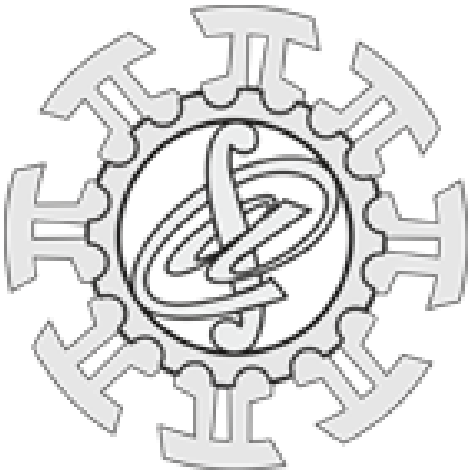

Wydział Matematyki Stosowanej

PRACA INŻYNIERSKA

Temat: Porównanie wybranych algorytmów heurystycznych w rozwiązywaniu zagadnień odwrotnych

Dyplomant: Kamil Kryus
Kierunek studiów: Informatyka
Specjalizacja: Programowanie Aplikacji Mobilnych

Opiekun pracy: dr inż. Adam Zielonka

Oświadczenie kierującego projektem inżynierskim

Potwierdzam, że niniejszy projekt został przygotowany pod moim kierunkiem i kwalifikuje się do przedstawienia go w postępowaniu o nadanie tytułu zawodowego: inżynier.

Data Podpis kierującego projektem

Oświadczenie autora

Świadomy odpowiedzialności karnej oświadczam, że przedkładany projekt inżynierski na temat: Porównanie wybranych algorytmów heurystycznych w rozwiązywaniu zagadnień odwrotnychzostał napisany przez autra samodzielnie.
Jednocześnie oświadczam, że ww. projekt:

-nie narusza praw autorskich w rozumieniu ustawy z dnia 4 lutego 1994 roku o prawie autorskim i prawach pokrewnych (Dz. U. z 2000 r. Nr 80, poz. 904 z późn. zm.) oraz dóbr osobistych chronionych prawem cywilnym, a także nie zawiera danych i informacji, które uzyskałem w sposób niedozwolony,

-nie była wcześniej podstawą żadnej innej urzędowej procedury związanej z nadawaniem dyplomów wyższej uczelni lub tytułów zawodowych,

-nie zawiera fragmentów dokumentów kopiowanych z innych źródeł bez wyraźnego zaznaczenia i podania źródła.

Podpis autora projektu:

1. Kamil Kryus nr albumu: 246591 .....
podpis

Gliwice, dnia .....

Spis treści

Rozdział 1

Wprowadzenie

Rozdział 2

Opis problemu

Rozdział 3

Opis algorytmów

3.1 Algorytm symulowanego wyżarzania

Algorytm ten został stworzony wzorując się na zjawisku wyżarzania w metalurgii, które polega na nagrzaniu elementu stalowego do odpowiedniej temperatury, przetrzymaniu go w tej temperaturze przez pewien czas, a następnie powolnym jego schłodzeniu. Sam algorytm natomiast bazuje na metodach Monte-Carlo i w pewnym sensie może być rozważany jako algorytm iteracyjny.

Główną istotą i zarazem zaletą tego algorytmu jest wykonywanie pewnych losowych przeskoków do sąsiednich rozwiązań, dzięki czemu jest w stanie uniknąć wpadania w lokalne minimum. Algorytm ten najczęściej jest używany do rozwiązywania problemów kombinatorycznych, jak np. problemu komiwojażera.

3.1.1 Parametry

Początkowa konfiguracja

W tym kroku powinniśmy zainicjalizować naszą temperaturę wysoką wartością oraz znaleźć początkowe losowe rozwiązanie naszego problemu.

Temperatura

Temperatura jest zarówno czynnikiem iteracyjnym, jak i jest związana z funkcją prawdopodobieństwa zamiany gorszego rozwiązania z lepszym. Zatem zakres temperatury powinien być taki, aby na początku działania naszego algorytmu dawał wysoką możliwość zamian, a wraz z postępem iteracji te prawdopodobieństwo się zmniejszało i pod koniec było bliskie zeru.

Końcowa temperatura

Jest to bardzo mała wartość, która sugeruje, iż gdy wartość temperatury spadnie do tego poziomu, zakończył się proces wyżarzania i rozwiązanie zostało znalezione. Wartość ta powinna być na tyle mała, by temperatura będąc niewiele większa prowadziła do bardzo niskiego prawdopodobieństwa, a jednocześnie nie wymagało to zbyt dużej ilości iteracji.

Powtarzanie pewną ilość razy dla zadanej temperatury

Wartość ta powinna być z góry ustalona i powinna dać nam możliwość sprawdzenia wielu sąsiadów obecnego rozwiązania, równocześnie nie powodując zbyt dużego obciążenia dla algorytmu.

Znajdowanie losowego sąsiada poprzedniego rozwiązania

Funkcja ta powinna nam pozwalać przejrzeć jak najszerszy zakres rozwiązań, a jednocześnie pozwolić na przeszukiwanie coraz to bliższych sąsiadów obecnie najlepszego rozwiązania, zatem warto uzależnić tą funkcję od stopnia ukończenia globalnych iteracji.

Funkcja kosztu

Poprzez funkcję kosztu rozumiemy różnicę pomiędzy obecnie najlepszym rozwiązaniem, a nowym. W tym wypadku nowe rozwiązanie zawsze jest gorsze (a tym samym przy poszukiwaniu minimum globalnego posiada wartość większą), to w tym wypadku wartość jest ujemna.

Prawdopodobieństwo zamiany P

Prawdopodobieństwo jest potrzebne do decyzji czy zamienić nasze nowe i gorsze rozwiązanie z wcześniejszym, lepszym.

Prawdopodobieństwo zależy od różnicy pomiędzy starą wartością, a nową, zwaną tutaj dystansem, oraz obecnej temperatury. Prawdopodobieństwo zatem można przedstawić w następujący sposób:

$$P = e^{\frac{\Delta E}{T}}$$

gdzie:

$$\Delta E - funkcja\ kosztu \tag{3.1}$$

T - obecna wartość temperatury

Prawdopodobieństwo to wraz ze spadkiem wartości funkcji kosztu maleje (gdyż jest zawsze ujemne), natomiast wyższa wartość temperatury zwiększa prawdopodobieństwo. Decydując o tym czy powinniśmy zamienić nasze gorsze rozwiązanie z lepszym powinniśmy porównać z wartością losową zawierającą się w zakresie [0,1].

Zmniejszanie temperatury

Szybkość zmniejszania nie powinna być zbyt duża, aby pozwolić algorytmowi na sprawdzenie jak największego zakresu możliwych rozwiązań, a jednocześnie niezbyt

wolna, gdyż może to spowodować zbyt wolny spadek prawdopodobieństwa, a tym samym finalnie pozwolić na pozostanie w gorszym rozwiązaniu. W większości opracowań można spotkać ten proces, jako mnożnik temperatury w zakresie [0.8;0.99].

3.1.2 Kroki algorytmu

Algorytm ten można również przedstawić za pomocą listy kroków:

1. Zainicjalizuj początkową konfigurację
2. Dopóki temperatura $>$ minimum, powtarzaj:
   (a) Powtórz pewną ilość razy dla danej temperatury
      i. Znajdź losowo sąsiada poprzedniego rozwiązania
      ii. Sprawdź czy rozwiązanie jest lepsze od poprzedniego (funkcja kosztu)
         A. Jeżeli jest, zamień rozwiązania
         B. Jeżeli nie jest, zamień rozwiązania z pewnym prawdopodobieństwem P
   (b) Zmniejsz temperaturę

Rozdział 4

Cel

Przetestowanie algorytmu wyżarzania, bla bla

Rozdział 5

Funkcje testowe

5.1 Funkcja kwadratowa dwóch parametrów

Funkcja ta ma następującą postać:

$$f(x, y) = x^2 + y^2$$

Cośo wykresie: <tutaj jakis wykres >

Funkcja ta przyjmuje jedynie wartości dodatnie, zakres dla potrzeb projektu został zawężony do [-10, 10], natomiast minimum globalne posiada w punkcie [0, 0] i wynosi 0.

5.2 Funkcja Rastrigina

Funkcja Rastrigina jest funkcją ciągłą, skalowalną i multimodalną. Dzięki posiadaniu wielu minimum lokalnych funkcja ta jest często stosowana w testowaniu algorytmów optymalizacyjnych.

$$f(x) = An + \sum_{i=1}^{n} [x_i^2 - A\cos(2\pi x_i)]$$

gdzie:
A = ilość wymiarów

Parametry funkcji możemy znaleźć w zakresie [-5.12; 5.12], minimum globalne natomiast w punkcie [0,...,0] i wynosi 0.

5.3 Funkcja Rosenbrocka

Rosenbrock's valley function is known as the second function of De Jong. This test function is continuous, scalable, naturally nonseparable, nonconvex, and unimodal.

$$f(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2]$$

Rozdział 6

Dobór parametrów

Rozdział 7

Implementacja

Rozdział 8

Dostosowanie algorytmów do funkcji testowej zadań odwrotnych

Rozdział 9

Narzędzia i technologie

Aby zapewnić bezpieczeństwo oraz możliwość pracy w kilku miejscach nad jednym problemem, zastosowałem kilka rozwiązań.

9.1 Użyte narzędzia

9.1.1 System kontroli wersji

Git

9.1.2 Podział pracy na mniejsze zadania

Git

9.2 Użyte technologie

.NET Framework

Rozdział 10

Podsumowanie

10.1 Dalsze kierunki rozwoju

Foobar

10.2 Źródła

https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4538776/