Równoległe algorytmy sztucznej inteligencji
Lista 2.

Należy zaprojektować za pomocą biblioteki MPI lub PVM wybrany równoległy algorytm metaheurystyczny (np. **algorytm koewolucyjny**, **scatter serach**, **algorytm mrówkowy**) znajdujący dobre rozwiązanie przybliżone jednego z poniższych problemów. Dane dla problemu należy pobrać ze strony
OR-LIBRARY:
http://people.brunel.ac.uk/~mastjjb/jeb/info.html
dla problemów 1 i 2,
ze strony TSPLIB:
http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/
dla problemu TSP,
ze strony QAPLIB:
http://www.opt.math.tu-graz.ac.at/qaplib/
dla problemu QAP oraz
ze strony
http://wojciech.bozejko.staff.iiar.pwr.wroc.pl/benchmarks.html
dla pozostałych problemów.

1. Problem jednomaszynowy

Dany jest zbiór n ponumerowanych zadań $N=\{1,2, \ldots ,n\}$, które należy wykonać, bez przerywania, na jednej maszynie. Maszyna ta, w dowolnej chwili, może wykonywać co najwyżej jedno zadanie. Dla zadania i ($i=1,2, \ldots ,n$), niech p_i, w_i, d_i będą odpowiednio: *czasem wykonywania*, *wagą funkcji kosztów* oraz *linią krytyczną*. Jeżeli ustalona jest kolejność wykonywania zadań oraz C_i ($i=1,2,...,n$) jest terminem zakończenia wykonywania zadania i, to $T_i = \max\{0, C_i - d_i\}$ nazywamy *opóźnieniem*, a $f_i(C_i)=w_i*T_i$ *kosztem opóźnienia* zadania. Rozważany problem polega na wyznaczeniu takiej kolejności wykonywania zadań, która minimalizuje *sumę kosztów opóźnień*, tj. $\sum w_i T_i$.

Niech będzie Π zbiorem permutacji elementów z N. Dla permutacji $\pi \in \Pi$ przez:

$$F(\pi) = \sum_{i=1}^{n} w_{\pi(i)} T_{\pi(i)},$$

oznaczamy *koszt permutacji* (tj. sumę kosztów opóźnień, gdy zadania są wykonywane w kolejności występowania w π). Rozważany problem sprowadza się do wyznaczenia permutacji optymalnej (o minimalnym koszcie) w zbiorze wszystkich permutacji Π.

2. Problem przepływowy (Flow shop)

Dany jest zbiór n zadań $J=\{1,2,\ldots,n\}$ oraz zbiór m maszyn $M=\{1,2,\ldots,m\}$. Zadanie $j \in J$ jest ciągiem m operacji O_{j1}, O_{j2},…, O_{jm} Operację O_{jk} należy wykonać, bez przerywania, na maszynie k w czasie p_{jk}. Wykonywanie zadania na maszynie k (dla $k=2, \ldots ,m$) może się rozpocząć dopiero po zakończeniu wykonywania tego zadania na maszynie k-1. Należy wyznaczyć kolejność, minimalizującą czas wykonania wszystkich zadań.
Niech $\pi =(\pi(1), \pi(1),\ldots,\pi(n))$ będzie permutacją zadań $\{1,2,\ldots,n\}$, a Π zbiorem wszystkich takich permutacji. Każda permutacja $\pi \in \Pi$ wyznacza jednoznacznie kolejność wykonywania zadań na maszynach (na każdej taką samą). Tak więc w omawianym problemie należy wyznaczyć permutację $\pi^* \in \Pi$ taką, że:

$$C_{\max}(\pi^*) = \max_{\pi \in \Pi} C_{\max}(\pi),$$

gdzie $C_{max}(\pi)$ jest czasem zakończenia wykonywania wszystkich zadań na maszynach, gdy są one wykonywane w kolejności π (tj. zadanie $\pi(i)$ jest wykonywane jako i- te w kolejności, $i=1,2,...,n$).

3. Problem komiwojażera (TSP)

Danych jest *n* miast wraz z odległościami pomiędzy nimi (dane mogą być w postaci współrzędnych geograficznych miast). Należy ustalić trasę przejścia przez wszystkie miasta, przez każde raz, tak by suma pokonanych odległości była minimalna. Należy powrócić do miasta startowego (wyznaczyć cykl w odpowiednim grafie).

4. Kwadratowy problem przydziału (QAP)

Niech *n* będzie liczbą lokalizacji i *n* urządzeń ma być rozmieszczonych w tych lokalizacjach, każde w jednej. Niech c_{ij} będzie kosztem na jednostkę odległości pomiędzy urządzeniami *i* i *j* a d_{ij} odległością pomiędzy lokalizacjami *i* i *j*. Koszt który należy zminimalizować po wszystkich możliwych permutacjach :

$$f = \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij} d_{\pi(i)\pi(j)} .$$

5. Problem gniazdowy (Job shop)

Dany jest zbiór zadań $J=\{1,2,\ldots,n\}$, które należy wykonać na maszynach ze zbioru $M=\{1,2,\ldots,m\}$. Zadanie jest ciągiem pewnych operacji. Każdą operację należy wykonać bez przerywania na odpowiedniej maszynie w ustalonym czasie (operacja ma przypisaną maszynę na której ma być wykonana). Problem polega na wyznaczeniu kolejności wykonywania operacji na każdej maszynie, minimalizującej czas wykonania wszystkich zadań.

6. Jednomaszynowy problem szeregowania zadań z przezbrojeniami

Dany jest zbiór *n* ponumerowanych zadań $N=\{1,2, \ldots ,n\}$, które należy wykonać, bez przerywania, na jednej maszynie. Maszyna ta, w dowolnej chwili, może wykonywać co najwyżej jedno zadanie. Dla zadania *i* ($i=1,2, \ldots ,n$), niech p_i, w_i, d_i będą odpowiednio: *czasem wykonywania*, *wagą funkcji kosztów* oraz *żądanym terminem zakończenia*. Dane są także przezbrojenia s_{ij}, $i, j \in N$, reprezentujące czas potrzebny na przygotowanie maszyny do wykonywania zadani *j*, jeżeli bezpośrednio przed *j* było wykonywane zadanie *i*. Ponadto, s_{0i} jest czasem przygotowania maszyny, jeżeli zadanie *i* jest wykonywane jako pierwsze.

Dla ustalonej kolejności wykonywania zadań, niech C_i ($i=1,2,\ldots,n$) będzie terminem zakończenia wykonywania zadania *i*. Wówczas, $T_i = \max\{0, C_i - d_i\}$ nazywamy *opóźnieniem*, a $f_i(C_i)=w_i*T_i$ *kosztem opóźnienia*. Rozważany problem sprowadza się do wyznaczenia takiej kolejności wykonywania zadań, która minimalizuje *sumę kosztów opóźnień*, tj. $\sum w_i T_i$.
Niech Π będzie zbiorem permutacji elementów z *N*. Dla permutacji $\pi \in \Pi$, przez:

$$F(\pi) = \sum_{i=1}^{n} f_{\pi(i)}(C_{\pi(i)}),$$

oznaczamy *koszt permutacji* (tj. sumę kosztów opóźnień, gdy zadania są wykonywane w kolejności występowania w π), gdzie: $C_{\pi(i)} = \sum_{j=1}^{i} (s_{\pi(j-1)\pi(j)} + p_{\pi(j)})$, $\pi(0) = 0$. Rozwiązanie problemu sprowadza się więc do wyznaczenia permutacji optymalnej (o minimalnym koszcie) w zbiorze wszystkich permutacji Π.

6. Jednomaszynowy problem szeregowania E/T

Niech $J = \{1, 2, \ldots, n\}$ będzie zbiorem zadań do wykonania bez przerywania, na jednej maszynie, która w dowolnej chwili wykonuje co najwyżej jedno zadanie. Przez p_i oznaczamy czas wykonywania zadania, a przez e_i oraz d_i odpowiednio żądany najwcześniejszy i najpóźniejszy termin

zakończenia wykonywania zadania $i \in J$. Jeżeli ustalona jest kolejność wykonywania zadań oraz C_i jest *terminem zakończenia* zadania *i*, to $E_i = \max\{0, e_i - C_i\}$ nazywamy *przyśpieszeniem* (earliness), a $T_i = \max\{0, C_i - d_i\}$ *opóźnieniem* (tardiness) zadania. W przypadku, gdy $E_i = 0$ oraz $T_i = 0$, to zadanie jest nazywane *terminowym*. Wyrażenie $u_i E_i + w_i T$ jest *kosztem wykonania* zadania, gdzie u_i oraz w_i ($i \in J$) są nieujemnymi współczynnikami funkcji kosztu.
Problem minimalizacji sumy kosztów zadań nieterminowych (w skrócie ***TWET***), polega na minimalizacji funkcji:

$$\sum_{i=1}^{n} (u_i E_i + w_i T_i).$$

W literaturze jest on oznaczany przez $1||\sum(u_i E_i + w_i T_i)$ i należy on do klasy problemów silnie NP-trudnych. W optymalnym rozwiązaniu mogą występować przestoje maszyny (zadania nie muszą być wykonywane bezpośrednio jedno po drugim), czyli $C_{i+1} - p_{i+1} \geq C_i$, $i = 1, 2, \ldots, n-1$. Rozwiązanie problemu sprowadza się więc do ustalenia kolejności wykonywania zadań oraz momentów ich rozpoczęcia.