RÓWNOLEGŁE ALGORYTMY POPULACYJNE DLA TRÓJWYMIAROWEGO PROBLEMU PAKOWANIA

Wojciech BOŻEJKO, Łukasz KACPRZAK, Mieczysław WODECKI

Streszczenie: Praca rozważa wariant problemu trójwymiarowego pakowania (3D-BPP),w którym wewnątrz pojedynczego kontenera należy umieścić ładunek o maksymalnej objętości. Do rozwiązania problemu użyte zostały dwa równoległe algorytmy populacyjne, opierające się na oddzielnej ewolucji współpracujących ze sobą podpopulacji możliwych rozwiązań. Eksperymenty obliczeniowe mają na celu porównanie jakości rozwiązań dostarczonych przez oba algorytmy w określonym czasie oraz zbadanie wpływu zrównoleglenia na ich wartość.

Słowa kluczowe: problem pakowania, MPI, algorytm równoległy, algorytm ewolucyjny, lokalne przeszukiwanie

1. Wprowadzenie

Trójwymiarowy problem pakowania (ang. *three-dimensional Bin packing problem*, w skrócie: 3D-BPP) stanowi istotne zagadnienie przemysłowe oraz posiada szereg zastosowań praktycznych. W wariancie maksymalizacji, problem sprowadza się do wypełnienia wolnej przestrzeni wewnątrz pojedynczego prostopadłościanu (ang. *bin*, dalej nazywanego kontenerem), pudełkami (ang. *box*) o jak największej objętości. Dzięki właściwemu zarządzaniu dostępną przestrzenią możliwa jest redukcja kosztów, związanych z magazynowaniem towarów oraz ich transportem. Trójwymiarowy problem pakowania jest trudnym zagadnieniem optymalizacyjnym, zaliczającym się do klasy problemów *NP-trudnych*.

Do rozwiązania problemu wykorzystane zostały dwa równoległe algorytmy populacyjne, oparte na strategii ewolucyjnej: algorytm hybrydowy, wykorzystujący metody lokalnego przeszukiwania (ang. *local search*) oraz algorytm ewolucyjny z dodatkowym operatorem mutacji. Równoległość przedstawionych algorytmów polega na podziale populacji możliwych rozwiązań na oddzielnie ewoluujące podpopulacje. Każda podpopulacja tworzona jest niezależnie od pozostałych i komunikuje się z innymi podpopulacjami w celu wymiany części rozwiązań. Umożliwia to dywersyfikację własnych osobników oraz stopniowe dostosowywanie się do pozostałych podpopulacji. Eksperymenty obliczeniowe mają na celu porównanie jakości rozwiązań dostarczonych przez oba algorytmy w określonym czasie oraz zbadanie wpływu zrównoleglenia na ich wartość.

Dalsza część pracy posiada podaną organizację. Rozdział 2 zawiera podstawowe informacje dotyczące algorytmów ewolucyjnych, w Rozdziale 3 przybliżone zostały wybrane pozycje literaturowe. Rozdział 4 zawiera definicję problemu, opisuje przyjęte założenia i uwzględnione ograniczenia. Metoda rozwiązania została przedstawiona w Rozdziale 5. Rozdział 6 zawiera wyniki przeprowadzonych badań eksperymentalnych, w Rozdziale 7 zamieszczono wnioski oraz wskazano kierunek dalszego rozwoju.

2. Algorytmy ewolucyjne

Algorytmy ewolucyjne należą do grupy algorytmów metaheurystycznych, swoje działanie wzorują na zjawisku ewolucji występującym w biologii.

Przykładem algorytmu ewolucyjnego jest, przedstawiony po raz pierwszy w pracy Holland [9], algorytm genetyczny. Pracuje on w oparciu o populację, będącą zbiorem możliwych rozwiązań danego problemu. Pojedyncze rozwiązanie nazywane jest osobnikiem. Każdy osobnik posiada chromosom, składający się z genów, które reprezentują konkretne cechy danego rozwiązania i jednoznacznie je określają. Miarą jakości osobnika jest wartość funkcji adaptacji (przystosowania) i najczęściej jest nią wartość funkcji celu.

Działanie klasycznego algorytmu genetycznego można podzielić na pięć etapów: Inicjalizację, Selekcję, Wybór rodziców, Krzyżowanie i opcjonalną Mutację. W kroku Inicjalizacji tworzona jest populacja początkowa, stanowiąca pierwszą generację osobników. Następujące po niej etapy wykonywane są do momentu spełnienia warunków stopu. Głównym zadaniem Selekcji jest określenie, które osobniki przetrwają do kolejnej generacji (iteracji). Odbywa się to poprzez ocenę każdego rozwiązania, ustalając wartość jego funkcji adaptacji oraz zastosowanie odpowiedniego sposobu wyboru osobników. Krok Wyboru rodziców, nazywany również Reprodukcją, odpowiada za wyznaczenie określonej ilości osobników (rodziców), które poddane zostaną procesowi Krzyżowania. Podczas Krzyżowania tworzone są nowe rozwiązania. Realizowane jest to poprzez wymianę genów, pomiędzy osobnikami rodzicielskimi. Mutacja zapobiega utkwieniu algorytmu w lokalnym optimum. Etap ten może być realizowany np. poprzez dokonanie pewnych, niewielkich zmian, w strukturze dotychczasowych rozwiązań.

Hybrydowe algorytmy ewolucyjne, zwane również memetycznymi, angażują metody lokalnego przeszukiwania do procesu ewolucyjnego. Metody te mają za zadanie poprawić zbieżność algorytmu w kierunku rozwiązań o większej wartości. Ich bazą jest analiza wybranych lub wszystkich rozwiązań leżących w pewnym bliskim otoczeniu $N(x) \subseteq X$ wybranego rozwiązania x [3]. Ma to na celu zastąpienie rozwiązania aktualnego rozwiązaniem o wyższej wartości funkcji przystosowania.

Istnieje wiele technik łączenia metod lokalnego przeszukiwania i algorytmów ewolucyjnych. Odnosząc się do przedstawionego schematu algorytmu genetycznego, przeszukiwanie lokalne może zostać użyte w kroku inicjalizacji, do poprawy jakości populacji początkowej, po krzyżowaniu, dla nowego pokolenia czy do ulepszenia populacji końcowej.

3. Przegląd literatury

Algorytmy ewolucyjne znajdują zastosowanie dla poszukiwania przybliżonych rozwiązań problemów, zarówno jedno-, dwu- i trójwymiarowego pakowania. W związku z tym jednak, że są one metodami ogólnymi i nie wykorzystują szczegółowych informacji o problemie, znaczna część badaczy uzupełnia je o dodatkowe heurystyki, odzwierciedlające specyfikę zagadnienia i przyjęte ograniczenia. Heurystyki takie, odpowiadają zazwyczaj za znajdowanie dozwolonych pozycji dla przedmiotów w pojemnikach (pakowanie 2D), bądź pudełek w kontenerach (pakowanie 3D) i są nazywane procedurami dekodującymi (ang. *decoding procedure*). Sposób działania procedury dekodującej określa strategia pakowania (*packing strategy*). Rola algorytmu ewolucyjnego, użytego wraz z procedurą dekodującą, może sprowadzać się do odnalezienia jak najlepszej kolejności (sposobu) pakowania, w której przedmioty, bądź pudełka, zostaną wykorzystane przez heurystykę.

Stawowy [14] opracował algorytm ewolucyjny dla zagadnienia jednowymiarowego pakowania. Celem pracy było stworzenie możliwie prostego algorytmu, rozwiązującego problem na poziomie zbliżonym do bardziej wyspecjalizowanych algorytmów. Potomstwo tworzone jest za pomocą operatorów mutacji, bez użycia krzyżowania. Tan i in. [13] zaproponowali algorytm ewolucyjny dla dwuwymiarowego problemu pakowania. Autorzy reprezentują dopuszczalne rozwiązanie nie jako wektor, ale jako strukturę, nazwaną cząstką, których zbiór tworzy rój (ang. *particle swarm*). Pojedyncza cząstka zawiera informacje dotyczące ilości użytych pojemników (ang. *bin*), rozmieszczonych w nich prostokątnych przedmiotach oraz najlepszych wynikach. Praca formułuje model matematyczny dla zagadnienia dwuwymiarowego pakowania wraz z dodatkowymi ograniczeniami. Pod uwagę wzięty został środek ciężkości całkowitego ładunku.

W pracach: He i in. [8], Wu i in. [16] oraz Karabulut i Mustafa [12] przedstawiony został problem trójwymiarowego pakowania, w którym pudełka umieszczane są w kontenerze o stałej długości i szerokości, ale o zmiennej wysokości. W wariancie problemu rozważanym przez He i in. [8] i Wu i in. [16] pudełka mogą posiadać różne rozmiary i dowolną rotację. Pojedyncze rozwiązanie reprezentowane jest jako chromosom, zawierający informację o kolejności pakowania pudełek i rodzaju rotacji każdego z nich. Procedura dekodująca wykorzystuje koncepcję punktów odniesienia (ang. *extreme points*, *reference points*). Autorzy [8] rozwinęli ideę przedstawionego w [16] współczynnika, nazwanego indeksem, służącego do wyboru pozycji na której pudełko zostanie umieszczone w kontenerze. Dodatkowo stworzony został framework (*global search framework*, - GSF), wykorzystujący koncepcję gradientu ewolucyjnego (ang. *evolutionary gradient*). Karabulut i Mustafa [12] zaproponowali metodę znajdowania pozycji dla pudełek w kontenerze, nazwaną *Deepest Bottom Left with Fill Method* (DBLF), będącą rozszerzeniem procedury przedstawionej przez Hopper'a [10], dla dwuwymiarowego problemu pakowania. Metoda wstawia pudełko na pierwszej dozwolonej pozycji, znajdującej się możliwie najgłębiej (*deepest*), najniżej (*bottom*) i najbliżej lewej strony kontenera (*left*). Pudełka nie posiadają możliwości rotacji.

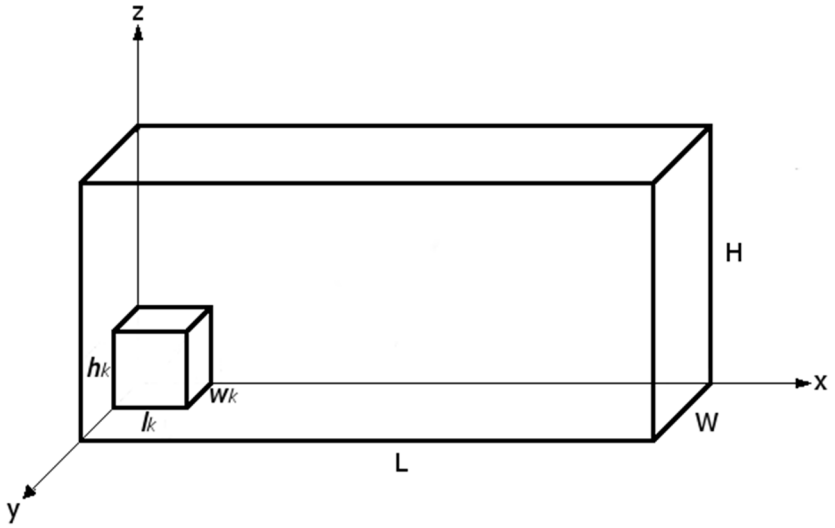
W pracach Goncalves i Resende [7], Bortfeld i Gehring [2], Kang i in. [11] autorzy rozważają problem załadunku pojedynczego kontenera o stałych wymiarach. Kang i in. [11] rozwijają strategię pakowania przedstawioną przez Karabulut'a i Mustafę [12]. Wprowadzone zostało pojęcie prostopadłościennego obiektu przestrzennego (ang. *cuboid space object*). Obiekt ten reprezentuje przestrzeń wewnątrz kontenera, w której możliwe jest zapakowanie pudełek. Wykorzystanie obiektów ma na celu, m.in. skrócenie czasu obliczeń. Pudełka o objętości i wymiarach nie pasujących do przestrzeni zajmowanej przez obiekt są pomijane na danym etapie pracy algorytmu. Goncalves i Resende [7] opracowali równoległy algorytm, w którym chromosom zawiera informacje dotyczące kolejności pakowania pudełek do kontenera oraz warstwy w której dane pudełko może zostać umieszczone. Warstwa (ang. *layer*) jest prostokątną strukturą (pionową lub poziomą), składającą się z pudełek tego samego typu, wykorzystywaną do uzupełniania pustych przestrzeni wewnątrz kontenera (ang. *maximal spaces*). Autorzy przedstawili procedurę, służącą do łączenia wolnych przestrzeni (*MaxJoin*). Bortfeld i Gehring [2] przedstawili algorytm genetyczny, którego strategia pakowania opiera się na technice warstwowej. Praca rozważa wiele praktycznych ograniczeń problemu, tj. określona ilość rotacji pudełek, ograniczenia dotyczące stabilności ładunku, jego wagi czy balansu.

Bischoff i Ratcliff [1] przeprowadzili dyskusję dotyczącą ówczesnej metodologii rozwiązywania problemu kontenerowego (ang. *Container loading problem*). Omówiono szereg ograniczeń praktycznych zagadnienia oraz zaprojektowano procedurę generującą różnorodne instancje problemu, co umożliwiło lepszą ocenę jakości algorytmów

opracowanych przez późniejszych badaczy. Sudholt [15] dokonał teoretycznej analizy algorytmów memetycznych. Podjęty został temat wpływu częstotliwości i głębokości (ilości iteracji) lokalnego przeszukiwania na efektywność i czas działania algorytmu. Przedstawione zostały klasy funkcji, dla których nawet niewielka zmiana parametrów, może mieć znaczący wpływ na wydajność. Ceschia, Schaerf i in. [5] przedstawione zostały techniki lokalnego przeszukiwania dla problemu pakowania połączonego z problemem wyznaczaniem tras (ang. *routing*). Zaproponowane podejście oparte zostało na strategii łączącej symulowane wyżarzanie z przeszukiwaniem sąsiedztwa o dużym rozmiarze.

4. Definicja problemu

Problem trójwymiarowego pakowania (3D-BPP), w rozważanym wariancie, polega na zapakowaniu takiej liczby pudełek do pojedynczego kontenera, aby przy spełnieniu wszystkich przyjętych ograniczeń, całkowita objętość ładunku była jak największa.



Rys. 1. Kontener

Kontener *K* (patrz Rys. 1.), ma postać prostopadłościanu o stałych wymiarach: długości *L*, szerokości *W*, wysokości *H* i objętości *V*. Dostępny jest zbiór pudełek $P = \{p_0, p_1, \dots, p_{n-1}\}$, typów $T = \{t_0, t_1, \dots, t_{m-1}\}$ i rotacji $R = \{0,1,2,3,4,5\}$, reprezentujących możliwe obroty pudełek. Każde pudełko $p_i \in P$ posiada określony typ u_i, definiujący jego wymiary i możliwe rotacje, tj. $u_i = t_k$, dla $k \in \{0,1,\dots,m-1\}$, $i = 0,1,\dots,n-1$, $u \in U = \{u_0, u_1, \dots, u_{n-1}\}$. Dany typ $t_k \in T$ można zapisać w postaci: $t_k = (l_k, w_k, h_k, MR_k)$, gdzie $MR_k \subseteq R$ a także określa dostępne rotacje, natomiast l_k, w_k, h_k oznaczają odpowiednio: długość, szerokość i wysokość pudełka. W zbiorze MR_k może znajdować się od jednej do sześciu rotacji.

Rozwiązanie $e = (PZ_e, q_e)$ rozważanego problemu pakowania definiuje zbiór zapakowanych pudełek $PZ_e \subseteq P$, $PZ_e = \{c_0, c_1, \dots, c_{q_e}\}$, gdzie q_e jest liczbą pudełek w zbiorze PZ_e. Należy znaleźć takie rozwiązanie *e* aby suma objętości zapakowanych pudełek:

$$\max_{0 \le c_0 \le c_1 \le \cdots \le c_{q_e-1} \le n-1} \sum_{z=0}^{q_e-1} v_z \quad (1)$$

była maksymalna i spełnione były ograniczenia:

- pudełko c_z musi znajdować się całkowicie wewnątrz kontenera, równolegle do jego ścian bocznych, w jednej z dostępnych dla danego typu rotacji,
- pudełko c_z nie może zajmować przestrzeni zajętej przez wcześniej zapakowane pudełka,
- pudełko c_z musi zostać umiejscowione na spodzie kontenera, bądź na górnej części innego pudełka.

Do określania pozycji pudełek wykorzystywane są współrzędne w kartezjańskim układzie odniesienia. Zbiór pudełek *P* może mieć dowolny charakter, począwszy od słabo heterogenicznego (ang. *weakly heterogeneous*) aż do silnie heterogenicznego (ang. *strongly heterogeneous*). Jeżeli zbiór jest słabo heterogeniczny oznacza to, że w instancji problemu zawiera się niewiele typów, z dużą ilością pudełek dla każdego z nich. Na zbiór silnie heterogeniczny składa się wiele typów pudełek, z niewielką ich ilością dla każdego typu.

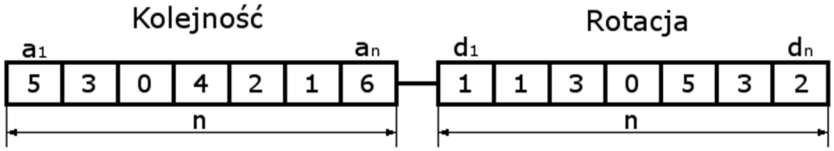
5. Opis użytych algorytmów

Zaproponowane algorytmy łączą w sobie cechy algorytmu ewolucyjnego oraz heurystyki, użytej jako procedura dekodująca. Rozdział opisuje przyjętą dla rozwiązania reprezentację (5.1), sposób ustalania pozycji pudełek w kontenerze (5.2) oraz schemat działania opracowanych algorytmów (5.3).

5.1. Sposób reprezentacji rozwiązania

Osobnik reprezentowany jest jako chromosom, składający się z dwóch części: Kolejności i Rotacji (zobacz Rys. 2.). Pierwsza z nich zawiera sekwencję pudełek, druga odpowiadający każdemu z nich numer rotacji, użytej do zapakowania pudełka do kontenera. Każda część chromosomu posiada długość równą ilości pudełek w zbiorze *P*, niezależnie od tego, ile z nich zostanie wykorzystanych w rozwiązaniu. Dla liczby pudełek równej n, długość chromosomu wynosi $2n$. Numer rotacji zapisany na pozycji d_s odpowiada pudełku o numerze zapisanym w miejscu a_s chromosomu. Podział chromosomu na dwie części został wykorzystany również w pracach [8, 16].

Na podstawie chromosomu, przy użyciu procedury dekodującej, ustalana jest rzeczywista postać rozwiązania *e* (wraz ze współrzędnymi pudełek w przestrzeni i ich rotacją). W związku z tym że $n \geq q_e$, co oznacza, że nie wszystkie pudełka zakodowane w chromosomie muszą znaleźć się zbiorze PZ_e, osobnik reprezentuje rozwiązanie *e* w sposób przybliżony. Przypadek, w którym $q_e = n$ oznacza, że rozwiązanie e jest optymalne. W dalszej części pracy pojęcia rozwiązanie, chromosom i osobnik będą tożsame.



Rys. 2. Chromosom

5.2. Procedura dekodująca

Zadaniem procedury dekodującej jest znalezienie dozwolonych pozycji w kontenerze, dla jak największej ilości pudełek, których sekwencja zakodowana jest za pomocą chromosomu. W praktyce, dla rozwiązania *e*, oznacza to odnalezienie q_e elementów tworzących zbiór PZ_e . Użyta do tego celu heurystyka bazuje na wersji metody przedstawionej w pracy [12], dla problemu pakowania ze zmienną wysokością kontenera (ang. *strip packing*). Użyta została ona również przez Kang'a i in. w pracy [11].

Procedura, w postaci pseudokodu, przedstawiona została na Rysunku 3. Podczas Obliczeń wstępnych pudełka są obracane, w zależności od zapisanego w chromosomie sposobu rotacji. Następnie procedura dąży do umieszczenia pudełek w kontenerze, według podanej sekwencji. Pętla znajdująca się w linii 4 jest wykonywana do momentu wykorzystania wszystkich dostępnych pudełek, bądź braku możliwości wstawienia. Istotnym elementem jest lista pozycji, służąca do przetrzymywania aktualnie dostępnych współrzędnych, które są rozważane przy poszukiwaniu miejsca, w którym pudełko b_s może zostać zapakowane. Jeżeli w kontenerze nie znajduje się jeszcze żadne pudełko, lista pozycji zawiera jedynie współrzędne (0,0,0), stanowiące początek kontenera.

Współrzędne znajdujące się na liście pozycji są sortowane według porządku: głębokość (*deep*, jak najmniejsza wartość na osi y), spód (*bottom*, jak najmniejsza wartość na osi z), lewo (*left*, jak najmniejsza wartość na osi x). W celu określenia, czy możliwe jest wstawienie rozważanego pudełka b_s na wybranej pozycji *j*, heurystyka sprawdza, czy będzie się ono całkowicie mieścić w kontenerze i czy nie narusza przestrzeni zajmowanej przez wcześniej zapakowane pudełka.

Jeżeli wstawienie jest możliwe, pozycja *j* zostaje usunięta z listy (Aktualizacja), a na jej miejsce dodawane są trzy nowe, powstałe dzięki umiejscowieniu pudełka b_s. Pozycje te posiadają współrzędne: $(x_j + l_s, y_j, z_j)$, $(x_j, y_j + w_s, z_j)$, $(x_j, y_j, z_j + h_s)$ i są nazywane punktami referencyjnymi bądź skrajnymi (ang. *reference points*, *extreme points*) [6, 8, 16]. Następnie, gdy pozostały jeszcze pudełka, procedura rozważa kolejne z nich, sortując zaktualizowaną listę dostępnych pozycji i sprawdzając możliwość wstawienia, w przeciwnym przypadku, kończy swoje działanie.

Jeżeli wstawienie pudełka b_s na pozycji *j* nie jest możliwe, heurystyka próbuje umieścić je na kolejnych pozycjach z listy, aż do jej skończenia. Spełnienie warunku *lista pozycji* $\neq\emptyset$ (Rys. 3. linia 9) oznacza, że na *liście pozycji* pozostały jeszcze współrzędne nierozważone w kontekście pudełka b_s. Gdy nie jest możliwe wstawienie pudełka na żadną z dostępnych pozycji, procedura przerywa działanie (warunek w linii 21).

W momencie zakończenia pracy przez procedurę dekodującą, obliczana jest wartość funkcji adaptacji dla danego rozwiązania *e*:

$$F(e) = \frac{\sum_{z=0}^{q_e-1} v_z}{V} 100\% \quad (2)$$

dla: $e = (PZ_e, q_e)$
gdzie: v_z jest objętością każdego kolejnego zapakowanego pudełka $c_z \in PZ_e$,
$|PZ_e| = q_e$, V jest objętością kontenera K.

Informacje dotyczące wartości funkcji adaptacji oraz zapakowanych pudełek (ich wymiary, współrzędne w kontenerze, rodzaj rotacji) są zapamiętywane i bezpośrednio powiązane z danym osobnikiem, w związku z czym, procedura zostaje ponownie uruchomiona dla chromosomu tylko w przypadku jego modyfikacji.

```
1 Obliczenia wstępne:
2      n:= ilość pudełek;  s:= 0; koniec = false;
3
4 While ( s ≠ n && koniec = false )
5  {
6      Dla pudełka b_s sortuj lista_pozycji; j:=0;
7      wstawienie = false;
8
9      While (lista_pozycji ≠ ∅ && wstawienie = false)
10     {
11            If (b_s pasuje na pozycję j)
12            {
13                   Aktualizacja; s:= s + 1; wstawienie := true;
14            }
15            Else
16            {
17                   j := j+1;
18            }
19
20     }
21     If ( lista pozycji = ∅)
22            koniec = true;
23
24 }
25 Oblicz F(e);
```

Rys. 3. Procedura dekodująca

5. 3. Schemat działania opracowanych algorytmów

Opracowane algorytmy są wielościeżkowymi algorytmami równoległymi, bazującymi na migracyjnym modelu wyspowym. Badają one przestrzeń rozwiązań, za pomocą równolegle działających wątków poszukiwań. Procesy wymieniają pomiędzy sobą informacje uzyskane w trakcie eksploracji własnych trajektorii, w związku z czym algorytmy zaliczają się do podklasy algorytmów kooperujących [4].

Schemat działania algorytmu hybrydowego, dla pojedynczego procesu, został przedstawiony w postaci pseudokodu na Rysunku 4, natomiast algorytmu ewolucyjnego na

Rysunku 5. Wszystkie etapy ich pracy, poza krokiem Wymiany, realizowane są przez dany wątek niezależnie od pozostałych. Algorytmy mogą zostać również uruchomione w postaci sekwencyjnej, w takim wypadku są one traktowane jako algorytm równoległy z jednym procesem (Wymiana jest pomijana).

Poniżej omówione zostaną poszczególne kroki wspólne obu algorytmom. Istotną różnicą w ich działaniu jest, znajdujący się w linii 16 (Rys. 4. i Rys. 5.), operator mutacji dla algorytmu ewolucyjnego i lokalnego przeszukiwania dla algorytmu hybrydowego.

Poprzez Inicjalizację wczytywane są dane problemu oraz zostają nadane początkowe wartości niezbędnym zmiennym. Następnie każdy proces tworzy własną, startową podpopulację losowych osobników. Dla wszystkich chromosomów, poszczególnych podpopulacji początkowych, wywoływana jest procedura dekodująca (Dekodowanie, Rozdział 5.2). W kolejnym kroku możliwe są dwie sytuacje:

- *licznik* $\neq f$, co oznacza, że nie nastąpił moment wymiany osobników pomiędzy procesami i wykonywane dalsze operacje po których nowe, bądź zmodyfikowane osobniki zostają poddane Dekodowaniu.
- *licznik* $= f$, w związku z czym dochodzi do migracji osobników pomiędzy poszczególnymi wątkami.

W kroku Selekcji, każdy proces bada średnią jakoś rozwiązań własnej podpopulacji i odrzuca wszystkie posiadające wartość funkcji adaptacji poniżej przeciętnej.

Krzyżowanie odbywa się za pomocą dwupunktowej metody PMX. W większości przypadków rodzice wybierani są z puli osobników, które przetrwały proces selekcji, możliwy jest jednak, z pewnym prawdopodobieństwem, wybór rodzica z grona rozwiązań odrzuconych, co ma stanowić jedną z metod zapobiegania stagnacji.

Mutacja jest operatorem wykorzystanym jedynie w algorytmie ewolucyjnym (Rys 5. linia 16), mogącym przebiegać na dwa sposoby. Ponieważ kolejność pudełek zakodowanych w chromosomie ma znaczenie dla procedury dekodującej, pierwszy z nich polega na losowej zamianie miejscami dwóch pól z numerami pudełek i odpowiadających im rotacji. Druga metoda zmienia wartość pola rotacji w pojedynczym chromosomie, w związku z czym związane z nim pudełko posiada zmienioną rotację podczas próby umieszczenia go w kontenerze. Nowa wartość rotacji nie może być zabroniona przez typ do którego należy pudełko.

Przeszukiwanie lokalne, będące cechą algorytmu hybrydowego (Rys. 4. linia 16) wywoływane jest w każdej iteracji głównej pętli algorytmu, w celu poprawienia jakości nowopowstałych

```
1 Inicjalizacja:
2     Dane;  licznik := 0;  f;
3
4 Twórz podpopulacje;
5 Dekodowanie podpopulacji;
6
7 While ( warunek_stopu = false )
8 {
9     If (licznik = f)
10    {
11          Wymiana osobników pomiędzy procesami;
12          licznik := 0;
13    }
14    Else
15    {
16          Selekcja; Krzyżowanie; Przeszukiwanie lokalne;
17          Dekodowanie nowych osobników;
18          licznik := licznik + 1;
19    }
20 }
```

Rys. 4. Algorytm hybrydowy z lokalnym przeszukiwaniem

```
1 Inicjalizacja:
2     Dane;  licznik := 0;  f;
3
4 Twórz podpopulacje;
5 Dekodowanie podpopulacji;
6
7 While ( warunek_stopu = false )
8 {
9     If (licznik = f)
10    {
11          Wymiana osobników pomiędzy procesami;
12          licznik := 0;
13    }
14    Else
15    {
16          Selekcja; Krzyżowanie; Mutacja;
17          Dekodowanie nowych osobników;
18          licznik := licznik + 1;
19    }
20 }
```

Rys. 5. Algorytm ewolucyjny z operatorem mutacji

osobników. Dla każdego rozwiązania generowane jest jego otoczenie, poprzez tworzenie rozwiązań do niego podobnych i obliczanie wartości ich funkcji adaptacji. Ponieważ sprawdzenie jakości każdego nowego osobnika jest operacją kosztowną obliczeniowo, otoczenie generowane jest metodą krok po kroku. Oznacza to, że wartość każdego sąsiada rozwiązania bieżącego ustalana jest zaraz po jego stworzeniu. Jeżeli reprezentuje on

rozwiązania o wyższej jakości, zastępuje aktualne rozwiązanie, kończąc przeszukiwanie. Przeszukiwanie lokalne dla danego osobnika może zostać przeprowadzone wielokrotnie w danej iteracji.

Wymiana jest kluczowym elementem obu algorytmów. Została ona zaimplementowana za pomocą biblioteki MPI. Procesy łączone są w pary, w ramach których zachodzi komunikacja. Każdy z dwójki procesorów wydziela część swojej podpopulacji i przesyła ją do drugiego wątku. Wysłane rozwiązania zastępowane są nowo przybyłymi. Komunikacja pomiędzy procesorami ma charakter nieblokujący (funkcje: *MPI_Isend()*,*MPI_Irecv())*, a grupy osobników przesyłane są pomiędzy nimi w postaci spakowanych pakietów danych (funkcje: *MPI_Pack()*,*MPI_Unpack())*). Głównym zadaniem Wymiany, jest dywersyfikacja poszczególnych podpopulacji.

6. Eksperymenty obliczeniowe

Obliczenia wykonano na komputerze wyposażonym w 6-rdzeniowy procesor Intel Core i7 CPU X980 (3.33GHz) i system operacyjny Linux Ubuntu 12.04.3 LTS.

Jako dane testowe użyto instancje problemu, uzyskane za pomocą procedury przedstawionej w pracy [BIR]. Do eksperymentów obliczeniowych wybrano siedem przykładów, różniących się rodzajem heterogeniczności. Ilość możliwych typów pudełek dla każdej instancji wyniosła od trzech do dwudziestu. Kontener posiadał stałe wymiary, niezależnie od rozważanego przykładu. Badane algorytmy uruchomione zostały w następujący sposób:

- sekwencyjnie, z użyciem jednego wątku (procesu),
- równolegle, z użyciem dwóch wątków (procesów),
- równolegle, z użyciem sześciu wątków (procesów).

Do badań przyjęto następujące założenia. W kroku Selekcji odrzucane są wszystkie osobniki, których wartość funkcji adaptacji jest niższa niż wartość średnia dla podpopulacji danego wątku. Wymiana następuje tylko w momencie, gdy algorytm uruchomiony jest w wersji równoległej. Podczas Wymiany wątki dobierane są w pary i przesyłają między sobą dziesięć procent własnych osobników. Osobniki wysłane są całkowicie zastępowane osobnikami otrzymanymi. Warunkiem stopu jest czas obliczeń równy 120 minut. Wielkość podpopulacji jest zmniejszana wraz ze wzrostem ilości użytych do obliczeń wątków.

Wyniki obliczeń przedstawiono w poniższej tabeli (patrz Tab. 1.). Nazwy przykładów testowych zostały zakodowane w formie bench *x_xx*, gdzie *x* zmienia się wraz ze wzrostem ilości typów, a *xx* jest numerem porządkowym danego przykładu. Tabela przedstawia informacje dotyczące objętości upakowanych w kontenerze pudełek, w zależności od liczby wątków, rodzaju instancji i użytego algorytmu. Algorytm LP oznacza algorytm hybrydowy wykorzystujący lokalne przeszukiwanie, algorytm M algorytm ewolucyjny z operatorem mutacji.

Tab. 1. Wyniki obliczeń przeprowadzonych z użyciem jednego procesora

problem	ilość typów	algorytm LP		algorytm M	
		ilość wątków	objętość [%]	ilość wątków	objętość [%]
bench1_03	3	1	69,0	1	71,2
bench2_01	5	1	78,2	1	83,4
bench3_01	8	1	79,5	1	86,8
bench4_02	10	1	72,3	1	78,9
bench5_17	12	1	76,4	1	79,8
bench6_01	15	1	71,2	1	74,9
bench7_02	20	1	70,0	1	75,8
bench1_03	3	2	72,2	2	74,6
bench2_01	5	2	79,8	2	85,9
bench3_01	8	2	78,1	2	84,4
bench4_02	10	2	73,4	2	81,4
bench5_17	12	2	77,5	2	81,5
bench6_01	15	2	73,2	2	78,3
bench7_02	20	2	70,8	2	79,9
bench1_03	3	6	70,0	6	75,6
bench2_01	5	6	77,7	6	85,3
bench3_01	8	6	76,4	6	85,6
bench4_02	10	6	73,0	6	81,6
bench5_17	12	6	74,5	6	81,6
bench6_01	15	6	72,7	6	79,2
bench7_02	20	6	72,9	6	78,8

Algorytm hybrydowy (LP) uzyskał gorszy wynik od algorytmu M dla każdej instancji problemu i każdej ilości wątków obliczeniowych. Objętość zapakowanych pudełek była w tym przypadku średnio o 6,0% niższa, przy czym najmniejsza różnica wyniosła 2,2% (bench1_03, jeden wątek) a największa 9,2% (bench3_01, sześć wątków). Upakowanie kontenera, uzyskane za pomocą algorytmu M, dla poszczególnych instancji problemu, było lepsze średnio o: 4,9% dla uruchomienia sekwencyjnego, 5,9% przy użyciu dwóch wątków, i 7,2% dla sześciu. Użycie dwóch wątków obliczeniowych sześciokrotnie pozwoliło znaleźć rozwiązanie lepsze niż przy wykorzystaniu jednego procesu, w przypadku obu algorytmów. Rozwiązanie gorsze uzyskano tylko dla problemu bench3_01. Obliczenia z wykorzystaniem sześciu procesów, przy użyciu algorytmu M, zwiększyły wartość upakowania w sześciu przypadkach, względem wyników uzyskanych dla jednego wątku (gorzej bench3_01) i w pięciu przypadkach względem wyników dla dwóch wątków (gorzej bench2_01 i bench7_02). W przypadku algorytmu LP, dla czterech instancji problemu objętość zapakowanych pudełek była większa przy użyciu sześciu wątków niż przy użyciu jednego

procesu (bench1_03, bench4_02, bench6_01, bench7_02) i tylko raz niż dla dwóch wątków (bench7_02).

7. Wnioski i dalsze badania

W pracy przedstawiono dwa, oparte na modelu wyspowym, równoległe algorytmy populacyjne dla trójwymiarowego problemu pakowania. Badania przeprowadzono dla instancji problemu o różnym poziomie heterogeniczności. Lepsze wyniki (o większym zajęciu przestrzeni wewnątrz kontenera) uzyskano przy pomocy algorytmu wykorzystującego operator mutacji, niż przy użyciu algorytmu angażującego lokalne przeszukiwanie. Prawdopodobnie jest to spowodowane dużą ilością czasu potrzebną do ustalenia wartości każdego nowego rozwiązania (konieczność znalezienia pozycji w kontenerze dla poszczególnych pudełek). Aby zwiększyć efektywność algorytmu hybrydowego, niezbędne jest wprowadzenie, wykorzystującej odpowiednie własności zagadnienia, akceleracji bądź dalsze zrównoleglenie obliczeń, np. z wykorzystaniem GPU.

Zwiększenie ilości wątków obliczeniowych, w większości przypadków, pozwalało znaleźć lepsze upakowania. Pogorszenie, dla większości instancji problemu, nastąpiło jedynie przy zwiększeniu liczby procesów z dwóch do sześciu, dla algorytmu hybrydowego, co może być związane ze zmniejszeniem ilości osobników w poszczególnych podpopulacjach.

Literatura

1. Bischoff E. E., Ratcliff M.S.W.: Issues in the Development of Approaches to Container Loading. Omega, Int. J. Mgmt Sci. Vol. 23, No. 4/1995, 377-390.
2. Bortfeld A., Gehring H.: A hybryd genetic algorithm for the container loading problem. European Journal of Operational Research 131/2001, 143-161.
3. Bożejko W., Pempera J. (red.), Optymalizacja dyskretna w informatyce, automatyce i robotyce, Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław 2012, ISBN 978-83-7493-743-6
4. Bożejko W.: A New Class of Parallel Scheduling Algorithms. Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław 2010.
5. Ceschia S., Schaerf A., Stützle T.: Local search techniques for a routing-packing problem. Computers & Industrial Engineering 66/2013, 1138–1149.
6. Crainic TG, Perboli G, Tadei R.: Extreme point-based heuristics for three- dimensional bin packing. INFORMS Journalon Computing 20 /2008, 368–384.
7. Goncalves J. F., Resende M. G.C.: A parallel multi-population biased random-key genetic algorithm for a container loading problem. Computers & Operations Research 39 /2012, 179–190.
8. He Y., Wu Y., de Souza R.: A global search framework for practical three-dimensional packing with variable carton orientations. Computers & Operations Research 39/2012, 2395–2414.
9. Holland J.H.: Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence. University of Michigan Press 8/1975), 183.
10. Hopper E.: Two-dimensional Packing Utilising Evolutionary Algorithms and other

Meta- Heuristic Methods. Ph.D.Thesis, University of Wales 2000.
11. Kang K., Moon I., Wang H.: A hybrid genetic algorithm with a new packing strategy for the three-dimensional bin packing problem. Appl. Math. Comput. (2012), http://dx.doi.org/10.1016/j.amc.2012.07.036
12. Karabulut K., Mustafa M.M.: A hybrid genetic algorithm for packing in 3D with deepest bottom left with fill method. ADVIS 2004: Advances in Information Systems 2004, 441–450.
13. Liu D.S., Tan * K.C., Huang S.Y., Goh C.K., Ho W.K.: On solving multiobjective bin packing problems using evolutionary particle swarm optimization. European Journal of Operational Research 190/2008, 357–382.
14. Stawowy A.: Evolutionary based heuristic for bin packing problem. Computers & Industrial Engineering 55/2008, 465–474.
15. Sudholt D.: The impact of parametrization in memetic evolutionary algorithms. Theoretical Computer Science 410 /2009, 2511-2528.
16. Wu Y., Li W., Goh M., de Souza R.: Three-dimensional bin packing problem with variable bin height. European Journal of Operational Research 202 /2010, 347–355.

Prof. nadzw. dr hab. Wojciech BOŻEJKO
Mgr inż. Łukasz KACPRZAK
Instytut Informatyki, Automatyki i Robotyki
Politechnika Wrocławska
Janiszewskiego 11-17, 50-372 Wrocław, Polska
e-mail: wojciech.bozejko@pwr.wroc.pl
e-mail: lukasz.kacprzak@pwr.wroc.pl

Prof. nadzw. dr hab. Mieczysław WODECKI
Instytut Informatyki, Uniwersytet Wrocławski
Joliot-Curie 15, 50+383 Wrocaw, Polska
e-mail: mwd@ii.uni.wroc.pl