1. Wprowadzenie
   1. Geneza pracy

Jako dziecko uwielbiałem przeglądać atlasy geograficzne i samemu rysować mapy. Nigdy do końca z tego nie wyrosłem, jednak to „hobby” przerodziło się w rozważanie, na jakiej podstawie zostają budowane dane drogi a inne poczekać muszą na lepsze czasy. Kilka lat takiego zastanawiania się nad optymalnością (i co ona oznacza) rozmieszczenia dróg, łączenia miast doprowadziła mnie do napisania tej pracy. Zagadnienie Steinera jest bowiem pewnym modelem matematycznym będącym uogólnieniem moich rozmyślań. Wierzę, że znajdując metodę rozwiązywania tego podstawowego problemu, nie będzie trudnym odpowiednie przeniesienie jej i zmodyfikowanie by móc rozwiązywać także bardziej złożone zagadnienia, zaspokajając moją młodzieńczą ciekawość . Opis czym jest zagadnienie Steinera i sposób w jaki można go rozbudować znajdzie się na dalszych stronach tejże pracy.

Zagadnienie Steinera wraz z szerokim wachlarzem modyfikacji ma w mojej ocenie mnogość zastosowań. Jednak znalezienie jego optymalnego rozwiązania należy do klasy problemów NP-trudnych (przypis). Dlatego wybór metody rozwiązującej to zagadnienie padł na algorytmy przybliżone, a dokładniej algorytm genetyczny.

* 1. Cel pracy

Celem mojej pracy inżynierskiej, poza stworzeniem tego dokumentu, jest implementacja własnego programu, który na wejściu otrzymuje instancję problemu Steinera, a jako wynik zwraca obliczone przybliżone jego rozwiązanie. Założenia co do stworzonego algorytmu i jego implementacji są następujące:

- algorytm ma znajdować lepsze rozwiązania dla problemu Steinera niż rozwiązanie MST(przypis) dla tych samych danych wejściowych,

- implementacja ma być modułowa – poszczególne jej elementy (np. operatory genetyczne, generatory rozwiązań, metryki, funkcja celu) powinny być w łatwy sposób dodawane i podmieniane,

-implementacja powinna dać możliwość w przyszłości łatwej adaptacji do problemów pochodnych od podstawowego zagadnienia Steinera,

- program uruchomiony na średniej jakości komputerze PC powinien znaleźć dopuszczalne(przypis)jakościowo rozwiązanie dla problemu rzędu 100 punktów w czasie nie dłuższym niż minuta (bez konieczności zmiany parametrów takich jak ilość iteracji lub wielkość populacji)

-stworzony program powinien mieć możliwość łatwej generacji danych testowych oraz parametrów algorytmu – powinien pozwalać w łatwy sposób znaleźć optymalny zbiór parametrów dla danej klasy problemu

Praca pisemna ma natomiast zawierać: teoretyczny opis stworzonego algorytmu, uwagi implementacyjne, analizę złożoności, dyskusję nad osiągniętymi wynikami oraz dobór optymalnych parametrów dla danej klasy problemu, określenie jakości stworzonej metody, jej ewentualne zastosowanie i dalszą rozbudowę.

1. Wstęp teoretyczny
   1. Aktualny stan badań tematu

???????????????????????????????????????????????????????????????????????

* 1. Model zagadnienia

W celu zdefiniowania naszego problemu, przydatnym będzie wprowadzenie poniższych terminów:

T – zbiór punktów zwany także dalej „Terminalami”

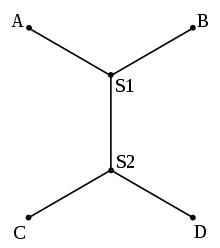
S – zbiór punktów zwany także dalej „Punktami Steinera”

V = T∪S

E – zbiór wszystkich krawędzi między wierzchołkami V (graf pełny)

d(E) – metryka euklidesowa

Euklidesowy Problem Drzewa Steinera (ESTP)

 Mając dany zbiór punktów T, wyznaczyć zbiór punktów S, taki że minimalne drzewo rozpinające M na grafie ważonym, nieskierowanym G(V, E) będzie minimalne, tzn.

będzie minimalne.

Problem ten powinien zobrazować rysunek (#nr rysunku). Dane są punkty T={A,B,C,D}. Zadaniem jest znaleźć najkrótsze drzewo łączące te punkty przy założeniu, że drzewo to może składać się z węzłów nie należących do T. Tymi pomocniczymi punktami są w tym przypadku S={S1,S2}.

W rozwiązaniu powyższego problemu przydatne może być kilka kluczowych obserwacji:

1. Wielkość zbioru S wynosić będzie maksymalnie n-1 wierzchołków, gdzie n oznacza moc zbioru T.
2. Jedynymi liśćmi w drzewie Steinera są terminale. Usunięcie punktów Steinera będących liśćmi może jedynie poprawić rozwiązanie.
3. . Jest to niejako uogólnienie warunku powyższego. Innymi słowy punkty Steinera o stopniu mniejszym niż 3 nie poprawiają rozwiązania. Punkty o stopniu 1 możemy wyeliminować na podstawie punktu 2., natomiast punkty o stopniu 2 na podstawie nierówności trójkąta.
4. Wszystkie punkty Steinera muszą leżeć wewnątrz otoczki wypukłej zbioru terminali. Jest to naturalny wniosek z poprzedniej obserwacji.

1.5 Przegląd modyfikacji problemu bazowego

Podstawowe zagadnienie znane w literaturze pod nazwą ESTP (Euclidean Steiner Tree Problem) ma cały szereg nazwanych i badanych uogólnień i przypadków szczególnych. Dosyć łatwo nasuwającym się uogólnieniem problemu może być wyjście poza metrykę euklidesową. Daje to szereg nowych problemów, których rozwiązania znacząco się różnią. Pośród tych zagadnień warto przytoczyć problem RSTP (Rectlinear Steiner Tree Problem) osadzony w metryce miejskiej. Istotnie inną klasą problemów są zagadnienia osadzone w przestrzeni grafów z krawędziami o ustalonych wagach. W tym wypadku mamy najczęściej do czynienia z grafem G(V,E) z wyszczególnionym zbiorem S (inkluzja) V. Zadaniem jest wyznaczenie zbioru V1(inkluzja)V takiego, który zminimalizuje sumę wag minimalnego drzewa rozpinającego S(suma)V1 .

1. Opis rozwiązania
   1. Część algorytmiczna
      1. Algorytm genetyczny – wstęp teoretyczny (2 strony)

Algorytm genetyczny wchodzi w skład szerszej grupy algorytmów ewolucyjnych (jakiś przypis). Jest to zbiór algorytmów, które znajdują rozwiązanie na drodze ewolucji, ulepszając je wraz z biegiem iteracji. Algorytmy genetyczne symulują naturalne zmiany zachodzące w populacjach (np. zwierząt bądź roślin). Polega to na tym, że każdy osobnik dysponuje zbiorem cech. Członkowie populacji krzyżując się między sobą tworzą nowe osobniki (z nowymi lecz odziedziczonymi cechami). Czasem na drodze przypadkowych mutacji, materiał genetyczny zostaje zmodyfikowany. Osobniki są poddane selekcji (w naturze mogą to być warunki atmosferyczne, działalność drapieżników itd.). Większe szanse na przeżycie (i dalsze propagowanie swojego kodu genetycznego) mają osobniki najlepiej przystosowane (w wypadku roślin i zwierząt może to być kolor, grubość sierści lub korzeni, długość ogona bądź łodygi itd.). Z biegiem czasu naturalna selekcja powoduje ewoluowanie puli genów (a tym samym zbioru cech) w kierunku lepszego przystosowania. Algorytm genetyczny, czerpiąc z natury, próbuje dokonań tego samego.

Pierwszym jego krokiem jest pozyskanie z zewnątrz bądź wygenerowanie pierwszej populacji. Następnie aż do wypełnienia kryterium stopu (o którym w szczegółach będzie mowa później) następuje tworzenie nowych pokoleń/populacji na podstawie krzyżowania osobników i mutacji. Nowe rozwiązania są poddawane ocenie i na tej podstawie jest dokonywana selekcja. Sposobów na wykonanie każdego z tych kroków jest całe mnóstwo. Wybór metody selekcji, krzyżowania (w tym doboru osobników do tej operacji), mutacji w dużej mierze decyduje o sukcesie takiego algorytmu.

W wielu problemach można próbować zastosować jakąś wersję algorytmu genetycznego. Jedynym warunkiem jest możliwość zamodelowania rozwiązania w postaci chromosomu – zbioru jego składowych (często binarnych ale również ciągłych, dyskretnych). Jednak jakość rozwiązania dostarczonego tą metodą zależy istotnie od problemu. Trudnym zadaniem jest oszacowanie jakości wyników zwróconych przez ten algorytm. Czasem drobna zmiana w sposobie krzyżowania może diametralnie zmienić kształt rozwiązań.

Dlatego nazwa „algorytm genetyczny” jest jedynie bardzo ogólnym zbiorem zasad i terminów. Natomiast jego implementacje mogą się bardzo fundamentalnie od siebie różnić – nie mówiąc o zwracanych wynikach. Należy bowiem wciąż pamiętać, że jest to algorytm przybliżony, a mówiąc o jakości rozwiązań mamy na myśli bliskość takich rozwiązań do rozwiązań optymalnych (obliczonymi drogą analityczną).

W najprostszej wersji, schemat algorytmu genetycznego przedstawia rysunek poniżej.

* + 1. Opis głównego algorytmu (2 strony)

Zaadoptowanie algorytmów genetycznych do rozwiązania zagadnienia Steinera nie jest rzeczą nową ale jak wyjaśniłem w poprzednim podrozdziale, stworzenie dwóch działających identycznie rozwiązań jest praktycznie niemożliwe. W skład implementacji wchodzą takie składowe jak: algorytmy selekcji, mutacji, krzyżowanie, doboru osobników do krzyżowania i mutacji, funkcja oceny rozwiązania, generatory rozwiązań i całych populacji, metody optymalizacji lokalnej, kryterium stopu oraz cały szereg parametrów – prawdopodobieństwa wykonania operatorów genetycznych, wielkość populacji, parametry kryterium stopu(ilość iteracji, czas obliczeń, próg wartości zadowalającego rozwiązania itd.). W następnych podrozdziałach skupię się szczegółowo na temat rozwiązania każdego z tych zagadnień. Użyte metody zostały wybrane metodą doświadczalną, sprawdzając zachowania kilku różnych implementacji. Należy bowiem pamiętać, że jakość każdej z nich jest mocno związana z postawionym problemem.

* + 1. Opis rozwiązania podproblemów
       1. Operatory genetyczne

Jak już wspomniano wyżej, dwoma typami operatorów genetycznych zaadoptowanych z natury do algorytmów ewolucyjnych są krzyżowanie i mutacja.

O ile mutacji są poddawane (z pewnym prawdopodobieństwem) pojedyncze rozwiązania, o tyle sprawa z krzyżowaniem zaczyna się mocno komplikować już na samym początku. Ten operator potrzebuje bowiem w standardowej wersji dwóch rozwiązań-rodziców i na ich podstawie tworzy kolejne, zawierające skrzyżowane cechy swoich przodków. Stąd nasuwa się pytanie: jak dobrać pary rodziców oraz ile ich powinno być? Drugie pytanie jest raczej kwestią doboru parametrów algorytmu . Należy przy tym jednak pamiętać, że w zależności od stosowanego operatora krzyżowania, z pary rodziców powstaje 1,2 lub nawet więcej potomków. W przypadku pierwszego pytania problem zdaje się być znacznie obszerniejszy. Znanych jest kilka podejść do tego tematu (przypis):

-dla każdej możliwej pary rodziców oblicza się z pewnym prawdopodobieństwem (np. rzędu 50%) czy para ta będzie podlegała krzyżowaniu ( w tym wypadku można np. poddawać parę testom prawdopodobieństwa dla każdego posiadanego operatora krzyżowania osobno, odpowiednio dobierając wartości prawdopodobieństwa),

-wybranie określonej z góry liczby par na podstawie losowania obu rodziców, tj. tworząc pierwszą parę losujemy z puli rodziców dwa osobniki z różnym prawdopodobieństwem a po losowaniu trafiają one z powrotem do puli.

W obu przypadkach można rozwinąć te strategie w stronę większej elitarności, zmieniając prawdopodobieństwa wylosowania w zależności od jakości przystosowania (fenotypu) rodziców. Podobną funkcję zazwyczaj(przypis – w przypadku selekcji nie-czysto losowej (metoda ruletki itd) pełni sam etap selekcji.

Jeśli chodzi o dobór par, została wybrana opcja numer jeden, tj. losowanie elementów do określonej ilości par z równym prawdopodobieństwem .

Ciekawszym elementem dotyczącym krzyżowania jest to, co dzieje się już gdy znane są oba osobniki rodziców. W moim programie zawarłem dwa różne sposoby, choć w tym wypadku można wymyślić z pewnością znacznie więcej. Te, które zostały stworzone są jednak najbardziej intuicyjne. Podczas działania programu, oba operatory mogą działać jednocześnie.

Zgłębiając literaturę opisującą temat nie natrafiłem na zaproponowane przeze mnie operatory. Zaprojektowałem je osobiście, stąd nie podaję ich źródła. Nie wykluczam jednak, iż ktoś mógł wykorzystać podobne metody wcześniej i je opisać. Jak już wspomniałem, oba wydają się być mocno intuicyjne i samonasuwające się.

Pierwszym algorytmem krzyżowania jest losowy dobór połowy punktów z pierwszego rodzica, połowy punktów z drugiego i połączenie ich w jedno, nowe rozwiązanie. Jest to operator wnoszący pewien chaos do nowego rozwiązania. Nie bierze on bowiem pod uwagę lokalizacji punktów i wybiera je zupełnie na ślepo. W specyficznym przypadku z obu rodziców mogą zostać pobrane punkty leżące obok siebie. Operator ten jednak ma pewien istotny plus: pozwala on na wyskoczenie z rozwiązań z minimum lokalnego. Jeśli rozwiązanie powstałe na skutek działania tego operatora, zostanie w jednej z kolejnych iteracji poddane np. operatorowi mutacji polegającej na optymalizacji lokalnej rozwiązania – z dużym prawdopodobieństwem usunięte zostaną punkty leżące blisko siebie. Ten operator zostanie opisany w innym punkcie (przypis -numer strony z operatorem mutacji)

Działanie tego operatora przedstawi rysunek.

Operator 1 - rysunek

Drugi operator znacznie bardziej przypomina standardowe metody krzyżowania. To co go odróżnia to oczywiście fakt, że odnosi się do innego kodowania genotypu. W przeciwieństwie do poprzednio zaproponowanego, ten operator zwraca uwagę na lokalizację punktów, natomiast podobnie jak on pobiera część z nich z genotypu pierwszego rodzica a część z drugiego. W tym wypadku ilość punktów pochodzących od rodziców może się różnić. Zasada ich doboru jest prosta. Na początku losowany jest miejsce podziału w zakresie dopuszczalnych lokalizacji punktów problemu. Następnie z genotypu pierwszego rodzica pobierane są punkty znajdujące się powyżej miejsca podziału, a z drugiego te znajdujące się poniżej. Operator może mieć też wersję symetryczną (na lewo – na prawo), bądź taką, w której miejsce podziału wyznacza dwa charakterystyczne obszary widoczne na rysunku

Operator 2 - rysunek

(schemat blokowy operatora)

a)góra – dół, b)lewo-prawo, c)lewy górny róg - reszta

W wypadku tego operatora zdecydowałem się na generowanie dwóch rozwiązań-dzieci z jednej pary rodziców. Drugie dziecko jest tworzone analogicznie tylko przy założeniu zamiany rodziców, tj. dla przypadku a) – górna część pobierana z rodzica nr 2 a dolna z rodzica nr 1.

Operatory mutacji

Jak już wspomniałem, operatory mutacji mają na ogół mniejsze znaczenie (i występują rzadziej) w algorytmach genetycznych. Jednakowoż pełnią one bardzo istotną rolę. Pozwalają bowiem na wygenerowania w trakcie działania algorytmu pewnego istotnie różnego od pozostałych rozwiązania, np. poprzez mutację genu, który już od dłuższego czasu pozostawał w stabilnym, stałym stanie. Ma to na celu wydostanie się fenotypu z ekstremum lokalnego. W przypadku rozważanego przeze mnie zagadnienia oraz charakterystycznego kodowania genotypu, mutacja polega na zmianach należących do niego punktów. Może to być dodanie zupełnie nowego punktu (genu) bądź jego usunięcie. Może to być również przesunięcie już istniejących punktów.

W moim rozwiązaniu stworzyłem dwa takie operatory. Pierwszy z nich do zadanego genotypu dodaje nowy, losowo wybrany punkt, powiększając tym samym genotyp. W zasadzie powinno to wystarczyć za jego opis. Można teraz przejść do drugiego, znacznie ciekawszego operatora, który istotnie zmienia rozwiązanie. Polega on bowiem na optymalizacji lokalnej. Do optymalizacji lokalnej wykorzystuję obserwacje poczynione w punkcie dotyczącym teorii nt. zagadnienia Steinera (przypis – odnośnik, nr strony). Dla przypomnienia podam wykorzystane wnioski raz jeszcze: punkty Steinera o stopniu mniejszym niż 3 nie poprawiają rozwiązania. Punkty o stopniu 1 i 2 można od razu wyeliminować.

Natomiast punkty o stopniu 3 można poddać łatwej, analitycznej optymalizacji. Sąsiadów takiego punku można traktować jako wierzchołki pewnego trójkąta. Interesująca nas składowa funkcji celu, którą możemy poprawić to suma tych trzech krawędzi, czyli w planimetrii suma odległości danego punktu od wierzchołków trójkąta. Z definicji punkt Fermata-Torricellego jest punktem minimalizującym tę sumę. Wystarczy zatem omawiany przez nas punkt zamienić punktem Fermata-Torricellego i otrzymana suma odległości z pewnością nie będzie większa. Metoda znajdowania wspomnianego punktu została zaczerpnięta z pozycji (przypis – książka o problemie Webera). Niestety nie istnieją proste analityczne wzory (jak w przypadku środka ciężkości trójkąta) do znalezienia go. Metoda znajdowania punktu F-T polega na konstrukcji trójkątów równobocznych na bokach zadanego trójkąta i znalezieniu przecięcia ich trzecich (nie wchodzących w skład trójkąta bazowego) wierzchołków z wierzchołkami w trójkącie bazowym przeciwstawnymi do tych, na których został zbudowany trójkąt równoboczny. Opis słowny jest zawiły, jednak rysunek powinien znacznie ułatwić zrozumienie tej konstrukcji.

Rysunek : konstrukcja punktu Fermata

W moim algorytmie konstrukcja ta przebiega następująco:

Schemat blokowy konstrukcji punktu Fermata

Taka optymalizacja lokalna

Rysunek: przykład działania lokalizacji z punktami o stopniu 1,2 i 3

Wracając do operatora mutacji, w moim rozwiązaniu stosuje on wspomnianą metodę konstrukcji punktu F-T dla każdego punktu o stopniu 3. Punkty o stopniach 1 i 2 są odrzucane z rozwiązania, a punkty o stopniu większym od 3 przepisywane.

Wydawałoby się, że taką optymalizację lokalną należałoby stosować przy każdej okazji, np. po każdym krzyżowaniu rozwiązanie dziecko powinno przejść taką mutację. Powód, dla którego tego nie czynię w moim rozwiązaniu jest bardzo prosty: takie działanie mogłoby np. sprawić utratę punktów o niższych stopniach, które w dalszych iteracjach (i w ostatecznym rozwiązaniu) mogłyby pełnić istotną rolę, bądź spowodować niechciane przesunięcia punktów szybko prowadząc do pewnego minimum lokalnego.

Dlatego w mojej implementacji optymalizacja lokalna jest traktowana głównie jako operator mutacji działający z pewnym prawdopodobieństwem. W celach testowych, operator został użyty m.in. z prawdopodobieństwem 100% w celu zaobserwowania wpływu na generowanie rozwiązań.

Optymalizacja lokalna będzie natomiast jeszcze użyta do zoptymalizowania ostatecznie znalezionego rozwiązania, gdy maksymalne zbliżenie do ekstremum lokalnego będzie pożądane. Warto zwrócić okazję, ze raz zoptymalizowane rozwiązanie, można poddać optymalizacji ponownie ( w zasadzie w nieskończoność), ponieważ wspomniana implementacja nie znajduje minimum lokalnego całego rozwiązania a jedynie pojedynczych punktów. Podczas kolejnej optymalizacji będziemy zatem operować na nieco zmodyfikowanym grafie itd.

Metody selekcji

Metoda wyboru następnej populacji – zapamiętywanie najlepszego rozwiązania