Wtorki 16:50
Grupa I3
Kierunek Informatyka
Wydział Informatyki
Politechnika Poznańska

Algorytmy i struktury danych

Sprawozdanie z zadania w zespołach nr. 4
prowadząca: dr hab. inż. Małgorzata Sterna, prof PP

Programowanie dynamiczne

autorzy:
Piotr Więtczak nr indeksu 132339
Tomasz Chudziak nr indeksu 136691

5 czerwca 2018

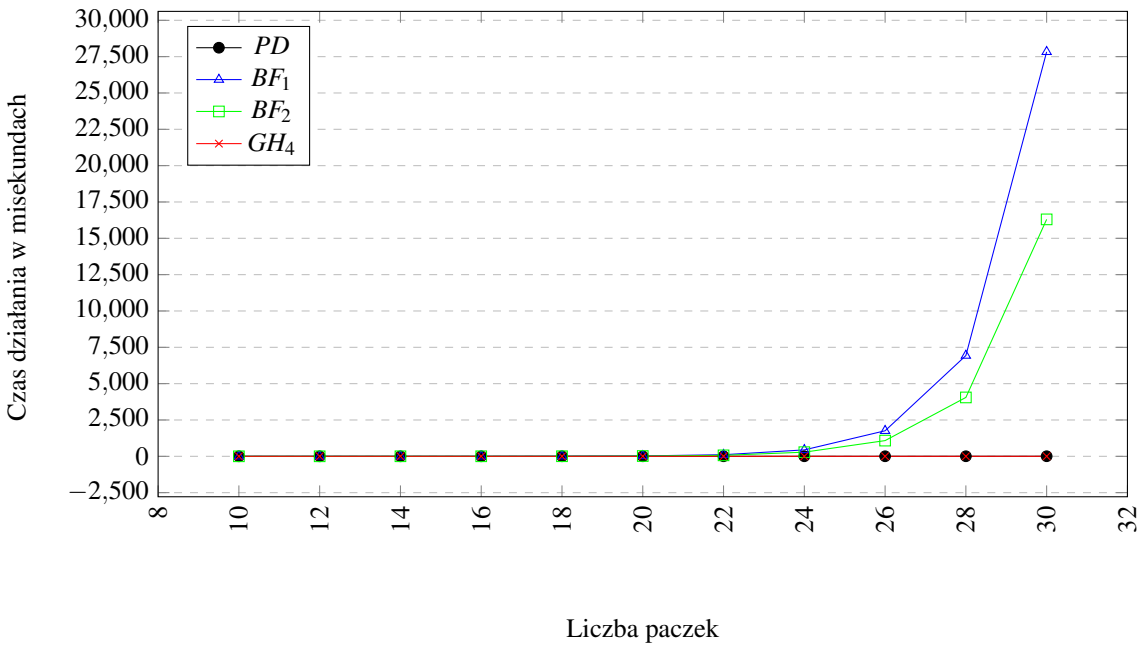
1 Implementacja

Do implementacji algorytmów wspomagających wybór paczek użyliśmy języka C++, a do pomiarów czasu klasy *std* :: *chrono* :: *hight_resolution_clock*, z biblioteki *chrono*. W implementacji zastosowano algorytmy korzystające z siedmiu różnych strategii wyboru paczek:

- PD - programowania dynamicznego,
- BF_1 - pełnego przeglądu,
- BF_2 - pełnego przeglądu z eliminacją rozwiązań niedopuszczalnych,
- GH_1 - heurystycznej z losowym wyborem paczek,
- GH_2 - heurystycznej z wyborem paczek o minimalnym ciężarze,
- GH_3 - heurystycznej z wyborem paczek o maksymalnej wartości,
- GH_4 - heurystycznej z wyborem paczek o maksymalnej zależności wartość/ciężar.

2 Zależność czasu obliczeń t od liczby paczek n dla PD, BF_1, BF_2, GH_4 dla $b = 50\% \sum s(a_i)$.

Wykres przedstawiający czasy działania algorytmów dla $b = 0.50\%$, skala liniowa



Wykres przedstawiający czasy działania algorytmów dla $b = 0.50\%$, skala logarytmiczna

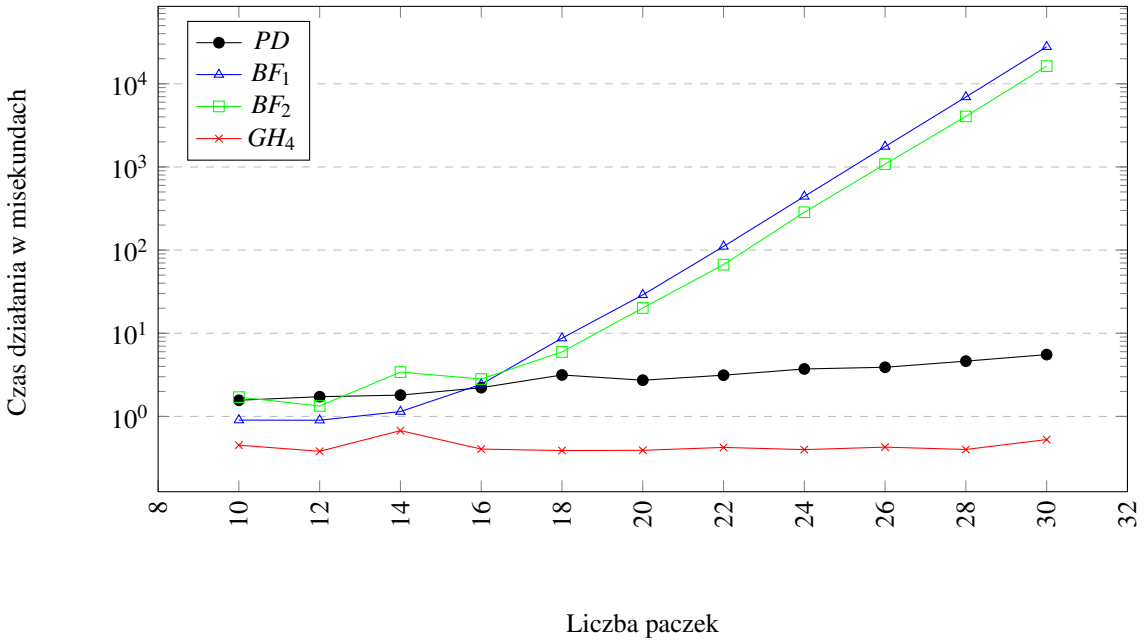


Tabela przedstawiająca czasy działania algorytmów dla $b = 0.50\%$

Liczba paczek	Czas trawania algorytmu $[ms]$			
	PD	BF_1	BF_2	GH_4
10	1.564	0.903	1.706	0.450
12	1.726	0.899	1.335	0.380
14	1.803	1.143	3.413	0.675
16	2.215	2.435	2.816	0.404
18	3.151	8.730	5.965	0.388
20	2.725	28.947	20.130	0.391
22	3.131	110.618	66.851	0.423
24	3.722	441.764	285.001	0.398
26	3.900	1757.558	1081.185	0.427
28	4.621	6933.151	4051.269	0.400
30	5.540	27831.445	16303.147	0.526

Wersja decyzyjna problemu plecakowego należy do klasy problemów NP-zupełnych, oznacza to że optymalizacyjny problem plecakowy również będzie należał do NP-trudnych, czyli będzie co najmniej tak trudny jak jego wersja dyskretna. Pseudowielomianowy algorytm problemu plecakowego klasyfikujemy jako problem NP-zupełny, a

problem nierozwiązywalny w czasie pseudowielomianowym nazywamy silnie NP-zupełny, dlatego klasa problemów trudnych nie jest jednorodna. Zastosowanie metody programowania dynamicznego jest możliwe dla problemów optymalizacyjnych, czyli takich których rozwiązanie polega na znalezieniu największej, albo najmniejszej wartość pewnego parametru problemu.

Metody BF_1 i BF_2 zwracają optymalny wynik, jednak przez zastosowanie pełnego przeglądu złożoność metody BF_1 wynosi $O(2^n)$, a metoda BF_2, eliminująca rozwiązania niedopuszczalne, działa nie wolniej od metody BF_1

Złożoność obliczeniowa algorytmu wykorzystującego programowanie dynamiczne, wynosi $O(n \cdot b)$, gdzie n - liczba dostępnych paczek, b - maksymalna ładowność plecaka. Metoda ta, korzysta z możliwości podziału problemu na zależne od siebie podproblemy, oraz zwraca optymalny wynik, jednak działa szybciej od metod BF. Nie jest jednak zależna tylko od ilości paczek, ale również od pojemności plecaka.

Strategie heurystyczne nie gwarantują otrzymania wyniku optymalnego, jednak działają znacznie szybciej od poprzednich metod, złożoność dla metody GH_1 wynosi $O(n)$, a dla pozostałych metod heurystycznych $O(2n)$, przez zastosowanie sortowania przez zliczanie.

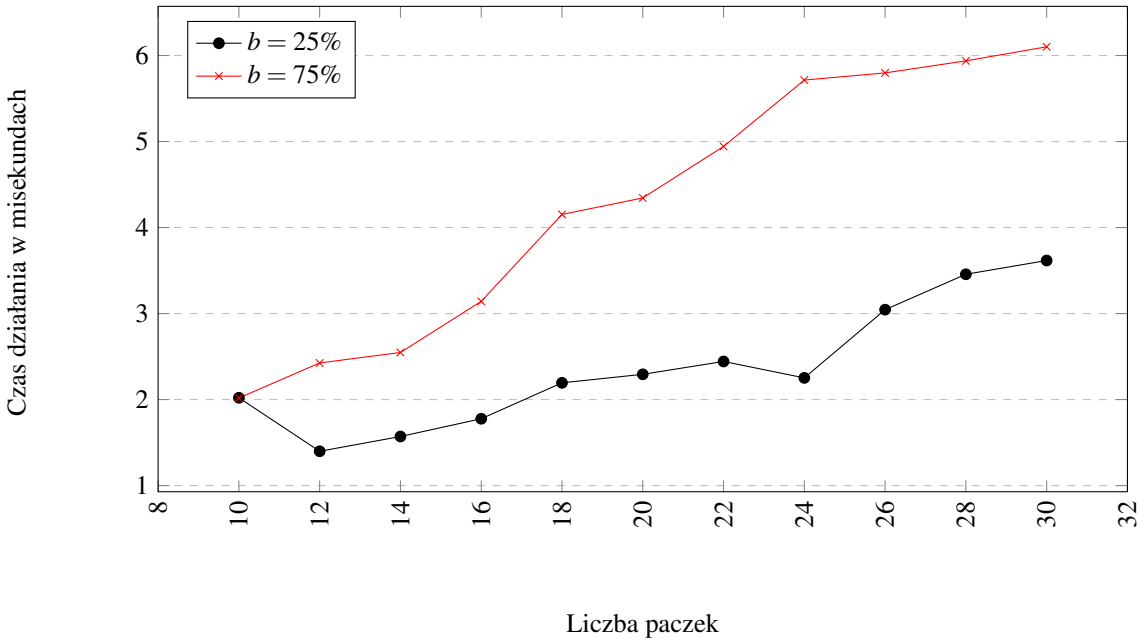
Wybór strategii rozwiązywania problemu zależy od tego czy chcemy poznać optymalny wynik, od wielkości plecaka, oraz od ilości dostępnego czasu. Jeżeli nie zależy nam na dokładnym wyniku, tylko na krótkim czasie poszukiwania, na pewno warto wybrać strategię heurystyczną, szczególnie przy dużej ilości paczek. Chcąc poznać optymalny wynik trzeba zdecydować między pełnym przeglądem, a programowaniem dynamicznym. Przy stosunkowo niskiej ilości paczek do rozmiaru plecaka metoda korzystająca z pełnego przeglądu może działać szybciej, w innych przypadkach poradzi sobie gorzej. Programowanie dynamiczne wymaga także dodatkowej pamięci na tablice przechowującą rozwiązania podproblemów.

3 Zależność czasu obliczeń t od liczby paczek n dla metody PD, BF_1, BF_2, GH_4 dla $b = 25\% \sum s(a_i)$ i $b = 75\% \sum s(a_i)$.

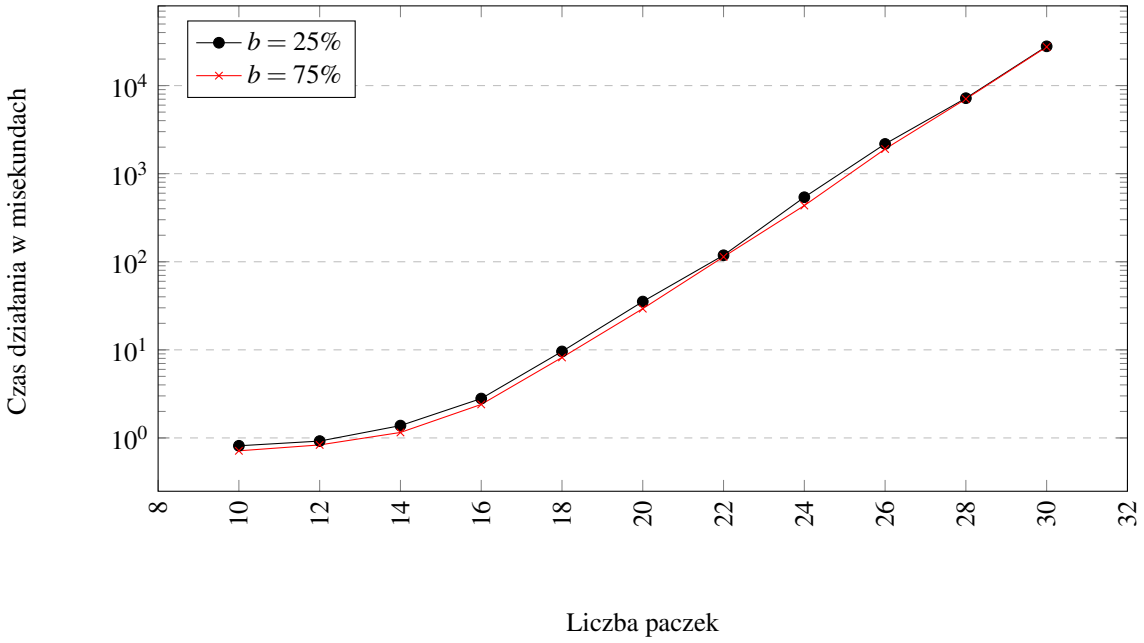
Tabela przedstawiający czasy działania algorytmów dla różnych b

Liczba	Czas trwania algorytmów dla $b = 25\%$ $[ms]$				Czas trwania algorytmów dla $b = 75\%$ $[ms]$			
paczek	PD	BF_1	BF_2	GH_4	PD	BF_1	BF_2	GH_4
10	2.020	0.814	1.233	0.349	2.018	0.714	1.857	0.430
12	1.398	0.921	1.454	0.429	2.425	0.832	1.604	0.407
14	1.570	1.380	1.510	0.433	2.547	1.153	2.252	0.478
16	1.777	2.799	1.817	0.485	3.141	2.400	3.321	0.434
18	2.194	9.581	2.395	0.556	4.152	8.192	9.354	0.480
20	2.293	35.322	4.151	0.528	4.343	29.341	30.594	0.460
22	2.443	118.491	6.802	0.489	4.942	113.809	120.644	0.481
24	2.252	540.134	20.763	0.497	5.714	434.529	468.442	0.467
26	3.045	2175.095	71.987	0.507	5.797	1905.247	1905.068	0.417
28	3.456	7193.456	114.979	0.435	5.937	7045.246	7187.442	0.479
30	3.616	27904.347	318.769	0.411	6.101	27492.279	28313.462	0.404

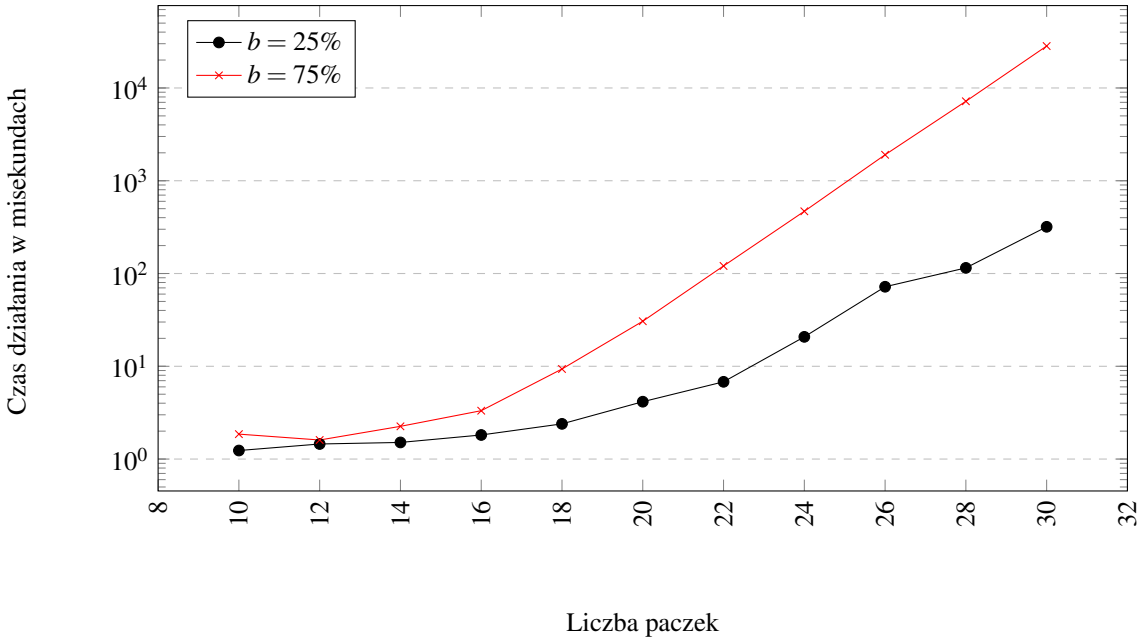
Wykres przedstawiający czasy działania algorytmu PD dla różnych b, skala liniowa



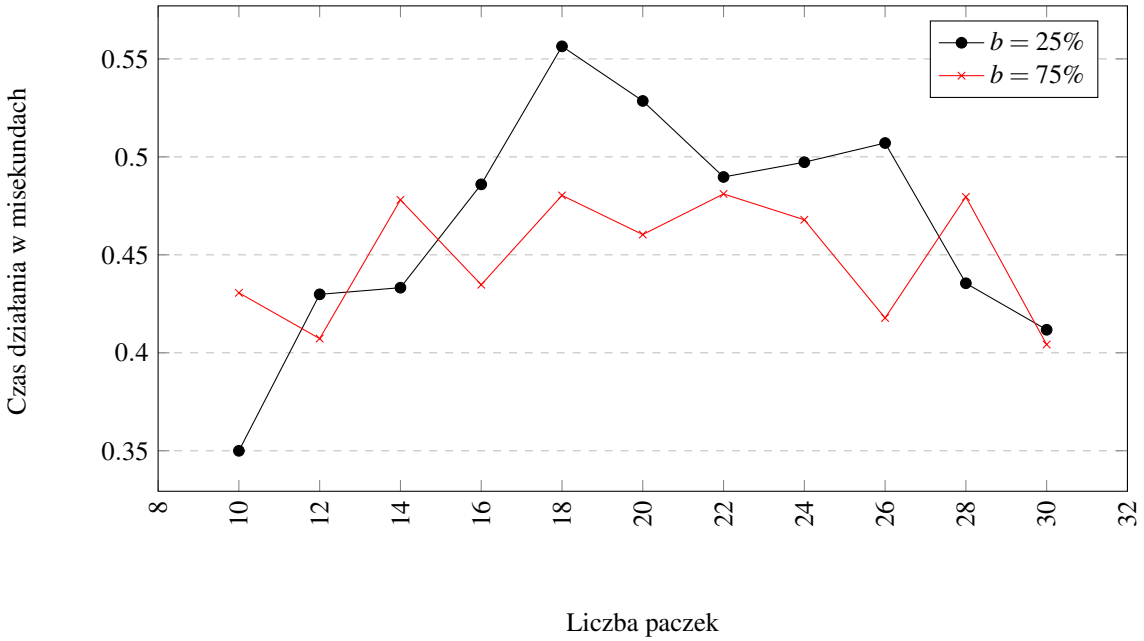
Wykres przedstawiający czasy działania algorytmu BF_1 dla różnych b, skala logarytmiczna.



Wykres przedstawiający czasy działania algorytmu BF_2 dla różnych b, skala logarytmiczna.



Wykres przedstawiający czasy działania algorytmu GH_4 dla różnych b, skala liniowa.



Czas działania wszystkich algorytmów rośnie wraz z ilością paczek, jednak jest to najmniej widoczne w przypadku GH_4. Rozmiar plecaka nie ma wpływu na metody BF_1 i GH_4. Natomiast w algorytmie BF_2 mniejszy rozmiar plecaka pozwala na odrzucenie większej ilości rozwiązań niedopuszczalnych co skraca czas jego działania. Przy metodzie wykorzystującej programowanie dynamiczne widać że pojemność plecaka ma duży wpływ na czas działania, przy wzroście ładowności wzrasta długość czasu działania, co jest zgodne ze złożonością obliczeniową

metody $O(n \cdot b)$, b - pojemność plecaka.

4 Obliczanie średniego błędu popełnionego przez poszczególne heurystyki.

Algorytm heurystyczny jest to schemat postępowania, który nie gwarantuje odnalezienia najlepszego rozwiązania, lecz dopuszczalnie dobrego. Oznacza to, że krótszy czas poświęcony na uzyskanie go, pozwala zrekompensować korzyści płynące z najlepszego. Algorytm zachłanny charakteryzuje się tym, że w każdym kroku decyzje podejmuje zachłannie. Innymi słowy, stara się on podjąć jak najlepszą decyzję na daną chwilę, nie dokonując oceny czy będzie to korzystne w kolejnych krokach. Algorytmy listowe wpierw porządkują dane wg jakiegoś kryterium (priorytet, wartość). Później rozpatruje je wg ustalonej kolejności.

W naszym badaniu testowaliśmy cztery heurystyki różniące się regułami wyboru paczek: losową (GH_1), min{s(a_i)} (GH_2), max{w(a_i)} (GH_3) i max{w(a_i)/s(a_i)} (GH_4). Sprawdzaliśmy błąd popełniany przez poszczególne heurystyki dla różnych ładowności pojazdu.

Tabela prezentująca błędy popełniane przez poszczególne heurystyki, dla $b = 25\%$ $[\%]$

Liczba paczek	GH_1	GH_2	GH_3	GH_4
100	29.517	17.292	10.757	17.737
120	55.085	14.799	23.620	5.293
140	33.744	22.452	30.514	0.660
160	65.074	19.323	10.267	0.568
180	56.951	23.347	30.512	0.000
200	56.475	15.871	33.060	3.666
220	30.272	20.988	18.109	4.806
240	46.826	20.291	20.235	0.000
260	52.659	19.769	18.362	1.450
280	54.999	22.358	23.737	0.000
300	40.436	22.810	17.967	1.200
średnia	52.204	21.930	23.717	3.538

Tabela prezentująca błędy popełniane przez poszczególne heurystyki, dla $b = 50\%$ $[\%]$

Liczba paczek	GH_1	GH_2	GH_3	GH_4
100	67.716	16.479	17.978	0.000
120	42.943	24.659	16.232	0.928
140	18.229	18.180	12.734	0.279
160	53.454	20.226	10.377	9.068
180	51.352	13.265	5.277	4.216
200	41.081	13.335	5.313	0.865
220	36.206	14.890	3.998	3.678
240	29.008	15.524	6.085	5.808
260	49.571	12.330	7.905	1.467
280	24.484	13.115	2.275	0.000
300	41.770	17.116	11.775	2.809
średnia	45.581	17.912	9.995	2.912

Tabela prezentująca błędy popełniane przez poszczególne heurystyki, dla $b = 75\%$ $[\%]$

Liczba paczek	GH_1	GH_2	GH_3	GH_4
100	47.273	0.828	1.407	2.236
120	55.092	0.637	1.081	1.719
140	16.002	0.583	2.465	3.048
160	22.028	10.494	0.000	0.000
180	9.766	6.336	1.827	2.259
200	16.053	7.627	0.925	1.291
220	28.255	6.443	0.519	0.563
240	12.068	8.440	2.403	3.051
260	17.421	18.183	7.957	0.440
280	16.721	5.510	0.137	0.937
300	26.328	10.172	1.116	0.433
średnia	26.699	7.525	1.984	1.598

Tabela podsumowująca wyniki

śr. błąd	GH_1	GH_2	GH_3	GH_4
$b = 25\%$	52.204	21.931	23.717	3.538
$b = 50\%$	45.581	17.912	9.995	2.912
$b = 75\%$	26.699	7.525	1.984	1.598
średnia	46.802	15.789	11.899	2.703

Wraz ze wzrostem ładowności pojazdu algorytmy radzą sobie coraz lepiej. Jest to spowodowane tym, że mniejsza ilość paczek nie zostanie użyta. Dla heurystyk ich wydajność niewydane się być skorelowana z ilością paczek, z którą ma do czynienia.

Jak widać w tabeli powyżej, jakość rozwiązanie zależy od użytej metody. Spośród badanych metod najmniej efektywna okazała się GH_1, następnie GH_2. Najlepiej wypadła GH_4 a po niej GH_3. Wydajność GH_1 zależy tylko i wyłącznie od przypadku. Skuteczność GH_2 i GH_3 jest zbliżona, pierwsza sortuje wg rozmiaru rosnąco, natomiast drugi wg rozmiaru malejąco. GH_2 będzie radził sobie gorzej dla dużej ilości małych paczek o niskiej wartości, a GH_3 dla stosunkowo dużej ilości dużych paczek o wysokiej wartości. GH_4 bierze pod uwagę stosunek wartości do rozmiaru paczki, co w konsekwencji daje wynik najbardziej zbliżony do optymalnego.

Metody zachłanne można wykorzystywać nie tylko do rozwiązywania problemu plecakowego (np. algorytm Kruskala służący do wyznaczania minimalnego drzewa rozpinającego dla grafu nie skierowanego).

Spis treści