Patryk Bandyra

Raport

Opis zadania:

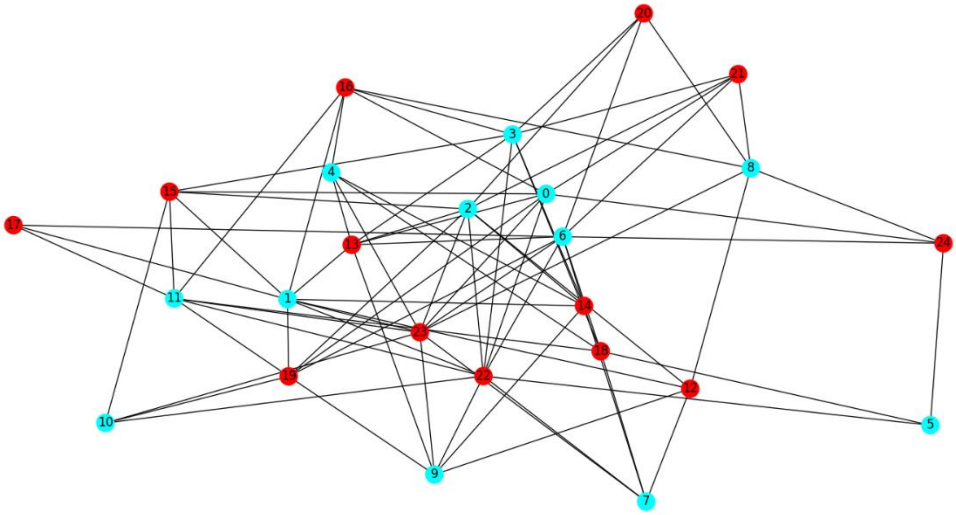
Galeria sztuki Surrealizmu Klasycznego planuje zakup kamer w celu zabezpieczenia swoich zasobów, które wystawione są na korytarzach (krawędź w grafie). Domyślnie potrzeba by zakupić tyle kamer, ile jest korytarzy w galerii, ale z uwagi na cięcia z budżetu miasta konieczna jest optymalizacja kosztów. Jeżeli kamera znajduje się w miejscu przecięcia korytarzy (węzeł w grafie) to jest w stanie pokryć wszystkie korytarze, które do niego dochodzą. Dla zadanego rozkładu galerii sztuki zaproponuj najmniejsze rozmieszczenie kamer, czyli takie, że pokryje ono wszystkie korytarze. (minimalizacja liczby wierzchołków) N=25 - liczba wierzchołków w grafie.

Opis działania programu:

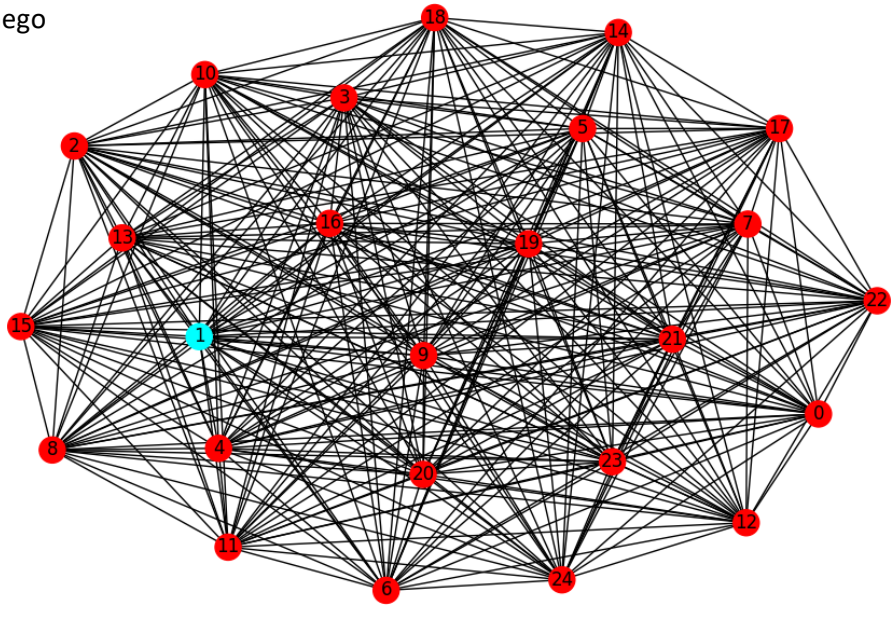
Uruchamiane są funkcje testujące algorytm ewolucyjny odpowiednio dla grafów: pełnego, losowego dwudzielnego oraz losowego (może być niespójny – w zależności od parametrów). Funkcje po zakończeniu wykonywania algorytmu wyświetlają odpowiednie grafy z pokolorowanymi wierzchołkami, gdzie wierzchołek czerwony odpowiada wierzchołkowi pokrytemu, a niebieski niepokrytemu.

Przykładowy wynik działania programu dla:

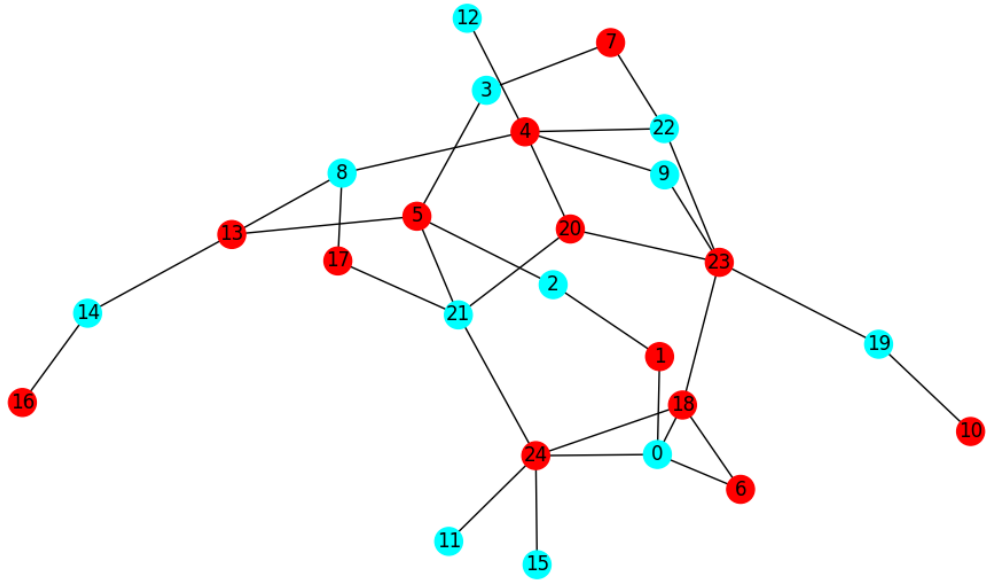
- losowego grafu dwudzielnego



- grafu pełnego



- losowego grafu (może być niespójny)



Opis działania algorytmu:

*przyjąłem założenie, że nie chcemy w populacji osobników niespełniających warunku pokrycia, gdyż może się to skończyć tym, że w końcowej generacji będziemy mieć same osobniki niespełniające warunku, ponieważ funkcja oceny bierze pod uwagę jak najmniejszą liczbę pokrytych wierzchołków

Populacja startowa jest inicjowana tylko członkami spełniającymi warunek pokrycia grafu. Następnie rozpoczyna się pętla algorytmu ewolucyjnego, gdzie warunkiem wyjścia jest osiągnięcie maksymalnej liczby wywołań funkcji oceny. Wewnątrz pętli:

1. Dokonywana jest selekcja turniejowa, gdzie w każdym turnieju może brać udział k członków. Liczba turniejów: math.ceil(len(population)/k). Członkowie, którzy brali udział w turnieju, ale go nie wygrali, nadal mogą wziąć udział w kolejnym.
2. Na populacji wyłonionej z selekcji turniejowej dokonywana jest mutacja. Zajście mutacji dla danego osobnika jest zależne od parametru prawdopodobieństwa. Mutacja osobnika polega na znalezieniu w jego liście wierzchołków wierzchołka pokrytego i jego odkryciu. Następnie sprawdzany jest warunek pokrycia danego grafu dla zmodyfikowanego członka. Jeśli warunek jest niespełniony następuje cofnięcie mutacji oraz kontynuuje się próbę zmodyfikowania danego osobnika, aż będzie spełniał warunek pokrycia lub zostanie osiągnięta maksymalna liczba prób mutacji.
3. Następnie łączy się populację bazową z populacją, która uległa mutacji.
4. Na podstawie tej połączonej populacji za pomocą funkcji oceny tworzy się populację potomną, która dla kolejnej iteracji algorytmu staje się populacją bazową (lub zostaje zwrócona gdy warunek stopu zostanie spełniony).

Algorytm przetestowano na grafach: pełnym, losowym dwudzielnym, losowym (z losową liczbą krawędzi; z możliwą niespójnością)

Wnioski:

1. Wpływ liczby osobników populacji na jakość uzyskanych rozwiązań:

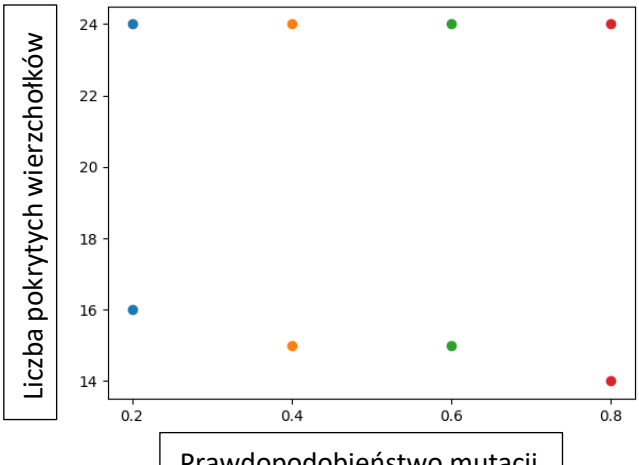
*na wykresie dla każdej liczebności populacji – najgorszy i najlepszy osobnik ostatniej generacji (jeśli jeden punkt -> najgorszy = najlepszy)

Jeśli zwiększamy liczebność populacji, a warunek stopu (maksymalna liczba wywołań funkcji oceny) pozostaje bez zmian, to oznacza mniejszą liczbę generacji populacji, czyli zmniejszamy możliwość modyfikowania poszczególnych osobników – jakość znalezionych rozwiązań będzie niska. Jeśli za bardzo zmniejszymy populację, to nie będzie ona wstanie dokonać właściwej eksploracji, mimo dużej ilości mutacji – jakość wyników będzie niska. Dlatego, aby zwiększyć jakość uzyskiwanych rozwiązań trzeba odpowiednio dobrać liczebność populacji i odpowiednio zwiększyć warunek stopu. Najlepiej jest wtedy, gdy populacja jest stosunkowo duża, a warunek stopu pozwala na wygenerowanie kilkudziesięciu/kilkuset generacji (w zależności jak długo chcemy czekać na wynik/jaki mamy sprzęt). Podsumowując, nie tylko liczebność populacji wpływa na jakość wyników, ale również stosunek liczebności populacji do warunku stopu (maksymalnej liczby wywołań funkcji oceny).

2. Jakość działania w zależności od typu grafu:

Największe rozbieżności wśród jakości osobników końcowej generacji występują przy grafach losowych (które mogą być niespójne) – im większa niespójność (większa liczba podgrafów, tym większa rozbieżność końcowych wyników). Dla grafu pełnego w znakomitej większości wszystkie osobniki mają tą samą ilość pokrytych wierzchołków. Dla grafu dwudzielnego występują pewne różnice, ale nie są tak znaczne jak w przypadku grafów losowych.

3. Wpływ prawdopodobieństwa mutacji na zachowanie algorytmu:



*na wykresie dla każdego prawdopodobieństwa – najgorszy i najlepszy osobnik ostatniej generacji (jeśli jeden punkt -> najgorszy = najlepszy)

Algorytm dla większych prawdopodobieństw mutacji osiąga lepsze wyniki. Wszystkie parametry poza prawdopodobieństwem mutacji były takie same.

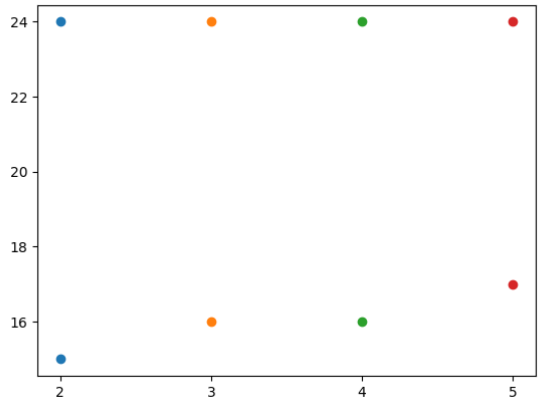
4. Wpływ warunku stopu na zachowanie algorytmu:

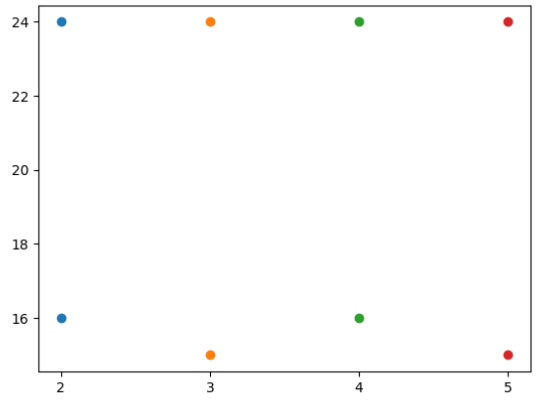
Jeśli reszta parametrów stała – im więcej możliwych wywołań funkcji oceny, tym lepiej (ograniczeniem tylko zasoby czasu i sprzętu). Jeśli liczebność populacji zmienna – patrz punkt 1 (należy odpowiednio dobrać warunek stopu i liczbę populacji – odpowiedni stosunek).

5. Wpływ liczby członków, biorących udział w turnieju (k):

Dla grafów losowych, z przeprowadzonych doświadczeń, nie jestem w stanie wyciągnąć jednoznacznych wniosków. Moim zdaniem, wszystko zależy od poszczególnego grafu. Dlatego ten parametr dla tego algorytmu należy dobierać w zależności od danego grafu.

*OX – k, OY – liczba pokrytych wierzchołków





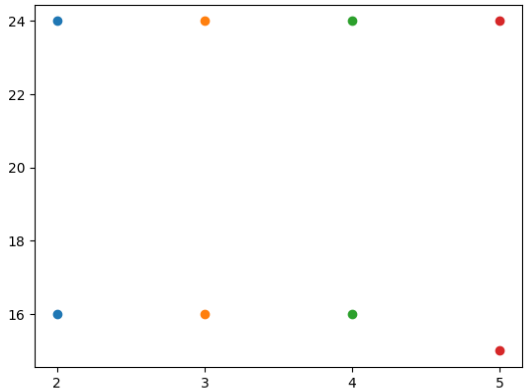


Tabela z przykładowymi wynikami algorytmu dla 3 typów grafów (N=25)

Dla tabel: poszczególne grafy takie same, różne parametry

Parametry algorytmu: populacja = 10, stop = 100, k = 2, p = 0.5

Graf	Wynik f. celu	Średnia	Odchylenie stand.	Czas [s]
Pełny	24	24	0	0.0319
Dwudzielny	14	14	0	0.0100
Losowy	14	14	0	0. 0040

Parametry algorytmu: populacja = 100, stop = 1000, k = 2, p = 0.5

Graf	Wynik f. celu	Średnia	Odchylenie stand.	Czas [s]
Pełny	24	24	0	0.3221
Dwudzielny	13	13	0	0.0837
Losowy	14	15.43	2.0363	0.0538

Parametry algorytmu: populacja = 1000, stop = 10000, k = 5, p = 0.6

Graf	Wynik f. celu	Średnia	Odchylenie stand.	Czas [s]
Pełny	24	24	0	3.7081
Dwudzielny	13	13.332	1.0568	1.2626
Losowy	13	14.209	1.2045	1.0531

Z powyższych tabel można wysnuć następujące wnioski:

- dla grafu pełnego funkcja znajduje optimum globalne już przy małej populacji; zwiększanie parametrów populacji, stopu nie przyniesie żadnych korzyści

- dla grafu dwudzielnego przy najmniejszej populacji i stopie, funkcja nie znajduje optimum globalnego; dopiero dla zwiększonych populacji i parametru stopu udaje się je odnaleźć (tabela 2)

- dla losowego grafu globalne optimum udaje się znaleźć dopiero w trzeciej próbie (dla największej populacji, największego parametru stopu i lekko zmodyfikowanych k i p); w tym przypadku opłaca się spędzić trochę więcej na poszukiwaniu optimum