Odporna optymalizacja problemu pakowania prostokątów z rotacją

Ireneusz Placek

May 2024

1 Wstęp do problemu

1.1 Klasyczna definicja problemu

Problem pakowania prostokątów (Rectangle Packing Problem), to zagadnienie optymalizacyjne polegające na ułożeniu, w jak najbardziej efektywny sposób, prostokątnych obiektów (reprezentujących przedmioty do zapakowania) na możliwie najmniejszym obszarze, tak aby prostokąty nie nachodziły na inne.

Dla zestawu prostokątów reprezentowanych przez wektor szerokości i wysokości:

$$\mathcal{P} \in \mathcal{N}^2, \quad \mathcal{P} = \{(w_1, h_1), (w_2, h_2), ..., (w_n, h_n)\} \tag{1}$$

gdzie n oznacza ilość prostokątów; chcemy wyznaczyć możliwie jak najmniejsze pole prostokąta, na którym możemy ułożyć wszystkie prostokąty:

$$\min(AB) = \min(\max_{i \in N}(w_i x_i) \max_{j \in N}(h_i y_i)), \quad N = 1, 2, ..., n \tag{2}$$

gdzie x i y reprezentują wartości położenia lewego dolnego punktu prostokąta w dwuwymiarowej rzeczywistości:

$$\mathcal{W} \in \mathcal{N}^2, \quad \mathcal{W} = \{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}, \quad \forall i \in N: \quad x_i, y_i \geq 0 \tag{3}$$

z warunkiem nienakładania się prostokątów:

$$\forall i, j \in N, i \neq j: \quad (w_i + x_i \leq x_j) \vee (h_i + y_i \leq y_j) = 1 \tag{4}$$

1.2 Uwzględnienie rotacji

Dodatkowo możemy uwzględnić rotację prostokątów, aby móc lepiej dopasowywać minimalne pole. Użyjemy w tym przypadku binarnej rotacji, która pozwoli nam zamieniać wysokość i szerokość miejscami:

$$r_i \in \{0, 1\}, \quad i \in N \tag{5}$$

gdzie 0 oznacza brak rotacji, a 1 oznacza rotację o 90°. Wtedy, równania minimalizacji i wykrywania kolizji mają taką postać:

$$\min(AB) = \min(\max_{i \in N}(x_i + w_i(1 - r_i) + h_i r_i) \max_{j \in N}(y_i + h_i(1 - r_i) + w_i r_i)), \quad N = 1, 2, ..., n \tag{6}$$

$$\forall i, j \in N, i \neq j: \quad (x_i + w_i(1 - r_i) + h_i r_i) \vee (y_i + h_i(1 - r_i) + w_i r_i) = 1 \tag{7}$$

1.3 Uodpornienie problemu

Możemy zastosować uodpornienie problemu pakowania prostokątów wobec nieznanych wartości wielkości i szerokości prostokątów:

$$\mathcal{P} \in \mathcal{N}^2, \quad \mathcal{P} = \{(w_1, h_1), (w_2, h_2), ..., (w_n, h_n)\} \tag{8}$$

$$\forall i \in N: \quad w_i \in (\underline{w_i}, \overline{w_i}), \quad 0 < \underline{w_i} \leq w_i \leq \overline{w_i} \tag{9}$$

$$\forall i \in N: \quad h_i \in (\underline{h_i}, \overline{h_i}), \quad 0 < \underline{h_i} \leq h_i \leq \overline{h_i} \tag{10}$$

gdzie wszystkie możliwe kombinacje $\mathcal{P}$ są reprezentowane przez zbiór L:

$$L = \{\mathcal{P}^{(1)}, \mathcal{P}^{(2)}, \mathcal{P}^{(3)}, ...\} \tag{11}$$

a pojedyncza kombinacja jest reprezentowana przez l:

$$l \in L, \quad l = \mathcal{P}^{(l)} = \{(w_1^{(l)}, h_1^{(l)}), (w_2^{(l)}, h_2^{(l)}), ..., (w_n^{(l)}, h_n^{(l)})\} \tag{12}$$

2 Algorytmy

Uznajmy iż istniejemy na razie w obrębie jednej kombinacji l.

2.1 Algorytm doskonały

W celu uzyskania najlepszego wyniku użyte zostało narzędzie DocPlex (wraz z pakietem CPLEX), z minimalnie zmienionymi funkcjami, ze względu na ograniczenia narzędzia:

$$\mathcal{P} \in \mathcal{N}^2, \quad \mathcal{P} = \{(w_1, h_1), (w_2, h_2), ..., (w_n, h_n)\} \tag{13}$$

$$\forall i \in \{1, 2, ..., n-1\} \forall j \in \{i+1, ..., n\} : \tag{14}$$
$$(x_i + w_i(1 - r_i) + h_i r_i) + \tag{15}$$
$$(x_j + w_j(1 - r_j) + h_j r_j) + \tag{16}$$
$$(y_i + h_i(1 - r_i) + w_i r_i) + \tag{17}$$
$$(y_j + h_j(1 - r_j) + w_j r_j) \geq 1 \tag{18}$$

$$A = \max_{i \in N}(x_i + w_i(1 - r_i) + h_i r_i) \tag{19}$$
$$B = \max_{j \in N}(y_i + h_i(1 - r_i) + w_i r_i) \tag{20}$$

Wartość szukana została zmieniona, ze względu na to iż multiplikacja tych dwóch liczb jest nieliniowa. Został zastosowany zamiennik w postaci logarytmów:

$$\min(\log(A) + \log(B)) \tag{21}$$

Problem dalej zostaje spełniony ponieważ logarytm to funkcja stale rosnąca w przedziale $(0, \infty)$, oraz:

$$Z = AB \tag{22}$$
$$\log(Z) = \log(AB) = \log(A) + \log(B) \tag{23}$$
$$\exp(\log(Z)) = \exp(\log(A) + \log(B)) = \exp(\log(AB)) \tag{24}$$
$$Z = AB \tag{25}$$

Zatem jeżeli chcemy uzyskać z powrotem wartość pola prostokąta, musimy zastosować funkcję eksponencjalną.

Aby uzyskać rozwiązanie używamy komponentu CP (Constraint Programming) z narzędzia DocPlex, które to wykonuje operację branch & bound w celu znalezienia optymalnego rozwiązania.

2.2 Algorytm aproksymacyjny

Jest to algorytm zachłanny bazujący na tym, iż "wkładane" prostokąty stanowią podstawy siatki przeszukiwania:

1. Ustaw odpowiednio wysokie, lecz możliwie najmniejsze wartości W i H (szerokość i wysokość prostokąta w którym pakujemy).
2. Posortuj prostokąty po określonym wymiarze, np. wysokości:
$$sort(\mathcal{P}) \longrightarrow h_1 \geq h_2 \geq ... \geq h_n \tag{26}$$
3. Zainicjalizuj dynamiczną siatkę z jednym polem o wielkości (W x H).
4. Dla każdego prostokąta (h_i, w_i) który chcemy zapakować; sprawdzając każdy prostokąt ($\dot{h_j}$, $\dot{w_j}$) w siatce od najmniejszego do najmniejszego:
   - Jeżeli prostokąt nie mieści się w siatce; ($h_i > \dot{h_j} \lor w_i > \dot{w_j}$) :
     Przejdź do następnego, ale jeżeli to był ostatni prostokąt siatki, to znaczy że wymiary (W, H) prostokąta do pakowania są zbyt niskie.
   - Jeżeli prostokąt mieści się w siatce; ($h_i \leq \dot{h_j} \land w_i \leq \dot{w_j}$):

(a) Umieść prostokąt jak najbliżej punktu (0, 0) czyli w lewym dolnym rogu;

(b) Tworzymy nowy podział siatki poprzez położenie prostokąta;

(c) Ponownie sprawdzamy wszystkie prostokąty, które są obok siebie siatki przy warunku:
$\forall j, k; j \neq k : \quad (\dot{h}_j = \dot{h}_k \vee \dot{w}_j = \dot{w}_k)$
Dla każdej kombinacji spełniającej ten warunek, łączymy prostokąty siatki w jeden.

Aby zastosować rotację, stosujemy ten algorytm h razy, gdzie za każdym razem rotujemy losowo pojedyńczy prostokąt (zamieniamy miejscami w_i i h_i).

3 Minimalizacja maksymalnego żalu

Naszym celem jest minimalizacja maksymalnego żalu. Polega ona na podejmowaniu decyzji w warunkach niepewności w taki sposób, aby zminimalizować największy możliwy żal wynikający z dokonania złego wyboru.

Żal jest miarą straty wynikającej z wyboru jednej decyzji zamiast innej, lepszej decyzji, która mogła być podjęta, gdyby pełna informacja była dostępna. W przypadku tego zadania chcemy otrzymać taki zestaw x, y, aby dla dowolnego zestawu prostokątów z kombinacji l uzyskać możliwie najniższe pole prostokąta, do którego pakujemy:

$$\min_{\{x,y\} \in \mathcal{S}} (\max_{l \in L} (\min(A^{(l)} B^{(l)}) - \min(\widehat{A^{(l)}} \widehat{B^{(l)}}))) \tag{27}$$

gdzie:

- $\min(A^{(l)} B^{(l)})$ oznacza wynik z algorytmu aproksymacyjnego;
- $\min(\widehat{A^{(l)}} \widehat{B^{(l)}})$ oznacza wynik z algorytmu doskonałego;
- $\mathcal{S}$ to zestaw wszystkich kombinacji wektorów x, y dla wszystkich kombinacji l.

4 Wnioski

4.1 Ciekawość problemu

Problem minimalizacji maksymalnego żalu w kontekście zagadnienia pakowania prostokątów jest niezwykle interesujący z kilku powodów:

- **Teoretyczna złożoność**: Problemy optymalizacyjne, szczególnie te związane z pakowaniem, są znane ze swojej złożoności i trudności w znalezieniu optymalnych rozwiązań. Dodanie warunku minimalizacji maksymalnego żalu dodaje dodatkowy wymiar do problemu, co czyni go jeszcze bardziej złożonym i fascynującym.
- **Różnorodność metod**: Problem można rozwiązywać przy użyciu różnych podejść, takich jak metody dokładne (np. DOcplex) i heurystyczne (np. algorytmy przybliżone), co pozwala na eksperymentowanie z różnymi technikami optymalizacyjnymi.
- **Regret theory**: Wprowadzenie koncepcji żalu (regret) w problemach optymalizacyjnych jest interesującym podejściem, które ma swoje korzenie w teorii decyzji. Pozwala to na uwzględnienie niepewności i ryzyka w procesie podejmowania decyzji.

4.2 Zastosowanie rozwiązania

Rozwiązania tego problemu mogą mieć szerokie zastosowanie w różnych dziedzinach:

- **Logistyka i zarządzanie łańcuchem dostaw**: Optymalne pakowanie prostokątów może być zastosowane do efektywnego rozmieszczania towarów w magazynach, kontenerach czy pojazdach transportowych, co prowadzi do oszczędności przestrzeni i kosztów.
- **Inżynieria i produkcja**: W przemyśle produkcyjnym, optymalizacja rozmieszczenia elementów na arkuszach materiału (np. blachy, tkaniny) może prowadzić do minimalizacji odpadów i zwiększenia wydajności produkcji.
- **Grafika komputerowa i projektowanie**: W grafice komputerowej i projektowaniu, problem pakowania prostokątów może być zastosowany do efektywnego rozmieszczania tekstur na modelach 3D, co wpływa na wydajność renderowania i jakość wizualną.

4.3 Poziom trudności w rozwiązaniu

Rozwiązanie tego problemu niesie ze sobą kilka wyzwań:

- **Wysoka złożoność obliczeniowa**: Problemy pakowania prostokątów są znane ze swojej złożoności obliczeniowej, co sprawia, że znalezienie optymalnego rozwiązania w rozsądnym czasie jest trudne, zwłaszcza dla dużych instancji problemu.
- **Zróżnicowane podejścia**: Wymaga znajomości różnych technik optymalizacyjnych, zarówno metod dokładnych, jak i heurystycznych, co może być trudne dla osób nieposiadających specjalistycznej wiedzy w tym zakresie.
- **Balansowanie między optymalnością a czasem obliczeń**: Często istnieje konieczność kompromisu między jakością rozwiązania a czasem obliczeń, co wymaga umiejętności w doborze odpowiednich metod i parametrów.