Zastosowanie programowania ewolucyjnego w celu rozwiązania problemu komiwojażera

Maciej Wieczór-Retman, Kacper Stojek

1 Wstęp

Problem komiwojażera (ang. *Traveling salesman problem*) jest problemem minimalizacji, polegającym na odnalezieniu minimalnego cyklu Hamiltona w pełnym grafie ważonym. Cykl Hamiltona to taki cykl w grafie, w którym każdy wierzchołek grafu jest odwiedzony dokładnie raz. Czyli rozwiązaniem problemu komiwojażera jest cykl o najmniejszej sumie wag krawędzi grafu. Liczba różnych cyklów Hamiltona H dla pełnego grafu nieskierowanego o n wierzchołkach jest wyrażana równaniem eq. (1).

$$H = \frac{(n-1)!}{2} \tag{1}$$

Z uwagi na liczbę różnych cyklów, przeszukiwanie wszystkich rozwiązań jest fizycznie niemożliwe dla większych grafów. Przykładowo stosunkowo mały graf o stu wierzchołkach, daje w przybliżeniu $4,7 \times 10^{155}$ rozwiązań. Dla przybliżenia zakładając, że sprawdzenie jednej ścieżki zajmuje $1\,\mu s$, przeanalizowanie wszystkich ścieżek zajęłoby $1,48 \times 10^{142}\,lat$. Z tego powodu ważne jest zastosowanie bardziej skomplikowanych algorytmów minimalizacyjnych, takich jak właśnie programowanie genetyczne. Dobrze skonstruowany algorytm powinien być w stanie znaleźć minimum globalne dla większości przypadków lub przynajmniej optymalne rozwiązanie.

Napisany algorytm będzie testowany na danych z www.math.uwaterloo.ca w celu weryfikacji jego działania.

2 Implementacja

Jak zostało wspomniane na wstępie, rozwiązaniem problemu komiwojażera jest najkrótszy cykl Hamiltona dla danego pełnego, ważonego i nieskierowanego grafu. Z tych informacji wynikają szczegóły implementacji takie jak funkcja przystosowania, reprezentacja osobników oraz ograniczenia co do sposobu mutacji, oraz krzyżowania osobników.

Ponieważ szukany jest cykl Hamiltona, w rozwiązaniu każdy wierzchołek grafu pojawi się tylko raz, to znaczy, że mając graf G o N wierzchołkach:

$$V(G) = \{v_1\, v_2\, \cdots v_N\}, \tag{2}$$

rozwiązaniem będzie pewna permutacja zbioru $V(G)$. Z uwagi na to, najłatwiej jest wykorzystać kodowanie całkowitoliczbowe. Z tym kodowaniem każdy wierzchołek dostaje swój indeks, a następnie każdy osobnik składa się z pewnej permutacji wierzchołków grafu, reprezentując pojedynczy cykl Hamiltona.

Mając zdefiniowany sposób zapisu osobników, można w oparciu o to zdefiniować funkcję przystosowania $f(x)$ dla osobnika x. Najpierw jednak należy zdefiniować funkcję kosztu $u(x)$. Ponieważ pojedynczy osobnik składa się z kolejnych indeksów wierzchołków, funkcją kosztu jest suma wag krawędzi pomiędzy kolejnymi wierzchołkami, oraz dodatkowo pomiędzy pierwszym a ostatnim wierzchołkiem w osobniku (szukany jest cykl grafu). Następnie funkcja przystosowania $f(x)$ jest następująca:

$$f(x) = C_{max} - u(x), \tag{3}$$

gdzie:

$f(x)$ - funkcja przystosowania,
$u(x)$ - funkcja kosztu,
C_{max} - parametr nie mniejszy niż maksymalna wartość $u(x)$ dla danej aktualnej populacji,
x - osobnik.

Z uwagi na naturę rozwiązania, metody krzyżowania osobników są ograniczone, ponieważ allele w nowopowstałym osobniku nie mogą się powtarzać. Dlatego przykładowo krzyżowanie arytmetycznie się nie nadaje. Możliwe sposoby krzyżowania to:

- PMX,
- OX,
- CX.

W projekcie użyto krzyżowania OX, ponieważ ta metoda krzyżowania najbardziej zachowuje kolejność alleli w osobniku. Zamysł jest taki, aby osobniki wymieniały się krótkimi trasami, zachowując ich kolejność. Dany osobnik na początku działania algorytmu będzie miał zoptymalizowany tylko fragment trasy. Krzyżując dwa osobniki, jest spora szansa, że ten fragment się zachowa i trafi do kolejnego osobnika. Za mieszanie kolejności raczej powinny być odpowiedzialne mutacje osobników. Metody krzyżowania PMX oraz OX za bardzo zmieniają szyk alleli, przez co nie zostały wykorzystane.

Podobnie jak w przypadku krzyżowania, osobniki po mutacji dalej muszą tworzyć poprawny ciąg Hamiltona. Z tego powodu mutacje ograniczają się do podmian elementów w strukturze osobnika, bez zmian ich wartości. Dopuszczalne sposoby mutacji to:

- inwersja (ang. inversion),
- wstawianie (ang. insertion),
- przenoszenie (ang. displacement),
- wzajemna wymiana (ang. reciprocal exchange).

W projekcie użyto inwersji oraz wstawiania. Przy mutacji, szczególnie w początkowej fazie działania algorytmu, ważne jest, aby mutacje miały charakter globalny. Inwersja polega na wylosowaniu podciągu w osobniku i odwrócenie w nim kolejności alleli, natomiast wstawianie polega na wylosowaniu pojedynczego allela, i wstawienie go w innym, losowo wybranym miejscu. Wstawianie pozwala na stworzenie nowych tras w osobnikach, tak zmiany są małe i algorytm przy tylko takim rodzaju mutacji, szybko się zbiega do minimum lokalnego. Dlatego, aby troch zwiększyć szansę na znalezienie innego, lepszego minimum, została dodana inwersja. Ponieważ zmiany przy inwersji są bardziej drastyczne, jest większa szansa, że populacja nie utknie od razu w minimum lokalnym. To czy zostanie wykonana inwersja czy wstawianie też zależy od obecnej iteracji. Im wyższa iteracja, tym większa szansa, że zostanie wykonane wstawianie zamiast inwersji.

Przy tworzeniu nowych osobników również trzeba uważać, aby nie tworzyć elity. Zostawianie i krzyżowanie tylko najlepszych osobników, prowadzi do przedwczesnego zbiegania się algorytmu do nieoptymalnego rozwiązania. Dlatego osobniki są niszowane w ten sposób, że osobniki z gorszym przystosowaniu mają większą szansę na trafienie do puli rodzicielskiej. W odwrotnej konfiguracji, gdzie lepsze osobniki miały większą szansę na trafienie do puli rodzicielskiej, algorytm przedwcześnie się zbiegał do suboptymlanego rozwiązania. Następnie po przeprowadzeniu krzyżowania i wyznaczeniu nowych osobników, są one poddawane mutacji.

Tworzenie nowej populacji tutaj, tj:

- jak są dobierani rodzice i krzyżowani,
- jak są wybierane nowe osobniki z otrzymanych pul osobników.

3 Wyniki pracy