POLITECHNIKA WROCŁAWSKA
WYDZIAŁ ELEKTRONIKI

KIERUNEK: Automatyka i Robotyka
SPECJALNOŚĆ: Systemy informatyczne w automatyce (ASI)

PROJEKT INŻYNIERSKI

Algorytm optymalizacji załadunku pojazdów

Optimization algorithm for vehicle loading

AUTOR:
Łukasz Barczak

PROWADZĄCY PRACĘ:
Prof. dr hab. inż. Czesław Smutnicki

OCENA PRACY:

WROCŁAW 2012

Spis treści

1. Wstęp

Od momentu wynalezienia przez Sumerów koła, człowiek zaczął przewozić przedmioty w najróżniejsze strony świata. Mimo że cele przyświecające tej idei są odmienne, począwszy od transportu w celach usługowych, a kończąc na zaopatrzeniu wojsk na liniach frontu w czasach wielkich wojen, zawsze napotykano na ten sam problem. Mowa tutaj o załadunku jak największej ilości towaru do jak najmniejszej liczby kontenerów. Lepsze zagospodarowanie przestrzeni palety, na którą układane są pakunki, zwiększa ich liczbę na jednej palecie, co prowadzi do zminimalizowania ilości użytych kontenerów.

Maksymalne wykorzystanie dostępnej przestrzeni palety pozwala na duże oszczędności, ponieważ większą ilość przesyłek można zapakować w mniejszą ilość kontenerów, co przekłada się na mniejszą ilość pojazdów wykorzystanych do transportu tegoż towaru. To z kolei prowadzi do mniejszego zużycia zarówno paliwa jak i pojazdów. Także mniejsza liczba kierowców jest angażowana w realizację zadania. Optymalizacja załadunku minimalizuje koszty ponoszone przez firmy logistyczne i transportowe, dlatego wielkie przedsiębiorstwa inwestują bardzo duże pieniądze w jak najlepsze algorytmy oraz narzędzia.

Ze względu na NP-trudny charakter problemu [1], naukowcy, inżynierowie oraz pasjonaci z całego świata starają się od dziesiątek lat uzyskać rozwiązanie optymalne. Wyniki ich badań są mniej lub bardziej skuteczne, co prowadzi do sporej ilości algorytmów.

Dotychczas powstały trzy grupy algorytmów rozwiązujących podany problem:

- Heurystyczne,
- Meta-heurystyczne,
- Dokładne.

Problem załadunku można rozpatrywać w jednym wymiarze, w dwóch bądź w trzech wymiarach. Przedstawione zagadnienie często nazywane jest dystrybutorskim problemem ładowania, gdzie produkty o różnych wymiarach pakowane są na paletę warstwami. Pakunki w danej warstwie mają taką samą wysokość. Problem jest opisany jako problem pakowania mniejszych pojemników do jak najmniejszej ilości większych pojemników, które są identyczne [2].

2. Cel projektu

Celem niniejszego projektu inżynierskiego było zaimplementowanie, przy użyciu dowolnych narzędzi programistycznych, algorytmu optymalizującego załadunek floty pojazdów o identycznych kontenerach w 2D. Podstawowe cele, jakie należało zrealizować to:

- zapoznanie się z problemem załadunku,
- zapoznanie się z algorytmami rozwiązującymi problem załadunku,
- zaimplementowanie wybranych algorytmów,
- porównanie efektywności algorytmów,
- ocena algorytmów.

3. Zakres opracowania

W pracy zawarto opisy zagadnienia załadunku, algorytmów wykorzystanych w pracy oraz wiele rysunków i przykładów przedstawiających oraz obrazujących problem przedstawiony w niniejszym dokumencie, a także sposoby jego rozwiązania. W rozdziale czwartym przedstawiony został problem pakowania oraz sposoby oceny jakości upakowania. Rozdział ten zawiera opis problemu załadunku w 2D. Rozdział piąty został poświęcony opisom wykorzystanych w pracy metod rozwiązywania problemu załadunku w 2D. Rozdział szósty skupia się na sposobie zaimplementowania problemu załadunku w języku C++. Opis hierarchii oraz użytych struktur danych, na których operują algorytmy. Rozdział siódmy obejmuje opis powstałej aplikacji rozwiązującej problem pakowania oraz jej przykładowe działanie, a także opis środowiska Microsoft Visual Studio 2008, w którym stworzono aplikację. Badania eksperymentalne zaimplementowanych algorytmów, tj. Best Fit, Worst Fit oraz First Fit zostały przedstawione w rozdziale ósmym. Testy miały na celu sprawdzenie, która z wykorzystanych metod zwraca najlepszy wynik dla przykładowej instancji problemu oraz który algorytm jest najefektywniejszy w sensie średnim. W rozdziale dziewiątym – ostatnim – podsumowano uzyskane wyniki oraz sformułowano wnioski.

4. Problem

Rozdział ten został poświęcony opisowi matematycznemu problemu załadunku w 2D oraz ocenie jakości rozłożenia paczek na palecie.

4.1 Sformułowanie problemu

Dana jest skończona ilość palet (1) oraz skończona ilość paczek (5) (w dalszej części dokumentu nazywane także pudełkami, pakunkami bądź towarami), które należy załadować na jak najmniejszą liczbę palet. Długość oraz szerokość każdej palety wynoszą odpowiednio *L* i *W* (2), natomiast długość i szerokość każdej paczki – *l* i *w* (6) [2]. Wymiary zarówno palet (3) (4) jak i pudełek są liczbami całkowitymi dodatnimi (7) (8). Paczki można obracać w pionie i poziomie. Paczki nie mogą na siebie nachodzić ani wystawać poza krawędzie palety (9). Dla budowy modelu matematycznego przyjęto następujące oznaczenia:

$$P=\{p_1, p_2, \ldots, p_m\} \quad , \tag{1}$$

$$pi=(L_i, W_i) \quad \text{, gdzie i = 1, 2, …, m,} \tag{2}$$

P – skończony zbiór palet,

p_i – i-ta paleta o długości L_i oraz szerokości W_i ,

L_i – długość palety p_i , gdzie:

$$0 < L_i < \infty, \tag{3}$$

W_i – szerokość palety p_i , gdzie:

$$0 < W_i < \infty, \tag{4}$$

$$Z=\{a_1, a_2, \ldots, a_n\} \quad , \tag{5}$$

$$a_i=(l_i, w_i) \quad \text{, gdzie i = 1, 2, …, n,} \tag{6}$$

Z – skończony zbiór paczek składający się z n prostokątnych paczek,

a_i – i-ta paczka o długości l_i i szerokości w_i ,

l_i – długość paczki a_i , gdzie :

$$0 < l_i \leq L_i \quad , \tag{7}$$

w_i – szerokość paczki a_i , gdzie:

$$0 < w_i \leq W_i \quad , \tag{8}$$

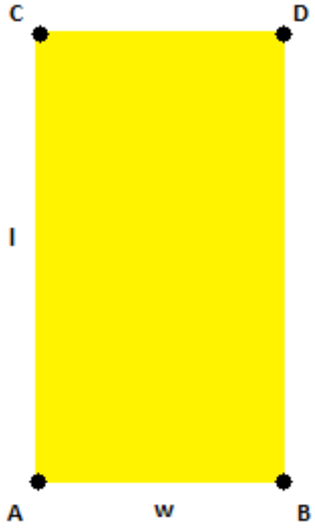
Dla każdego i = 1, 2, …, n zachodzi:

$$[(0<l_i\leq L)\wedge(0<w_i\leq D)\vee(0<w_i\leq L)\wedge(0<l_i\leq D)] \quad . \tag{9}$$

Rozmieszczenie paczki na palecie można opisać w sposób geometryczny, za pomocą współrzędnych x i y lewego, dolnego wierzchołka paczki oraz jej orientacji (poziomej bądź pionowej).

Rysunek 1. Wymiary i współrzędne paczki w orientacji poziomej.

Współrzędne poszczególnych wierzchołków paczki w orientacji poziomej (o_i = 0) przedstawionej na Rysunku 1. wynoszą odpowiednio $A=(x_i, y_i)$, $B=(x_i+l, y_i)$, $C=(x_i, y_i+w)$, $D=(x_i+l, y_i+w)$.



Rysunek 2. Wymiary i współrzędne paczki w orientacji pionowej.

Współrzędne poszczególnych wierzchołków paczki w orientacji pionowej (o_i = 1) przedstawionej na Rysunku 2. wynoszą odpowiednio $A=(x_i, y_i)$, $B=(x_i+w, y_i)$, $C=(x_i, y_i+l)$, $D=(x_i+w, y_i+l)$.

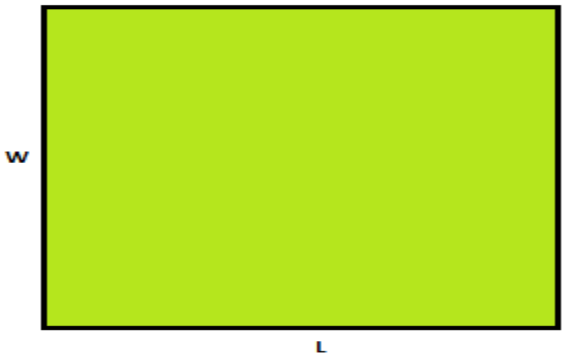
Paczki nie mogą wystawać poza krawędzie palety, na której się znajdują. Dopuszczalne rozmieszczenie paczek zostało przedstawione na Rysunku 3:

- dla $o_i = 0$:

$$(x_i \geq 0) \wedge (y_i \geq 0) \quad ,$$
$$(x_i + l_i \leq L) \wedge (y_i + w_i \leq W) \quad .$$

- dla $o_i = 1$:

$$(x_i \geq 0) \wedge (y_i \geq 0) \quad ,$$
$$(x_i + w_i \leq L) \wedge (y_i + l_i \leq W) \quad .$$

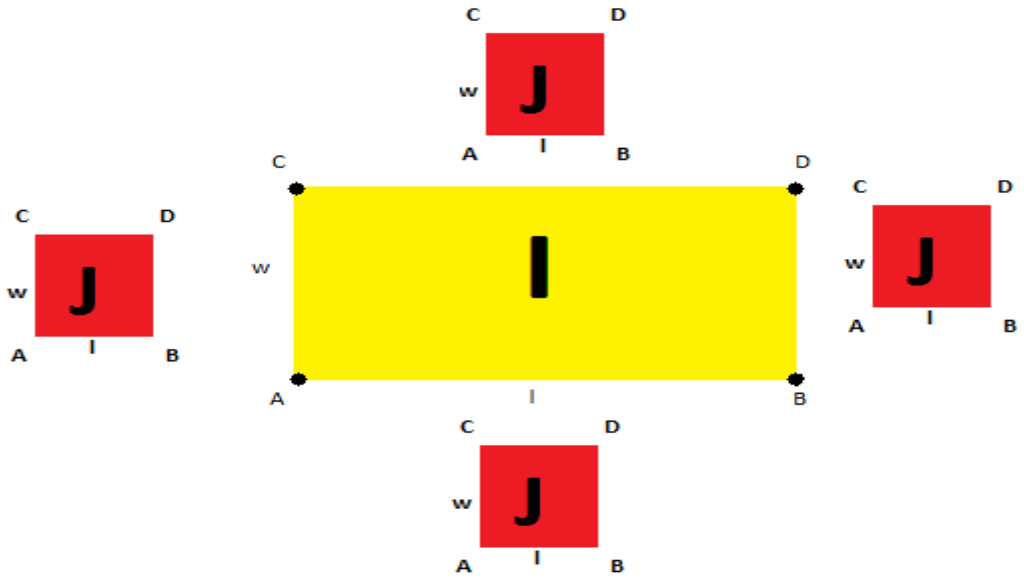


Rysunek 3. Dopuszczalne rozmieszczenie paczki na palecie.

Dowolne dwa pudełka nie mogą nachodzić na siebie, co obrazują Rysunki 4, 5, 6 i 7:

- dla $o_i = 0$ oraz $o_j = 0$:

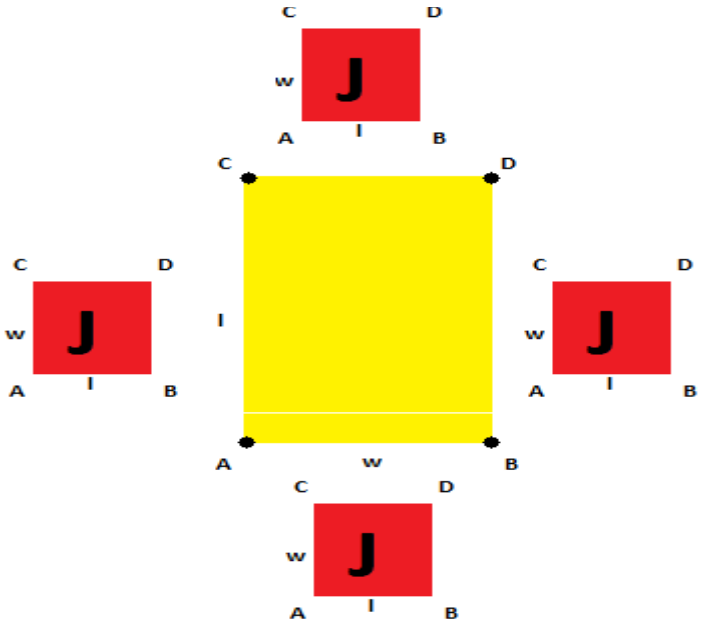
$$(x_j + l_j \leq x_i) \vee (y_j \geq y_i + w_i) \vee (y_j + w_j \leq y_i) \vee (x_j \geq x_i + l_i) \quad ,$$



Rysunek 4. Dopuszczalne rozmieszczenie dwóch dowolnych pudełek w orientacji poziomej.

- dla $o_i = 1$ oraz $o_j = 0$ (Rysunek 5):

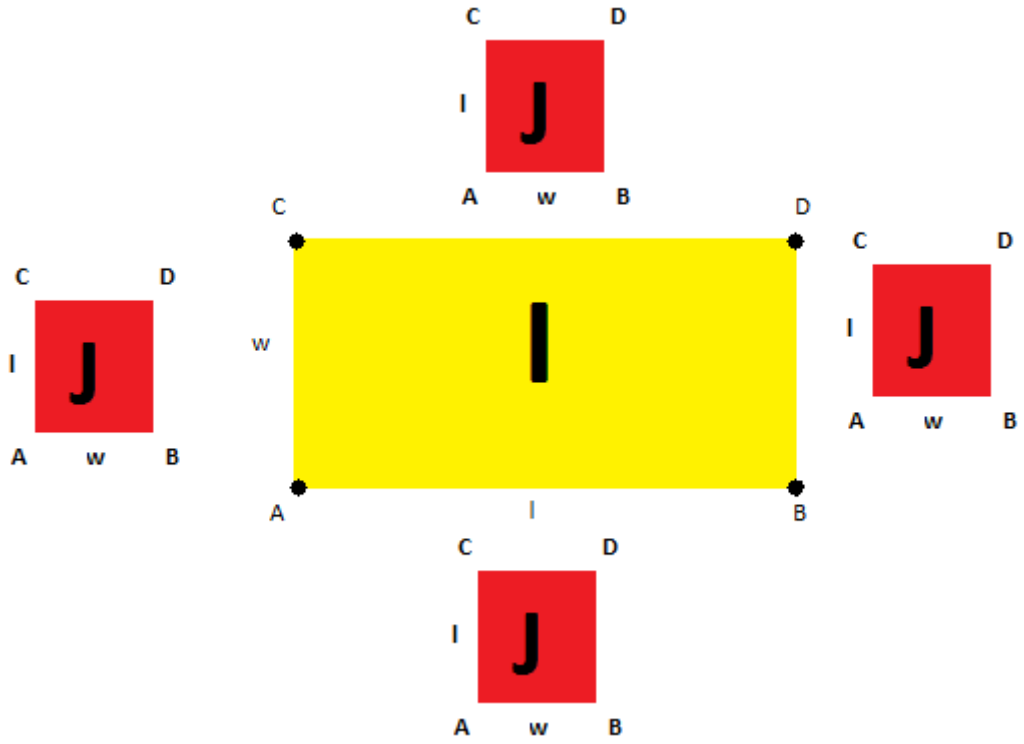
$$(x_j + l_j \leq x_i) \vee (y_j \geq y_i + l_i) \vee (y_j + w_j \leq y_i) \vee (x_j \geq x_i + w_i) \quad ,$$



Rysunek 5. Dopuszczalne rozmieszczenie dwóch dowolnych pudełek w orientacji pion-poziom.

- dla $o_i = 0$ oraz $o_j = 1$ (Rysunek 6):

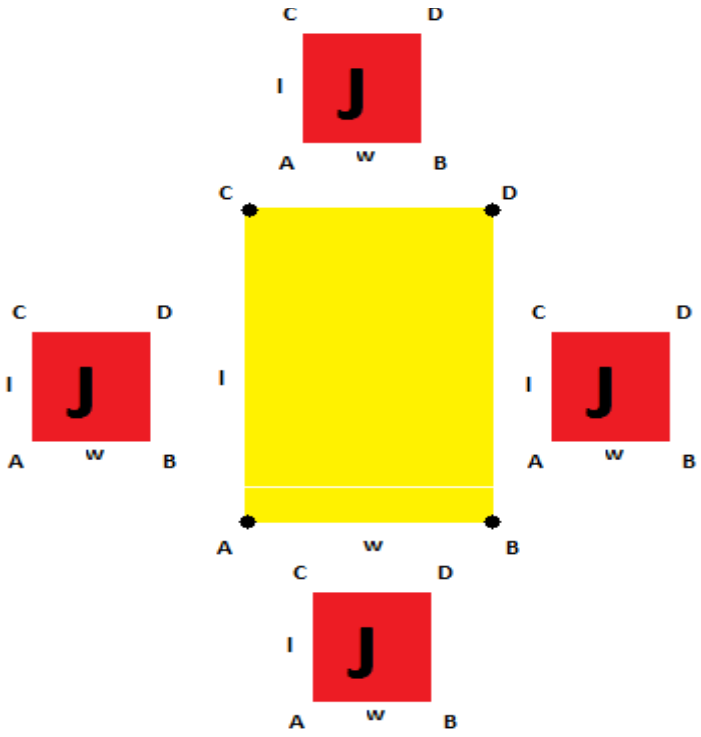
$$(x_j + w_j \leq x_i) \vee (y_j \geq y_i + w_i) \vee (y_j + l_j \leq y_i) \vee (x_j \geq x_i + l_i) \quad ,$$



Rysunek 6. Dopuszczalne rozmieszczenie dwóch dowolnych pudełek w orientacji poziom-pion.

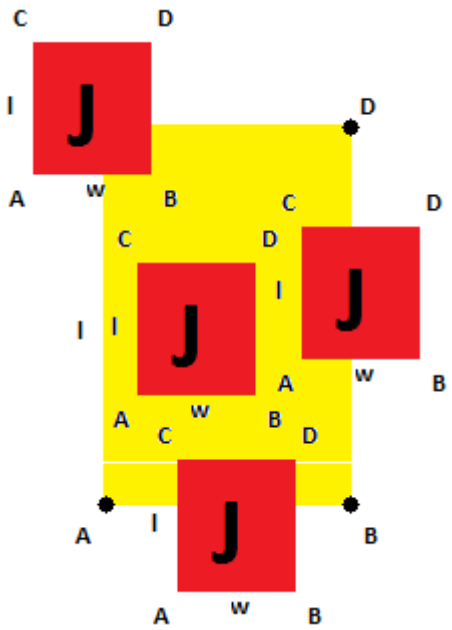
- dla $o_i = 1$ oraz $o_j = 1$ (Rysunek 7):

$$(x_j + w_j \leq x_i) \vee (y_j \geq y_i + l_i) \vee (y_j + l_j \leq y_i) \vee (x_j \geq x_i + w_i) \quad ,$$



Rysunek 7. Dopuszczalne rozmieszczenie dwóch dowolnych pudełek w orientacji pion-pion.

- dowolne dwa pudełka nie mogą nachodzić na siebie. Niedopuszczalne rozmieszczenie paczek zostało pokazane na Rysunku 8.



Rysunek 8. Niedopuszczalne rozmieszczenie dwóch dowolnych pudełek.

4.2 Ocena jakości

Podstawowym sposobem miary jakości rozmieszczenia paczek na palecie jest stosunek pól powierzchni paczek umieszczonych na palecie do pola powierzchni palety (10), na której znajdują się paczki, tzn.:

$$0 \le Q_i = \frac{\sum_{m=1}^{n} (l_m * w_m * Z_m(a_m))}{L_i * W_i} \le 1 \quad , \tag{10}$$

$Z_m(a_m)$ = 0, gdy paczka nie znajduje się na palecie p_i,

$Z_m(a_m)$ = 1, gdy paczka znajduje się na palecie p_i,

Q_i – współczynnik jakości rozmieszczenia paczek na palecie p_i.

Jednakże taki sposób oceny jakości rozmieszczenia paczek na palecie jest niewystarczający dla problemu przedstawionego w pracy, ponieważ obejmuje on większą liczbę palet niż jedną, dlatego należało poddać go modyfikacji. Celem jest takie rozmieszczenie paczek na palecie, aby minimalizować ilość palet wykorzystanych do załadunku (11), więc:

$$min(P_p) \quad , \tag{11}$$

P_p – zbiór palet wykorzystywanych do załadunku paczek.

Jeżeli $P_{pi} = P_{pj}$, to następnym krokiem oceny jakości jest porównanie liczby paczek znajdujących się na paletach ze zbioru P_{pi} z liczbą paczek zapakowanych na palety ze zbioru P_{pj} , a następnie wybranie wartości największej (12):

$$max(U_{ip}, U_{jp}) \quad , \tag{12}$$

P_{pi} – zbiór palet użytych przy algorytmie i,

P_{pj} – zbiór palet użytych przy algorytmie j,

U_{ip} – liczba paczek załadowanych na palety ze zbioru P_{pi} ,

U_{jp} – liczba paczek załadowanych na palety ze zbioru P_{pj} .

Jeżeli $U_{ip}=U_{jp}$, to kolejnym krokiem oceny jakości rozmieszczenia paczek na palecie jest porównanie sumy zajętych przestrzeni palet ze zbioru P_{pi} z sumą zajętych przestrzeni palet ze zbioru P_{pj} (13):

$$min(Z_{ip}, Z_{jp}) \quad , \tag{13}$$

Z_{ip} – suma zajętych przestrzeni palet ze zbioru P_{pi} ,

Z_{jp} – suma zajętych przestrzeni palet ze zbioru P_{pj} ,

zajęta przestrzeń – przestrzeń zajęta przez paczki oraz bloczki blokady (wytłumaczone w kolejnym rozdziale).

Wybranie minimum z wykorzystanych przestrzeni ma na celu wybranie tych palet, do których można by było upchać jeszcze więcej paczek. Innymi słowy, ocena jakości polega na upakowaniu wszystkich paczek na jak najmniejszej liczbie palet, wykorzystując jak najmniejszą ich powierzchnię.

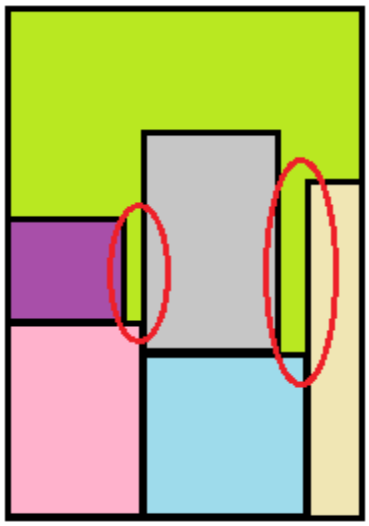
5. Metody rozwiązywania

Rozdział ten poświęcony jest algorytmom heurystycznym wykorzystanym do rozwiązania problemu załadunku w dwu-wymiarze. Zawarto opisy oraz grafiki wyjaśniające zagadnienia niezbędne do zrozumienia działania algorytmów. Przedstawiono także małe i proste instancje problemów, aby zobrazować działanie danego algorytmu. Kompleksowe badania algorytmów zostały przedstawione w rozdziale siódmym.

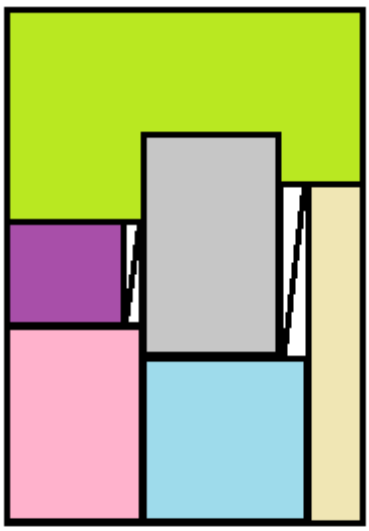
Do rozwiązania problemu wykorzystano kilka algorytmów, a mianowicie algorytmy Best Fit [3], Worst Fit oraz First Fit [4] w celu porównania osiąganych wyników oraz wybrania najlepszego spośród nich dla danej instancji problemu [2]. Algorytmy działają na podobnej zasadzie, wykorzystując puste przestrzenie występujące między paczkami na palecie, tzw. dziury. Metody te różnią się sposobem sortowania oraz dobierania paczek do danej dziury. Podane algorytmy są metodami heurystycznymi. Działają szybko, jednak ich wyniki nie zawsze są optymalne. Wykorzystanie heurystyk ma miejsce wtedy, kiedy klasyczne metody albo trwają za długo, albo nie zwracają dokładnego, optymalnego rezultat. W projekcie wykorzystano techniki heurystyczne, ponieważ są dosyć proste w implementacji oraz dają zadowalające, przybliżone wyniki – są skuteczne – i nie wymagają dużych nakładów mocy obliczeniowej, aby się wykonać. Główna struktura użytych algorytmów pozostała taka sama, jednak pewna ich część uległa modyfikacji w celu uzyskania efektywniejszych rozwiązań.

5.1 Dziura, bloczki blokady

Dziura jest to pusta przestrzeń znajdująca się na palecie, do której umieszczamy paczki. Dziury przedstawiono na Rysunku 9. Jeśli paleta jest pusta, to dziura ma taką szerokość, jak szerokość palety. Po umieszczeniu paczki następuje poszukiwanie kolejnych dziur, czyli następuje obliczenie różnicy między szerokością palety a szerokością paczki lub między szerokościami dwóch paczek. Jeśli nie ma takiej paczki, którą można by było umieścić w danej dziurze, to dziura wypełniana jest bloczkami blokady, informującymi, że nie ma takiej paczki, którą można by do niej włożyć. Bloczki blokady – Rysunek 10 - są traktowane w obliczeniach jak paczki, tzn. mają swoją szerokość oraz wysokość.



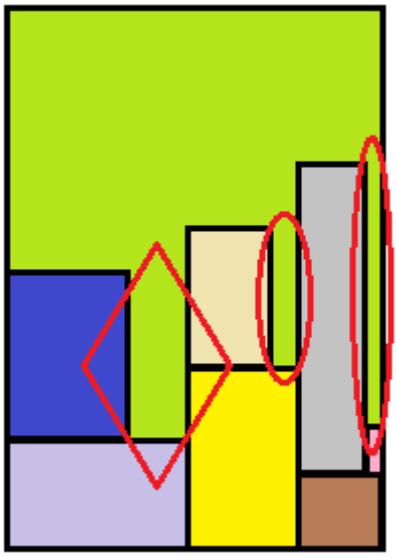
Rysunek 9. Dziury znajdujące się na palecie.



Rysunek 10. Bloczki blokady (białe prostokąty).

5.1.1 Najgłębsza dziura

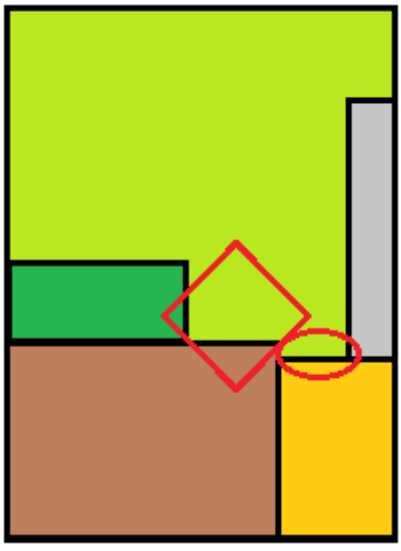
Najgłębsza dziura jest to taka dziura, której współrzędna y jest najmniejsza spośród współrzędnych y pozostałych dziur, tj. lewy dolny róg dziury leży najbliżej dna palety. Przypadek najgłębszej dziury zaprezentowano na Rysunku 11.



Rysunek 11. Najgłębsza dziura zaznaczona czerwonym rombem.

5.1.2 Najszersza dziura

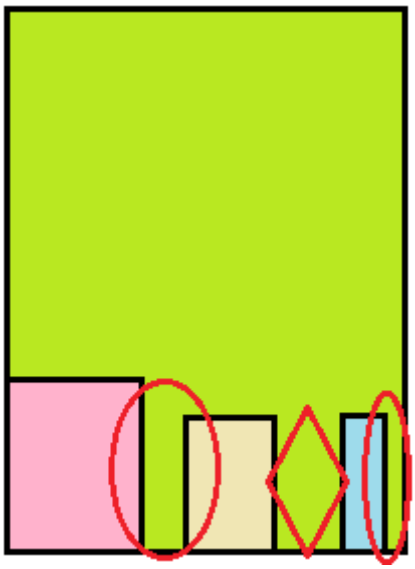
Najszersza dziura, znajdująca się na Rysunku 12, jest to taka dziura, której l – szerokość – jest największa spośród szerokości pozostałych dziur.



Rysunek 12. Najszersza dziura zaznaczona czerwonym rombem.

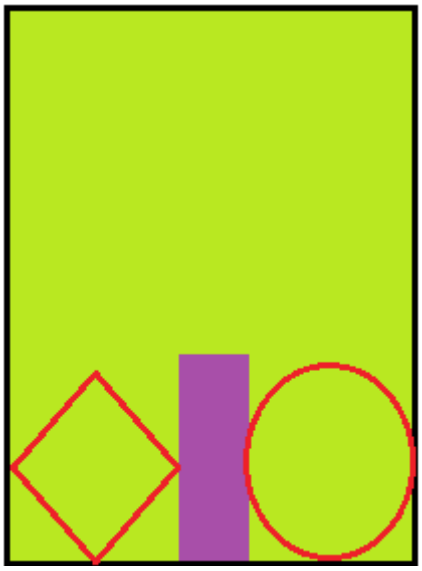
5.1.3 Przypadek kilku najgłębszych dziur

Jeżeli zdarzy się sytuacja przedstawiona na Rysunku 13, że na palecie pojawi się kilka dziur należących do kategorii najgłębszych, to wybierana jest ta, która jest najszersza.



Rysunek 13. Najszersza dziura z najgłębszych zaznaczona czerwonym rombem.

Jeżeli nie istnieje dziura, która jest najszersza, to wybierana jest dziura, której lewy dolny róg znajduje się najbliżej lewej krawędzi palety. Pokazano to na Rysunku 14.

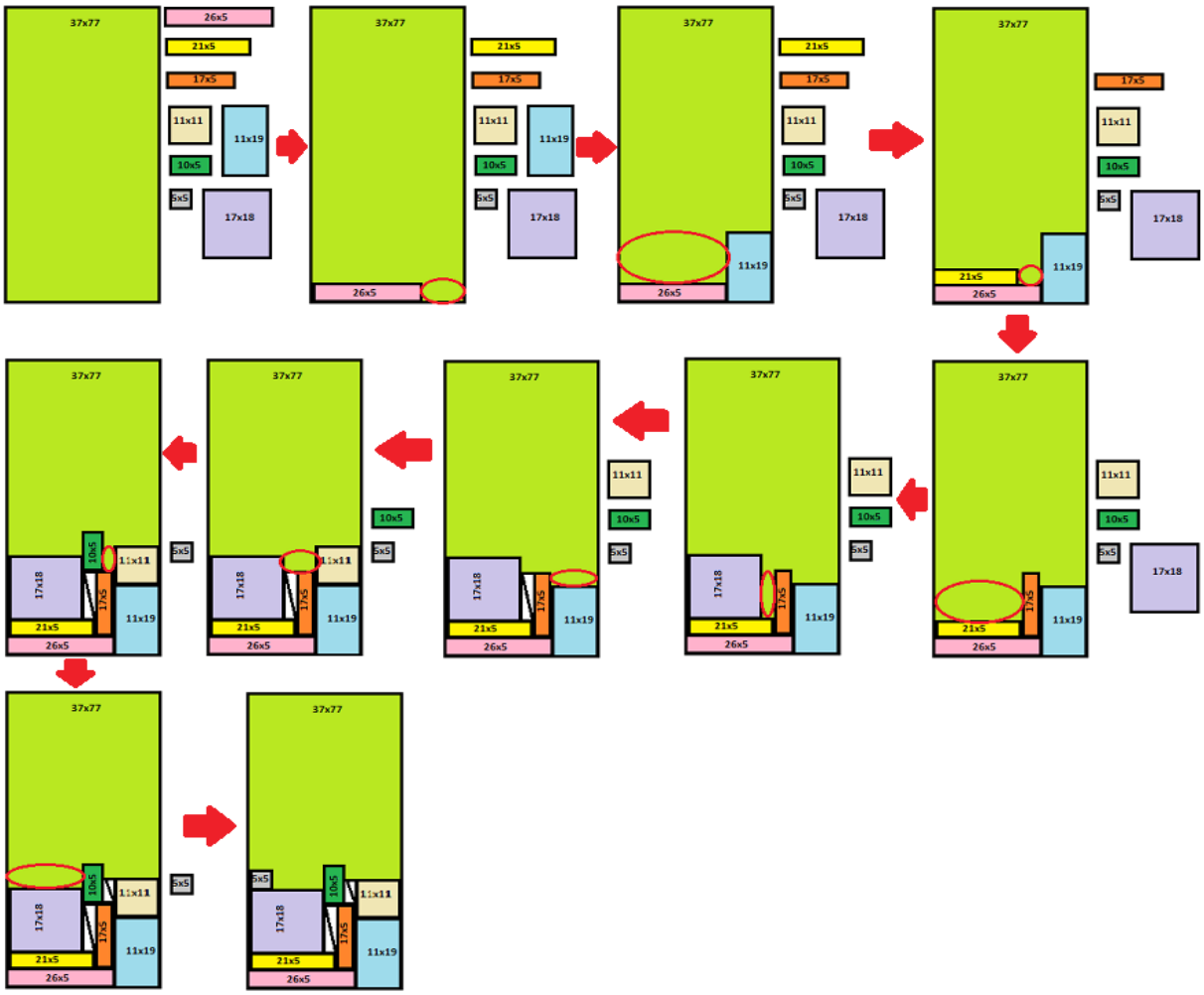


Rysunek 14. Dziura najbardziej z lewej zaznaczona czerwonym rombem.

5.2 Algorytm Best Fit

Algorytm Best Fit jest algorytmem najlepszego dopasowania paczki (wybiera krawędź paczki najlepiej pasującą) do powstałej pustej przestrzeni (dziury) na palecie. Działanie algorytmu, które przedstawiono na Rysunku 15, można opisać w trzech krokach:

- algorytm wyszukuje najgłębszą dziurę na palecie, a następnie wybiera najszerszą z najgłębszych, jeśli ilość najgłębszych dziur jest większa niż jeden,
- wybiera taką paczkę z listy paczek nieułożonych (paczek, które nie znajdują się na palecie), której szerokość bądź długość najlepiej pasuje do szerokości znalezionej dziury (najmniejsza jest różnica między szerokością dziury a konkretnym parametrem paczki, ale nie mniejsza od zera) i wstawia w miejsce danej dziury - w miejsce wskazywane przez współrzędne dziury (lewy, dolny róg paczki),
- powtarza powyższe kroki, aż do zapełnienia palety. Kiedy zapełnienie nastąpi, użytą paletę wrzuca do listy używanych palet, a następnie tworzy nową paletę o takich samych wymiarach, co poprzednia.

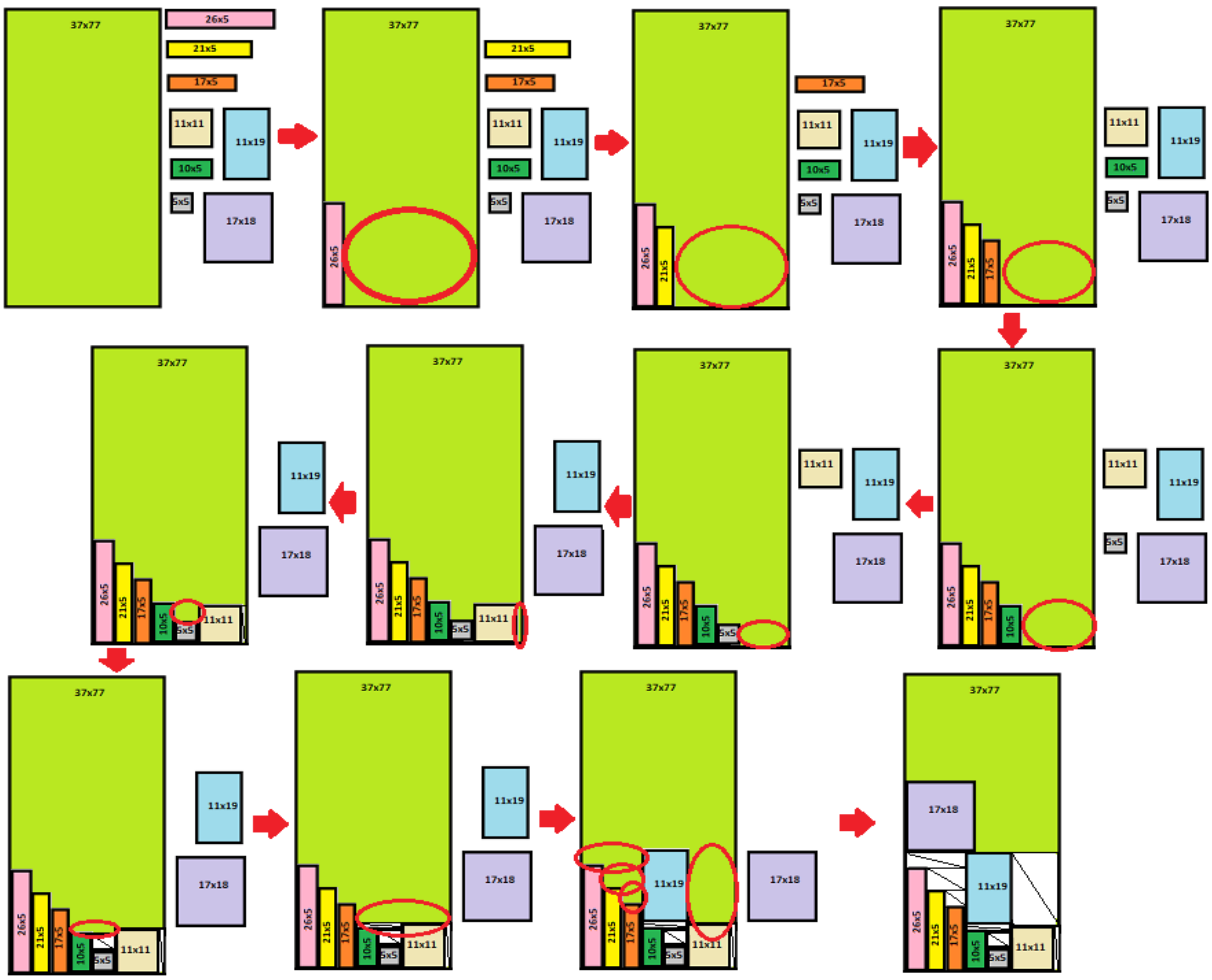


Rysunek 15. Działanie algorytmu Best Fit dla przykładowej instancji problemu.

5.3 Algorytm Worst Fit

Algorytm Worst Fit jest algorytmem najgorszego dopasowania paczki (wybiera krawędź paczki najmniej pasującą) do powstałej pustej przestrzeni (dziury) na palecie. Działanie algorytmu, które przedstawiono na Rysunku 16, można opisać w trzech krokach:

- algorytm wyszukuje najgłębszą dziurę na palecie, a następnie wybiera najszerszą z najgłębszych, jeśli ilość najgłębszych dziur jest większa niż jeden,
- wybiera taką paczkę z listy paczek nieułożonych (paczek, które nie znajdują się na palecie), której szerokość bądź długość najmniej pasuje do szerokości znalezionej dziury (największa jest różnica między szerokością dziury a konkretnym parametrem paczki, ale nie mniejsza od zera) i wstawia w miejsce danej dziury - w miejsce wskazywane przez współrzędne dziury (lewy, dolny róg paczki),
- powtarza powyższe kroki, aż do zapełnienia palety. Kiedy zapełnienie nastąpi, użytą paletę wrzuca do listy używanych palet, a następnie tworzy nową paletę o takich samych wymiarach, co poprzednia.

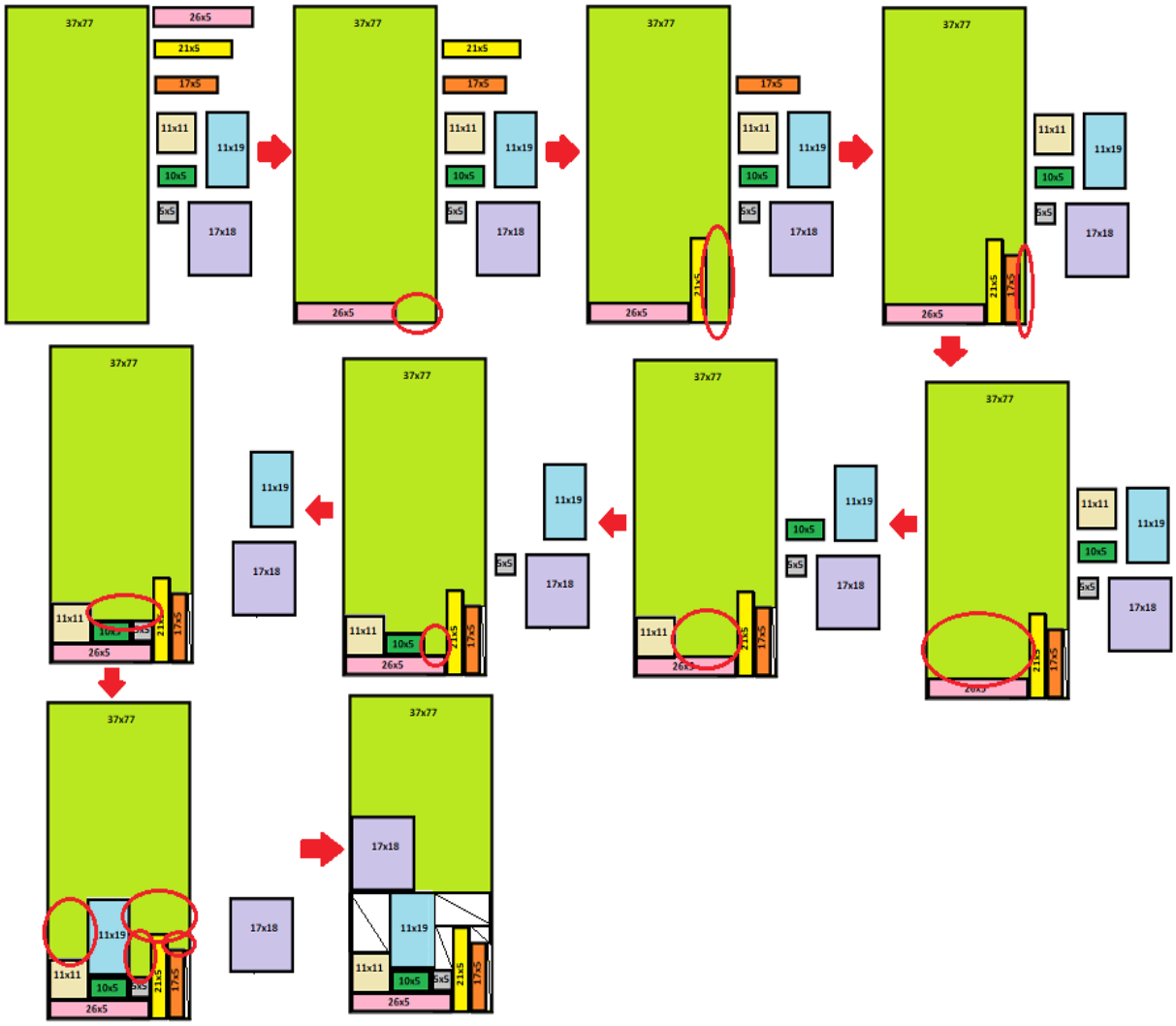


Rysunek 16. Działanie algorytmu Worst Fit dla przykładowej instancji problemu.

5.4 Algorytm First Fit

Algorytm First Fit jest algorytmem polegającym na wkładaniu paczek jedna po drugiej na paletę. Jeśli dana paczka nie mieści się w daną dziurę – algorytm wybiera kolejną paczkę z listy paczek nieułożonych na palecie. Działanie algorytmu, które przedstawiono na Rysunku 17, można opisać w trzech krokach:

- wyszukuje najgłębszą dziurę, a następnie najszerszą z najgłębszych,
- wybiera pierwszą paczkę z listy paczek nieułożonych, której szerokość bądź długość pasuje do szerokości znalezionej dziury i wstawia w miejsce danej dziury - w miejsce wskazywane przez współrzędne dziury (lewy, dolny róg paczki). Jeśli dana paczka nie pasuje – bierze kolejną.
- powtarza powyższe kroki, aż do zapełnienia palety. Kiedy zapełnienie nastąpi, użytą paletę wrzuca do listy używanych palet, a następnie tworzy nową paletę o takich samych wymiarach, co poprzednia.



Rysunek 17. Działanie algorytmu First Fit dla przykładowej instancji problemu.

6. Implementacja i struktury danych

W rozdziale tym opisano szkielet programu, z czego program jest zbudowany oraz jak wygląda jego implementacja. Przedstawiono pseudokody algorytmów wykorzystanych do rozwiązania problemu, a także wyjaśniono struktury danych wykorzystywane przez te algorytmy. W ramach projektu zaimplementowano trzy algorytmy heurystyczne:

- algorytm najlepszego dopasowania Best Fit (Rysunki 18 i 19),
- algorytm najgorszego dopasowania Worst Fit (Rysunki 20 i 21),
- algorytm kolejnego dopasowania – First Fit (Rysunki 22 i 23).

Program został podzielony na kilka modułów, w celu ułatwienia implementacji, poprawienia czytelności kodu oraz większej podatności na przyszłe modyfikacje bądź aktualizacje. W programie wyróżniono pięć klas:

- klasa Punkt,
- klasa Dziura,
- klasa Paczka,
- klasa Paleta,
- klasa Algo.

Działanie programu opiera się na plikach wejściowych oraz wyjściowych. Pliki wejściowe pobierają od użytkownika dane niezbędne do działania programu, natomiast pliki wyjściowe służą do przechowywania wyników.

Na pliki wejściowe składają się pliki zawierające niezbędne dane do stworzenia list:

- palet,
- paczek.

Do kategorii plików wyjściowych należą pliki:

- wynik działania algorytmu Best Fit,
- wynik działania algorytmu Worst Fit,
- wynik działania algorytmu First Fit,
- wyniki działania wszystkich algorytmów, służące do porównań.

W części głównej programu – main – następuje otwieranie istniejących już plików. Pliki są tworzone, jeśli nie istnieją. Niektóre z nich są dostępne tylko w trybie czytania bądź pisania, inne natomiast w obydwu trybach.

Main zawiera funkcję menu, która odpowiada za działanie całego programu. Funkcja menu pobiera pliki oraz listy paczek i palet jako argumenty. W menu znajduje się instrukcja switch odpowiedzialna za wyświetlanie menu w trybie tekstowym widocznym dla użytkownika programu.

6.1 Pseudokody

W podrozdziale przedstawiono pseudokody wykorzystanych algorytmów wyjaśniające oraz obrazujące przebieg ich działania.

6.1.1 Algorytm Best Fit

```
tworzy nową paletę o długości L i szerokości W
jeżeli lista paczek jest pusta to
    należy pobrać listę paczek
w przeciwnym razie
    jeżeli liczba elementów w liście paczek jest większa niż zero i liczba palet jest
    mniejsza niż podana liczba palet to
        szukaj dziury na aktualnej palecie
        jeżeli lista paczek nie jest pusta to
            przeszukuj tę listę w celu znalezienia paczek mieszczących się do
            znalezionej dziury
            jeżeli paczka się mieści w dziurze to
                ustaw flagę możliwości umieszczenia paczki w dziurze na true
                zwiększ licznik paczek pasujących
            koniec warunku
            sprawdź, który bok paczki lepiej pasuje do dziury i jeśli trzeba, to
            obróć paczkę
            sortuj paczki w liście po najdłuższych bokach
            sortuj paczki w liście po fladze możliwości umieszczenia paczki
        koniec warunku
        jeżeli licznik pasujących jest równy zero to
            wypełnij aktualną dziurę bloczkami blokady
        w przeciwnym razie
            jeżeli paczka nie jest umieszczona na żadnej palecie to
```

Rysunek 18. Pseudokod algorytmu Best Fit - część pierwsza.

```
                    wrzuć paczkę do dziury
                koniec warunku
                aktualizuj zajętą powierzchnie palety o położenie paczki
                zmień status paczki na umieszczoną
                przerzuć paczkę z listy paczek nieumieszczonych do listy paczek ułożonych
                wyzeruj flagi możliwości umieszczenia innych paczek
            koniec warunku
        koniec warunku
        wyczyść listę znalezionych dziur
        jeśli bieżąca paleta jest pełna to
            zapisz informacje o palecie do pliku
            zapisz ułożenie paczek na palecie do pliku
            wrzuć paletę do listy palet użytych
            jeżeli pozostały jeszcze jakieś paczki to
                twórz nową paletę o długości L i szerokości W
            koniec warunku
        koniec warunku
        zapisz wynik działania algorytmu do pliku
koniec warunku
```

Rysunek 19. Pseudokod algorytmu Best Fit - część druga.

6.1.2 Algorytm Worst Fit

```
tworzy nową paletę o długości L i szerokości W
jeżeli lista paczek jest pusta to
    należy pobrać listę paczek
w przeciwnym razie
    jeżeli liczba elementów w liście paczek jest większa niż zero i liczba palet jest mniejsza niż podana liczba palet to
        szukaj dziury na aktualnej palecie
        jeżeli lista paczek nie jest pusta to
            przeszukuj tę listę w celu znalezienia paczek mieszczących się do znalezionej dziury
```

Rysunek 20. Pseudokod algorytmu Worst Fit - część pierwsza.

jeżeli paczka się mieści w dziurze **to**

ustaw flagę możliwości umieszczenia paczki w dziurze na true

zwiększ licznik paczek pasujących

koniec warunku

sprawdź, który bok paczki najmniej pasuje do dziury i jeśli trzeba, to obróć paczkę

sortuj paczki w liście po najkrótszych bokach

sortuj paczki w liście po fladze możliwości umieszczenia paczki

koniec warunku

jeżeli licznik pasujących jest równy zero **to**

wypełnij aktualną dziurę bloczkami blokady

w przeciwnym razie

jeżeli paczka nie jest umieszczona na żadnej palecie **to**

wrzuć paczkę do dziury

koniec warunku

aktualizuj zajętą powierzchnie palety o położenie paczki

zmień status paczki na umieszczoną

przerzuć paczkę z listy paczek nieumieszczonych do listy paczek ułożonych

wyzeruj flagi możliwości umieszczenia innych paczek

koniec warunku

koniec warunku

wyczyść listę znalezionych dziur

jeśli bieżąca paleta jest pełna **to**

zapisz informacje o palecie do pliku

zapisz ułożenie paczek na palecie do pliku

wrzuć paletę do listy palet użytych

jeżeli pozostały jeszcze jakieś paczki **to**

twórz nową paletę o długości L i szerokości W

koniec warunku

koniec warunku

zapisz wynik działania algorytmu do pliku

koniec warunku

Rysunek 21. Pseudokod algorytmu Worst Fit - część druga.

6.1.3 Algorytm First Fit

```
tworzy nową paletę o długości L i szerokości W
jeżeli lista paczek jest pusta to
    należy pobrać listę paczek
w przeciwnym razie
    jeżeli liczba elementów w liście paczek jest większa niż zero i liczba palet jest mniejsza niż podana liczba palet to
        szukaj dziury na aktualnej palecie
        jeżeli lista paczek nie jest pusta to
            przeszukuj tę listę w celu znalezienia paczki mieszczącej się w znalezionej dziurze
            jeżeli paczka się mieści w dziurze to
                ustaw flagę możliwości umieszczenia paczki w dziurze na true
                zwiększ licznik paczek pasujących
                sprawdź, które ułożenie paczki pozwala na umieszczenie jej w dziurze i ułóż ją w ten sposób
                sortuj paczki w liście po fladze możliwości umieszczenia paczki
                przerwij szukanie
            koniec warunku
        koniec warunku
        jeżeli licznik pasujących jest równy zero to
            wypełnij aktualną dziurę bloczkami blokady
        w przeciwnym razie
            jeżeli paczka nie jest umieszczona na żadnej palecie to
                wrzuć paczkę do dziury
            koniec warunku
            aktualizuj zajętą powierzchnie palety o położenie paczki
            zmień status paczki na umieszczoną
            przerzuć paczkę z listy paczek nieumieszczonych do listy paczek ułożonych
            wyzeruj flagi możliwości umieszczenia innych paczek
        koniec warunku
    koniec warunku
```

Rysunek 22. Pseudokod algorytmu First Fit – część pierwsza.

```
        wyczyść listę znalezionych dziur
        jeśli bieżąca paleta jest pełna to
                zapisz informacje o palecie do pliku
                zapisz ułożenie paczek na palecie do pliku
                wrzuć paletę do listy palet użytych
                jeżeli pozostały jeszcze jakieś paczki to
                        twórz nową paletę o długości L i szerokości W
                koniec warunku
        koniec warunku
        zapisz wynik działania algorytmu do pliku
koniec warunku
```

Rysunek 23. Pseudokod algorytmu First Fit – część druga.

6.2 Struktura klas

W tym podrozdziale opisano charakterystyki poszczególnych klas oraz jakie ważniejsze pola oraz metody zawierają.

6.2.1 Klasa Punkt

Obiekty tej klasy wykorzystywane są do określenia lewego, dolnego rogu dziur oraz paczek.

Pola:

- współrzędna x punktu, która jest wartością całkowitą,
- współrzędna y punktu, która jest wartością całkowitą.

Metody:

- konstruktor parametryczny – tworzy punkt o podanych parametrach, niepodanie parametrów skutkuje stworzeniem punktu (0, 0),
- metoda zwracająca współrzędną x punktu,
- metoda zwracająca współrzędną y punktu,
- metoda ustawiająca współrzędną x punktu na konkretną wartość – metoda wykorzystywana w aktualizacji współrzędnych,
- metoda ustawiająca współrzędną y punktu na konkretną wartość – metoda wykorzystywana w aktualizacji współrzędnych.

6.2.2 Klasa Dziura

Obiekty tej klasy reprezentują dziury (wolne przestrzenie) znajdujące się na palecie, do których można bądź nie wkładać zdefiniowane paczki. Dziury zostały obszerniej opisane w rozdziale czwartym.

Pola:

- głębokość dziury będąca liczbą całkowitą,
- szerokość dziury będąca liczbą całkowitą,
- współrzędne dziury, które są reprezentowane przez obiekt typu Punkt.

Metody:

- konstruktor domyślny, tworzący dziurę o parametrach zerowych,
- konstruktor parametryczny, tworzący dziurę o podanych parametrach (głębokość, szerokość, współrzędna x, współrzędna y),
- metoda ustawiająca głębokość dziury na podaną wartość,
- metoda ustawiająca szerokość dziury na podaną wartość,
- metoda modyfikująca współrzędne dziury,
- metoda zwracająca głębokość dziury,
- metoda zwracająca szerokość dziury,
- metoda zwracająca współrzędną x dziury,
- metoda zwracająca współrzędną y dziury,
- metoda sortująca dziury po głębokości (od najgłębszej do najpłytszej),
- metoda sortująca dziury po szerokości (od najszerszej do najwęższej),
- metoda usuwająca dziury płytsze niż najgłębsza,
- metoda usuwająca dziury węższe od najszerszej.

6.2.3 Klasa Paczka

Obiekty tej klasy reprezentują fizyczne paczki w postaci prostokątów układanych na palecie. Można je obracać. Dzielą się na paczki umieszczone na jakiejś palecie i na paczki jeszcze niewłożone. Paczki zostały szerzej opisane w poprzednich rozdziałach.

Pola:

- szerokość paczki będąca wartością całkowitą,
- długość paczki będąca wartością całkowitą,
- numer paczki będący wartością całkowitą (numer paczki nie może być równy zero (pusta przestrzeń na palecie) oraz dwa (bloczek blokady),

- powierzchnia paczki będąca zmienną typu zmiennoprzecinkowego,
- flaga obrotu (0 – paczka w takim ustawieniu, w jakim ją pobrano, 1 – paczka obrócona względem położenia początkowego o 90^{o}),
- status paczki określający czy paczka została umieszczona na jakiejś palecie czy nie (0 – nie, 1 – tak),
- flaga mówiąca o tym, czy paczka pasuje w miejsce danej dziury (0 – nie, 1 – tak).

Metody:

- konstruktor parametryczny tworzący paczkę o zdefiniowanych parametrach,
- metoda zwracająca szerokość paczki,
- metoda zwracająca długość paczki,
- metoda zwracająca numer paczki,
- metoda zwracająca powierzchnię paczki,
- metoda ustawiająca szerokość paczki na zadaną wartość,
- metoda ustawiająca długość paczki na zadaną wartość,
- metoda ustawiająca flagę obrotu,
- metoda zmieniająca status paczki,
- metoda ustawiająca flagę mówiącą o tym, czy paczka pasuje w miejsce dziury na 1,
- metoda ustawiająca flagę mówiącą o tym, czy paczka pasuje w miejsce dziury na 0,
- metoda zwracająca obrót paczki,
- metoda zwracająca status paczki,
- metoda zwracająca flagę mówiącą o tym, czy paczka pasuje w miejsce danej dziury.

6.2.4 Klasa Paleta

Obiekty tej klasy reprezentują palety, na których układane są paczki. Przechowuje ilość paczek znajdujących się na palecie, wyszukuje dziury, przechowuje dziury, układa paczki, obraca paczki na palecie w zależności od zastosowanego algorytmu.

Pola:

- długość palety będąca wartością całkowitą,
- szerokość palety będąca wartością całkowitą,

- liczba palet będąca wartością całkowitą,
- tablica dwuwymiarowa rozłożenia paczek na palecie (0 – wolna przestrzeń, 2 – bloczek blokady, inny numer będący większy i różny od 0 oraz większy od 2 – paczka),
- powierzchnia palety będąca zmienną typu zmiennoprzecinkowego,
- powierzchnia zajęta palety będąca zmienną zmiennoprzecinkową,
- obiekt typu Dziura reprezentujący aktualną dziurę,
- lista dziur znajdujących się na palecie,
- lista paczek ułożonych na palecie.

Metody:

- konstruktor parametryczny tworzący paletę o zdefiniowanych parametrach,
- metoda zwracająca długość oraz metoda zwracająca szerokość palety,
- metoda zwracająca ilość paczek znajdujących się na palecie,
- metoda zwracająca powierzchnię palety,
- metoda zwracająca zajętą powierzchnie palety,
- metoda sprawdzająca, czy paleta jest pełna,
- metoda aktualizująca powierzchnię palety o powierzchnię wkładanej paczki,
- metoda szukająca dziury na palecie,
- metoda operująca na liście dziur (sortuje, usuwa dziury),
- metoda wykonująca algorytm Best Fit na palecie,
- metoda wykonująca algorytm Worst Fit na palecie,
- metoda wykonująca algorytm First Fit na palecie,
- metoda zapisująca obraz palety do pliku,
- metoda obracająca paczkę na palecie dla algorytmu Best Fit,
- metoda obracająca paczkę na palecie dla algorytmu Worst Fit,
- metoda zapisująca informacje o palecie do pliku.

6.2.5 Klasa Algo

Obiekty tej klasy przechowują wyniki uzyskane przez poszczególne algorytmy. Następnie dane przechowywane przez obiekty tego typu służą do porównywania wyników uzyskanych przez wykonane algorytmy.

Pola:

- string przechowujący nazwę algorytmu,
- tablica dwuwymiarowa przechowująca liczbę wykorzystanych palet, ogólną zajętość powierzchni na paletach oraz liczbę paczek znajdujących się na wykorzystanych paletach.

Metody:

- konstruktor parametryczny tworzący obiekt o zdefiniowanym parametrze (nazwa),
- metoda zwracająca nazwę,
- metoda zwracająca liczbę wykorzystanych palet,
- metoda zwracająca całkowitą zajętość powierzchni na paletach,
- metoda zwracająca liczbę zapakowanych paczek,
- metoda pobierająca i zapisująca do tablicy dwuwymiarowej liczbę wykorzystanych palet, całkowitą zajętość powierzchni na paletach oraz liczbę paczek zapakowanych,
- metoda sortująca po liczbie wykorzystanych palet od najmniejszej ich ilości do największej,
- metoda sortująca po zajętości powierzchni (od najmniejszej wartości do największej),
- metoda sortująca po liczbie zapakowanych paczek (od największej do najmniejszej),
- metoda usuwająca obiekty przechowujące większą liczbę palet niż najmniejsza uzyskana,
- metoda usuwająca obiekty przechowujące większą wartość zajętości powierzchni od najmniejszej uzyskanej,
- metoda usuwająca obiekty przechowujące mniejszą ilość paczek niż maksymalna.

6.3 Pliki

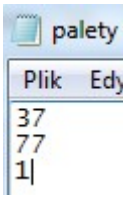
W rozdziale tym opisano strukturę, działanie, przeznaczenie plików oraz obsługę plików wejściowych. Wszystkie pliki są formatu .txt.

6.3.1 Pliki wejściowe

Pliki wejściowe przechowują i podają zdefiniowane przez użytkownika parametry do programu. Pliki te są niezbędne do uzyskania wyników działania algorytmów.

5.3.1.1 Plik palety.txt

Plik ten zawiera parametry palet, które mają zostać wykorzystane w programie. Niepodanie żadnych danych do pliku skutkuje niewykonaniem się programu. Podanie parametrów niezgodnych z wymaganiami wystosuje odpowiedni komunikat. Należy pamiętać, aby kursor po wpisaniu danych był ustawiony na miejscu po ostatnim parametrze.



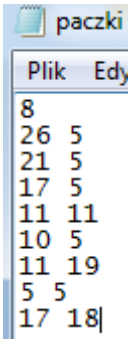
Rysunek 24. Przykładowy plik palety.txt.

Pierwszy parametr pliku podanego na Rysunku 24 definiuje szerokość palety. Druga liczba podana w pliku reprezentuje długość palety, natomiast trzecia liczba określa liczbę palet, jaka ma być wykorzystana w programie. Przed uruchomieniem programu należy stworzyć plik palety.txt i podać trzy parametry. Kursor ustawić tak, jak jest przedstawione na powyższym rysunku – nie może być żadnego pustego miejsca, po tym miejscu; jeżeli takowe istnieje, to należy je usunąć przy pomocy klawisza „backspace”. Wpisywać tylko wartości całkowite. Podanie parametrów paczki jako litery zostanie wyłapane przez program, który wyświetli odpowiedni komunikat.

6.3.1.2 Plik paczki.txt

Plik ten przechowuje dane paczek, które następnie zostaną wprowadzone do programu. Niepodanie żadnych danych do pliku skutkuje niewykonaniem się programu. Podanie paczek niezgodnych z wymaganiami, tj. paczki o wartościach ujemnych bądź większych od parametrów palety nie zostaną pobrane. Należy pamiętać, aby kursor po

wpisaniu danych był ustawiony na miejscu po ostatnim parametrze. Podanie parametrów paczki jako litery zostanie wyłapane przez program, który wyświetli odpowiedni komunikat.



Rysunek 25. Przykładowy plik paczki.txt.

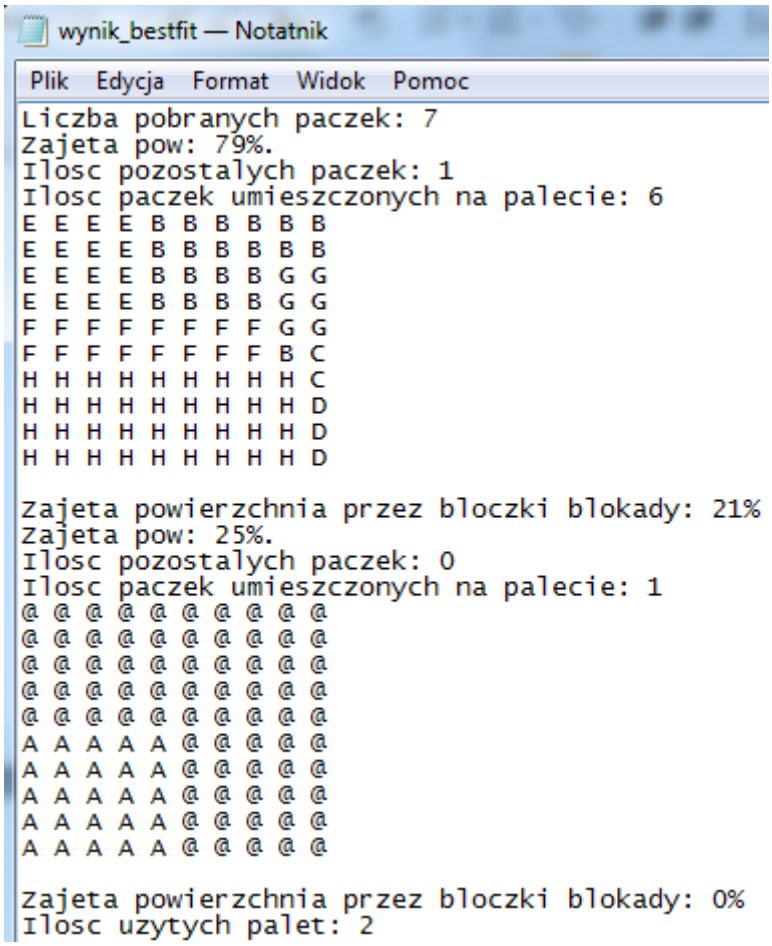
Pierwszy parametr pliku podanego na Rysunku 25 definiuje liczbę paczek, drugi szerokość, natomiast trzeci reprezentuje jej wysokość. Liczba paczek jest równa ilości podanych paczek w pliku minus paczki niespełniające wymagań (szerokość lub długość mniejsza bądź równa zero lub większa od parametrów palety). Przed uruchomieniem programu należy stworzyć plik palety.txt i podać trzy parametry. Kursor ustawić tak, jak jest przedstawione na powyższym rysunku – nie może być żadnego pustego miejsca, po tym miejscu - jeżeli takowe istnieje, to należy je usunąć przy pomocy klawisza „backspace”. Wpisywać tylko wartości całkowite. Podanie parametrów palety jako litery zostanie wyłapane przez program, który wyświetli odpowiedni komunikat.

6.3.2 Pliki wyjściowe

W plikach tych przechowywane są wyniki uzyskane poprzez zastosowanie konkretnych algorytmów. Na wyniki te składają się liczba pobranych paczek, zajęta powierzchnia przez paczki wyrażona w procentach, procentowa zajętość powierzchni palety przez same bloczki blokady, ilość pozostałych paczek po upakowaniu danej palety, ilość paczek umieszczonych na danej palecie oraz liczba użytych palet oraz rozłożenie paczek na palecie. W podrozdziale tym zostaną przedstawione i omówione dwa przykładowe pliki wynikowe.

Do plików wyjściowych należą pliki:

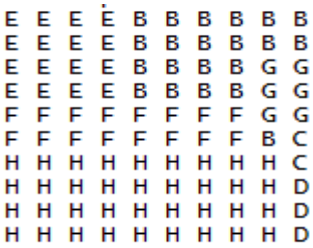
- wynik_bestfit.txt – zawiera wyniki działania algorytmu Best Fit,
- wynik_worstfit.txt – zawiera wyniki działania algorytmu Worst Fit,
- wynik_Firstfit.txt – zawiera wyniki działania algorytmu First Fit,
- wyniki.txt – zawiera wyniki wszystkich algorytmów niezbędne do porównania.



Rysunek 26. Przykładowy plik wynik_bestfit.txt.

Omówienie poszczególnych elementów pliku wyjściowego wynik_bestfit.txt (Rysunek 26):

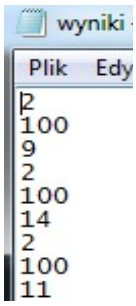
- liczba pobranych paczek – jest to liczba paczek, które zostały pobrane przez program. Do liczby tej nie wliczają się paczki niespełniające wymagań,
- zajeta pow – obszar palety zajęty przez paczki wyrażony w procentach,
- ilosc pozostalych paczek – liczba paczek pozostałych po zapełnieniu danej palety,
- ilosc paczek umieszczonych na palecie – liczba paczek znajdujących się na palecie, zajmujących część powierzchni tej palety,
- zajeta powierzchnia przez bloczki blokady – obszar palety zajęty przez bloczki blokady wyrażony w procentach,
- ilosc uzytych palet – jeśli użytkownik podał liczbę palet większą niż wymagana do upakowania wszystkich paczek, to jest to ilość użytych palet do zapakowania wszystkich paczek. W przeciwnym razie jest to liczba palet wykorzystanych do upakowania części paczek.



Rysunek 27. Przykładowe upakowanie palety.

Opis symboli wykorzystywanych przy wizualizacji palet przedstawionej na Rysunku 27. Każdy symbol jest rozmiarów 1x1:

- symbol @ - symbol ten reprezentuje pustą przestrzeń znajdującą się na palecie. Dziura składa się z co najmniej jednego takiego symbolu,
- symbol B – symbol ten reprezentuje bloczek blokady. Bloczki blokady zostały wytłumaczone w rozdziale czwartym. Wystąpienie bloczków blokady na palecie informuje o braku paczki w danym miejscu, ponieważ nie było takiej paczki, która mogłaby się zmieścić miejsce wskazywane przez daną dziurę,
- pozostałe symbole – reprezentują paczki. Paczka składa się z co najmniej jednego takiego samego symbolu, np. H.



Rysunek 28. Przykładowy plik wyniki.txt.

Omówienie poszczególnych elementów pliku wyjściowego wyniki.txt (Rysunek 28):

- pierwszy, czwarty oraz siódmy wiersz – liczba palet wykorzystanych przez odpowiednio algorytm Best Fit, Worst Fit, First Fit,
- wiersze drugi, piąty oraz ósmy – całkowita powierzchnia zajęta przez paczki oraz bloczki blokady po wykonaniu się odpowiednio algorytmu Best Fit, Worst Fit, First Fit. Wartość ta jest liczona jako suma powierzchni paczek upakowanych na wykorzystane palety oraz powierzchni wszystkich bloczków blokady znajdujących się na wykorzystanych paletach,
- wiersze trzeci, szósty oraz dziewiąty – liczba ułożonych paczek na paletach.

7. Aplikacja i opis technologii

W rozdziale tym przedstawiono środowisko Microsoft Visual Studio 2008, przy użyciu którego powstała aplikacja rozwiązująca problem załadunku w 2D oraz opisano sposób w jaki działa aplikacja oraz jak poprawnie z niej korzystać. Zawarto przykłady obrazujące zachowanie programu dla różnych przypadków, dla różnych działań podejmowanych przez użytkownika.

Aplikacja została napisana, uruchamiana oraz testowana na komputerze ASUS z wgranym systemem Windows 7 Premium Service Pack 1, kartą graficzną GeForce GT540M, procesorem Intel Core i7-2630QM 2.0 GHz oraz z 4GB pamięci RAM.

7.1 Microsoft Visual Studio 2008

Microsoft Visual Studio 2008 jest to zintegrowane środowisko programistyczne (IDE), które służy do tworzenia oprogramowania konsolowego oraz graficznego przy użyciu bibliotek graficznych, np. SDL czy OpenGL. MVS2008 pozwala na pisanie programów w różnych językach programowania (C#, Visual Basic czy C++), dzięki zawartych w nim zestawom narzędzi programistycznych.

Cechuje się szerokim wachlarzem możliwości, który jednak zależy od aktualnie posiadanej edycji, z których najbardziej rozbudowaną i zaawansowaną jest edycja Visual Studio Team System. W pracy wykorzystano edycję Visual Studio Professional, która udostępnia opcje zdalnego debugowania, co ułatwia implementację i szukanie potencjalnych błędów w powstającym programie. Środowisko MVS umożliwia zainstalowanie pożądanych pakietów językowych, które są udostępniane jako bezpłatne dodatki.

Jak wcześniej wspomniano, aplikacja powstała w oparciu o edycję Visual Studio Professional zawierającą pakiet Microsoft Visual C++, co pozwoliło na implementację w języku C++.

7.2 Aplikacja

W tym podpunkcie przedstawiono poprawny sposób korzystania z aplikacji, opis menu i opcji, jakie się tam znajdują. Wyjaśniono w jaki sposób czytać uzyskane wyniki i w jakim miejscu na komputerze program je generuje. Na koniec przedstawiono przykładową instancję problemu oraz jej rozwiązanie.

7.2.1 Warunki poprawnego korzystania z aplikacji

- program należy umieścić w dowolnym folderze na komputerze,
- w tym samym folderze należy stworzyć plik „paczki.txt” oraz wypełnić go poprawnymi danymi, czyli liczbami całkowitymi występującymi w parach oddzielonymi za pomocą spacji. Przykładowy plik został przedstawiony w rozdziale piątym oraz dalszej części tego rozdziału,
- w tym samym folderze należy stworzyć plik „palety.txt” oraz wypełnić go poprawnymi danymi, czyli liczbami całkowitymi występującymi pojedynczo. Przykładowy plik został przedstawiony w rozdziale piątym oraz dalszej części tego rozdziału,
- w obu plikach należy umieścić kursor bezpośrednio po ostatniej wartości, jak zostało to przedstawione w rozdziale piątym oraz w dalszej części tego rozdziału,
- pliki wynikowe są generowane przez program w tym samym folderze, w którym się znajduje. Czytanie wyników zostało omówione w rozdziale piątym.

7.2.2 Menu

Rysunek 29. Menu aplikacji.

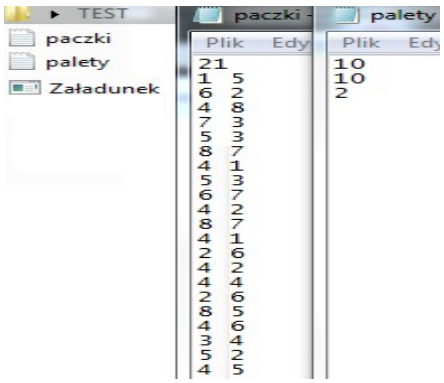
Jak widać na powyższym rysunku (Rysunek 29), menu składa się z siedmiu różnych opcji. Aby wybrać którąś z opcji należy wybrać jej numer poprzez wpisanie danej wartości z klawiatury. Podanie wartości nieznajdującej się w menu wyboru, spowoduje ponowne wyświetlenie menu bez żadnych konsekwencji.

Opis dostępnych opcji:

- „pobierz palety z pliku” – pobiera palety z pliku „palety.txt”. Jeśli taki plik nie istnieje, to program zgłosi błąd i wtedy należy stworzyć taki plik w tym samym folderze, w którym znajduje się aplikacja, a następnie wypełnić go danymi,
- „pobierz paczki z pliku” – pobiera paczki z pliku „paczki.txt”. Jeśli taki plik nie istnieje, to program zgłosi błąd i wtedy należy stworzyć taki plik w tym samym folderze, w którym znajduje się aplikacja, a następnie wypełnić go danymi,
- „algorytm Best Fit” – wybór tej opcji skutkuje wykonaniem algorytmu Best Fit dla wprowadzonych danych z pliku,
- „algorytm Worst Fit” – wybór tej opcji skutkuje wykonaniem algorytmu Worst Fit dla wprowadzonych danych z pliku,
- „algorytm First Fit” – wybór tej opcji skutkuje wykonaniem algorytmu First Fit dla wprowadzonych danych z pliku,
- „porownanie wynikow” – opcja ta porównuje wyniki uzyskane przez wykonane algorytmu. Aby uzyskać wynik zwracany przez tę opcję, należy wykonać wszystkie algorytmy – w przeciwnym wypadku zostanie wystosowany odpowiedni komunikat. Poprawne wykonanie się zadań wywołanych przez tę opcje powoduje zakończenie programu,
- „wyjscie” – kończy działanie programu.

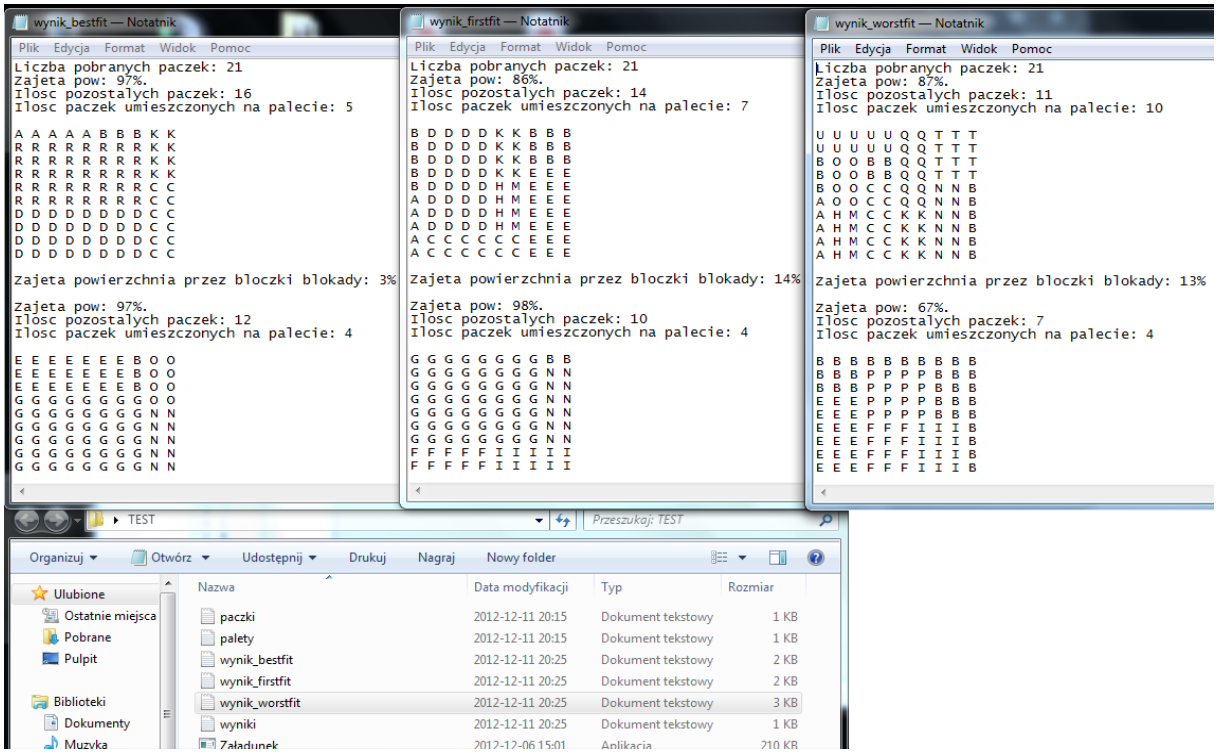
7.2.3 Przykładowe działanie aplikacji

Aplikację umieszczono w folderze o nazwie „TEST”, w którym stworzono dwa pliki: paczki.txt oraz palety.txt i wypełniono je danymi spełniającymi wymagania poprawności. Wynik takiego działania zaprezentowano na Rysunku 30.

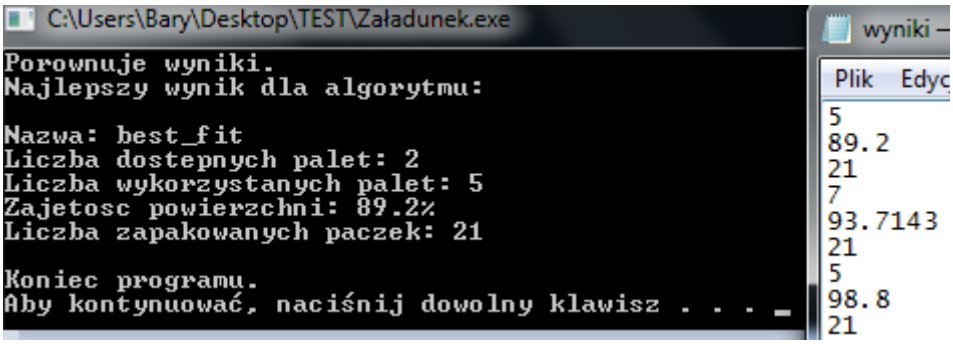


Rysunek 30. Folder TEST oraz pliki wraz z ich zawartościami.

Następnie uruchomiono aplikację poprzez podwójne kliknięcie lewym przyciskiem myszy na jej ikonkę. Kolejnymi krokami było wczytanie danych z plików „palety.txt” oraz „paczki.txt”, a następnie wykonanie po kolei wszystkich algorytmów. Wykonanie się jednego algorytmu powoduje wyzerowanie listy wczytanych paczek, więc po działaniu któregokolwiek z algorytmów trzeba ponownie wczytać paczki z pliku – wybrać opcję o numerze 2. Po wykonaniu powyższych kroków, program zapisał wyniki uzyskane po działaniu algorytmów do konkretnych plików, co pokazuje Rysunek 31. Porównanie uzyskanych wyników przedstawia Rysunek 32.



Rysunek 31. Folder TEST oraz jego pliki po wykonaniu algorytmów.



Rysunek 32. Porównanie otrzymanych wyników.

8. Badania eksperymentalne

Celem przeprowadzonych badań eksperymentalnych była ocena jakości rozwiązań generowanych przez wykorzystane algorytmy: Best Fit, Worst Fit oraz First Fit, a także wyłonienie najlepszego algorytmu w sensie średnim.

Algorytmy badano porównując zwracane przez nie wyniki, tj. zajętość powierzchni wykorzystanych palet przez paczki (z.p.p.p), zajętość powierzchni wykorzystanych palety przez bloczki blokady (z.p.p.b.b), liczba palet wykorzystanych (maksymalna ilość podawana jest przez użytkownika, l.p.w), liczba palet potrzebnych do upakowania wszystkich paczek (l.p.p) oraz liczba paczek upakowanych na użyte palety (l.p.u).

Testy prowadzone były na komputerze o takiej samej specyfikacji, jak ta podana w rozdziale szóstym, czyli na komputerze ASUS z wgranym systemem Windows 7 Premium Service Pack 1, kartą graficzną GeForce GT540M, procesorem Intel Core i7-2630QM 2.0 GHz oraz z 4GB pamięci RAM.

Algorytmy badano dla pięciu różnych zestawów danych testowych:

- pierwszy zestaw – jedna paleta 10x10, dziesięć paczek (tabela 1),
- drugi zestaw – dwie palety 10x10, dwadzieścia paczek (tabela 2),
- trzeci zestaw – pięć palet 10x10, trzydzieści paczek (tabela 3),
- czwarty zestaw – osiem palet 10x10, czterdzieści paczek (tabela 4),
- piąty zestaw – dziesięć palet 10x10, pięćdziesiąt paczek (tabela 5).

Najlepszym algorytmem był ten algorytm, który uzyskał najmniejszy stosunek użytych palet do palet dostępnych. Jeśli istniało kilka algorytmów o takim samym stosunku wykorzystanych palet do załadunku podanych paczek do palet dostępnych, to lepszym algorytmem był ten, który ekonomiczniej wykorzystał powierzchnię tych palet, czyli mniejsza była średnia zajętość palet przez paczki.

8.1 Dane

Badania zostały przeprowadzone dla pięciu przykładowych zestawów danych testowych:

- dziesięć paczek,

PALETY (W, L, n)	PACZKI (w, l)
(10, 10, 1)	(4, 6)
	(7, 2)
	(1, 5)
	(4, 2)
	(3, 2)
	(1, 6)
	(6, 2)
	(4, 4)
	(1, 1)
	(2, 5)

Tabela 1. Pierwszy zestaw danych testowych.

- dwadzieścia paczek,

PALETY (W, L, n)	PACZKI (w, l)
(10, 10, 2)	(4, 6)
	(7, 2)
	(1, 5)
	(4, 2)
	(3, 2)
	(1, 6)
	(6, 2)
	(4, 4)
	(1, 1)
	(2, 5)
	(4, 2)
	(1, 4)
	(3, 1)
	(2, 3)
	(6, 7)
	(5, 3)
	(4, 5)
	(3, 2)
	(6, 8)
	(1, 3)

Tabela 2. Drugi zestaw danych testowych.

- trzydzieści paczek,

PALETY (W, L, n)	PACZKI (w, l)
(10, 10, 5)	(4, 6)
	(7, 2)
	(1, 5)
	(4, 2)
	(3, 2)
	(1, 6)
	(6, 2)
	(4, 4)
	(1, 1)
	(2, 5)
	(4, 2)
	(1, 4)
	(3, 1)
	(2, 3)
	(6, 7)
	(5, 3)
	(4, 5)
	(3, 2)
	(6, 8)
	(1, 3)
	(2, 4)
	(2, 1)
	(3, 5)
	(6, 7)
	(4, 3)
	(5, 5)
	(4, 2)
	(1, 2)
	(2, 2)
	(4, 2)

Tabela 3. Trzeci zestaw danych testowych.

- czterdzieści paczek,

PALETY (W, L, n)	PACZKI (w, l)
(10, 10, 8)	(4, 6)
	(7, 2)
	(1, 5)
	(4, 2)
	(3, 2)
	(1, 6)
	(6, 2)
	(4, 4)
	(1, 1)
	(2, 5)
	(4, 2)
	(1, 4)
	(3, 1)
	(2, 3)
	(6, 7)
	(5, 3)
	(4, 5)
	(3, 2)
	(6, 8)
	(1, 3)
	(2, 4)
	(2, 1)
	(3, 5)
	(6, 7)
	(4, 3)
	(5, 5)
	(4, 2)
	(1, 2)
	(2, 2)
	(4, 2)
	(1, 4)
	(3, 2)
	(5, 6)
	(4, 2)
	(4, 5)
	(2, 1)
	(7, 8)
	(5, 3)
	(1, 2)
	(5, 2)

Tabela 4. Czwarty zestaw danych testowych.

- pięćdziesiąt paczek,

PALETY (W, L, n)	PACZKI (w, l)
(10, 10, 10)	(4, 6)
	(7, 2)
	(1, 5)
	(4, 2)
	(3, 2)
	(1, 6)
	(6, 2)
	(4, 4)
	(1, 1)
	(2, 5)
	(4, 2)
	(1, 4)
	(3, 1)
	(2, 3)
	(6, 7)
	(5, 3)
	(4, 5)
	(3, 2)
	(6, 8)
	(1, 3)
	(2, 4)
	(2, 1)
	(3, 5)
	(6, 7)
	(4, 3)
	(5, 5)
	(4, 2)
	(1, 2)
	(2, 2)
	(4, 2)
	(1, 4)
	(3, 2)
	(5, 6)
	(4, 2)
	(4, 5)
	(2, 1)
	(7, 8)
	(5, 3)
	(1, 2)
	(5, 2)
	(5, 6)
	(1, 2)
	(4, 3)
	(5, 7)
	(2, 3)
	(5, 3)
	(6, 8)
	(4, 5)
	(1, 8)
	(2, 4)

Tabela 5. Piąty zestaw danych testowych.

8.2 Wyniki

W tym podrozdziale przedstawiono w tabelach oraz omówiono uzyskane wyniki dla poszczególnych algorytmów. Czerwona czcionka oznacza najlepszy wynik.

	Best Fit				
	z.p.p.p (1 - max)	z.p.p.b.b (1 - max)	l.w.p	l.p.p	l.p.u
Pierwszy zestaw	0,8600	0,1400	1	2	9
Drugi zestaw	0,9950	0,0050	2	3	15
Trzeci zestaw	0,9575	0,0425	4	4	30
Czwarty zestaw	0,8933	0,1067	6	6	40
Piąty zestaw	0,9000	0,1000	8	8	50
Suma	4,6058	0,3942	21	23	144
MAX	5	5	26		150
Średnia	0,9212	0,0788	0,8077		0,9600

Tabela 6. Wyniki uzyskane dla algorytmu Best Fit.

	Worst Fit				
	z.p.p.p (1 - max)	z.p.p.b.b (1 - max)	l.w.p	l.p.p	l.p.u
Pierwszy zestaw	0,6200	0,3800	1	2	8
Drugi zestaw	0,8350	0,1650	2	4	18
Trzeci zestaw	0,6820	0,3180	5	6	29
Czwarty zestaw	0,6000	0,4000	8	9	39
Piąty zestaw	0,6160	0,3840	10	12	48
Suma	3,3530	1,6470	26	33	142
MAX	5	5	26		150
Średnia	0,6706	0,3294	1,0000		0,9467

Tabela 7. Wyniki uzyskane dla algorytmu Worst Fit.

	First Fit				
	z.p.p.p (1 - max)	z.p.p.b.b (1 - max)	l.w.p	l.p.p	l.p.u
Pierwszy zestaw	0,9200	0,0800	1	2	9
Drugi zestaw	0,8700	0,1300	2	4	17
Trzeci zestaw	0,7160	0,2840	5	6	29
Czwarty zestaw	0,7657	0,2343	7	7	40
Piąty zestaw	0,8000	0,2000	9	9	50
Suma	4,0717	0,9283	24	28	145
MAX	5	5	26		150
Średnia	0,8143	0,1857	0,9231		0,9667

Tabela 8. Wyniki uzyskane dla algorytmu First Fit.

Najgorsze wyniki uzyskano dla algorytmu Worst Fit (Tabela 7), który nie dość, że wykorzystywał największą ilość dostępnych palet, to jeszcze upakowywał najmniej ekonomicznie, co prowadziło do zapakowania mniejszej liczby paczek na użyte palety niż w przypadku innych algorytmów. Jednakże dla zestawu drugiego uzyskał najlepszy wynik w liczbie upakowanych paczek na użyte palety, ale aby upakować wszystkie paczki, potrzebowałby więcej kontenerów niż algorytm Best Fit.

Algorytm Best Fit (Tabela 6) potrzebował najmniejszej liczby palet do upakowania wszystkich dostępnych paczek. Stosunek liczby paczek zapakowanych do dostępnych nie był największy, ale stosunek liczby użytych palet do dostępnych był najmniejszy - algorytm First Fit (Tabela 8) zapakował jedną paczkę więcej kosztem trzech palet więcej, co prowadzi do wniosku, że algorytm Best Fit wykorzystując taką samą liczbę palet, jak algorytm First Fit, zapakowałby jeszcze więcej paczek. Algorytm Best Fit najgorzej zachowuje się przy małej ilości palet, mając do dyspozycji wiele dużych paczek, ponieważ wkłada najpierw te większe (najlepiej pasujące do szerokości pustej palet). Liczba użytych bloczków blokady także jest najmniejsza w przypadku algorytmu Best Fit, co świadczy o dużej ekonomiczności sposobu układania przez niego paczek.

9. Podsumowanie

W pracy przedstawiono trzy wybrane algorytmy wykorzystywane do rozwiązywania problemu załadunku w wersji 2D. Zaimplementowane algorytmy Best Fit, Worst Fit oraz First Fit cechują się prostotą, szybkością działania oraz zwracaniem zadowalających wyników.

Napisana aplikacja jest aplikacją dosyć prostą, którą można by w przyszłości rozbudować o kompletny interfejs graficzny, wykorzystując biblioteki graficzne, np. SDL dla problemu 2D czy OpenGL dla problemu 3D.

W dzisiejszych czasach załadunek pojazdów jest głównie rozwiązywany jako problem trójwymiarowy. Jednak jest to problem cięższy do zaimplementowania ze względu na bardziej rozbudowane mechanizmy przeszukiwania oraz podziału przestrzeni, a także same algorytmy rozwiązujące problem są dużo bardziej skomplikowane.

Algorytm Worst Fit można by zoptymalizować poprzez wybieranie dziur nie najszerszych, a najwęższych, dzięki czemu minimalizowano by marnowanie przestrzeni.

Algorytmy Best Fit, Worst Fit oraz First Fit mogą być skutecznie zaadaptowane do problemu załadunku 3D – należy je odpowiednio zmodyfikować, są także wykorzystywane w zarządzaniu pamięcią.

Programy optymalizujące załadunek znajdują zastosowanie w transporcie oraz dystrybucji, m.in. dzięki opcjom wyświetlania układu upakowania, co oszczędza wiele czasu oraz pieniędzy.

Z zastosowanych trzech algorytmów, najlepszym w sensie średnim okazał się algorytm najlepszego dopasowania, czyli algorytm Best Fit, natomiast najmniej efektywnym algorytmem okazał się algorytm najgorszego dopasowania – Worst Fit.

Spis rysunków

Spis tabel

Bibliografia

1: Adam Stawowy, Algorytm hybrydowy dla problemu pakowania, [w:] Współczesne problemy zarządzania przedsiębiorstwem, Wydział Zarządzania AGH, Kraków, str. 152-158, 2000.
2: Andrea Lodi, Alghorithms for Two-Dimensional Bin Packing and Assignment Problems [online], University of Bologna, Italy, http://tinyurl.com/ctoxmdp, 2000.
3: Matt Perdeck, Fast Optimizing Rectangle Packing Algorithm for Building CSS Sprites [online], http://tinyurl.com/blr7tde, 2011.
4: Alan Kaminsky, Heaps and Garbage Collection - Lecture Notes [online], Department of Computer Science RIT, Rochester, http://tinyurl.com/brlpfce, 2001.

Dodatek

Dodatek zawiera płytę CD z zawartością:

1. wersja elektroniczna niniejszego dokumentu,
2. źródłowa wersja aplikacji,
3. wykonawcza wersja aplikacji wraz z przykładowymi plikami wejściowymi.