do pojemnika β, jak i zamiany elementów między pojemnikami α i β, zmiana wartości funkcji celu Δf zależy jedynie od zmiany $l(\alpha)$ i $l(\beta)$.

Maksymalna wielkość przeszukiwanego sąsiedztwa ustalona została na $k^{\max} = 20$. Powyżej tej wartości rozwiązania ulegają bardzo niewielkiej poprawie, a czas przetwarzania rośnie znacząco. Dodatkowo algorytm jest zatrzymywany, jeśli liczba pojemników w rozwiązaniu jest równa dolnemu ograniczeniu na liczbę pojemników, którego wartość jest obliczana z wykorzystaniem metod podanych przez Fekete i Schepers [1].

6. Testy algorytmów

Wszystkie algorytmy zostały zaprogramowane w Borland Delphi 5.0. Testy zostały wykonane na komputerze z procesorem Pentium II 400MHz pod systemem Windows NT 4.0.

Do testów wykorzystane zostały dane z OR Library (http://www.ms.ic.ac.uk/info.html) oraz dane, którymi posłużyli się Scholl, Klein i Jürgens testując inne algorytmy [7]. W OR Library dostępnych jest 8 zbiorów po 20 problemów każdy, o znanych rozwiązaniach optymalnych: problemy w zbiorach U120, U250, U500 i U1000 zawierają odpowiednio 120, 250, 500 i 1000 elementów o wielkościach równomiernie rozmieszczonych w przedziale [20, 100], które muszą być upakowane w pojemnikach o wielkości 150 każdy. Problemy w zbiorach T60, T120, T249 i T501 zawierają odpowiednio 60, 120, 249 i 501 elementów o wielkościach w zakresie [25, 50], które muszą być upakowane w pojemnikach o wielkości 50 każdy. Problemy w klasie T zostały wygenerowane w taki sposób, że istnieje rozwiązanie optymalne wypełniające całkowicie każdy pojemnik dokładnie trzema elementami.

Dane testowe używane przez Scholla, Kleina i Jürgensa [7] podzielone są na 3 zbiory, które w dalszej części nazywane są B1, B2 i B3. B1 zawiera 720 problemów, z których 704 zostało rozwiązanych optymalnie. B2 zawiera 480 problemów, a optymalne rozwiązania znane są dla 477 z nich. Problemy ze zbiorów B1 i B2 zawierają od 50 do 500 elementów, a wielkość elementów i pojemników jest zróżnicowana (dokładny opis znajduje się w [7]). Zbiór B3 składa się z 10 trudnych problemów. Każdy z nich zawiera 200 elementów o wielkościach z przedziału od 20000 do 35000, które muszą być upakowane w pojemnikach o wielkości 100000. Tylko dla 3 problemów z B3 znane są optymalne rozwiązania.

Wyniki obliczeń testowanych algorytmów porównywane są z wartościami najlepszych znanych ograniczeń dolnych na liczbę pojemników. Tylko w 26 problemach na 1370 problemów nie są to wartości optymalne. Kompletne wyniki przeprowadzonych testów znajdują się w [2]. W poniższym tekście przedstawione są ich skrót oraz główne wnioski.

W tablicy 1 znajdują się wyniki testów algorytmów. Dla MBS i MBS' podane są liczby rozwiązanych problemów w poszczególnych klasach, a dla pozostałych heurystyk podane są liczby problemów rozwiązanych ponad te, które zostały rozwiązane przez MBS'.

W klasach U i T heurystyki MBS i MBS' uzyskały podobne wyniki. Natomiast w zbiorze B1 znacznie lepsze rezultaty uzyskała MBS', podczas gdy w zbiorze B2 nieco lepsze rezultaty uzyskała MBS. Potwierdza to spodziewaną własność braku dominacji między MBS i MBS' z jednoczesną statystyczną przewagą MBS'.

Tablica 1

Liczba problemów rozwiązanych przez testowane algorytmy w poszczególnych zbiorach

Zbiór	Liczba problemów	MBS	MBS'	Relaxed MBS'	Perturbation MBS'	Sampling MBS'	VNS
U120	20	12	11	+8	+0	+8	+9
U250	20	10	12	+4	+1	+4	+7
U500	20	11	11	+5	+3	+6	+8
U1000	20	7	7	+6	+9	+12	+13
T60	20	0	0	+0	+20	+0	+0
T120	20	0	0	+0	+20	+0	+0
T249	20	0	0	+0	+20	+0	+0
T501	20	0	0	+0	+20	+0	+0
B1	720	251	629	+52	+8	+34	+65
B2	480	387	381	+44	+24	+83	+93
B3	10	0	0	+2	0	0	+2
Wszystkie	1370	678	1051	+121	+125	+147	+197

Pozostałe heurystyki uruchamiane są z rozwiązaniami początkowymi uzyskanymi z MBS'. Relaxed MBS', Sampling MBS' i VNS poprawiają znaczną liczbę rozwiązań w klasach U i B. Wśród nich wyraźnie dominuje VNS, który dla czterech problemów z klasy B1 znajduje rozwiązania lepsze niż najlepsze znane dotąd. Zaskakujące jest, że żaden z powyższych algorytmów nie rozwiązuje ani jednego problemu z klasy T, natomiast Perturbation MBS' rozwiązuje je wszystkie. Wobec powyższego skonstruowany został końcowy algorytm hybrydowy, wywołujący kolejno MBS', Perturbation MBS' i VNS.

Tablica 2 przedstawia rezultaty testów wszystkich prezentowanych heurystyk. Podane są liczby problemów rozwiązanych i nie rozwiązanych, bezwzględne i względne średnie i maksymalne odchylenia od ograniczeń dolnych oraz średni i maksymalny czas obliczeń.

Tablica 2

Wyniki testów algorytmów dla wszystkich danych testowych

Algorytm	Problemy rozwiązane	Problemy nie rozwiązane	Odchylenie bezwzględne		Odchylenie względne [%]		Czas [s]	
			Srednie	Maks.	Srednie	Maks.	Sredni	Maks.
MBS	678	692	1.02	9	1.42	16.67	0.05	8.99
MBS'	1051	319	0.38	9	0.61	14.29	0.02	4.57
Relaxed MBS'	1172	198	0.16	6	0.31	14.29	0.09	5.88
Perturbation MBS'	1176	194	0.22	5	0.38	14.29	0.08	4.57
Sampling MBS'	1198	172	0.14	2	0.24	5.56	0.10	16.63
VNS	1248	122	0.09	2	0.18	5.00	0.09	4.57
Pert. MBS' + VNS	1329	41	0.03	2	0.04	2.94	0.14	5.05

MBS' znacznie poprawia średnią jakość rozwiązań w stosunku do MBS, przy jednoczesnym około dwukrotnym spadku czasu obliczeń. Kolejne heurystyki poprawiają rozwiązania MBS' wprowadzając tylko niewielki wzrost czasu przetwarzania. Perturbation

MBS' + VNS wyraźnie dominuje nad wszystkimi pozostałymi heurystykami, uzyskując rozwiązania w zakresie 3% od optimum przy tylko nieznacznym wzroście czasu obliczeń.

Na podstawie problemów z klasy B w tablicy 3 algorytmy MBS' i Perturbation MBS' + VNS są porównane z innymi znanymi algorytmami rozwiązywania problemu *bin-packing*. Dane porównawcze zostały zaczerpnięte z badań opisanych w [7]. MBS' wyraźnie dominuje nad prostymi heurystykami. Perturbation MBS' + VNS osiąga znacznie lepsze rezultaty od *tabu-search* (DualTabu). Również porównanie z metodą podziału i oszacowań BISON wypada bardzo korzystnie dla VNS. W bardzo długim czasie (chociaż na nieco wolniejszym komputerze) BISON osiąga rezultaty porównywalne z Perturbation MBS' + VNS.

Tablica 3

Liczba problemów rozwiązanych przez różne znane algorytmy – porównanie (TL – limit czasu na 486 66MHz)

Zbiór	Liczba problemów	FFD	BFD	WFD	B2F	FFD-B2F	MBS'	DualTabu	BISON TL=50s	BISON TL=1000s	Pert. MBS' + VNS
B1	720	547	548	409	545	617	629	666	695	697	694
B2	480	236	236	192	292	319	381	466	472	473	474
B3	10	0	0	0	0	0	0	3	3	3	2

7. Podsumowanie

W referacie zaprezentowano szereg heurystyk do rozwiązywania problemu *bin-packing*, bazujących na heurystyce MBS oraz metaheurystyce VNS. Mimo iż heurystyki bazujące na MBS mają wysoką pesymistyczną złożoność obliczeniową, w testach okazują się bardzo wydajne. Uzyskany algorytm hybrydowy Perturbation MBS' + VNS okazał się bardzo wydajny w porównaniu z innymi metodami heurystycznymi i dokładnymi, uzyskując w krótkim czasie rozwiązania bardzo bliskie optymalnych.

LITERATURA

1. Fekete S. P., Schepers J.: New classes of lower bounds for bin packing problems. Lecture notes in computer science, vol. 1412, 1998, pp. 257–270.
2. Fleszar K., Hindi K. S.: New Heuristics for One-Dimensional Bin-Packing. Computers and Operations Research, Vol 29/7, 2002, pp. 821-839.
3. Garey M. R., Johnson D. S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman, 1979.

4. Gupta J. N. D., Ho J. C.: A new heuristic algorithm for the one-dimensional bin-packing problem. Production Planning & Control, vol. 10, no. 6, 1999, pp. 598–603.
5. Hansen P., Mladenović N.: An Introduction to Variable Neighbourhood Search. Rozdział w Voss S. (redaktor): Metaheuristics, Advances and Trends in Local Search Paradigms for Optimization, pp. 433–458. Kluwer, 1999.
6. Mladenović N., Hansen P.: Variable Neighbourhood Search. Computers & Operations Research, vol. 24, no. 11, 1997, pp. 1097–1100.
7. Scholl A., Klein R., Jürgens C.: BISON: A fast hybrid procedure for exactly solving the one-dimensional bin packing problem. Computers & Operations Research, vol. 24, no. 7, 1997, pp. 627–645.
8. Valério de Carvalho J. M.: Exact solution of bin-packing problems using column generation and branch-and-bound. Annals of Operations Research, vol. 86, 1999, pp. 629-659.

Recenzent: Dr hab. inż. Konrad Wala, Prof. AGH

Abstract

A bin-packing problem considered here can be stated as follows: given a number of items of known sizes, what is the minimum number of bins, each having the same capacity, necessary to pack all items.

Several new heuristics for solving the one-dimensional bin packing problem are presented. Some of these are based on the minimum bin slack (MBS) heuristic of Gupta and Ho. MBS is bin-focused. At each step, it finds a set of items that fits the bin capacity as much as possible by enumerating sets of items. New algorithms based on MBS include: MBS' (fixes one item before enumeration), Relaxed MBS' (accepts sets of items that leave a small slack in a bin), Perturbation MBS' (repeatedly updates MBS' solution by finding improved sets of items), Sampling MBS' (builds many solutions with different order of enumeration).

A different algorithm is an application of the variable neighbourhood search (VNS) metaheuristic of Mladenović and Hansen to the bin-packing problem. It improves a given solution by repeatedly "shaking" (randomly moving items between bins) and locally optimising the solution (moving items between bins to improve the solution).

The most effective algorithm turned out to be one composed of MBS', Perturbation MBS' and VNS, where the former algorithm provides an initial solution for the latter. When tested on 1370 benchmark problem instances from two sources, this last hybrid algorithm proved capable of achieving the optimal solution for 1329 instances, and could find for 4 instances solutions better than the best known. This is remarkable performance when set against other methods, both heuristic and optimum seeking.