AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE

**WYDZIAŁ ELEKTROTECHNIKI, AUTOMATYKI,
INFORMATYKI I INŻYNIERII BIOMEDYCZNEJ**

KATEDRA AUTOMATYKI I INŻYNIERII BIOMEDYCZNEJ

Praca dyplomowa magisterska

Algorytmy przybliżone dla zagadnienia przydziału kwadratowego
Approximation algorithms for qadratic assignment problem

Autor: *Stefan Kultys*
Kierunek studiów: *Automatyka i Robotyka*
Opiekun pracy: *dr inż. Wojciech Chmiel*

Kraków, 2014

Oświadczam, świadomy(-a) odpowiedzialności karnej za poświadczenie nieprawdy, że niniejszą pracę dyplomową wykonałem(-am) osobiście i samodzielnie i nie korzystałem(-am) ze źródeł innych niż wymienione w pracy.

Serdeczne podziękowania dla bla bla bla

Spis treści

1. Wstęp

Nadejście rewolucji przemysłowej spowodowało powstanie wielkiej liczby firm, przedsiębiorstw, które były dużo większe niż znane wcześniej zakłady rzemieślnicze. Ich rozmiar powodował również rozrost złożoności problemów związanych z organizacją tychże firm. W związku z kompleksowością pojawił się problem jak najlepszego przydziału dostępnych zasobów, najwłaściwszej organizacji pracy. Zaistniała więc potrzeba stworzenia różnych metod, sposobów, dzięki którym można by powyższe problemy w jakiś sposób rozwiązać. Ta potrzeba doprowadziła do powstania badań operacyjnych.

Chociaż początki badań operacyjnych faktycznie związane są z rewolucją przemysłową, to jednak pojęcie badań operacyjnych, które znamy obecnie, związane jest z działaniami podejmowanymi przez agencje wojskowe już na początku drugiej wojny światowej. Można by stwierdzić, że pewną ironią losu jest fakt, iż wiele wykorzystywanych dzisiaj odkryć i wynalazków, które bardzo ułatwiają nam codziennie życie, zostało powołanych do życia w związku z działaniami, które najczęściej kojarzą się z cierpieniem i przemocą. Brytyjskie i amerykańskie organizacje wojskowe zatrudniły ogromną liczbę naukowców, by ci wdrożyli naukowe podejście do spraw związanych z efektywnym zbrojeniem się, zarządzaniem zasobami oraz taktycznymi i strategicznymi problemami związanymi z prowadzeniem działań wojennych.

Mówi się, że podjęte wysiłki miały duży wpływ na takie znane wydarzenia jak Bitwa o Anglię, czy też Bitwa o Atlantyk.

Sukces, jaki odniosły badania operacyjne w wojskowości, zachęcił ludzi związanych z przemysłem do zaadaptowania ich również w samym przemyśle. Ożywienie w gospodarce, spowodowane zakończeniem wojny, doprowadziło do wzrostu złożoności działalności firm, a więc badania operacyjne idealnie nadawały się jako narzędzie wspierające organizację i zarządzanie tymi przedsiębiorstwami.

Niewątpliwie, następujący szybki rozwój badań operacyjnych miał swą przyczynę w tym, że wielu naukowców, którzy parali się nimi podczas wojny, szukając pracy w swojej branży, chętnie zajęło się dalszymi studiami nad badaniami operacyjnymi w dziedzinach związanych nie tylko z wojskowością. Oczywiście nie oznaczało to, że wojsko całkowicie zrezygnowało z badań operacyjnych. Również postęp związany z powstaniem komputerów dał odpowiednie narzędzia do analizy coraz bardziej złożonych problemów. Wiele problemów związanych z podejmowaniem decyzji, wyborem najlepszego rozwiązania można było rozwiązać podpierając się matematycznym modelem. Mając więc problem w sformalizowanej postaci można zaproponować algorytm, który rozwiąże dane zagadnienie. Sam algorytm jako ciąg kolejnych instrukcji, które należy wykonać, by osiągnąć dany cel, bardzo dobrze nadaje się do

zaimplementowania i wykonania na komputerze. Coraz szybsze komputery o coraz pojemniejszych pamięciach, a także wykorzystanie technik programowania równoległego i współbieżnego pozwalają na rozwiązywanie coraz bardziej złożonych problemów w rozsądnym czasie. Z czasem więc zaczęły się pojawiać kolejne algorytmy, ale też nowe problemy. Również dokonywane odkrycia naukowe pozwoliły na wykorzystanie występujących w naturze procesów do tworzenia nowatorskich metod rozwiązywania skomplikowanych zagadnień.

Niestety, istnieje wiele problemów, w przypadku których można jedynie powiedzieć, że mają optymalne rozwiązanie, nie da się jednak znaleźć go przy wykorzystaniu obecnie dostępnej technologii. Poprzez oszacowanie złożoności obliczeniowej algorytmów można tylko stwierdzić, że potrzebny czas do znalezienia rozwiązania problemu przy ich wykorzystaniu jest niejednokrotnie dłuższy niż przeciętny czas życia człowieka. Przykładem takiego zagadnienia jest tzw. problem przydziału kwadratowego, polegającego na przydziale pewnej liczby placówek do takiej samej liczby miejsc. Wynika z tego, że dla n placówek możliwe jest w sumie *n!* wszystkich permutacji. Wraz ze wzrostem liczby placówek, które należy przydzielić, ilość możliwych rozwiązań rośnie bardzo szybko. Już dla stosunkowo małej ilości placówek możliwa jest ogromna liczba rozwiązań. Istnieje więc wiele algorytmów przybliżonych, inaczej nazywanych aproksymujących, które znajdują jedynie przybliżone rozwiązanie postawionego problemu. Nie oznacza to jednak, że zwrócone przez algorytm rozwiązanie nie może być faktycznie optymalne, jednak nie da się przeważnie tego sprawdzić.

Jak już zostało to nadmienione wyżej, istnieje wiele algorytmów wykorzystujących analogie do zachowań występujących w przyrodzie. Przykładem są algorytmy genetyczne, których działanie wzorowane jest na ewolucji biologicznej - spośród znalezionych w danym pokoleniu rozwiązań, wybierane są najlepsze z nich (według pewnych ustalonych dla danego problemu kryteriów), traktowane są jako rodzice dla następnego pokolenia, które dziedziczy po rodzicach ich cechy. Wykorzystywane są również różnego rodzaju operatory mutacji, katastrofy itp.

1.1. Cel pracy

Celem niniejszej pracy jest dokonanie przeglądu wybranych algorytmów przybliżonych, ich wad i zalet oraz przedstawienie ich wykorzystania w kontekście problemu przydziału kwadratowego. Następnie, przy użyciu specjalnie napisanej na potrzeby pracy aplikacji, która rozwiązuje problem przydziału kwadratowego, należy zaprezentować rezultaty przeprowadzonych eksperymentów oraz opisać zastosowane scenariusze testowe i dokonać analizy otrzymanych wyników.

1.2. Zawartość pracy

Rozdział nr 2 zawiera opis problemu przydziału kwadratowego, obszar jego zastosowań i jego model matematyczny. Rozdziały od trzeciego do piątego poświęcone są wybranym algorytmom przybliżonym, ich wadom i zaletom. Opisane są sposoby działania, oraz to w jaki sposób mogą być wykorzystane

do rozwiązania problemu QAP. Tematyka następnych rozdziałów związana jest z napisaną na potrzeby niniejszej pracy aplikacją rozwiązującą problem przydziału kwadratowego przy zastosowaniu algorytmu genetycznego i jego kwantowej modyfikacji. Przedstawione są przeprowadzone testy, opisane są wybrane instancje testowe, oraz dokonana jest analiza uzyskanych rezultatów. Ostatnio rozdział poświęcony jest na podsumowanie całej pracy i wnioskom z niej płynącym.

2. Zagadnienie przydziału kwadratowego

2.1. Opis problemu

Zagadnienie przydział kwadratowego (Qadratic Assignment Problem - QAP) jest jednym najtrudniejszych problemów optymalizacji kombinatorycznej. Należy on klasy problemów NP - trudnych i dla rozmiarów o wartości większej niż 30 wymagane jest stosowanie algorytmów przybliżonych w celu jego rozwiązania. Zagadnienia przydziału kwadratowego zostało przedstawione przez Koopmansa i Beckmanna w roku 1957 do rozwiązania zagadnień ekonomicznych. Problem ten jest matematycznym modelem sytuacji, w której chcemy przydzielić pewną ilość placówek do takiej samej ilości lokalizacji (miejsc) znając przy tym odległości pomiędzy danymi lokalizacjami oraz przepływu miedzy placówkami. Przydziału tego należy dokonać minimalizując koszt tej operacji, który jest proporcjonalny do przepływu pomiędzy placówkami pomnożonego przez odległość między miejscami, do których te placówki zostały przydzielone. Istnieją również wersje tego problemu, w których podany jest również koszt samego przydziału placówki do lokalizacji. Z racji, iż trudność rozwiązania tego problemu jest duża oraz, że modeluje on wiele faktycznych zagadnień, wielu autorów poświęciło mu dużo uwagi, przez co znaleźć można wiele różnych publikacji traktujących o problemie QAP. Niewątpliwie postępujący rozwój w dziedzinie informatyki i elektroniki pozwolił na analizę coraz bardziej złożonych problemów i tworzenie nowych metod, które dotychczas nie byłyby możliwe do wykorzystania.

2.2. Obszary zastosowań

Przy pomocy problemu przydziału kwadratowego można modelować wiele różnych zagadnień, które występują w otaczającym nas świecie. Do dziedzin, w których zagadnienie QAP znajduje zastosowanie, należą m. in:

- ekonomia,
- informatyka,
- elektronika,
- logistyka,

– mechanika,

– architektura.

Do wybranych problemów z spośród wymienionych wyżej dziedzin należą m. in:

– projektowanie zagospodarowania przestrzennego w nowopowstających miastach,

– projektowanie układów elektroniki,

– właściwa lokalizacja fabryk,

– organizacja biur, oddziałów szpitalnych,

– wyważanie turbin w silnikach odrzutowych.

2.3. Model matematyczny

Model matematyczny zagadnienia przydziału kwadratowego może być przedstawiony w następujący sposób:

Dany jest zbiór:

$$N = \{1, ..., n\} \tag{2.1}$$

oraz następujące macierze o wymiarach $n \times n$*:*

$$A = (a_{ij}),\ B = (b_{ij}),\ C = (c_{ij}) \tag{2.2}$$

gdzie macierz A *jest macierzą odległości pomiędzy lokalizacjami. Z tego powodu często macierz ta oznacza jest też literą* D*, od angielskiego słowa distance, oznaczającego odległość. Macierz* B *jest macierzą określającą pewne powiązania pomiędzy placówkami, np. przepływ informacji, ilość połączeń, ilość towaru jaką należy przetransportować z jednej lokalizacji do drugiej, itp. Macierz ta jest też oznaczana literą* F *(ang. flow - przepływ). Macierz* C *określa koszt przydziału placówki do lokalizacji. Dana jest również funkcja celu, będąca określona w następującej sposób:*

$$\Phi(\pi) = \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij} \cdot b_{\pi(i),\pi(j)} + \sum_{i=1}^{n} c_{\pi(i),i} \tag{2.3}$$

gdzie π *jest permutacją* $\pi = (\pi(1), \pi(2), ..., \pi(n))$*, a* $\pi(i)$ *oznacza numer placówki przydzielonej do i-tej lokalizacji. Funkcja celu określa więc ogólny koszt przydziału i eksploatacji przydzielonego systemu. Szukana jest zatem permutacja minimalizująca funkcję celu, czyli taka, dla której wspomniany koszt jest najmniejszy.*

2.4. Złożoność obliczeniowa

Rozwiązanie problemu QAP jest permutacją. Należy przydzielić n placówek do n miejsc. Wynika stąd, że wszystkich możliwości przydziału jest $n!$. Jak zostało wspomniane wcześniej zagadnienie przydziału kwadratowego jest problemem NP-trudnym, czyli zadaniem o złożoności co najmniej wykładniczej. Zadanie o złożoności silni jest zadaniem o złożoności jeszcze większej niż wykładnicza. Wynika z tego fakt, iż już dla stosunkowo małych rozmiarów problemu czas znalezienia rozwiązania poprzez wykorzystanie algorytmów znajdujących dokładne rozwiązanie staje się praktycznie niemożliwe. Zjawiska modelowane zagadnieniem QAP mają rozmiary nierzadko liczony w setkach i większe. Znalezienie dokładnego rozwiązania mogłoby wtedy zająć czas nawet dłuższy niż znany wiek Wszechświata. Chcąc więc znaleźć rozwiązanie postawionego problemu należy stosować algorytmy, które poradzą sobie w czasie zdecydowanie krótszym. Receptą są algorytmy przybliżone, inaczej zwane aproskymacyjne. Zagadnieniu algorytmów przybliżonych poświęcony jest następny rozdział niniejszej pracy.

3. Algorytmy przybliżone

Złożoność otaczającego nas świata powoduje, że bardzo często występujące problemy, które chcielibyśmy rozwikłać są w rzeczywistości bardzo trudne do rozwiązania. Dotyczy to praktycznie każdej sfery ludzkiego życia. W wielu sytuacjach natura problemu nie pozwala na zastosowanie metod matematycznych, jednakże nawet w przypadku takich trudności, w których matematyka przychodzi z pomocą, można stwierdzić jedynie, że problem ma rozwiązanie i to nawet najlepsze z możliwych, optymalne, lecz znalezienie go jest praktycznie niewykonalne. Używając języka naukowego, wiele z tych problemów można nazwać NP-trudnymi. Złożoność obliczeniowa algorytmów pozwalających na rozwiązanie ich jest zbyt duża, by w ogóle warto było je stosować. Pojawia się więc potrzeba zastosowania czegoś, co pozwoli na znalezienie rozwiązania dobrego, przybliżającego chociaż rozwiązanie optymalne. I faktycznie jest grupa algorytmów, które pozwalają na uzyskanie takiego efektu. Są to algorytmy przybliżone, inaczej zwane aproksymującymi.

W przeciwieństwie do problemów optymalizacji, których rozwiązanie jest możliwe do znalezienia w czasie wielomianowym, problemy NP-trudne nie dają p̈unktu wyjściaḋo znalezienia rozwiązania optymalnego. Jednakże, niejednokrotnie istnieje p̈unkt wyjścia ,, który pozwala na dojście do rozwiązania znajdującego się w pobliżu rozwiązania najlepszego. W tym sensie algorytmy przybliżone podobne są do algorytmów dokładnych: również polegają na uchwyceniu istoty problemu i następnie na znalezieniu algorytmu, która pozwoli na wykorzystanie jej.

Ogromna ilość problemów, dla których nie jesteśmy w stanie znaleźć rozwiązania optymalnego, przyczyniła się do powstania wielu algorytmów aproksymacyjnych. Przy tworzeniu algorytmów dąży się do tego, by działały one jak najszybciej. W przypadku rozwiązywania przy ich użyciu problemów niejednokrotnie czas ich działania jest dosyć długi. Jednakże, pozwalają one na znalezienie dobrego rozwiązania w sytuacji, gdy użycie algorytmów dokładnych nie pozwoliłoby uzyskać rozwiązania w ogóle.

Ciekawą rzeczą związaną z algorytmami przybliżonymi jest fakt, że wiele z nich powstało na podstawie obserwacji zjawisk występujących w przyrodzie.

Poniżej zostaną przedstawione kilka algorytmów aproksymujących, zostaną przedstawione podstawowe informacje na ich temat, opisany schemat ich działania, a także to, w jaki sposób przy ich pomocy można by próbować rozwiązać problem QAP.

3.1. Particle Swarm Optimization

3.1.1. Geneza i opis algorytmu

Algorytm Particle Swarm Optimization (PSO), czyli algorytm optymalizacji rojem cząstek, po raz pierwszy został przedstawiony w pracy Jamesa Kennedy'ego i Russella Eberharta w 1995 roku, jako metoda optymalizacji nieliniowych funkcji ciągłych. Metoda powstała w oparciu o przeprowadzane symulacje uproszczonych modeli zachowań społecznych. Inspiracją dla autorów były przeprowadzane przez naukowców komputerowe symulacje zachowań stad ptaków czy ławic ryb.

Zachowania stad ptaków zawsze interesowały naukowców. Chcieli oni dociec w jaki sposób ptaki potrafią, latając w licznych stadach, lecieć w sposób synchroniczny, często zmieniając kierunek lotu czy też błyskawicznie się przegrupowując. Z czasem powstawały różnego rodzaju modele tychże zachowań, programy pozwalające na symulowanie ich. Również ciekawą rzeczą był fakt, że ptaki potrafią znaleźć sobie pożywienie, ominąć zagrożenie, mimo że nie posiadają początkowo wiedzy na ten temat. Pojawiły się tezy, że potrafią one wykorzystać zdobytą wiedzę przez inne osobniki, czy tez poprzednie pokolenia. Dążenie do znalezienia pokarmu, próby unikania sytuacji niebezpiecznych czy drapieżników są czynnikami decydującymi o poprawie sytuacji życiowejptaków. Jest to swego rodzaju optymalizacja dokonywana samoistnie przez naturę. Analiza tych zachowań stała się punktem wyjścia do tworzenia algorytmów pozwalających na rozwiązywanie wielu trudnych problemów.

Algorytm PSO w pewien sposób przypomina wspomniane wcześniej symulacje, lecz zawiera też parę istotnych różnic. W klasycznej wersji, algorytm zawiera rój cząstek poruszających się w wielowymiarowej przestrzeni, który inicjowany jest w sposób losowy. Cząstki te reprezentują rozwiązania problemu i scharakteryzowane są swoją prędkością i położeniem. Ruch cząstek w kolejnych iteracjach ma na celu przeszukiwanie przestrzeni rozwiązań. Każda z cząstek zapamiętuje znalezioną przez siebie dotychczas najlepszą pozycję. W oparciu o te pozycje, w każdej iteracji cząstki mają aktualizowaną swoją prędkość i położenie.

Algorytm PSO posiada wiele zalet. Przede wszystkim jest bardzo prosty i wydajny, oraz pozwala na optymalizację wielu różnych funkcji. Aktualizacja prędkości i położenia cząstek wymaga jedynie podstawowych operacji matematycznych. Algorytm nie wymaga również zapamiętywania dużej ilości danych, dlatego jest wydajny z punktu widzenia szybkości działania i nie wymaga wielu zasobów pamięci. Ważną cechą jest również to, że jest on bardzo odporny na wpadnięcie do minimum lokalnego.

3.1.2. Model matematyczny algorytmu

Model matematyczny algorytmu PSO może być przedstawiony w następujący sposób: *Mamy dany rój cząstek, który składa się z n cząstek. Każda z nich porusza się w d-wymiarowej przestrzeni. Każda z cząstek opisana jest przez dwa wektory:*

– *wektor położenia:*

$$x_i = [x_{i1}, x_{i2}, ..., x_{id}] \tag{3.1}$$

– *wektor prędkości:*

$$v_i = [v_{i1}, v_{i2}, ..., v_{id}] \tag{3.2}$$

Ponadto, każda z cząstek zapamiętuje znalezioną przez siebie najlepszą dotychczas pozycję w wektorze:

$$x_i^b = [x_{i1}^b, x_{i2}^b, ..., x_{id}^b] \tag{3.3}$$

Zapamiętywana jest również w wektorze x^* najlepsza dotychczas pozycja w ogóle znaleziona przez wszystkie cząstki w roju.

Wartości prędkości i położenia w każdej iteracji algorytmu aktualizowane są odpowiednio według poniższych wzorów[odniesienie]:

$$v_{ij}(t) = w \cdot v_{ij}(t-1) + c_1 \cdot r_1 \cdot (x_{ij}^b(t-1) - x_{ij}(t-1)) + c_2 \cdot r_2 \cdot (x_j^*(t-1) - x_{ij}(t-1)) \tag{3.4}$$

$$x_{ij}(t) = x_{ij}(t-1) + v_{ij}(t) \tag{3.5}$$

gdzie liczby r_1 i r_2 są wybierane losowo z przedziału $[0, 1]$, natomiast współczynniki c_1 i c_2 odpowiadają za to, w jakim stopniu do aktualizacji prędkości brane są pod uwagę najlepsze znalezione dotychczas położenia każdej z cząstek z osobna i najlepsze położenie w ogóle. Parametr w określa bezwładność cząstek i z czasem maleje liniowo do 0.

3.1.3. Pseudokod dla algorytmu PSO

Poniżej znajduje się pseudokod, który opisuje jak krok po kroku działa algorytm optymalizacji rojem cząstek:

Wczytaj rozmiar roju n, wymiar d, ilość iteracji t i inne parametry;
while *nie wystąpił warunek stopu* **do**
  $t \leftarrow t + 1$;
  for $i \leftarrow 1$ **to** n **do**
    Policz dopasowanie cząstki x_i;
    if x_i *jest lepsza niż* x_i^b **then**
      $x_i^b \leftarrow x_i$
    end
    if x_i^b *jest lepsza niż* x^* **then**
      $x^* \leftarrow x_i^b$
    end
  end
  for $i \leftarrow 1$ **to** n **do**
    for $j \leftarrow 1$ **to** d **do**
      Zaktualizuj prędkość v_{ij};
      Zaktualizuj położenie x_{ij};
    end
  end
end

Algorithm 1: Algorytm PSO

3.1.4. Zastosowanie algorytmu PSO dla problemu QAP

Aby było możliwe zastosowanie algorytmu PSO do rozwiązania problemu przydziału kwadratowego, należy odpowiednio ująć problem QAP, by dało się go wpasować w model algorytmu. Przede wszystkim rozwiązaniami zagadnienia przydziału kwadratowego są permutacje, czyli jest to problem dyskretny. Pozycje cząstek w algorytmie PSO mogą zmieniać się w sposób ciągły, położenie nie musi być określone współrzędnymi całkowitymi. Również w permutacji liczby nie mogą się powtarzać. Natomiast nie stoi nic na przeszkodzie, by zwrócona przez algorytm pozycja cząstki była opisana w każdym kierunku przez współrzędne o tej samej wartości. Proste mapowanie: wartość położenia w $i - tym$ kierunku określa przydzielenie do $i - tej$ lokalizacji obiektu o tejże wartości może powodować, że dany obiekt będzie przydzielony wielokrotnie.

3.2. Algorytm Tabu Search

3.2.1. Geneza i opis algorytmu

Algorytm Tabu Search został zaproponowany przez Freda Glovera w roku 1986. Jest to metaheurystyka pozwalająca innym metodom optymalizacji unikać sytuacji, w których te wpadają w minima

lokalne. Dzięki metodzie tabu search udało się znaleźć optymalne lub prawie optymalne rozwiązania dla bardzo wielu problemów optymalizacji takich jak szeregowanie zadań, problem przydziału kwadratowego, rozpoznawanie charakteru, kolorowanie grafów.

Słowo tabu kojarzone jest przede wszystkim z czymś zakazanym, najczęściej na tle kulturowym. W przypadku algorytmu należy je rozumieć bardziej jako ograniczenie. W ogólności algorytm tabu search polega na zabranianiu wykonywania danej operacji modyfikującej rozwiązanie zwanej ruchem. Ruch jest funkcją, która transformuje dane rozwiązanie w inne, w przypadku permutacji może to być zamiana miejscami dwóch liczb. W danym momencie możliwy jest pewien podzbiór rozwiązań, w które inne może być przetransformowane. Z dostępnych ruchów wybierany jest ten, który powoduje polepszenie rozwiązania i ostatnio wykonany ruch dodawany jest do tablicy ruchów zabronionych na pewną określoną liczbę iteracji algorytmu. Mechanizm ten pozwalaj na wyjście z minimum lokalnego i pozwala uniknąć ruchów cyklicznych. Jednakże w pewnych określanych sytuacjach możliwe jest wykonanie ruchu zabronionego. Zdefiniowana jest specjalna funkcja, zwana funkcją aspiracji, która pozwala obliczyć, czy zabroniony ruch będzie jednak opłacalny.

Algorytm zatrzymuje się, gdy spełniony jest jeden z warunków zatrzymania. Takimi warunkami mogą być wykonanie z góry założonej iteracji algorytmu czy też wykonaniu ustalonej liczby ruchów, które nie prowadzą do dalszej poprawy rozwiązania.

3.2.2. Pseudokod algorytmu Tabu Search

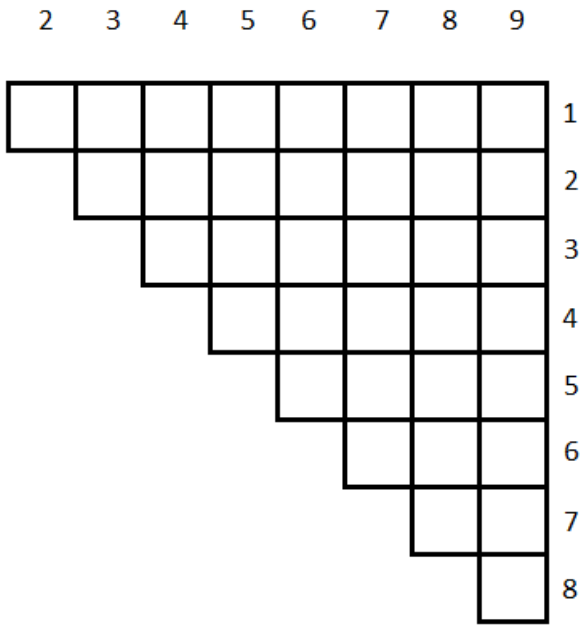
Inicjalizuj pierwsze rozwiązanie x;
Inicjalizuj rozwiązanie najlepsze x^b: $x^b \leftarrow x$;
while *nie wystąpił warunek stopu* **do**
    Przygotuj listę możliwych ruchów dla obecnego rozwiązania;
    Wybierz najlepszy możliwy ruch z uwzględnieniem tablicy tabu i kryterium aspiracji;
    Przypisz otrzymane w ruchu rozwiązanie do rozwiązania aktualnego x;
    if *rozwiązanie x jest lepsze od* x^b **then**
        $x^b \leftarrow x$
    end
    Zaktualizuj tablicę tabu i kryterium aspiracji;
end

Algorithm 2: Algorytm Tabu Search

3.2.3. Zastosowanie algorytmu Tabu Search dla problemu QAP

Rozwiązaniami problemu przydziału kwadratowego są permutacje określające przydział placówek do lokalizacji. Należy więc, mając dane aktualne rozwiązanie problemu QAP, określić w jaki sposób będzie wykonywany ruch w kolejnych iteracjach działania algorytmu. Zmiana aktualnego rozwiązania musi odbyć się w sposób, który nie spowoduje, że do danej lokalizacji zostanie przypisany więcej niż

jeden obiekt, jak również któryś z obiektów nie zostanie desygnowany do żadnego z miejsc. W przeciwieństwie do, przykładowo, algorytmu PSO, strategia Tabu Search pozwala na przeszukiwanie przestrzeni rozwiązań w sposób dosyć prosty. Istnieje wiele metod dokonywania ruchów w przypadku, gdy rozwiązanie jest permutacją. Najczęściej spotykanym w literaturze jest sposób polegający na zamianie miejscem dwóch elementów permutacji. Wynika stąd, że dla permutacji o długości n istnieje $\binom{n}{2}$ kombinacji takiego wyboru. Wykonane ruchu zapisywane są w tablicy tabu i trzymane są w niej przez określoną liczbę iteracji algorytmu. Poniżej znajduje się przykładowa tablica:



W powyższej tablicy w komórce o indeksie (i, j) wpisuje się liczbę iteracji algorytmu, podczas których zamiana obiektów o wartości (nie indeksie) nie można zamienić miejscami. Po każdej iteracji liczba ta jest zmniejsza o 1. Wpis dodawany jest, gdy nastąpiła zamiana miejscami obiektów o wartościach i i j.

Innymi metodami pozwalającymi na wykonanie ruchu w algorytmie TS są przykładowo wstawienie jednego z elementów permutacji w inne miejsce i przesunięcie pozostałych elementów, czy też inwersja wybranej grupy elementów permutacji o określonej szerokości.

3.3. Algorytm mrówkowy

3.3.1. Geneza i opis algorytmu

Algorytm mrówkowy (Ant Algoirthm) został stworzony przez Marco Dorigo, jak metoda rozwiązywania trudnych problemów optymalizacji jakimi są przykładowo problem komiwojażera (TSP, Travelling Salesman Problem) czy problem przydziału kwadratowego QAP. Inspiracją do powstania algorytmu

była obserwacja faktycznych, istniejących w naturze, rojów mrówek. Uwagę naukowców przykuło to, że mrówki, które same są dosyć prostymi stworzeniami, działając w grupie potrafią osiągnąć wysoki poziom organizacji, żyją w zhierarchiwizowanym społeczeństwie. Również ciekawą cechą w zachowaniu mrówek jest to, że nastawione są bardziej na przeżycie całej społeczności niż pojedynczego osobnika. Posiadają one także niespotykane umiejętności pozwalające im na znajdywanie najkrótszej drogi pomiędzy mrowiskiem a miejscem, w którym znajduje się pożywienie.

Ważnym czynnikiem pozwalającym na znajdywanie najkrótszej ścieżki do źródła pokarmu oraz zapamiętywania tejże drogi są substancje chemiczne wydzielana przez mrówki, zwane feromonami. Insekty te mają zdolność wyczuwania feromonów i dzięki temu najprawdopodobniej potrafią wybrać drogę, dla której stężenie feromonów jest największe. Pozwala to również innym osobnikom, na wykorzystanie informacji o lokacji pożywienia zdobytej przez inne mrówki.

Można w związku z powyższym metaforycznie stwierdzić, że mrówki posiadają zdolność wychodzenia z minimów lokalnych i wybierają minimum globalne.

3.3.2. Model matematyczny algorytmu

3.3.3. Zastosowanie algorytmu mrówkowego dla problemu QAP

3.4. Algorytmy ewolucyjne

Algorytmy ewolucyjne są najogólniej rzecz biorąc algorytmy optymalizacji, które bazują na stopniowym polepszaniu pewnej populacji rozwiązań danego problemu. Z powodzeniem są stosowane od wielu lat do rozwiązywania wielu zarówno praktycznych jak i teoretycznych problemów. Istnieje wiele różnorodnych implementacji algorytmów ewolucyjnych. Należą do nich między innymi[evolutionary PDF]:

– algorytmy genetyczne, stworzone przez Johna Henry'ego Hollanda,

– strategie ewolucyjne,opracowane przez Ingo Rechenberga oraz Hansa Paul Schwefela,

– programowanie ewolucyjne stworzone przez Lawrence'a Fogela.

Algorytmy ewolucyjne posiadają wiele cech, które odróżniają je od innych metod optymalizacji. Przede wszystkim zmianom poddawana jest zakodowana w łańcuchu znaków postać problemu , a nie jego parametry bezpośrednio i wykorzystywane jest doświadczenie poprzednich pokoleń. Łańcuch ten ma ustaloną długość i korzysta ze znaków ze skończonego alfabetu. Jak już zostało to wspomniane wcześniej, przetwarzana jest też pewna populacja rozwiązań, nie jedno. By ocenić dane rozwiązanie potrzebna jest jedynie funkcja celu, bądź też coś, co pozwoli porównać dwa rozwiązania i wyłonić lepsze z nich. Kolejnym elementem, który cechuje algorytmy ewolucyjne jest fakt, że stosowane są w nich niedeterministyczne, probabilistyczne reguły wyboru.

Algorytmy genetyczne są chyba najczęściej stosowanymi i najbardziej znanymi implementacjami algorytmów ewolucyjnych. Często, choć niepoprawnie, terminy *algorytmy ewolucyjne* oraz *algorytmy*

genetyczne stosowane są zamiennie. Z powodu ich popularności, dalsza część rozdziału będzie poświęcona tejże podgrupie algorytmów ewolucyjnych.

3.4.1. Geneza i opis algorytmów genetycznych

Algorytmy genetyczne zostały opracowane, jak już zostało to wspomniane, przez Johna Hollanda przy pomocy jego kolegów i studentów związanych z Uniwersytetem Michigan. Celami, które im przyświecały podczas tworzenia algorytmów były chęć opisania oraz wyjaśnienie istoty zjawisk zachodzących w świecie przyrody, dzięki którym możliwa jet adaptacja do różnych warunków, a także utworzenie oprogramowania mogącego symulować mechanizmy obecne w systemach biologicznych. Ważną cechą rzeczywistych systemów biologicznych jest odporność tych systemów. Potrafią one szybko zaadaptować się do zmieniających się warunków ich otaczających, posiadają duże zdolności regeneracyjne. Niewątpliwie są to cechy pożądane także przy projektowaniu różnego rodzaju systemów: inżynierskich, ekonomicznych. Skoro algorytmy genetyczne pojawiły się w oparciu o symulacje zjawisko zachodzących w naturze, może pojawić się pytanie czy one same posiadają podobne zdolności. Odpowiedź na nie podał Holland w roku 1975 udowadniając, że algorytmy genetyczne są odporną metodą poszukiwania rozwiązań nawet w skomplikowanych przestrzeniach. Inną ważną zaletą algorytmów genetycznych jest to, że stosowanie ich nie wymaga spełniania założeń dotyczących przestrzeni poszukiwań jakimi są przykładowo ciągłość czy różniczkowalność. Dzięki stosowaniu probabilistycznych technik wyboru, a także dzięki przetwarzaniu całej populacji rozwiązań, w przeciwieństwie do wielu innych analitycznych metod optymalizacji, zmniejszona znacznie jest szansa na zatrzymanie się w ekstremum lokalnym. W tym sensie metody analityczne przejawiają swój brak odporności. Jednakże, z faktu, iż algorytm genetyczny jest metodą przybliżoną, w przypadku optymalizowania wielu problemów nie jest możliwe stwierdzenie, czy znalezione rozwiązanie, zwrócone przez algorytm jest optymalne i jak daleko znajduje się od optimum. Nie można również zagwarantować, że start algorytmu przy pewnych początkowych ustawieniach zawsze zwróci ten sam rezultat.

W klasycznej, najprostszej wersji, algorytm genetycznych składa się z kilku podstawowych operacji. Mając wygenerowaną losowo populację początkową uruchamiamy algorytm na określoną liczbę operacji. W każdej iteracji dokonujemy oceny każdego z rozwiązań w populacji i w oparciu o ich wartość dopasowania rozwiązań dokonujemy selekcji osobników, na których będą wykonane operacje krzyżowania oraz mutacji. W zależności od postaci rozwiązania, a także od metody jego kodowania, operatory genetyczne mogą działać w różnoraki sposób. W ogólnym przypadki, krzyżowanie polega na wymianie informacji pomiędzy osobnikami wybranymi na drodze selekcji, a mutacja na losowej zmianie wybranego rozwiązania, mogącej zmienić dane rozwiązanie na lepsze bądź gorsze. Teoretycznie z każdą iteracją, ogólne dopasowanie całego pokolenia powinno się polepszać. Ogólną ideą algorytmu jest przetrwanie najsilniejszych i eliminacja słabszych.

3.4.2. Operatory selekcji

Już sam sposób doboru osobników, które posłużą za podstawę dla kolejnego pokolenia, ma kluczowe znaczenie. Istnieje wiele sposobów wyboru rozwiązań. Do takich metod można zaliczyć między innymi:

– metoda ruletki,

– metoda rankingowa,

– metoda turniejowa.

Metoda ruletki polega na "kręceniu"wirtualną ruletką, na której każde z rozwiązań z aktualnej populacji ma wyznaczony swój wycinek koła. Szerokość wycinka zależy od dopasowania danego rozwiązania. Dla każdego rozwiązania jest liczona wartość funkcji dopasowania, a także liczona jest suma wszystkich wartości dopasowania. Stosunek wartości dopasowania danego osobnika do sumy wszystkich dopasowań wyznacza szerokość wycinka ruletki dla danego rozwiązania. Metoda ta faworyzuje osobniki lepsze, nie pozbawiając jednak szansy wyboru tych gorszych. Jednak może to prowadzić w sytuacji, gdy jedno z rozwiązań jest wyraźnie lepsze od pozostałych, że osobniki wybrane do krzyżowania będą się składać głownie z tego jednego.

Metoda bazująca na rankingu rozwiązań nieco zmniejsza rolę dopasowania danego rozwiązania w kontekście jego szans do bycia wybranym na rodzica. W tej metodzie rozwiązania są szeregowane według swojego dopasowania, a o prawdopodobieństwie wyboru decyduje pozycja w rankingu. W oparciu o ranking definiuje się funkcję, która określa ile kopii danego rozwiązania jest brane pod uwagę. W tym sposobie dysproporcje między prawdopodobieństwami wyboru rozwiązań są mniejsze w stosunku do metody ruletkowej.

Metoda turniejowa polega na losowym wyborze dwóch rozwiązań z populacji i rozwiązanie lepsze z tej pary jest brane pod uwagę w dalszych operacjach. Rozwiązania lepsze mają taką samą szansę bycia wybranym do porównania co te słabsze, lecz i tak w przypadku pojedynku, to one wygrają.

Należy również wspomnieć, że funkcja dopasowania powinna zwracać liczby nieujemne i dla rozwiązania lepszego jego wartość dopasowania powinna być większa, niż dla gorszego. Nie zawsze da się taką informacje uzyskać bezpośrednio z funkcji celu. Znana jest własność dualności zadań minimalizacji kosztu i maksymalizacji zysku, lecz w przypadku funkcji przystosowania, zwracana wartość zawsze musi być nieujemna. Z tego powodu należy dokonać przekształcenia funkcji celu w funkcję dopasowania. Najczęściej dokonywane to jest poprzez odejmowanie wartości funkcji celu od pewnej liczby. Jeśli taka różnica jest ujemna, zwracamy 0. Wartość tej liczby może być ustalona w wieloraki sposób. Przykładowo, może to być z góry ustalona liczba, może też być modyfikowana wraz z kolejnymi iteracjami algorytmu, np. może przyjąć wartość największej znalezionej do tej pory wartości funkcji celu.

3.4.3. Operatory krzyżowania

Celem operatorów krzyżowania, zwanych inaczej mieszania, jest spowodowanie wymiany informacji pomiędzy wybranymi rozwiązaniami i utworzenie na tej podstawie kolejnych. Mając wyselekcjonowaną

populacje rozwiązań, łączymy w pary osobniki i dokonujemy ich krzyżowania. To w jaki sposób zostanie to wykonane zależy od postaci rozwiązania. Istnieje wiele różnych metod krzyżowania. Inne są wykorzystywane dla rozwiązań kodowanych w sposób binarny, inne dla kodów wykorzystujących liczby rzeczywiste itp. Do najbardziej znanych metod krzyżowania należą:

– krzyżowanie jednopunktowe,

– krzyżowanie wielopunktowe,

– krzyżowanie z częściowym odwzorowaniem PMX,

– krzyżowanie cykliczne CX,

– krzyżowanie z zachowaniem porządku OX.

Trzy ostatnie z wymienionych operatorów dotyczy rozwiązań będących permutacjami i ich użycie pozwala na zachowanie tej postaci po dokonaniu krzyżowania. Operatory jednopunktowy i wielopunktowy powodują wymianę pomiędzy osobnikami fragmentów kodu pomiędzy wylosowanymi punktami. Operacja krzyżowania wykonywana jest z pewnym określonym jako parametr algorytmu prawdopodobieństwem. Dla każdego z rozwiązań, wybranych na drodze selekcji, losowane jest czy będzie brane pod uwagę w dalszych działaniach. W zależności od przyjętych założeń, może zaistnieć potrzeba doboru dodatkowych rozwiązań w przypadku, gdy jest ich niewystarczająca ilość, by móc dokonać operacji krzyżowania.

3.4.4. Operatory mutacji

Operacja mutacji jest losową zmianą dokonywaną na rozwiązaniach wzorowaną na mutacjach występujących faktycznie w przyrodzie. Jest ona błądzeniem przypadkowych w przestrzeni ciągów kodowych[goldberg]. Operator mutacji pozwala na przywrócenie utraconych ważnych informacji w kodzie rozwiązania, bądź na uzyskanie dobrych (lub złych) cech w rozwiązaniu, których często nie dałoby się otrzymać na drodze krzyżowania. Z racji iż prawdopodobieństwo wystąpienia mutacji jest wielokrotne mniejsze niż wystąpienie krzyżowania, operacja ta odgrywa drugorzędną rolę w działaniu algorytmu, choć niejednokrotnie pozwala na uzyskanie zaskakujących rezultatów.

W przypadku binarnej postaci kodu rozwiązań operacja mutacji najczęściej polega na zamianie wartości bit z 0 na 1 i na odwrót. W innych przypadkach mutacja polega na zmianie wybranego elementu na inny dopuszczalny. W sytuacji, gdy rozwiązaniem jest permutacja, należy zadbać, by operator mutacji pozostawiał rozwiązanie w poprawnej postaci. Mutacja w takim przypadku może polegać przykładowo na zamianie miejscami dwóch elementów, czy też na inwersji pewnego fragmentu kodu.

3.4.5. Schemat działania algorytmu genetycznego

Poniżej znajduje się pseudokod dla klasycznej wersji algorytmu genetycznego: Inicjalizuj populację początkową;

while *nie wystąpił warunek stopu* **do**

Dokonaj selekcji rozwiązań będącej podstawą dla nowego pokolenia; Dokonaj operacji krzyżowania na wybranych osobnikach;

Dokonaj operacji mutacji na osobnikach otrzymanych na drodze krzyżowania;

Zaktualizuj populację w oparciu o otrzymane rozwiązania w wyniku działania operatorów genetycznych

end

Algorithm 3: Algorytm genetyczny

3.4.6. Zastosowanie algorytmu PSO dla problemu QAP

Algorytm genetyczny można w łatwy sposób zaimplementować dla problemu QAP. Postać rozwiązania problemu przydziału kwadratowego to permutacja. Należy więc na każdym etapie działania algorytmu dokonywać zmian w taki sposób, by zwracane w kolejnych iteracjach populacje rozwiązań były populacjami permutacji. Wyżej zostały wymieniowe metody krzyżowania, takie jak operatory OX, PMX, CX, i mutacji, które mogą być stosowane dla rozwiązań permutacyjnych. Optymalizacja problemu QAP polega na minimalizacji kosztu przydziału obiektów do lokalizacji. Z tego powodu funkcja celu (wzór) nie nadaje się wprost do oceny przystosowania osobników. Funkcję dopasowania można więc uzyskać z funkcji celu poprzez odejmowanie wartości funkcji celu od pewnej liczby, którą może być największa znaleziona dotychczas wartość funkcji celu, czy też najgorsza wartość dopasowania w ostatniej iteracji itp.

4. Zastosowanie algorytmu quantum EA dla zagadnienia QAP

Sposób w jaki działają algorytmy genetyczne, otwarty schemat działania, skłania do tworzenia wielu modyfikacji. Zmianom mogą podlegać operatory krzyżowani, mutacji, sposób kodowania rozwiązania. Można dodawać też nowe operatory o działaniu nieobjętym przez tradycyjne operatory. Z tego powodu, na przestrzeni lat, pojawia się wiele publikacji na temat algorytmów genetycznych i nowych sposób podejścia do tego tematu. W jednej z takich publikacji, autorzy Jinwei Gu, Xingsheng Gu i Manzhan Gu zaproponowali algorytm o nazwie *ä novel parallel quantum genetic algorithm¨ NPQGA* i przedstawili jego wykorzystanie dla problemu szeregowania zadań. Jednakże algorytm ten nadaje się także dla innych zastosowań, do jakich należy na przykład problem QAP i należy do grupy tak zwanych kwantowych algorytmów ewolucyjnych.

4.1. Opis algorytmu

Główną cechą kwantowych algorytmów ewolucyjnych jest zastosowanie w nich bitów kwantowych - kubitów. Wykorzystywane są one do reprezentacji rozwiązań algorytmów. Kubit w danym momencie może reprezentować teoretycznie nieskończenie wiele stanów będących superpozycją stanu 0 i 1. Obserwacja stanu bitu kwantowego pozwala dopiero na jednoznaczną ocenę jego stanu. Stan kubitu może być reprezentowany przez równanie:

$$|\psi >= \alpha|0 > +\beta|1 > \tag{4.1}$$

gdzie $|\alpha|^2$ jest prawdopodobieństwem, że kubit znajduje się w stanie 0, oraz $|\beta|^2$ jest prawdopodobieństwem, że kubit jest w stanie 1. α i β są liczbami zespolonymi. Obie liczby są znormalizowane, co znaczy, że:

$$|\alpha|^2 + |\beta|^2 = 1 \tag{4.2}$$

Kubit jest więc najmniejszą jednostką informacji w tych algorytmach i jest reprezentowany poprzez parę liczb $\lfloor{}^{\alpha}_{\beta}$. Podobnie jak w innych algorytmach ewolucyjnych, algorytm NPQGA bazuje na zmieniają-

cych się w czasie, dynamicznych populacjach rozwiązań i korzysta z funkcji oceniającej te rozwiązania wykorzystując własności bitów kwantowych. Oprócz stosowania tradycyjnie rozumianych operatorów selekcji, krzyżowania oraz mutacji, autorzy zaproponowali również operator katastrofy, a także operator związany z bramkami kwantowymi, służący do zmiany stanów kubitów.

4.1.1. Kodowanie rozwiązań

W publikacji, został zawarty został przykład obrazujący działanie algorytmu dla problemu szeregowania zadań. Został również przedstawiony sposób kodowania rozwiązań, który nadaje się również dla problemu przydziału kwadratowego. Ogólnie rozwiązanie problemu jest ciągiem kubitów i można je przedstawić w następujący sposób:

$$\left[\begin{array}{c|c|c|c} \alpha_1 & \alpha_2 & ... & \alpha_l \\ \beta_1 & \beta_2 & ... & \beta_l \end{array}\right] \tag{4.3}$$

gdzie

$$l = ([\log_2^n] + 1) \cdot n \tag{4.4}$$

a *n* oznacza rozmiar problemu, czyli ilu elementowa jest permutacja reprezentująca rozwiązanie problemu. Nawiasy kwadratowe oznaczają cechę liczby. Niestety z samego ciągu kubitów nie wynika od razu jaką permutację ten ciąg koduje. By uzyskać rozwiązanie permutacyjne, które jest używane dla problemu QAP, należy wykonać następujące kroki:

1. dla każdego kubitu wylosuj liczbę $\ni$ z przedziału *[0,1]*,
2. jeśli $\ni < |\alpha_i|^2$, to określ stan *i-tego* kubitu na 1, w przeciwnym przypadku na *0*,
3. dla utworzonego ciągu bitów, każde $[\log_2^n] + 1$ bitów zamień na postać dziesiętną,
4. mając ciąg liczb naturalnych posortuj go rosnąco z zapamiętaniem pozycji liczb w ciągu,
5. jeśli dwie kolejne liczby są różne, to mniejsza z nich reprezentuje przydzielony do placówki o numerze indeksu, obiekt o mniejszym numerze, a jeśli są równe, to liczba z mniejszym indeksem reprezentuje obiekt o niższym numerze.
6. ustaw rosnąco według indeksów powyższy ciąg liczb naturalnym zastępując te liczby odpowiadającymi im numerami przydzielonych obiektów.

W ten sposób uzyskana zostaje permutacja, w której pozycja określa numer lokalizacji, a wartość liczby na tej pozycji, określa przedzielony do niej obiekt.

4.1.2. Operatory genetyczne

Jako, że algorytm NPQGA jest modyfikacją algorytmu genetycznego, w swym działaniu korzysta z typowych operatorów genetycznych. Autorzy algorytmu w zaprezentowanym przykładzie zaproponowali selekcję ruletkową, operator krzyżowania CX, mutację polegającą na zamianie w losowym kubicie parametrów α i β oraz bramkę kwantową do zmiany stanów kubitów o nazwie *rotation gate*.

Pewnym nietypowym rozwiązaniem związanym z krzyżowaniem jest zmiana prawdopodobieństwa zajścia krzyżowania z czasem. Im więcej iteracji algorytmu minęło, tym mniejsza jest prawdopodobieństwo krzyżowania. Należy ustawić jako parametry algorytmu prawdopodobieństwa maksymalne i minimalne zajścia krzyżowania.

$$P_c^+ = \begin{cases} \frac{P_{cmax}}{1+\frac{t}{t_max}}, \ P_c^+ > P_cmin \\ P_{cmax}, \ P_c^+ < P_cmin \end{cases} \tag{4.5}$$

Operator CX działa w następujący sposób:

1. Wybierany jest dowolny element z pierwszego z rodziców, najczęściej jest to pierwszy element permutacji.
2. Sprawdzana jest wartość elementu w drugim rodzicu na pozycji tej samej co wybrany element w pierwszym rodzicu.
3. Znajdywany jest element w pierwszym rodzicu o wartości sprawdzonej w punkcie 2 i dla tego elementu powtarzamy krok 2.
4. Wykonywane są powyższe kroki, aż do dotarcia w pierwszym rodzicu do punktu startowego.
5. Uzyskane w ten sposób zestawy punktów w obu rodzicach przenoszone są do rozwiązań potomków z zachowaniem indeksów elementów permutacji w taki sposób, że elementy z rodzica pierwszego umieszczane są w potomku nr 2 i na odwrót.
6. Powtarzane jest szukanie punktów poczynając od pierwszego niewybranego punktu w rodzicu pierwszym i znalezione grupy punktów są kopiowane do potomków, lecz tym razem elementy z pierwszego rodzica zostają umieszczone w potomku pierwszym. W następnym wyszukiwaniu ponownie elementy z rodzica pierwszego kopiowane są do potomka drugiego.
7. Wyszukiwanie cykli elementów powtarza się aż wszystkie elementy zostaną wybrane.

Mutacja zachodzi wtedy dla danego osobnika, gdy wylosowana dla niego liczba z przedziału *[0,1]* jest mniejsza niż prawdopodobieństwo mutacji p_m. Wtedy losuje się, który kubit z rozwiązania poddany będzie modyfikacji, która wygląda w sposób następujący:

$$\begin{bmatrix} \alpha_i' \\ \beta_i' \end{bmatrix} = \begin{bmatrix} \beta_i \\ \alpha_i \end{bmatrix} \tag{4.6}$$

Po dokonaniu powyższej zamiany, należy ponownie sprawdzić stan kubitu, co może się wiązać ze zmianą liczby dziesiętnej, w której skład wchodzi zmodyfikowany kubit, co dalej może pociągać za sobą zmianę całej permutacji.

Operator bramki kwantowej jest tym elementem algorytmów kwantowych, który ma największy wpływ na zmianę stanów bitów kwantowych. Istnieje wiele różnych odmian bramek kwantowych, takie jak bramka NOT, CNOT, bramka Hadamarda. W algorytmie NPQGA została zaproponowana bramka rotacyjna - *rotation gate*. Uaktualnienie parametrów α i β następuje w następujący sposób:

$$\begin{bmatrix} \alpha_i' \\ \beta_i' \end{bmatrix} = \begin{matrix} \cos(\Theta_i) & -\sin(\Theta_i) \\ \sin(\Theta_i) & \cos(\Theta_i) \end{matrix} \tag{4.7}$$

Kąt Θ_i określony jest poprzez swoją wartość i kierunek obrotu:

$$\Theta_i = \Delta_I \cdot s(\alpha_i, \beta_i), \tag{4.8}$$

gdzie Δ_i określa o jaki kąt należy dokonać rotacji, a $s(\alpha_i, \beta_i)$ określa kierunek obrotu. Zarówno wartość kąta i jego kierunek odczytuje się z tablicy *Look Up*.

5. Modyfikacja algorytmu quatum QAP

6. Aplikacja rozwiązująca problem przydziału kwadratowego z wykorzystaniem kwantowego algorytmu ewolucyjnego

7. Metodyka eksperymentów

7.1. Instancje testowe

7.2. Scenariusze testowe

8. Eksperymenty obliczeniowe

9. Rezultaty działania algorytmu dla oraz przeprowadzone testy

10. Analiza uzyskanych wyników

11. Podsumowanie i wnioski

Bibliografia

[1] H. Partl: *German TEX*, TUGboat Vol. 9,, No. 1 ('88)