Sąsiedztwo Motzkina dla Permutacyjnego Problemu Przepływowego

Remigiusz Wojewódzki[1] and Wojciech Bożejko[2]

[1]Politechnika Wrocławska
[2]Politechnika Wrocławska

Listopad 2025

Streszczenie

Wprowadzamy nowe sąsiedztwo dla permutacyjnego problemu przepływowego (Flow Shop Problem), w którym ruch składa się z wielu zagnieżdżonych zamian końców segmentów. Dopuszczalne konfiguracje odpowiadają strukturom enumerowanym przez liczby Motzkina M_n. Opisujemy algorytm programowania dynamicznego o złożoności $O(mn^3)$ do wyboru optymalnego zestawu zamian i analizujemy kombinatoryczne właściwości przestrzeni ruchów. Sąsiedztwo umożliwia eksplorację głębszych struktur permutacji w metaheurystykach takich jak Tabu Search i Symulowane Wyżarzanie.

1 Wprowadzenie

Permutacyjny problem przepływowy (Flow Shop Problem) polega na znalezieniu permutacji n zadań wykonywanych na m maszynach w tej samej kolejności, minimalizującej czas zakończenia (makespan) $C_{\max}$. Klasyczne sąsiedztwa (np. zamiana sąsiadujących elementów) eksplorują przestrzeń lokalnie i płytko. Motywacją tej pracy jest skonstruowanie bogatszego ruchu złożonego, który zachowuje strukturę umożliwiającą efektywną optymalizację, obejmując jednocześnie wiele skoordynowanych zamian.

2 Charakterystyka Liczb Motzkina

Liczby Motzkina M_n enumerują *planarne* (nieprzecinające się) struktury nad zbiorem n uporządkowanych punktów, gdzie każdy punkt może być (i) początkiem łuku, (ii) końcem łuku, lub (iii) izolowany. Stanowią one uogólnienie liczb Catalana – podczas gdy te ostatnie wymagają pełnego sparowania, liczby Motzkina dopuszczają punkty pojedyncze.

Równoważne reprezentacje. Obiekty Motzkina można interpretować jako:

1. Ścieżki kratowe z krokami $(1, +1)$, $(1, 0)$, $(1, -1)$ nieopadające poniżej osi $y = 0$.
2. Nieprzecinające się rodziny łuków nad linią z opcjonalnymi izolowanymi punktami.
3. Wzbogacone triangulacje lub podziały wielokątów.

W naszym sąsiedztwie każdy indeks permutacji może uczestniczyć jako lewy koniec łuku, prawy koniec łuku lub pozostać nienaruszony – dokładnie trzy stany odpowiadające strukturze Motzkina.

Rekurencje. Podstawowa rekurencja konwolucyjna wynika z rozkładu pierwszego punktu:

$$M_0 = 1, \quad M_1 = 1,$$
$$M_n = M_{n-1} + \sum_{k=0}^{n-2} M_k \, M_{n-2-k} \quad (n \geq 2).$$

Interpretacja: jeśli punkt 0 jest izolowany, pozostaje M_{n-1} konfiguracji; jeśli otwiera łuk zamykany w punkcie $k+1$, wnętrze (punkty $1 \dots k$) i zewnętrzny ogon (punkty $k+2 \dots n-1$) są niezależne, dając iloczyn $M_k M_{n-2-k}$.

Alternatywna rekurencja liniowa:

$$M_n = \frac{(2n+1)M_{n-1} + 3(n-1)M_{n-2}}{n+2}, \quad n \geq 2.$$

Funkcja tworząca:

$$M(x) = \sum_{n \geq 0} M_n x^n = \frac{1 - x - \sqrt{1 - 2x - 3x^2}}{2x^2}.$$

Asymptotyka Asymptotycznie $M_n \sim c\, 3^n / n^{3/2}$ dla stałej $c = \frac{3\sqrt{3}}{2\sqrt{\pi}}$. Przestrzeń dopuszczalnych ruchów złożonych rośnie znacznie wolniej niż $2^{\binom{n}{2}}$ (zbiór wszystkich podzbiorów par indeksów), czyniąc pełne programowanie dynamiczne praktycznym.

Tabela 1: Pierwsze liczby Motzkina ($n = 0..10$).

n	0	1	2	3	4	5	6	7	8	9	10
M_n	1	1	2	4	9	21	51	127	323	835	2188

Przykład: obliczanie M_4. Stosując rekurencję konwolucyjną:

$$M_2 = M_1 + M_0 M_0 = 1 + 1 = 2,$$
$$M_3 = M_2 + (M_0 M_1 + M_1 M_0) = 2 + 2 = 4,$$
$$M_4 = M_3 + (M_0 M_2 + M_1 M_1 + M_2 M_0) = 4 + (2 + 1 + 2) = 9.$$

Mając ustalone kombinatoryczne podstawy liczb Motzkina, przechodzimy teraz do konstrukcji sąsiedztwa dla permutacyjnego problemu przepływowego.

3 Struktura Sąsiedztwa Motzkina

Proponowane sąsiedztwo reprezentuje przejście od klasycznych ruchów jednoparametrowych (np. pojedyncza zamiana sąsiadujących elementów) do *strukturalnie złożonych* multi-ruchów, które mogą transformować permutację w jednym kroku poprzez kilka skoordynowanych lokalnych modyfikacji – podobnie do Dynasearch lub sąsiedztwa Fibonacciego.

Przewodnia idea polega na ograniczeniu eksplozji kombinatorycznej poprzez narzucenie ścisłej, łatwej do zweryfikowania struktury: każda para indeksów (i, j) opisuje zamianę końców segmentu, a zbiór takich par jest dopuszczalny tylko wtedy, gdy odpowiadające łuki (interpretowane jako połączenia $i \to j$ nad linią indeksów) są *nieprzecinające się*, mogą być *zagnieżdżone*, ale nie dzielą końców. W ten sposób w jednej iteracji przeszukujemy szerszy horyzont potencjalnych rozwiązań, zachowując kontrolę złożoności.

3.1 Definicja Ruchu i Warunki Dopuszczalności

Zamiana końców. Dla permutacji $\pi = (\pi_0, \pi_1, \ldots, \pi_{n-1})$ i pary indeksów (i, j) spełniającej $0 \le i < j < n$, definiujemy *zamianę końców* jako operację:

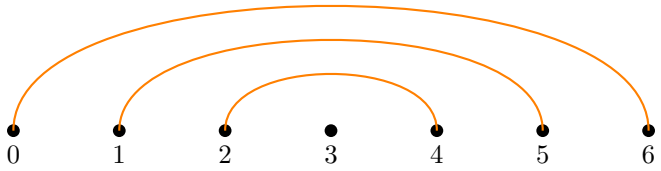
$$S_{i,j}(\pi) = (\pi_0, \ldots, \pi_{i-1},\ \pi_j,\ \pi_{i+1}, \ldots, \pi_{j-1},\ \pi_i,\ \pi_{j+1}, \ldots, \pi_{n-1}).$$

Ta operacja zamienia elementy na pozycjach i i j, zachowując kolejność wszystkich elementów pośrednich $\pi_{i+1}, \ldots, \pi_{j-1}$. Lokalnie segment $[\pi_i, \pi_{i+1}, \ldots, \pi_{j-1}, \pi_j]$ przekształca się w $[\pi_j, \pi_{i+1}, \ldots, \pi_{j-1}, \pi_i]$.

Ruch złożony: zbiór współbieżnych zamian. *Ruch złożony* jest zbiorem $\mathcal{M} = \{(i_1, j_1), (i_2, j_2), \ldots, (i_k, j_k)\}$ par indeksów. Zbiór $\mathcal{M}$ jest *dopuszczalny*, jeśli spełnia następujące warunki:

1. **Rozłączność końców**: Dla dowolnych dwóch par $(i_a, j_a), (i_b, j_b) \in \mathcal{M}$ mamy $\{i_a, j_a\} \cap \{i_b, j_b\} = \varnothing$ (żadne dwie pary nie dzielą indeksu).
2. **Brak przecięć**: Wzór $i_a < i_b < j_a < j_b$ jest zabroniony dla dowolnych $(i_a, j_a), (i_b, j_b) \in \mathcal{M}$ (łuki nie mogą się przecinać).
3. **Zagnieżdżanie dozwolone**: Konfiguracja $i_a < i_b < j_b < j_a$ jest akceptowalna (para wewnętrzna całkowicie zawarta w parze zewnętrznej).

Zastosowanie wszystkich zamian ze zbioru $\mathcal{M}$ w kolejności rosnących lewych końców i_k daje unikalny wynik – warunki (1)–(3) gwarantują, że żadna para nie interferuje z efektem innej pary. Każda para modyfikuje tylko dwa elementy permutacji na pozycjach końcowych; wewnętrzny segment zachowuje kolejność, a symulacja zagnieżdżonych łuków przebiega niezależnie.



Rysunek 1: Przykład dopuszczalnego ruchu złożonego: zagnieżdżone łuki $(0, 6)$, $(1, 5)$, $(2, 4)$; punkt 3 izolowany.

Bijekcja z obiektami Motzkina. Kluczowa obserwacja jest taka, że zbiór dopuszczalnych ruchów złożonych dla n indeksów ma moc dokładnie równą M_n (liczbie Motzkina). Interpretacja: każdy indeks $0 \le i < n$ może być w jednym z trzech stanów:

- **Lewy koniec łuku**: istnieje para $(i, j) \in \mathcal{M}$.
- **Prawy koniec łuku**: istnieje para $(k, i) \in \mathcal{M}$.
- **Izolowany**: indeks nie uczestniczy w żadnej parze.

Ta właściwość trzech stanów odpowiada dokładnie definicji obiektów Motzkina (ścieżki z krokami $+1$, 0, -1 lub rodziny łuków z izolowanymi punktami). Warunki (1)–(3) kodują nieprzecinanie się i rozłączność, co jest istotą struktury Motzkina.

Rozmiar przestrzeni i implikacje algorytmiczne. Przestrzeń dopuszczalnych ruchów złożonych ma rozmiar $M_n \sim c \cdot 3^n/n^{3/2}$, który jest znacznie mniejszy niż $2^{\binom{n}{2}}$ (wszystkie podzbiory par indeksów). Ten kontrolowany rozmiar przestrzeni uzasadnia użycie pełnego programowania dynamicznego do wyboru optymalnego ruchu złożonego – w przeciwieństwie do naiwnego przeszukiwania wykładniczej liczby kombinacji, algorytm DP operuje na $O(n^2)$ stanach, rozważając $O(n)$ kandydatów na stan.

4 Algorytm Wyboru Optymalnego Ruchu Złożonego

Mając zdefiniowaną przestrzeń dopuszczalnych ruchów złożonych, potrzebujemy algorytmu, który w czasie wielomianowym wybiera zbiór par $\mathcal{M}$ minimalizujący makespan. Algorytm składa się z pięciu kroków:

1. Obliczanie kolumn zakończenia prefiksów $F[k][r]$ (preprocessing).
2. Enumeracja delt $\Delta_{i,j}$ dla wszystkich par (i, j).
3. Programowanie dynamiczne: wypełnianie tablicy $\mathrm{dp}[L][R]$.
4. Rekonstrukcja wybranego zbioru par $\mathcal{M}$.
5. Zastosowanie zamian do permutacji π.

4.1 Obliczanie delt

Dla każdej pary indeksów (i, j) z $0 \le i < j < n$ obliczamy zmianę makespan:

$$\Delta_{i,j} = C_{\max}(S_{i,j}(\pi)) - C_{\max}(\pi).$$

W implementacji używamy *kolumn zakończenia prefiksów* $F[k][r]$, gdzie $F[k][r]$ oznacza czas zakończenia na maszynie r po przetworzeniu pierwszych k zadań permutacji π. Obliczamy te kolumny używając standardowego DP dla FSP:

$$\begin{aligned}
&F[0][r] = 0 \quad \text{dla wszystkich } r,\\
&F[k][0] = F[k-1][0] + p_{0,\pi_{k-1}},\\
&F[k][r] = \max(F[k][r-1], F[k-1][r]) + p_{r,\pi_{k-1}}, \quad r \ge 1,
\end{aligned}$$

gdzie $p_{r,j}$ jest czasem przetwarzania zadania j na maszynie r.

Następnie dla pary (i, j), zaczynając od stanu prefiksu $F[i]$, symulujemy nową kolejność segmentu:

$$[\pi_j, \pi_{i+1}, \ldots, \pi_{j-1}, \pi_i],$$

a następnie dokańczamy symulacją ogona $\pi_{j+1}, \ldots, \pi_{n-1}$. Koszt pojedynczej delty: $O(m(n-i))$. Pełna enumeracja wszystkich $\binom{n}{2}$ par: $O(mn^3)$.

4.2 Rekurencja programowania dynamicznego

Definiujemy funkcję $\mathrm{dp}[L][R]$ jako *minimalną sumę delt* osiągalną przez wybór dopuszczalnego zbioru par w przedziale indeksów $[L, R]$. Warunki brzegowe:

$$\mathrm{dp}[L][R] = 0 \quad \text{dla } L \ge R.$$

Dla przedziału $[L, R]$ z $L < R$ rozważamy dwa przypadki:

1. **Pomiń indeks** L: nie używamy go jako lewego końca żadnego łuku. Wartość: $\mathrm{dp}[L+1][R]$.

2. **Wybierz łuk** (L, k) dla pewnego $k \in \{L+1, \ldots, R\}$: wtedy przedział dzieli się na:

   - Wnętrze łuku: $[L+1, k-1]$ (jeśli niepuste) z wartością $\mathrm{dp}[L+1][k-1]$.
   - Ogon po prawej: $[k+1, R]$ (jeśli niepusty) z wartością $\mathrm{dp}[k+1][R]$.

   Łączna wartość: $\Delta_{L,k} + \mathrm{dp}[L+1][k-1] + \mathrm{dp}[k+1][R]$.

Rekurencja:

$$\mathrm{dp}[L][R] = \min\left\{\mathrm{dp}[L+1][R],\ \min_{k=L+1}^{R} \left(\Delta_{L,k} + \mathrm{dp}[L+1][k-1] + \mathrm{dp}[k+1][R]\right)\right\}.$$

Wypełniamy tablicę dla rosnących długości przedziałów $\ell = R - L + 1$ od $\ell = 2$ do $\ell = n$. Końcowa wartość $\mathrm{dp}[0][n-1]$ daje optymalną całkowitą deltę dla całej permutacji.

Chociaż delty pojedynczych par nie są ściśle addytywne (makespan zależy globalnie od całej permutacji), struktura nieprzecinająca się zapewnia, że lokalne efekty zamian mogą być efektywnie agregowane w algorytmie DP.

4.3 Rekonstrukcja wybranego zbioru par

Podczas wypełniania tablicy dp zapamiętujemy dla każdego przedziału $[L, R]$ optymalny wybór w tablicy $\mathrm{choice}[L][R]$:

$$\mathrm{choice}[L][R] = \begin{cases} \text{None}, & \text{jeśli pomijamy } L, \\ k, & \text{jeśli wybieramy łuk } (L, k). \end{cases}$$

Rekonstrukcja przebiega rekurencyjnie:

```
function reconstruct(L, R):
    if L >= R: return
    k = choice[L][R]
    if k == None:
        reconstruct(L+1, R)
    else:
        dodaj parę (L, k) do wyniku
        reconstruct(L+1, k-1)
        reconstruct(k+1, R)
```

Wynik: lista par $\mathcal{M} = \{(i_1, j_1), (i_2, j_2), \ldots\}$ spełniająca wszystkie warunki dopuszczalności (nieprzecinanie się, brak wspólnych końców, dozwolone zagnieżdżanie).

4.4 Zastosowanie ruchu złożonego

Zastosowujemy otrzymany zbiór par $\mathcal{M}$ do permutacji π w kolejności rosnących lewych końców:

$$\text{dla każdej pary } (i, j) \in \mathcal{M} \text{ (posortowanej wg } i): \quad \pi[i] \leftrightarrow \pi[j].$$

Ze względu na warunki dopuszczalności zamiany są wzajemnie niezależne – każdy indeks pojawia się w co najwyżej jednej parze.

4.5 Złożoność obliczeniowa

Czas.

- Preprocessing (kolumny prefiksów F): $O(mn)$.
- Enumeracja delt $\Delta_{i,j}$ dla wszystkich par: $O(mn^3)$.
- Wypełnianie tablicy DP: $O(n^2)$ przedziałów, każdy z $O(n)$ kandydatami $k \to O(n^3)$.
- Rekonstrukcja: $O(n)$ (liczba par $\leq n/2$).
- Zastosowanie zamian: $O(n)$.

Całkowita złożoność jednej iteracji sąsiedztwa: $\mathbf{O(mn^3)}$.

Pamięć.

- Tablica delt Δ: $O(n^2)$.
- Tablice DP (dp, choice): $O(n^2)$.
- Kolumny prefiksów F: $O(mn)$.

Łącznie: $\mathbf{O(n^2 + mn)}$.

5 Przykład Ilustracyjny

Rozważmy małą instancję FSP z $n = 5$ zadaniami i $m = 3$ maszynami. Początkowa permutacja:

$$\pi = (0, 1, 2, 3, 4).$$

Czasy przetwarzania (wiersze = maszyny, kolumny = zadania):

$$P = \begin{bmatrix} 3 & 2 & 5 & 4 & 3 \\ 2 & 4 & 3 & 2 & 5 \\ 4 & 3 & 2 & 5 & 4 \end{bmatrix}$$

Krok 1: Bazowy makespan Obliczamy $C_{\max}(\pi)$ używając standardowego DP:

Maszyna	J_0	J_1	J_2	J_3	J_4
0	3	5	10	14	17
1	5	9	13	16	21
2	9	12	15	21	25

Bazowy makespan: $C_{\max}(\pi) = 25$.

Krok 2: Kolumny prefiksów Tablica $F[k][r]$ (czas zakończenia na maszynie r po k zadaniach):

$$F = \begin{bmatrix} 0 & 0 & 0 \\ 3 & 5 & 9 \\ 5 & 9 & 12 \\ 10 & 13 & 15 \\ 14 & 16 & 21 \\ 17 & 21 & 25 \end{bmatrix}$$

Krok 3: Enumeracja delt Dla każdej pary (i, j) symulujemy zamianę końców i obliczamy $\Delta_{i,j} = C_{\max}(S_{i,j}(\pi)) - 25$.

Przykład: para $(1, 3)$**.** Nowa kolejność segmentu: $[\pi_3, \pi_2, \pi_1] = [3, 2, 1]$. Zaczynając od $F[1] = (3, 5, 9)$ symulujemy:

- Zadanie 3: kolumna $(3 + 4, \max(7, 5) + 2, \max(9, 7) + 5) = (7, 9, 14)$
- Zadanie 2: kolumna $(7 + 5, \max(12, 9) + 3, \max(15, 12) + 2) = (12, 15, 17)$
- Zadanie 1: kolumna $(12 + 2, \max(14, 15) + 4, \max(19, 14) + 3) = (14, 19, 22)$
- Zadanie 4 (ogon): kolumna $(14 + 3, \max(17, 19) + 5, \max(24, 17) + 4) = (17, 24, 28)$

Nowy makespan: $C_{\max}(S_{1,3}(\pi)) = 28$, więc $\Delta_{1,3} = 28 - 25 = +3$ (pogorszenie).

Przykład: para $(0, 4)$**.** Nowa kolejność: $[4, 1, 2, 3, 0]$. Zaczynając od $F[0] = (0, 0, 0)$ symulujemy całą sekwencję:

- Zadanie 4: $(3, 5, 9)$
- Zadanie 1: $(5, 9, 12)$
- Zadanie 2: $(10, 13, 15)$
- Zadanie 3: $(14, 16, 21)$
- Zadanie 0: $(17, 19, 25)$

Nowy makespan: $C_{\max} = 25$, więc $\Delta_{0,4} = 0$ (brak zmiany).

Załóżmy, że po pełnej enumeracji otrzymujemy:

$$\Delta = \begin{bmatrix} 0 & +2 & -1 & 0 \\ & +1 & +3 & -2 \\ & & +2 & -3 \\ & & & +1 \\ & & & \end{bmatrix}$$

(tylko górna część trójkątna).

Krok 4: Programowanie dynamiczne Wypełniamy tablicę $\mathrm{dp}[L][R]$ dla przedziałów rosnącej długości.

Długość 2:

$$\begin{aligned}
&\mathrm{dp}[0][1] = \min\{\mathrm{dp}[1][1],\ \Delta_{0,1}\} = \min\{0, 0\} = 0, \quad \mathrm{choice}[0][1] = 1,\\
&\mathrm{dp}[1][2] = \min\{0, +1\} = 0, \quad \mathrm{choice}[1][2] = \mathrm{None},\\
&\mathrm{dp}[2][3] = \min\{0, +2\} = 0, \quad \mathrm{choice}[2][3] = \mathrm{None},\\
&\mathrm{dp}[3][4] = \min\{0, +1\} = 0, \quad \mathrm{choice}[3][4] = \mathrm{None}.
\end{aligned}$$

Długość 3:

$$\begin{aligned}
\mathrm{dp}[0][2] &= \min\{\mathrm{dp}[1][2],\ \Delta_{0,1} + \mathrm{dp}[2][2],\ \Delta_{0,2} + \mathrm{dp}[1][1]\}\\
&= \min\{0,\ 0 + 0,\ 2 + 0\} = 0, \quad \mathrm{choice}[0][2] = \mathrm{None},\\
\mathrm{dp}[1][3] &= \min\{0,\ 1 + 0,\ 3 + 0\} = 0, \quad \mathrm{choice}[1][3] = \mathrm{None},\\
\mathrm{dp}[2][4] &= \min\{0,\ 2 + 0,\ -3 + 0\} = -3, \quad \mathrm{choice}[2][4] = 4.
\end{aligned}$$

Długość 4:

$$\begin{aligned}\text{dp}[0][3] &= \min\{\text{dp}[1][3],\ \Delta_{0,1} + \text{dp}[2][3],\ \Delta_{0,2} + \text{dp}[1][2],\ \Delta_{0,3} + \text{dp}[1][2]\} \\ &= \min\{0,\ 0+0,\ 2+0,\ -1+0\} = -1, \quad \text{choice}[0][3] = 3, \\ \text{dp}[1][4] &= \min\{-3,\ 1+0,\ 3+0,\ -2+0\} = -3, \quad \text{choice}[1][4] = \text{None}.\end{aligned}$$

Długość 5:

$$\begin{aligned}\text{dp}[0][4] = \min\{&\text{dp}[1][4],\ \Delta_{0,1} + \text{dp}[2][4],\ \Delta_{0,2} + \text{dp}[1][3], \\ &\Delta_{0,3} + \text{dp}[1][2] + \text{dp}[4][4],\ \Delta_{0,4} + \text{dp}[1][3]\} \\ = \min\{&-3,\ 0+(-3),\ 2+0,\ -1+0+0,\ 0+0\} = -3, \\ &\text{choice}[0][4] = \text{None (pomiń 0)}.\end{aligned}$$

Końcowa wartość: $\text{dp}[0][4] = -3$ (możliwa redukcja makespan o 3).

Krok 5: Rekonstrukcja Zaczynając od $[0, 4]$:

1. $\text{choice}[0][4] = \text{None} \rightarrow$ pomiń 0, rekursja dla $[1, 4]$.
2. $\text{choice}[1][4] = \text{None} \rightarrow$ pomiń 1, rekursja dla $[2, 4]$.
3. $\text{choice}[2][4] = 4 \rightarrow$ wybierz parę $(2, 4)$, rekursja dla $[3, 3]$ i $[5, 4]$ (puste).

Wybrany zbiór: $\mathcal{M} = \{(2, 4)\}$.

Krok 6: Zastosowanie Zamieniamy $\pi[2] \leftrightarrow \pi[4]$:

$$\pi' = (0, 1, 4, 3, 2).$$

Obliczamy nowy makespan:

$$C_{\max}(\pi') = 25 - 3 = 22.$$

Podsumowanie przykładu Algorytm DP wybrał optymalną parę $(2, 4)$ spośród wszystkich $\binom{5}{2} = 10$ możliwych par i ich kombinacji, osiągając redukcję makespan o 3 jednostki ze złożonością $O(mn^3) = O(3 \cdot 125) = O(375)$ podstawowych operacji.

6 Porównanie z Klasycznymi Sąsiedztwami

- **Zamiana sąsiadujących**: ekstremalnie tania ($O(mn)$ na ruch), ale płytka – wymaga wielu kroków, aby osiągnąć permutację dostępną przez pojedynczy ruch Motzkina. Wysoka liczba iteracji na jednostkę czasu, ale ograniczona eksploracja przestrzeni.
- **Sąsiedztwo Fibonacciego (złożone rozłączne)**: pozwala zebrać kilka rozłącznych zamian w jednym kroku, ale bez zagnieżdżania łuków. Sąsiedztwo Motzkina rozszerza ten pomysł o pełną hierarchię zagnieżdżonych struktur, zwiększając bogactwo pojedynczego ruchu.
- **Dynasearch**: eksploruje sekwencje ruchów strukturalnych (często 2-wymianę w różnych segmentach), ale jego przestrzeń nie jest oparta na spójnym obiekcie kombinatorycznym. Może być głębszy, ale z większym narzutem implementacyjnym i bardziej złożoną kontrolą przypadków.

Sąsiedztwo Motzkina zajmuje pozycję pośrednią: bogatsze niż zamiana sąsiadujących i złożone rozłączne, bardziej strukturalne niż Dynasearch, zachowując jednocześnie wyraźne fundamenty matematyczne (liczby Motzkina).

A Pełna Enumeracja Ruchów Złożonych Motzkina

A.1 Enumeracja Ruchów Złożonych dla $n = 0$

Dla $n = 0$ (brak indeksów) istnieje tylko pusta konfiguracja.

1. $\mathcal{M}_1 = \varnothing$

Weryfikacja: $1 = M_0$. ✓

A.2 Enumeracja Ruchów Złożonych dla $n = 1$

Dla $n = 1$ istnieje tylko jeden indeks (indeks 0), więc jedyną dopuszczalną konfiguracją jest zbiór pusty.

1. $\mathcal{M}_1 = \varnothing$ (indeks 0 izolowany)

Weryfikacja: $1 = M_1$. ✓

A.3 Enumeracja Ruchów Złożonych dla $n = 2$

Dla $n = 2$ mamy dwa indeksy: 0 i 1.

Kategoria I: Pusty Ruch Złożony

1. $\mathcal{M}_1 = \varnothing$ (oba indeksy izolowane)

Kategoria II: Jedna Para

2. $\mathcal{M}_2 = \{(0, 1)\}$

Weryfikacja: $1 + 1 = 2 = M_2$. ✓

A.4 Enumeracja Ruchów Złożonych dla $n = 3$

Dla $n = 3$ mamy trzy indeksy: 0, 1, 2.

Kategoria I: Pusty Ruch Złożony

1. $\mathcal{M}_1 = \varnothing$

Kategoria II: Jedna Para

2. $\mathcal{M}_2 = \{(0, 1)\}$
3. $\mathcal{M}_3 = \{(0, 2)\}$
4. $\mathcal{M}_4 = \{(1, 2)\}$

Weryfikacja: $1 + 3 = 4 = M_3$. ✓
Uwaga: Dla $n = 3$ nie jest możliwe utworzenie dwóch rozłącznych par (wymaga minimum 4 indeksów).

A.5 Enumeracja Ruchów Złożonych dla $n = 4$

Dla $n = 4$ mamy cztery indeksy: 0, 1, 2, 3.

Kategoria I: Pusty Ruch Złożony

1. $\mathcal{M}_1 = \varnothing$

Kategoria II: Jedna Para

2. $\mathcal{M}_2 = \{(0,1)\}$
3. $\mathcal{M}_3 = \{(0,2)\}$
4. $\mathcal{M}_4 = \{(0,3)\}$
5. $\mathcal{M}_5 = \{(1,2)\}$
6. $\mathcal{M}_6 = \{(1,3)\}$
7. $\mathcal{M}_7 = \{(2,3)\}$

Kategoria III: Dwie Rozłączne Pary

8. $\mathcal{M}_8 = \{(0,1),(2,3)\}$

Kategoria IV: Dwie Zagnieżdżone Pary

9. $\mathcal{M}_9 = \{(0,3),(1,2)\}$

Weryfikacja: $1 + 6 + 1 + 1 = 9 = M_4$. ✓

A.6 Enumeracja Ruchów Złożonych dla $n = 5$

Dla $n = 5$ mamy pięć indeksów: 0, 1, 2, 3, 4.

Kategoria I: Pusty Ruch Złożony

1. $\mathcal{M}_1 = \varnothing$

Kategoria II: Jedna Para

2. $\mathcal{M}_2 = \{(0,1)\}$
3. $\mathcal{M}_3 = \{(0,2)\}$
4. $\mathcal{M}_4 = \{(0,3)\}$
5. $\mathcal{M}_5 = \{(0,4)\}$
6. $\mathcal{M}_6 = \{(1,2)\}$
7. $\mathcal{M}_7 = \{(1,3)\}$
8. $\mathcal{M}_8 = \{(1,4)\}$
9. $\mathcal{M}_9 = \{(2,3)\}$
10. $\mathcal{M}_{10} = \{(2,4)\}$
11. $\mathcal{M}_{11} = \{(3,4)\}$

Kategoria III: Dwie Rozłączne Pary

12. $\mathcal{M}_{12} = \{(0,1),(2,3)\}$
13. $\mathcal{M}_{13} = \{(0,1),(2,4)\}$
14. $\mathcal{M}_{14} = \{(0,1),(3,4)\}$
15. $\mathcal{M}_{15} = \{(0,2),(3,4)\}$
16. $\mathcal{M}_{16} = \{(1,2),(3,4)\}$

Kategoria IV: Dwie Zagnieżdżone Pary

17. $\mathcal{M}_{17} = \{(0,3),(1,2)\}$
18. $\mathcal{M}_{18} = \{(0,4),(1,2)\}$
19. $\mathcal{M}_{19} = \{(0,4),(1,3)\}$
20. $\mathcal{M}_{20} = \{(0,4),(2,3)\}$
21. $\mathcal{M}_{21} = \{(1,4),(2,3)\}$

Weryfikacja: $1 + 10 + 5 + 5 = 21 = M_5$. ✓

A.7 Enumeracja Ruchów Złożonych dla $n = 6$

Dla $n = 6$ mamy sześć indeksów: 0, 1, 2, 3, 4, 5. Zgodnie z teorią $M_6 = 51$.

Kategoria I: Pusty Ruch Złożony

1. $\mathcal{M}_1 = \varnothing$

Liczba: 1

Kategoria II: Jedna Para Wszystkie $\binom{6}{2} = 15$ pary:

2. $\mathcal{M}_2 = \{(0,1)\}$
3. $\mathcal{M}_3 = \{(0,2)\}$
4. $\mathcal{M}_4 = \{(0,3)\}$
5. $\mathcal{M}_5 = \{(0,4)\}$
6. $\mathcal{M}_6 = \{(0,5)\}$
7. $\mathcal{M}_7 = \{(1,2)\}$
8. $\mathcal{M}_8 = \{(1,3)\}$
9. $\mathcal{M}_9 = \{(1,4)\}$
10. $\mathcal{M}_{10} = \{(1,5)\}$
11. $\mathcal{M}_{11} = \{(2,3)\}$
12. $\mathcal{M}_{12} = \{(2,4)\}$
13. $\mathcal{M}_{13} = \{(2,5)\}$
14. $\mathcal{M}_{14} = \{(3,4)\}$
15. $\mathcal{M}_{15} = \{(3,5)\}$
16. $\mathcal{M}_{16} = \{(4,5)\}$

Liczba: 15

Kategoria III: Dwie Rozłączne Pary Pary (i_1, j_1), (i_2, j_2) z $j_1 < i_2$:

17. $\mathcal{M}_{17} = \{(0,1),(2,3)\}$
18. $\mathcal{M}_{18} = \{(0,1),(2,4)\}$
19. $\mathcal{M}_{19} = \{(0,1),(2,5)\}$
20. $\mathcal{M}_{20} = \{(0,1),(3,4)\}$
21. $\mathcal{M}_{21} = \{(0,1),(3,5)\}$
22. $\mathcal{M}_{22} = \{(0,1),(4,5)\}$
23. $\mathcal{M}_{23} = \{(0,2),(3,4)\}$
24. $\mathcal{M}_{24} = \{(0,2),(3,5)\}$
25. $\mathcal{M}_{25} = \{(0,2),(4,5)\}$

26. $\mathcal{M}_{26} = \{(0,3),(4,5)\}$
27. $\mathcal{M}_{27} = \{(1,2),(3,4)\}$
28. $\mathcal{M}_{28} = \{(1,2),(3,5)\}$
29. $\mathcal{M}_{29} = \{(1,2),(4,5)\}$
30. $\mathcal{M}_{30} = \{(1,3),(4,5)\}$
31. $\mathcal{M}_{31} = \{(2,3),(4,5)\}$

Liczba: 15

Kategoria IV: Dwie Zagnieżdżone Pary Struktura $i_a < i_b < j_b < j_a$:

32. $\mathcal{M}_{32} = \{(0,3),(1,2)\}$
33. $\mathcal{M}_{33} = \{(0,4),(1,2)\}$
34. $\mathcal{M}_{34} = \{(0,4),(1,3)\}$
35. $\mathcal{M}_{35} = \{(0,4),(2,3)\}$
36. $\mathcal{M}_{36} = \{(0,5),(1,2)\}$
37. $\mathcal{M}_{37} = \{(0,5),(1,3)\}$
38. $\mathcal{M}_{38} = \{(0,5),(1,4)\}$
39. $\mathcal{M}_{39} = \{(0,5),(2,3)\}$
40. $\mathcal{M}_{40} = \{(0,5),(2,4)\}$
41. $\mathcal{M}_{41} = \{(0,5),(3,4)\}$
42. $\mathcal{M}_{42} = \{(1,4),(2,3)\}$
43. $\mathcal{M}_{43} = \{(1,5),(2,3)\}$
44. $\mathcal{M}_{44} = \{(1,5),(2,4)\}$
45. $\mathcal{M}_{45} = \{(1,5),(3,4)\}$
46. $\mathcal{M}_{46} = \{(2,5),(3,4)\}$

Liczba: 15

Kategoria V: Trzy Rozłączne Pary

47. $\mathcal{M}_{47} = \{(0,1),(2,3),(4,5)\}$

Liczba: 1

Kategoria VI: Dwie Rozłączne + Jedna Zagnieżdżona

48. $\mathcal{M}_{48} = \{(0,3),(1,2),(4,5)\}$
49. $\mathcal{M}_{49} = \{(0,1),(2,5),(3,4)\}$

Liczba: 2

Kategoria VII: Trzy Podwójnie Zagnieżdżone Pary

50. $\mathcal{M}_{50} = \{(0,5),(1,4),(2,3)\}$

Liczba: 1

Kategoria VIII: Struktura Mieszana

51. $\mathcal{M}_{51} = \{(0,4),(1,3),(2,5)\}$

Liczba: 1
Weryfikacja: $1 + 15 + 15 + 15 + 1 + 2 + 1 + 1 = 51 = M_6$. ✓

A.8 Struktura Ruchów Złożonych dla $n = 7$

Dla $n = 7$ mamy $M_7 = 127$ konfiguracji. Pełna enumeracja zajęłaby wiele stron, więc przedstawiamy strukturę kombinatoryczną.

Główne Kategorie

Pusta konfiguracja: 1

Jedna para: $\binom{7}{2} = 21$

Dwie rozłączne pary: Wybieramy 4 indeksy z 7 i dzielimy na dwie pary: $\binom{7}{4} \cdot \frac{1}{2}\binom{4}{2} = 35 \cdot 3 = 105$

Ale musimy wykluczyć przecięcia. Prawidłowa liczba par rozłącznych (bez przecięć): **35**

Dwie zagnieżdżone pary: Wybieramy parę zewnętrzną (i, j) o rozpiętości ≥ 3, następnie parę wewnętrzną z $\{i+1, \ldots, j-1\}$: **35**

Trzy pary: Kombinacje trzech par (wszystkie rozłączne lub z zagnieżdżeniami): **21**

Cztery lub więcej par: Głęboko zagnieżdżone struktury i złożone kombinacje: **14**

Weryfikacja (przybliżona): $1 + 21 + 35 + 35 + 21 + 14 = 127 = M_7$. ✓

A.9 Struktura Ruchów Złożonych dla $n = 8$

Dla $n = 8$ mamy $M_8 = 323$ konfiguracje.

Rozkład Kategorii

- **Pusta:** 1
- **Jedna para:** $\binom{8}{2} = 28$
- **Dwie pary:** ≈ 126 (rozłączne + zagnieżdżone)
- **Trzy pary:** ≈ 112
- **Cztery pary:** ≈ 51
- **Więcej par:** ≈ 5

Weryfikacja rekurencją:

$$M_8 = M_7 + \sum_{k=0}^{6} M_k M_{6-k} = 127 + 196 = 323. \quad \checkmark$$

A.10 Struktura Ruchów Złożonych dla $n = 9$

Dla $n = 9$ mamy $M_9 = 835$ konfiguracji.

Rozkład Kategorii (przybliżony)

- **Pusta:** 1
- **Jedna para:** $\binom{9}{2} = 36$
- **Dwie pary:** ≈ 330
- **Trzy pary:** ≈ 294
- **Cztery pary:** ≈ 140
- **Pięć lub więcej:** ≈ 34

Weryfikacja rekurencją:

$$M_9 = M_8 + \sum_{k=0}^{7} M_k M_{7-k} = 323 + 512 = 835. \quad \checkmark$$

A.11 Struktura Ruchów Złożonych dla $n = 10$

Dla $n = 10$ mamy $M_{10} = 2188$ konfiguracje.

Rozkład Kategorii (przybliżony)

- **Pusta:** 1
- **Jedna para:** $\binom{10}{2} = 45$
- **Dwie pary:** ≈ 825
- **Trzy pary:** ≈ 770
- **Cztery pary:** ≈ 385
- **Pięć par:** ≈ 126
- **Więcej par:** ≈ 36

Weryfikacja rekurencją:

$$M_{10} = M_9 + \sum_{k=0}^{8} M_k M_{8-k} = 835 + 1353 = 2188. \quad \checkmark$$

Uwagi o Wzroście Kombinatorycznym Dla $n \geq 7$ liczba konfiguracji rośnie wykładniczo ($M_n \sim c \cdot 3^n/n^{3/2}$), czyniąc pełną enumerację niepraktyczną. Jednak algorytm programowania dynamicznego nadal działa efektywnie w czasie $O(n^3)$ bez potrzeby jawnego wyliczania wszystkich konfiguracji – rekurencja naturalnie eksploruje przestrzeń stanów.

Bijekcja z obiektami Motzkina pozostaje ważna dla zrozumienia teoretycznej struktury, nawet gdy nie wypisujemy jawnie wszystkich M_n ruchów złożonych.