WOJSKOWA AKADEMIA TECHNICZNA

im. Jarosława Dąbrowskiego

WYDZIAŁ CYBERNETYKI

Techniki algorytmiczne

Sprawozdanie projektu – Problem pakowania

inż. Bartłomiej Pawelec

inż. Maciej Talecki

inż. Hubert Makowski

Grupa: WCY24KX1S4

Prowadzący: mgr inż. Stąpor Piotr

Spis treści

1. Teoretyczny opis problemu

Dany jest zbiór przedmiotów $I = \{1,2, \dots, n\}$ gdzie każdy przedmiot i ma rozmiar s_i taki, że

$0 < s_i \leq 1$ lub ogólnie $s_i \leq C$ dla pojemności C

oraz mamy nieskończoną liczbę pojemników o stałej pojemności 1. Celem jest rozdzielenie przedmiotów do jak najmniejszej liczby pojemników w taki sposób, aby suma rozmiarów przedmiotów w każdym pojemniku nie przekraczała 1 (lub C w wersji ogólnej).

Można sformalizować problem jako program całkowitoliczbowy (Integer Linear Programming – ILP):

- Niech x_{ij} będzie zmienną binarną przyjmującą wartość 1, jeśli przedmiot i zostaje umieszczony w pojemniku j, oraz 0 w przeciwnym przypadku.
- Niech y_j będzie zmienną decyzyjną równą 1, jeśli pojemnik j jest użyty, w przeciwnym przypadku 0.

Takie podejście minimalizuje liczbę użytych koszy przy zachowaniu ograniczenia, że w każdym z nich suma rozmiarów przedmiotów nie przekracza pojemności.

$$x_{ij} \in \{0,1\} \; y_j \in \{0,1\}$$

$$min. \sum_{j=1}^{n} y_j$$

$$s.t. \sum_{j=1}^{n} x_{ij} = 1, \; \forall i = 1, \dots, n$$

$$\sum_{j=1}^{n} s_i x_{ij} \leq y_j, \; \forall j = 1, \dots, n$$

2. Opis algorytmu dokładnego (pełne przeszukanie)

Algorytm dokonuje rekurencyjnego przeszukiwania z przycinaniem gałęzi:

```
Funkcja ExponentialBinPacking(items[0..n-1], C)
├─ bestBinCount ← n
├─ opCount ← 0
├─ Procedura SearchExp(idx, bins)
│ ├─ Jeśli idx == n to
│ │ ├─ Jeśli bins.size < bestBinCount to bestBinCount ← bins.size
│ │ └─ Zakończ SearchExp
│ ├─ w ← items[idx]
│ ├─ Dla i od 0 do bins.size-1 wykonuj
│ │ ├─ opCount ← opCount + 1
│ │ ├─ Jeśli bins[i] + w ≤ C to
│ │ │ ├─ bins[i] ← bins[i] + w
│ │ │ ├─ Call SearchExp(idx+1, bins)
│ │ │ ├─ bins[i] ← bins[i] - w
│ │ │ └─ Przerwij pętlę
│ ├─ bins.add(w)
│ ├─ opCount ← opCount + 1
│ ├─ Call SearchExp(idx+1, bins)
│ └─ bins.removeLast()
├─ Koniec procedury SearchExp
├─ Call SearchExp(0, empty list)
└─ Zwróć (bestBinCount, opCount)
```

3. Opis algorytmu heurystycznego First Fit

1. Przetwarzanie kolejności wejściowej bez sortowania.
2. Dla każdego przedmiotu sprawdzamy kolejno pojemniki i wstawiamy do pierwszego z wystarczającą wolną przestrzenią.
3. Jeśli żaden nie pasuje – otwieramy nowy pojemnik.

4. Implementacja

4a. Algorytm dokładny ExactBinPacking() pełne przeszukanie.
4b. Algorytm heurystyczny PackFirstFit() z prostym przydziałem.
4c. Generator danych:

• GenWorstCase(n): naprzemiennie duże/małe wagi (worst-case FF).
• GenRandomCase(n): losowe wagi 1..C.

4d. Pomiar złożoności:

• Czas: Stopwatch from System.Diagnostics.
• Liczba koszy i licznik operacji sumowania.

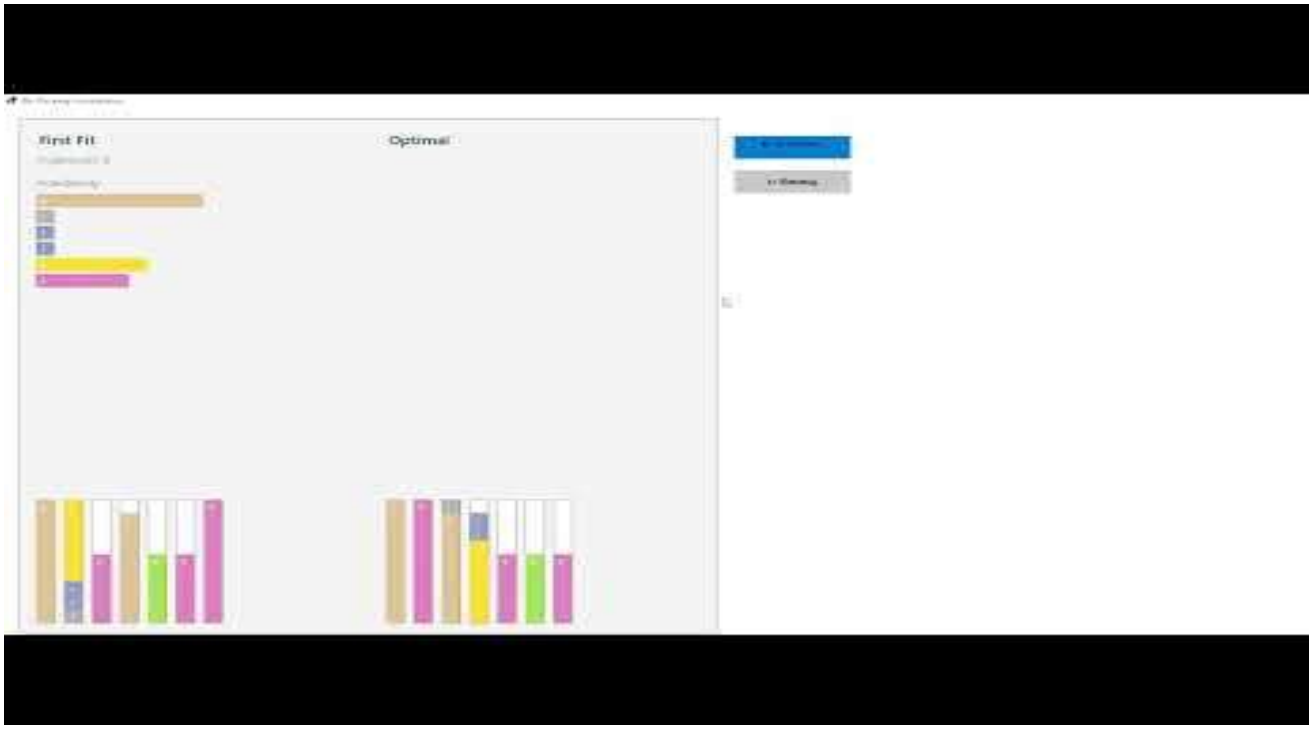
Projekt składa się z trzech elementów pod wspólną przestrzenią nazw:

1. **FirstFitPacker** – statyczna klasa, która implementuje prosty algorytm First–Fit: dla każdego przychodzącego przedmiotu (Item) próbuje go wstawić do pierwszego koszyka o dostępnej pojemności, a gdy żaden nie pasuje – tworzy nowy koszyk.
2. **ExactBinPacking** – wykładnicze rozwiązanie problemu bin-packing, który przeszukuje wszystkie możliwe przydziały przedmiotów, zwraca optymalną liczbę koszy oraz licznik operacji dodawania.
3. **ComplexityTester** – klasa do generowania danych testowych (pesymistycznych i losowych), uruchamiania zarówno First–Fit (FirstFitPacker), jak i (opcjonalnie) ExactBinPacking, mierzenia czasu wykonania, czasu CPU, zużycia pamięci (GC) i liczby operacji, a wyniki zapisuje do plików CSV.

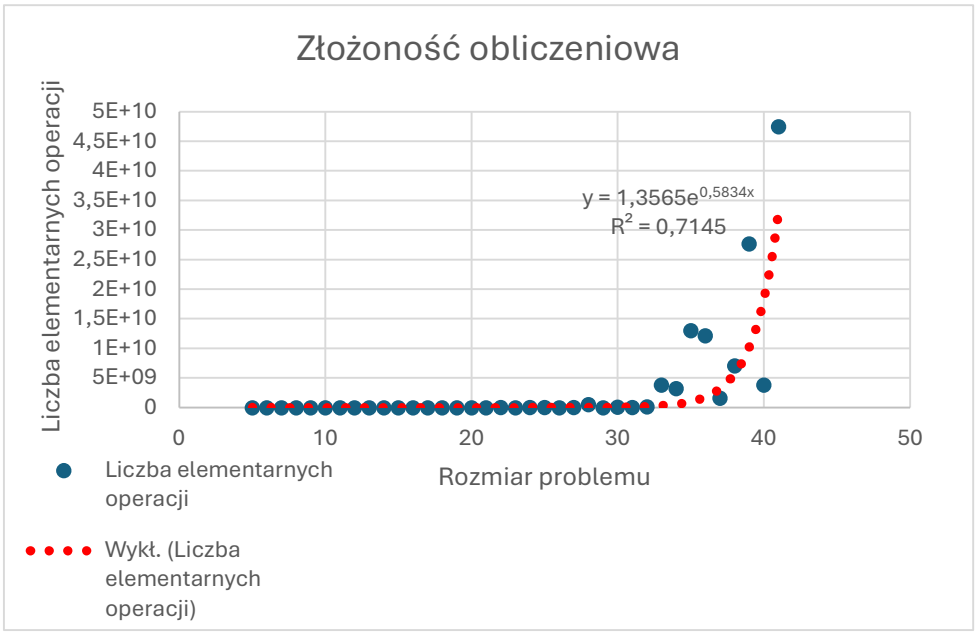
5. Teoretyczne oszacowania

Algorytm	T(n) pesym. / oczek.	Dokładność
First Fit	$O(n^2)$ / $\Theta(n^2)$	$\leq 7/4 \cdot OPT+1$
Optimal	$O(2^n)$ / $\Theta(2^n)$	1

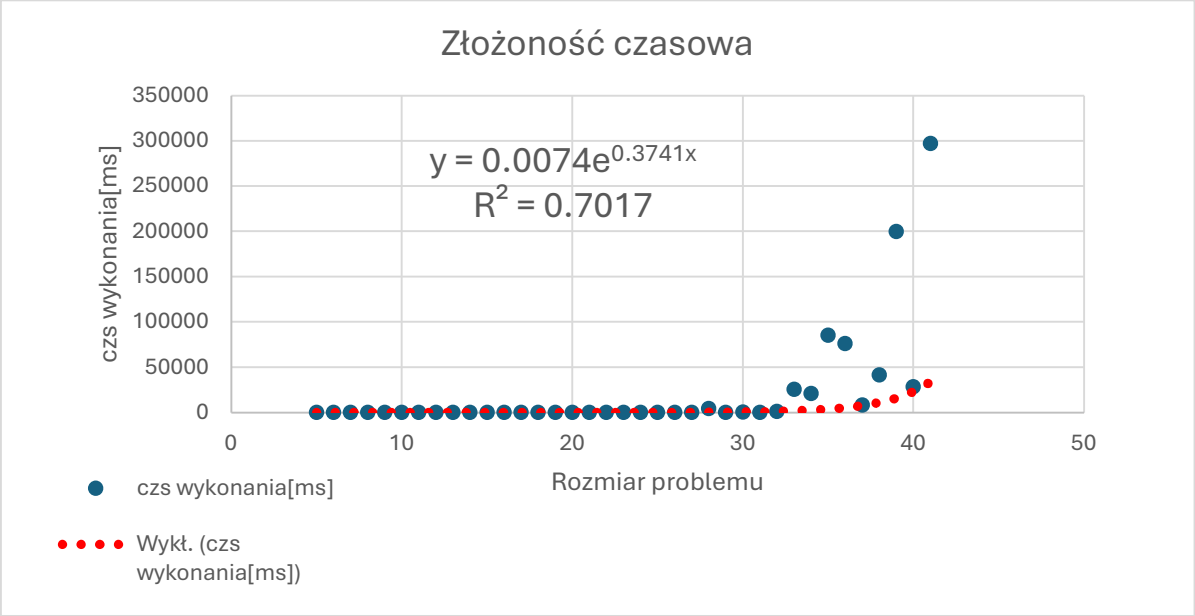
6. Wizualizacja rozwiązania problemu



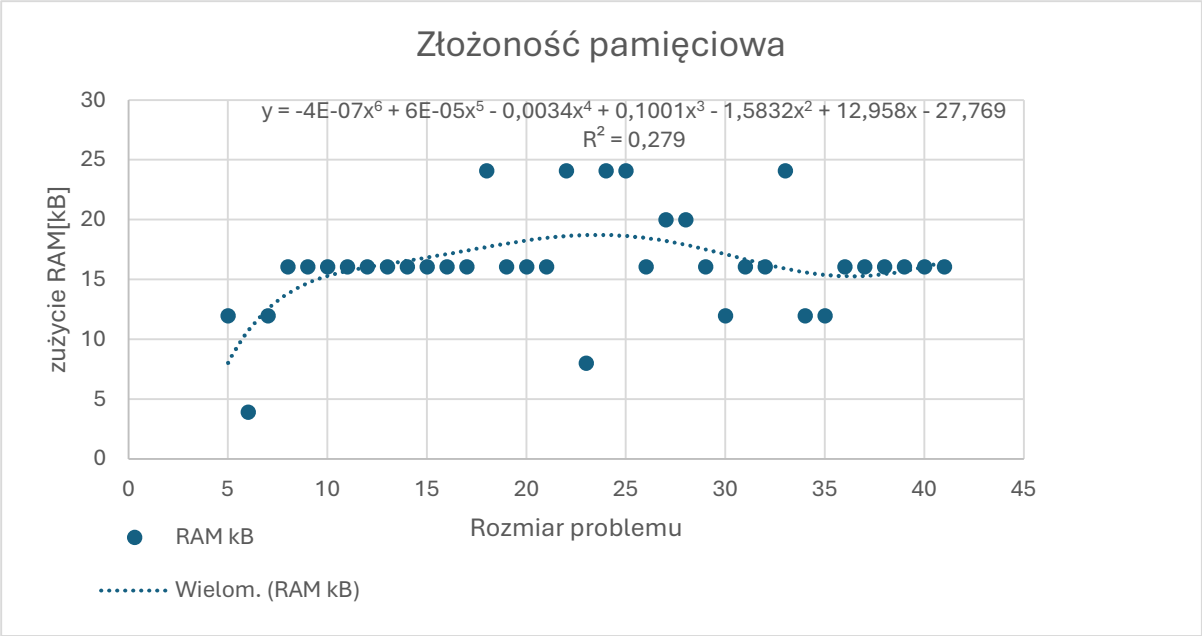
7. Wyniki eksperymentalne



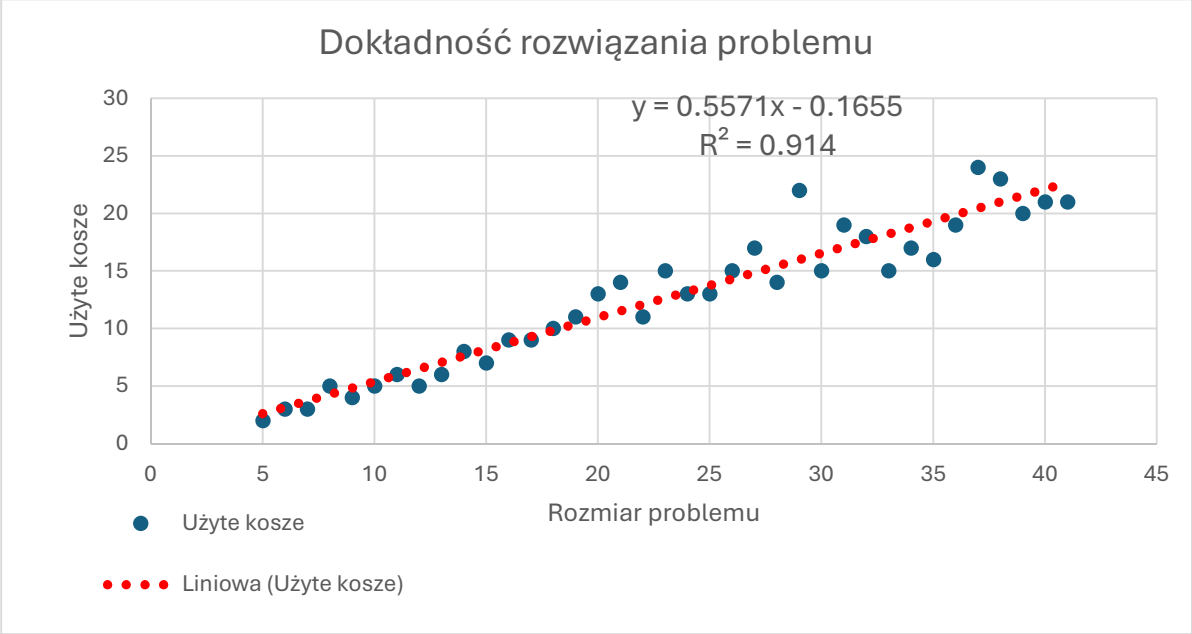
Rys. 1. Złożoność obliczeniowa – algorytm optymalny przypadek średni



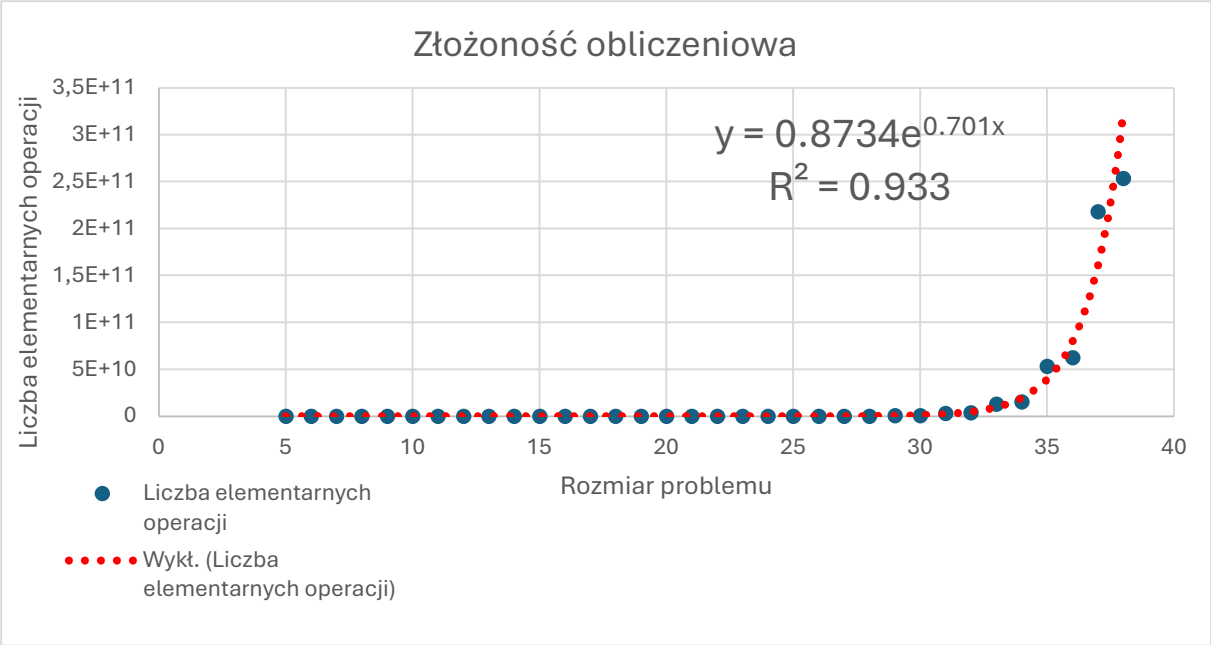
Rys. 2. Czas wykonania algorytmu względem problemu– algorytm optymalny przypadek średni



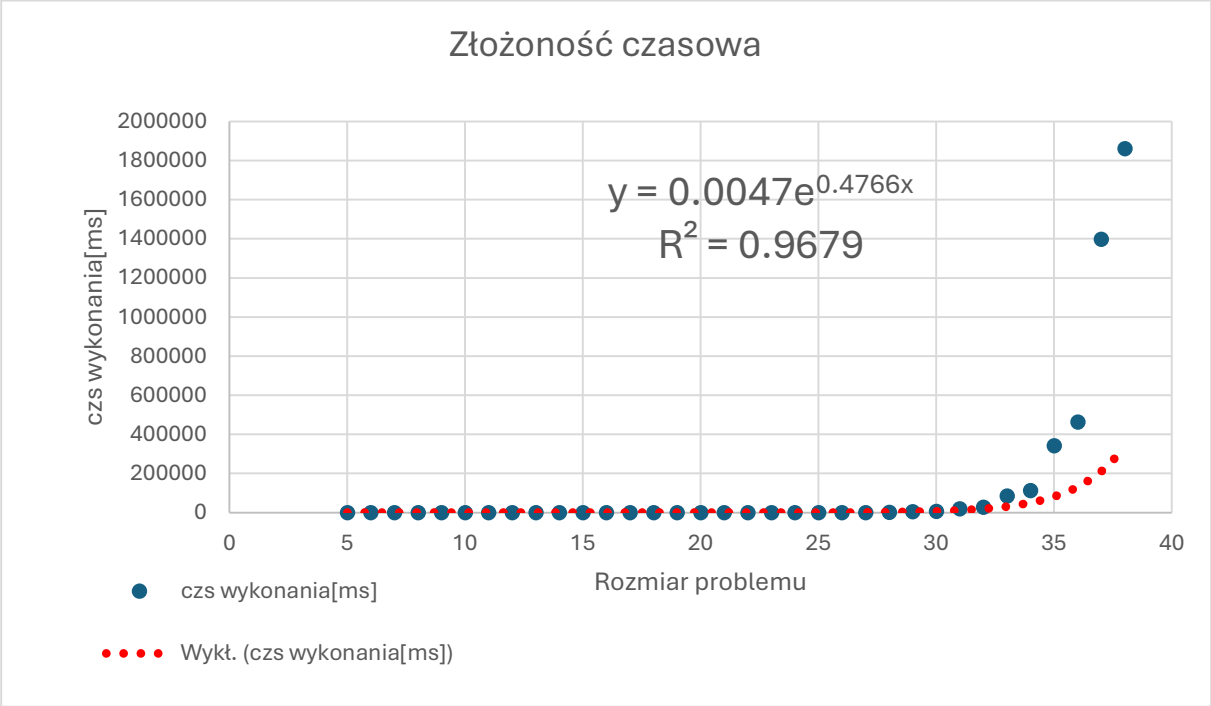
Rys. 3. Złożoność pamięciowa– algorytm optymalny przypadek średni



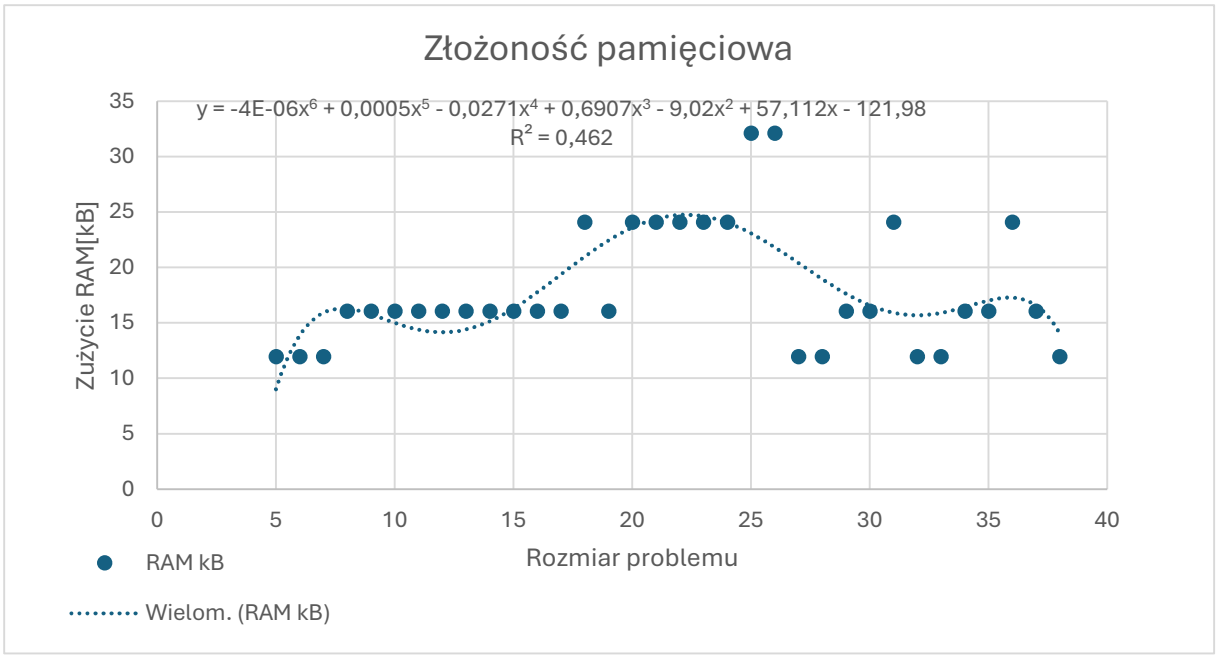
Rys. 4. Liczba użytych koszy do rozwiązania problemu – algorytm optymalny przypadek średni



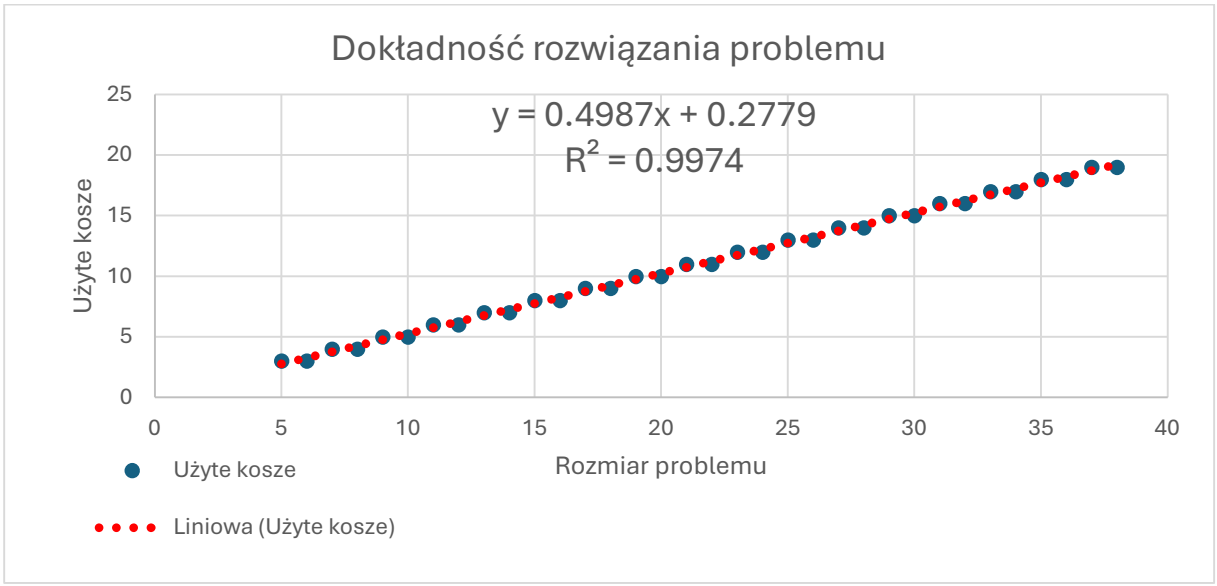
Rys. 5. Złożoność obliczeniowa– algorytm optymalny przypadek pesymistyczny



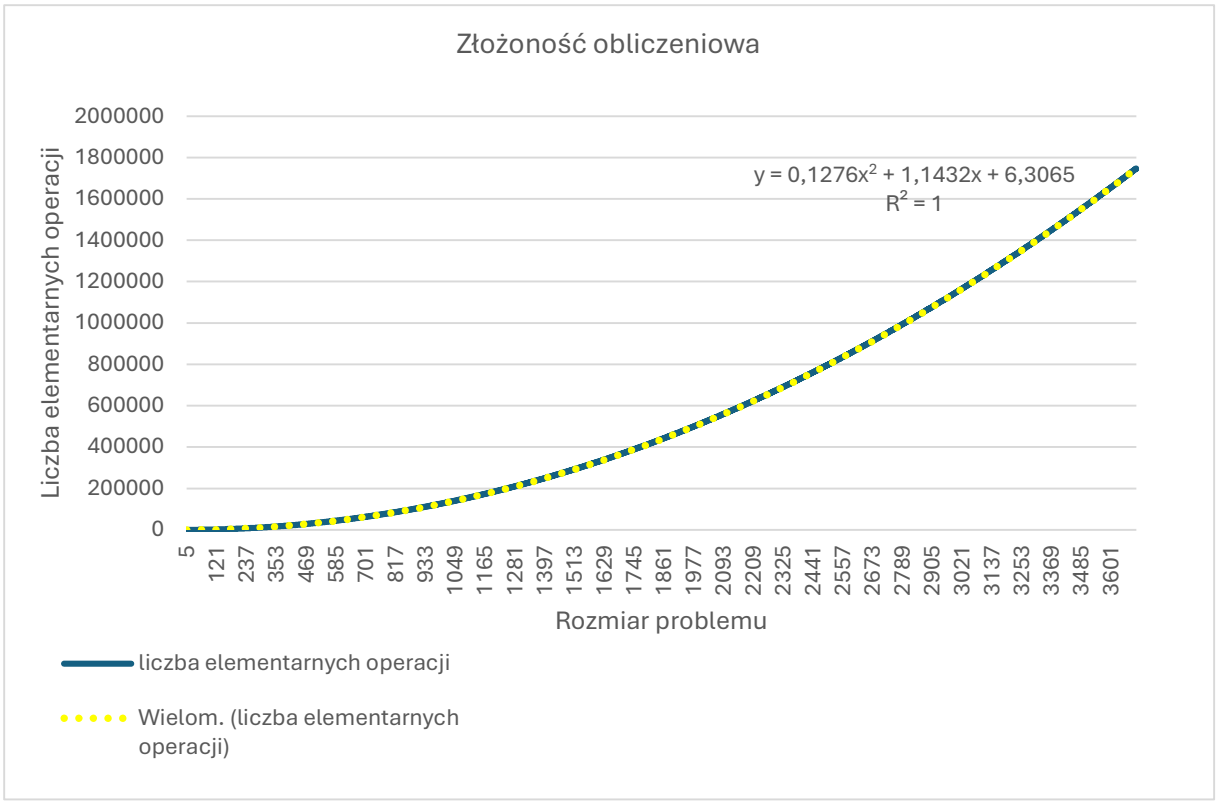
Rys. 6. Złożoność czasowa– algorytm optymalny przypadek pesymistyczny



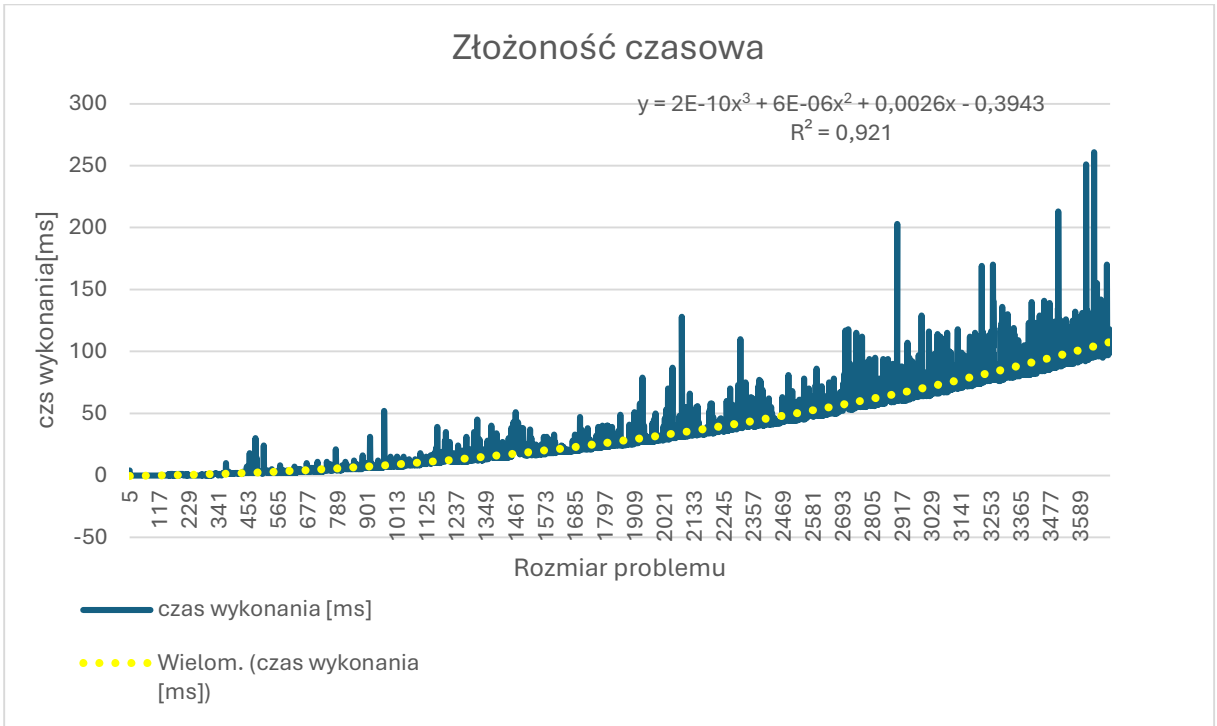
Rys. 7. Złożoność pamięciowa– algorytm optymalny przypadek pesymistyczny



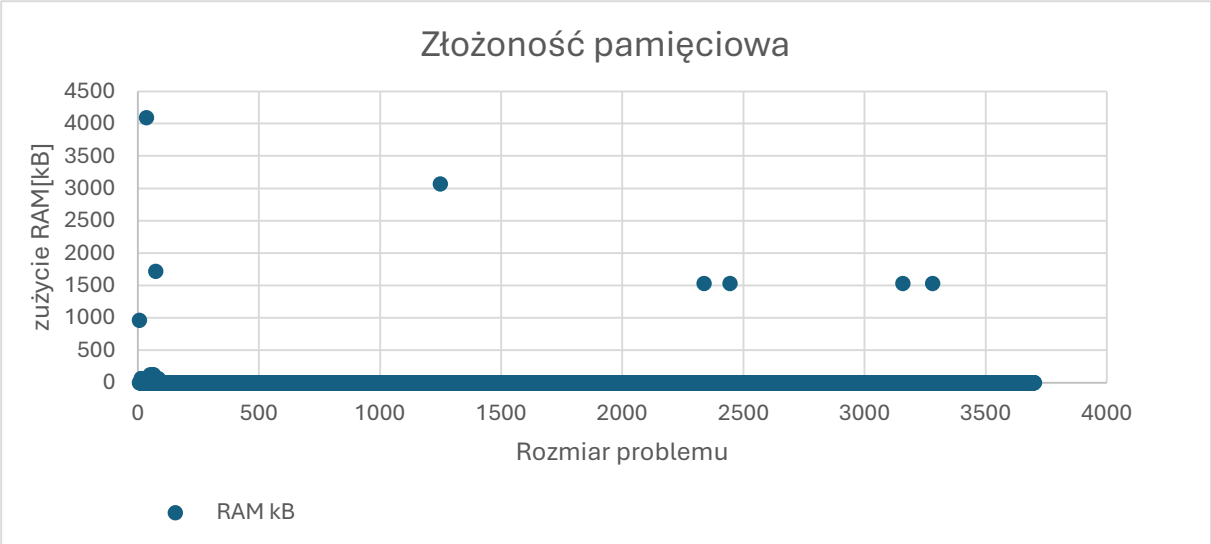
Rys. 8. Liczba użytych koszy do rozwiązania problemu – algorytm optymalny przypadek pesymistyczny



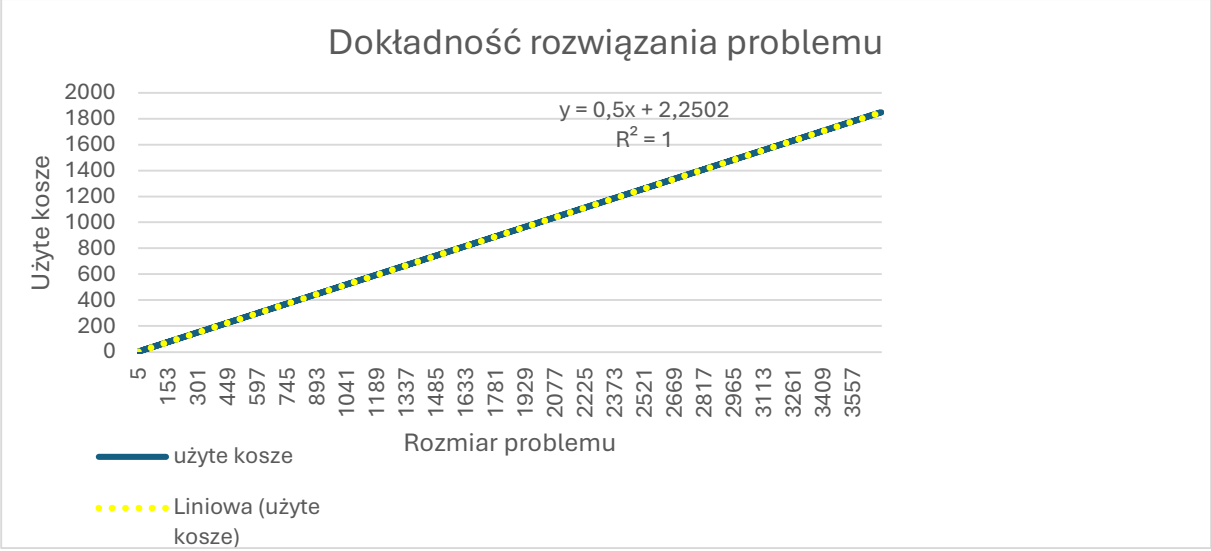
Rys. 9. Liczba użytych koszy do rozwiązania problemu – algorytm first fit przypadek pesymistyczny



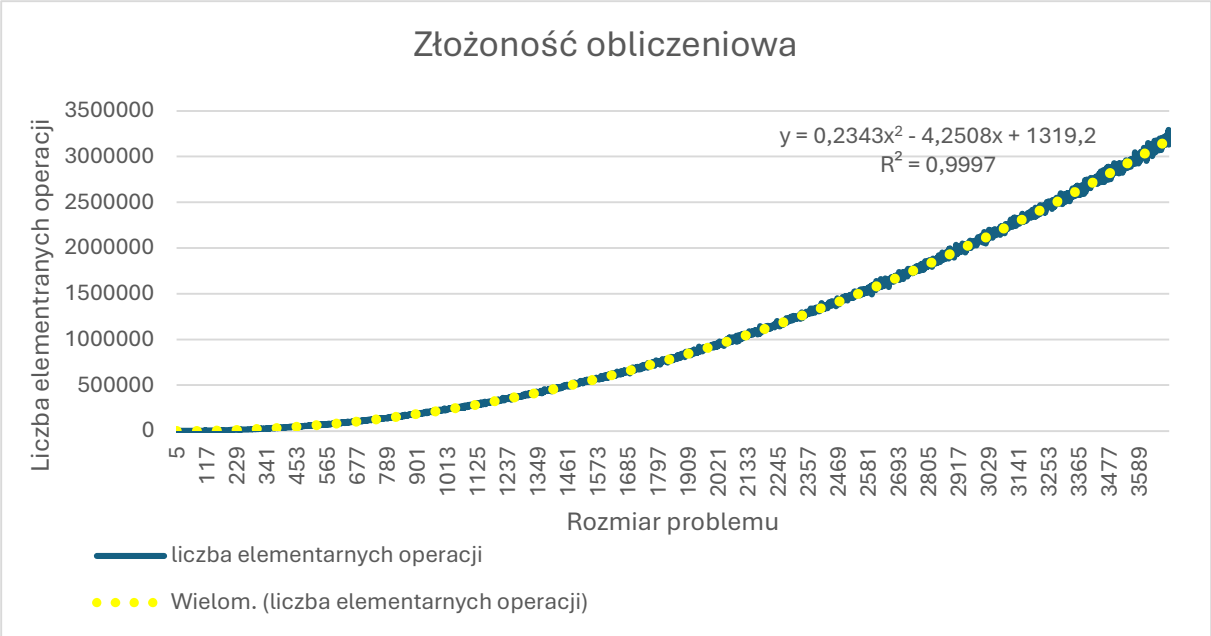
Rys. 10. Czas wykonania algorytmu względem problemu– algorytm first fit przypadek pesymistyczny



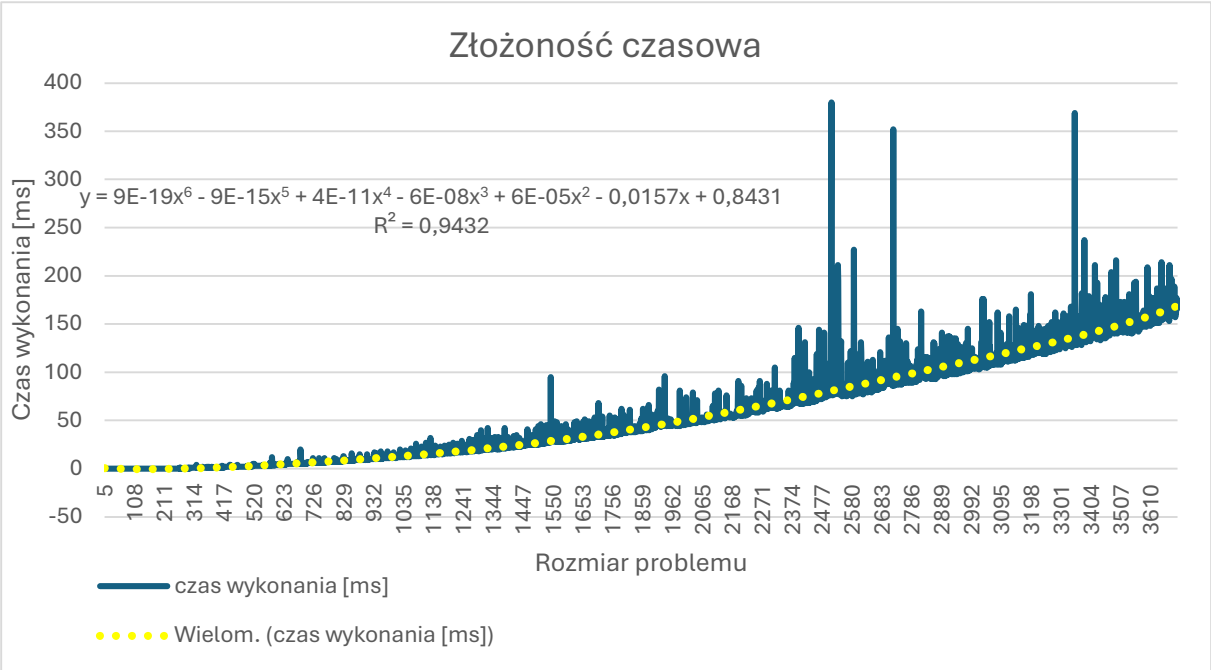
Rys. 11. Złożoność pamięciowa– algorytm first fit przypadek pesymistyczny Liczba użytych koszy do rozwiązania problemu – algorytm optymalny przypadek średni



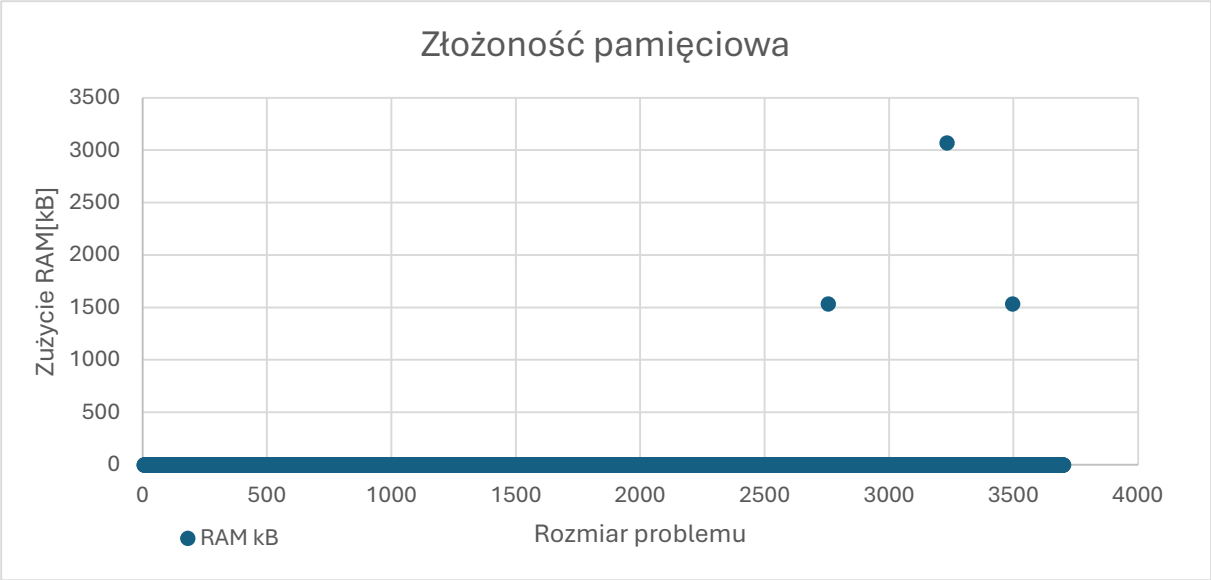
Rys. 12. Czas wykonania algorytmu względem problemu algorytm first fit przypadek pesymistyczny



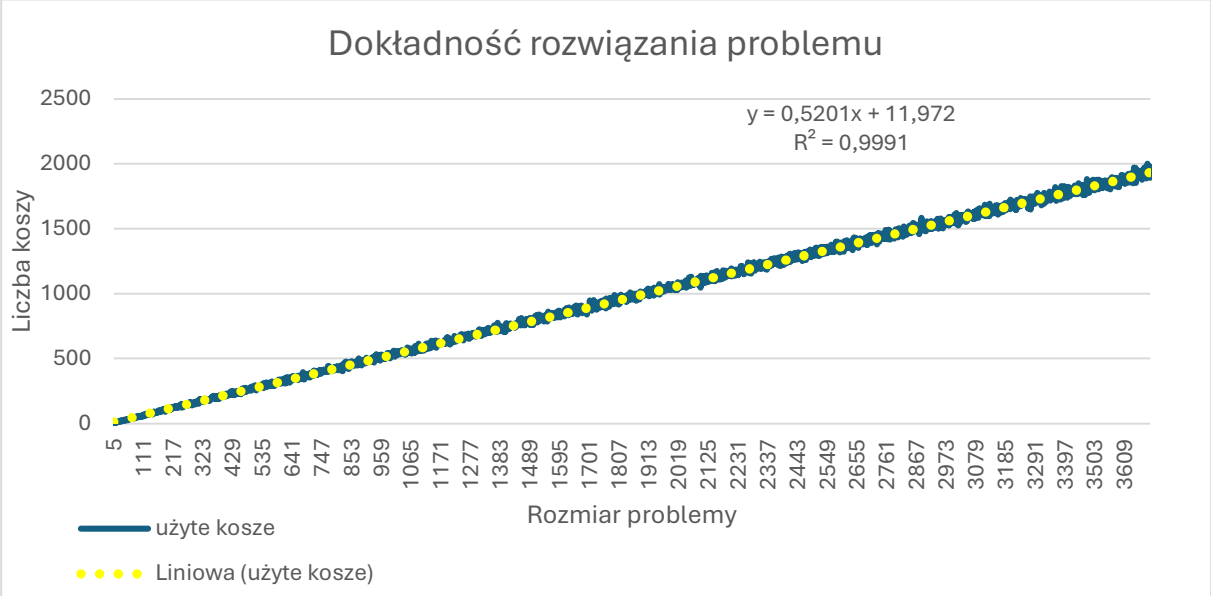
Rys. 13. Liczba użytych koszy do rozwiązania problemu – algorytm first fit przypadek średni



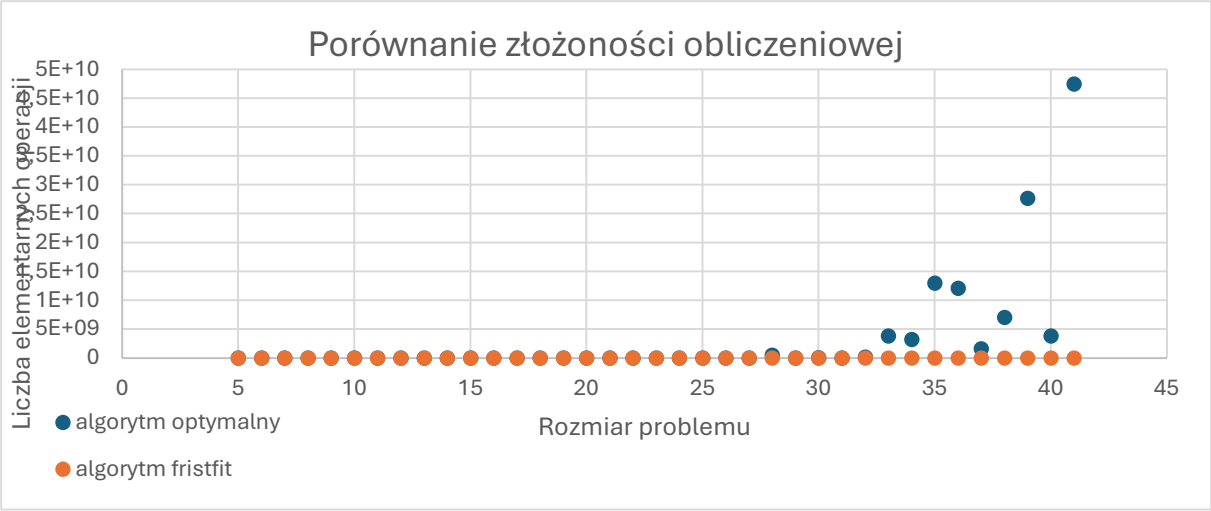
Rys. 14. Czas wykonania algorytmu względem problemu– algorytm first fit przypadek średni



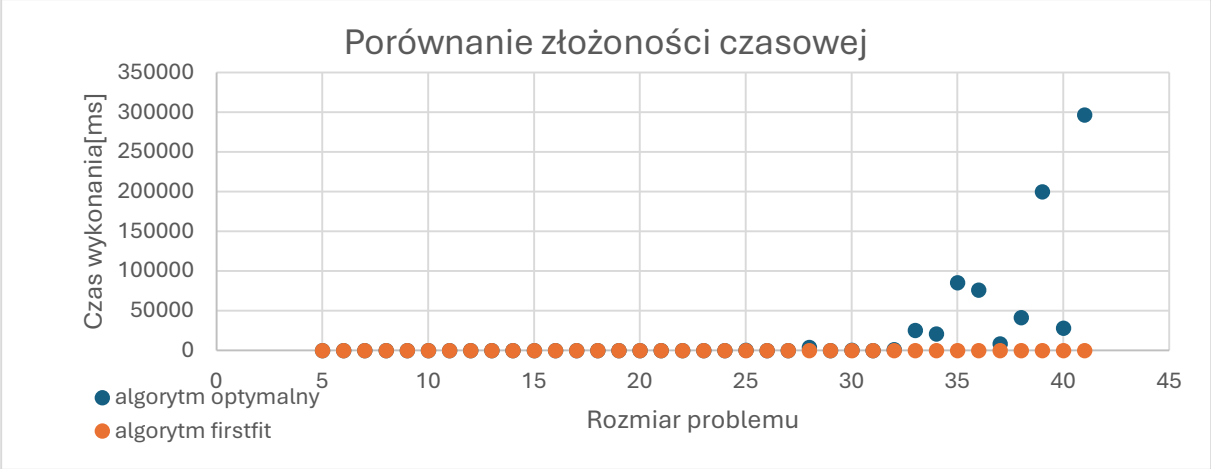
Rys. 15. Złożoność pamięciowa– algorytm first fit przypadek średni



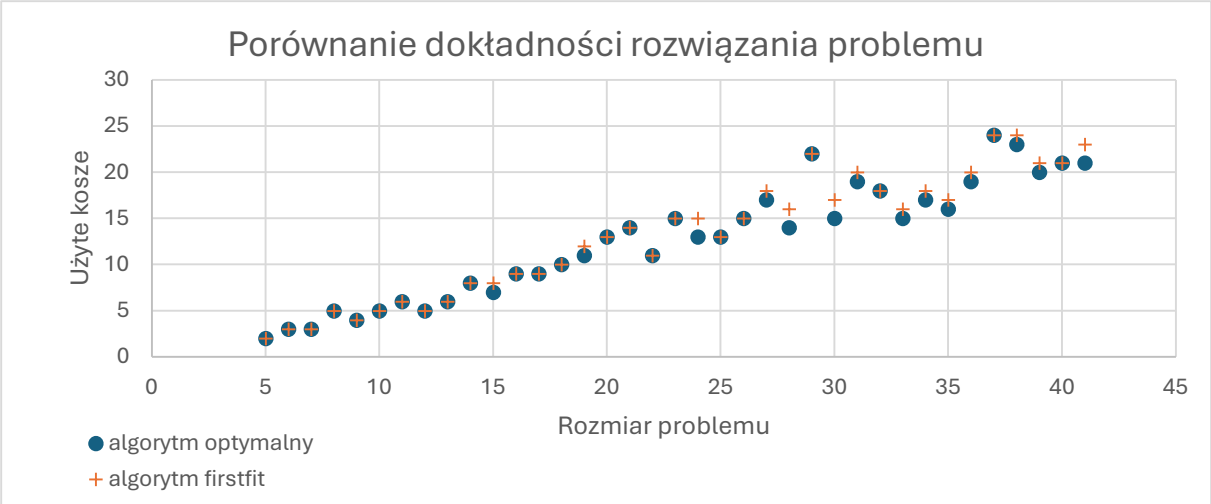
Rys. 16. Liczba użytych koszy do rozwiązania problemu – algorytm optymalny przypadek średni



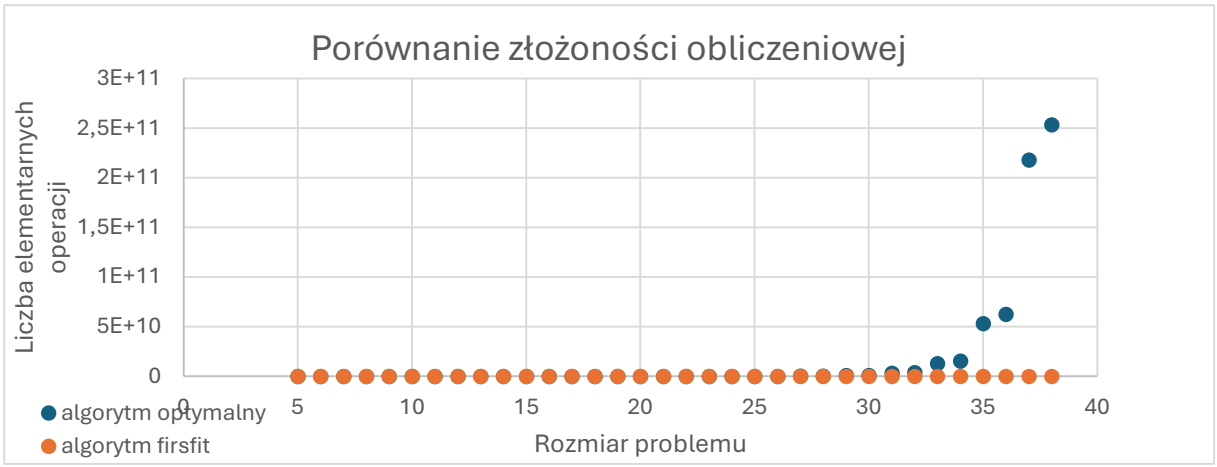
Rys. 17. Porównanie złożoności obliczeniowej algorytmu optymalnego i frist fit przypadek średni



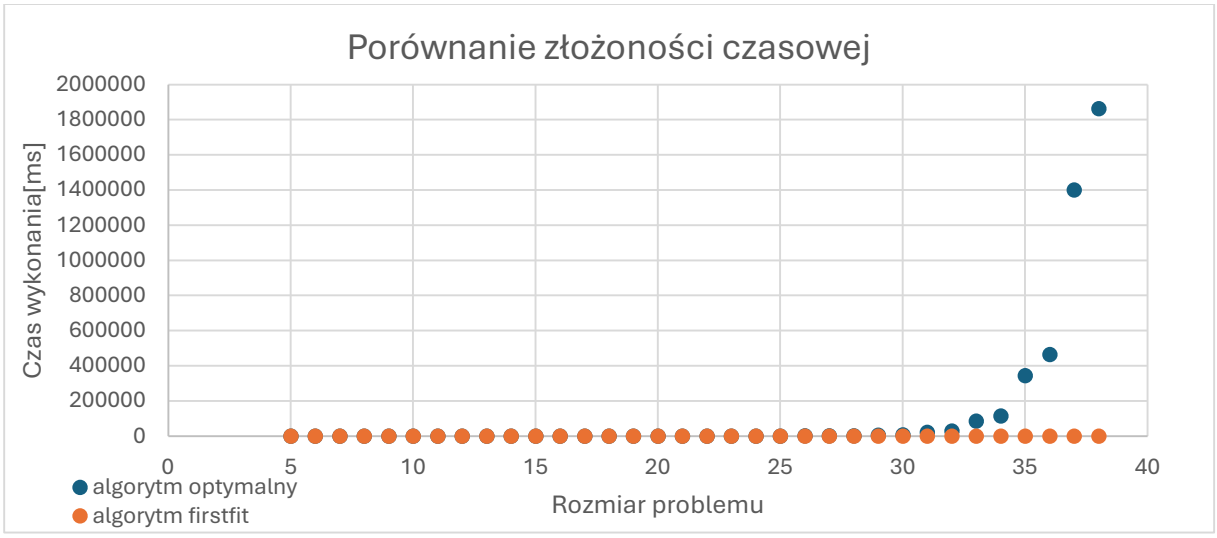
Rys. 18. Porównanie złożoności czasowej algorytmu optymalnego i frist fit przypadek średni



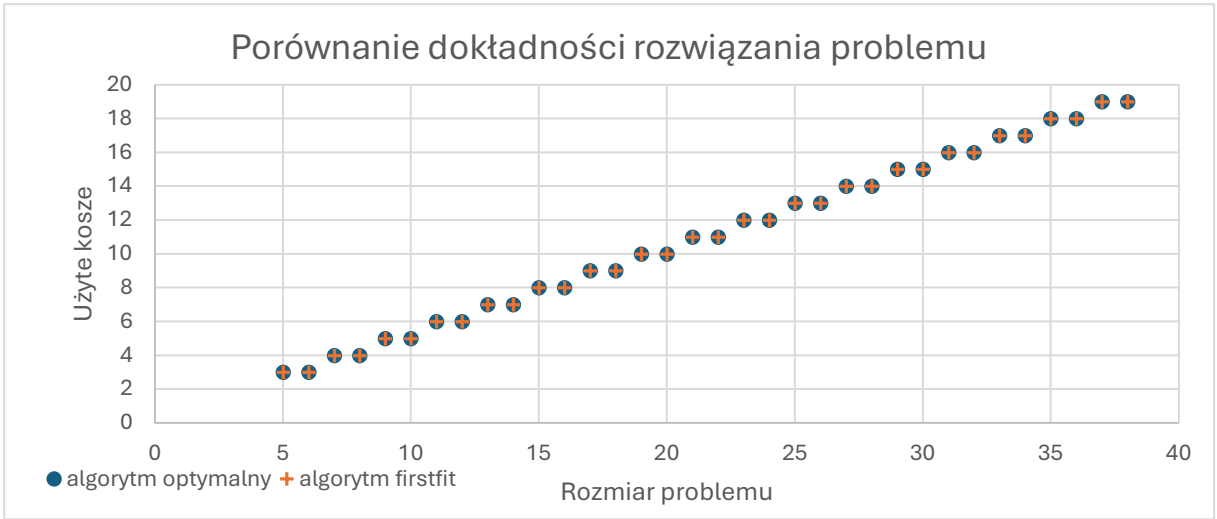
Porównanie dokładności rozwiązania problemu algorytmu optymalnego i frist fit przypadek średni



Rys. 19. Porównanie złożoności obliczeniowej algorytmu optymalnego i frist fit przypadek pesymistyczny



Rys. 20. Porównanie złożoności czasowej algorytmu optymalnego i frist fit przypadek pesymistyczny



Rys. 21. Porównanie dokładności rozwiązania problemu algorytmu optymalnego i frist fit przypadek średni

8. Wnioski

W trakcie realizacji projektu dotyczącego problemu pakowania przeanalizowano dwa podejścia algorytmiczne – algorytm dokładny (optymalny) oraz heurystykę First Fit. Na podstawie wyników eksperymentalnych oraz przeprowadzonych analiz można sformułować następujące wnioski:

1. Złożoność obliczeniowa i czas działania
   Algorytm First Fit, mimo swojej prostoty, charakteryzuje się znacznie niższą złożonością czasową. Działa w czasie rzędu $O(n^2)$, co umożliwia jego efektywne stosowanie nawet dla dużych rozmiarów problemu ($n > 1000$). W przeciwieństwie do niego, algorytm optymalny wykazuje wykładniczy wzrost złożoności obliczeniowej – jego użyteczność kończy się praktycznie przy $n \approx 20$, gdyż dalsze zwiększanie liczby przedmiotów powoduje drastyczny wzrost czasu wykonania.

2. Liczba operacji
   Eksperymenty pokazały, że liczba operacji (np. sumowań) w algorytmie First Fit jest o rząd wielkości mniejsza niż w algorytmie dokładnym. Potwierdza to teoretyczne przewidywania i wskazuje na znaczne oszczędności w zasobach obliczeniowych podczas jego stosowania.

3. Dokładność rozwiązania
   - W przypadku średnim heurystyka First Fit osiąga wyniki bardzo bliskie optimum – stosunek liczby użytych pojemników do rozwiązania optymalnego mieści się zazwyczaj w przedziale 1.00–1.05.
   - W przypadku pesymistycznym (np. sekwencja naprzemiennych dużych i małych przedmiotów) dokładność ulega pogorszeniu (1.10–1.20), ale nadal pozostaje w granicach akceptowalnych dla zastosowań praktycznych.
   - Algorytm optymalny, jak przystało na jego konstrukcję, zawsze znajduje najlepsze możliwe rozwiązanie, jednak czyni to kosztem czasu i pamięci operacyjnej.

4. Zastosowanie praktyczne
   W realnych scenariuszach, gdzie istotne są ograniczenia czasowe i zasobowe, a rozwiązanie nie musi być idealne, heurystyka First Fit jest zdecydowanie bardziej praktyczna. Algorytm optymalny sprawdza się wyłącznie w małych instancjach problemu lub jako punkt odniesienia do oceny jakości heurystyk.

5. Podsumowanie porównania
   Projekt potwierdził klasyczne twierdzenia z dziedziny algorytmiki: istnieje istotny kompromis pomiędzy dokładnością rozwiązania a jego kosztami obliczeniowymi. First Fit reprezentuje rozwiązanie szybkie i proste, a jednocześnie zadziwiająco skuteczne w praktyce, natomiast pełne przeszukiwanie gwarantuje dokładność kosztem wydajności.