Algorytmy optymalizacji	
Kierunek *Informatyczne systemy automatyki*	**Termin** *Środa TN 15:15*
Imię, nazwisko, numer albumu *Karol Jantas 259306, Marek Wlazło 259365*	**Data** *5 czerwca 2024*
Link do projektu `https://www.overleaf.com/4172127553gnrmwbqvmppf#deaf9c`	

Raport końcowy

1 Wstęp

Problem pakowania jest jednym z kluczowych problemów optymalizacyjnych, z którymi spotykamy się w różnych dziedzinach życia. Polega on na efektywnym umieszczeniu określonych obiektów w danej, ograniczonej przestrzeni tak aby zoptymalizować daną funkcję celu np. minimalizując zużycie miejsca, minimalizując koszty transportu, maksymalizując wartość pakowanego ładunku [2].

Główne wyzwania związane z problemem pakowania obejmują jego NP-trudność, co oznacza, że znalezienie optymalnego rozwiązania jest często trudne lub niemożliwe w skończonym czasie, zwłaszcza dla dużych instancji problemu. Ponadto, sposób reprezentacji problemu oraz zastosowane algorytmy mają istotny wpływ na efektywność rozwiązania. Istnieją dwie zasadnicze grupy algorytmów, które służą do rozwiązywania problemu pakowania:

1.1 Algorytmy dokładne

Algorytmy dokładne charakteryzują się następującymi własnościami:

- wysoka precyzja – dokładne algorytmy zapewniają dokładne rozwiązania dla danego problemu. Oznacza to, że po zakończeniu działania algorytmu uzyskujemy optymalne rozwiązanie [3].
- duża złożoność czasowa – często mają wyższą złożoność czasową, szczególnie dla problemów o dużej liczbie danych wejściowych lub złożoności obliczeniowej.

Przykładowe algorytmy dokładne obejmują metody takie jak przeszukiwanie całkowite, programowanie dynamiczne, czy algorytmy oparte na eliminacji wstecznej.

1.2 Algorytmy heurystyczne

Algorytmy heurystyczne cechują się poniższymi właściwościami:

- przybliżone rozwiązania – algorytmy heurystyczne zazwyczaj znajdują przybliżone rozwiązania, które mogą być zbliżone do optymalnych, ale nie gwarantują optymalności [1].
- złożoność czasowa – zazwyczaj mają niższą złożoność czasową niż algorytmy dokładne. Często stosuje się je do rozwiązywania problemów, które są zbyt trudne do rozwiązania w sposób dokładny w sensownym czasie.

Przykładowe algorytmy heurystyczne obejmują metody takie jak przeszukiwanie z nawrotami, algorytmy genetyczne, czy metody gradientowe.

2 Opis problemu

Problem pakowania w rozważanym wariancie, polega na takim rozmieszczeniu pudełek w danym kontenerze, aby, przy spełnieniu wszystkich przyjętych ograniczeń, całkowite pole jak i liczba zapakowanych pudełek była możliwie jak największa. Kontener posiada formę prostokąta o stałych wymiarach: szerokości W, wysokości H i polu P. Dany jest zbiór pudełek $Z = \{z_1, z_2, \ldots, z_n\}$. Każde pudełko $z_i \in$ Z posiada określoną szerokość w oraz wysokość h. Rozwiązanie rozważanego problemu określa zbiór zapakowanych pudełek stanowiący podzbiór lub całość dostępnego zbioru pudełek. Rozwiązanie postawionego problemu musi spełniać następujące rozwiązania:

- zapakowane pudełka nie mogą na siebie nachodzić,
- każde zapakowane pudełko musi znajdować się w całości wewnątrz kontenera, równolegle do jego ścian,
- każde zapakowane pudełko musi zostać umieszczone na innym zapakowanym pudełku bądź na spodzie kontenera.

Celem jest znalezienie dozwolonego rozwiązania, które maksymalizuje zarówno liczbę umieszczonych w kontenerze pudełek jak i obszar załadunku:

$$max \sum_{i=1}^{n} C_i \tag{1}$$

$$max \sum_{i=1}^{n} F_i \tag{2}$$

gdzie:

$$C_i = \begin{cases} 1 \; gdy \; Z_i \; jest \; wykoystane \\ 0 \; w \; przeciwnym \; razie \end{cases} \tag{3}$$

$$F_i = c_i p_i \tag{4}$$

$$p_i = w_i h_i \tag{5}$$

3 Opis metody rozwiązywania

Powyższy problem pakowania został skutecznie rozwiązany przy wykorzystaniu algorytmu genetycznego. Poniżej przedstawiony został szczegółowy opis głównych etapów zaimplementowanego algorytmu, które zostały przeprowadzone w celu znalezienia możliwie najlepszego rozwiązania:

- kodowanie rozwiązań – każdy chromosom reprezentuje jedno możliwe rozmieszczenie elementów w kontenerze. W tym przypadku jest to lista zawierająca wymiary pudełek (szerokość, wysokość) wraz z ich współrzędnymi (x, y) wewnątrz kontenera,
- inicjalizacja populacji – tworzona jest początkowa populacja chromosomów, gdzie każdy chromosom reprezentuje losowe rozmieszczenie elementów w kontenerze,
- funkcja oceny – ocenia każde rozmieszczenie elementów pod względem wykorzystania dostępnego miejsca oraz ilości pudełek w kontenerze. Im większa liczba zajętego miejsca oraz liczba umieszczonych pudełek w kontenerze, tym populacja jest lepiej oceniania,
- operatory genetyczne – stosowane są operatory genetyczne, takie jak selekcja i mutacja, aby wygenerować nowe rozwiązania. Zaimplementowana mutacja polega na losowych zmianach pozycji pudełek,
- ewolucja populacji – populacja jest ewoluowana poprzez iteracyjne stosowanie operacji genetycznych. Nowe pokolenia chromosomów są tworzone, a ich jakość jest oceniana za pomocą funkcji oceny,
- kryterium stopu – algorytm działa do momentu osiągnięcia kryterium stopu, czyli określonej liczby generacji,

- wybór najlepszego rozwiązania – po zakończeniu działania algorytmu genetycznego wybierane jest najlepsze znalezione rozwiązanie, które maksymalizuje liczbę elementów w kontenerze oraz zajęte pole, przy zachowaniu ograniczeń dotyczących dostępnej przestrzeni.

4 Opis algorytmu

4.1 Parametry:

- `population_size` (int): Rozmiar populacji, czyli liczba losowych rozwiązań tworzonych na początku algorytmu. Domyślnie 100.
- `generations` (int): Liczba pokoleń, przez które będzie ewoluować populacja. Domyślnie 1000.

4.2 Funkcje:

- `can_place_box(box, position, packed_boxes):`
  Sprawdza, czy dane pudełko może być umieszczone na danej pozycji w kontenerze bez zachodzenia na inne pudełka.
  Parametry:
  - `box` (tuple): Krotka zawierająca szerokość i wysokość pudełka.
  - `position` (tuple): Krotka zawierająca współrzędne x i y pozycji, gdzie pudełko jest próbowane umieścić.
  - `packed_boxes` (list): Lista krotek reprezentujących umieszczone już pudełka w kontenerze.

  Zwraca:
  `True`, jeśli pudełko może być umieszczone na danej pozycji; `False`, w przeciwnym razie.
- `evaluate_solution(solution):`
  Ocenia daną konfigurację pudełek (rozwiązanie) pod kątem liczby umieszczonych pudełek i całkowitej zajętej powierzchni.
  Parametr:
  - `solution` (list): Lista krotek, gdzie każda krotka zawiera informacje o jednym pudełku (szerokość, wysokość) i jego pozycji.

  Zwraca:
  Krotka zawierająca liczbę umieszczonych pudełek i całkowitą zajętą powierzchnię.
- `generate_random_solution():`
  Generuje losową konfigurację pudełek w kontenerze.
  Zwraca:
  Losową konfigurację pudełek w postaci listy krotek.
- `mutate(solution):`
  Mutuje losowo wybrane pudełko w danej konfiguracji.
  Parametr:
  - `solution` (list): Konfiguracja pudełek, która ma zostać zmutowana.
- `genetic_algorithm(population_size=100, generations=1000):`
  Implementuje algorytm genetyczny do rozwiązania problemu pakowania pudełek w kontenerze.
  Parametry:
  - `population_size` (int): Rozmiar populacji początkowej.
  - `generations` (int): Liczba pokoleń.

  Zwraca:
  Krotka zawierająca najlepszą znalezioną konfigurację pudełek i jej ocenę fitness.

5 Baza danych

W badaniach zaimplementowanego algorytmu użyta została baza danych "Bin Packing: Two-dimensional" z bilbioteki OR-Library, która zawiera instancje dwuwymiarowego problemu pakowania w pojemniki rozważanego w Hopper E. i Turton B. C. H., 2002, „An empirical study of meta-heuristics applied to 2D rectangular bin packing" Special Issue on Cutting, Packing and Knapsacking Problems, Studia Informatica, vol. 2, no. 1. ISBN 2-912590-13-2; ISSN Regular 1625-7545. Baza ta posiada trzy różne zestawy danych, każdy zestaw zawiera 5 zbiorów:

- 1 zestaw – M1a, M1b, M1c, M1d, M1e
- 2 zestaw – M2a, M2b, M2c, M2d, M2e
- 3 zestaw – M3a, M3b, M3c, M3d, M3e

Każdy zbiór składa się z szeregu par liczb – szerokości oraz wysokości pojedynczego pudełka. Poszczególne zestawy wyróżniają się następującymi cechami

- 1 zestaw – każdy zbiór posiada: 100 par liczb (szerokość,wysokość), suma pól wszystkich pudełek wynosi od 2500 do 2850, każda wysokość oraz szerokość zawiera się w przedziale od 1 do 9,
- 2 zestaw – każdy zbiór posiada: 100 par liczb (szerokość,wysokość), suma pól wszystkich pudełek wynosi od 24750 do 25750, każda wysokość oraz szerokość zawiera się w przedziale od 1 do 30,
- 3 zestaw – każdy zbiór posiada: 150 par liczb (szerokość,wysokość), suma pól wszystkich pudełek wynosi od 25700 do 30000, każda wysokość oraz szerokość zawiera się w przedziale od 1 do 30,

6 Badania

Badania zaimplementowanego algorytmu genetycznego zostały przeprowadzone dla każdego zbioru z 3 zestawów danych. Każdy zbiór został przetestowany trzy razy. We wszystkich tabelach występuje para liczb, pierwsza liczba odnosi się do liczby zapakowanych pudełek, natomiast druga określa całkowite pole zajęte przez zapakowane pudełka. Wyniki zaimplementowanego algorytmu genetycznego dla pierwszego zestawu danych można zaobserwować w tabeli 1, tabela 2 przedstawia wyniki dla drugiego zestawu, natomiast w tabeli 3 zostały umieszczone wyniki algorytmu dla zestawu trzeciego. We wszystkich badaniach zostały użyte takie same hiperparametry:

- wielkość populacji (pop_size) – 100,
- liczba generacji (generations) – 1000.

Tabela 1: Zestaw pierwszy (M1) – wielkość kontenera 10x10

	M1a	M1b	M1c	M1d	M1e
1	12,75	9,91	11,89	6,96	7,83
2	11,92	7,96	9,94	5,95	9,96
3	12,82	7,94	8,99	6,96	7,95

Tabela 2: Zestaw drugi (M2) – wielkość kontenera 50x50

	M2a	M2b	M2c	M2d	M2e
1	16,1215	14,1687	13,1554	12,1485	13,1671
2	15,1642	9,1845	13,1697	13,1441	14,1762
3	12,1752	10,1953	12,1706	11,1782	11,1590

Tabela 3: Zestaw trzeci (M3) – wielkość kontenera 50x50

	M3a	M3b	M3c	M3d	M3e
1	14,1395	15,1462	16,1551	10,2049	15,1777
2	13,2049	14,1651	15,1366	14,1552	16,1369
3	11,1561	19,1374	17,1387	14,1817	16,1723

7 Wnioski

Analizując wyniki badań dla pierwszego zestawu (tabela 1), można zauważyć, że liczba zapakowanych pudełek do kontenera zawiera się w zakresie od 5 do 12, natomiast ich całkowita powierzchnia zawiera się w przedziale od 75 do 99. W przypadku zestawu drugiego (tabela 2), zaobserwować można, że liczba zapakowanych pudełek oscyluje w okolicach 9-16 sztuk, których całkowita powierzchnia wynosi około 1215-1953. W trzecim zestawie (tabela 3) liczba pudełek w kontenerze zawiera się w zakresie od 10 do 19, których łączne pole wynosi od 1366 do 2049.

Zaimplemenetowany algorytm genetyczny dobrze radzi sobie dla pierwszego zestawu danych (M1). Zaobserwować można to po tym, że w większości przypadków zapełnia on prawie całe dostępne pole kontenera (wymiary 10x10 – pole 100). W przypadku zestawu drugiego oraz trzeciego algorytm działa wyraźnie słabiej, ponieważ ma wyraźne problemy z zapełnieniem całego pola kontenera (wymiary 50x50 – pole 2500). Przeważnie algorytmowi udało się spakować pudełka, których łączne pole wynosiło około 1500-1750 (zestaw drugi M2) oraz 1400-1650 (zestaw trzeci M3). Powodem takiego stanu rzeczy może być nieodpowiednie dostosowanie hiperparametrów. Zwiększenie liczby generacji oraz liczby populacji powinno pomóc w osiągnięciu lepszego wyniku dla zestawu drugiego oraz trzeciego. Należy jednak pamiętać, że im większa wartość wyżej wymienionych hiperparametrów, tym algorytm potrzebuje znacznie więcej czasu, aby się wykonać.

Literatura

[1] Natallia Kokash. An introduction to heuristic algorithms. *Department of Informatics and Telecommunications*, pages 1–8, 2005.

[2] Andrea Lodi, Silvano Martello, Michele Monaci, and Daniele Vigo. *Two-Dimensional Bin Packing Problems*, chapter 5, pages 107–129. John Wiley Sons, Ltd, 2014.

[3] Czesław Smutnicki. Algorytmy szeregowania zadań. *Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław*, 2012.