Projektowanie efektywnych algorytmów
Projekt

263934 Michał Pawlus

20.12.2023

Ant Colony Optimization

Spis treści	strona

Sformułowanie zadania
Zadanie polega na opracowaniu i implementacji algorytmu ACO, który rozwiązywać będzie problem znalezienia minimalnego cyklu Hamiltona w grafie ważonym. Problem ten nazywa się problemem komiwojażera i jest problemem NP-trudnym.

Wielkość błędu bezwzględnego, w zależności od wielkości instancji, nie może przekraczać:
- dla $n < 24$, 0%,
- dla $25 < n < 74$, 50%,
- dla $75 < n < 449$, 100%
- dla $450 < n < 2500$, 150%.

Metoda
Algorytm ACO (Ant Colony Optimization) to metaheurystyka inspirowana zachowaniami kolonii mrówek w poszukiwaniu najlepszej trasy do jedzenia. Należy do grupy algorytmów inteligencji stadnej, które bazują na kolektywnym zachowaniu jednostek w samoorganizujących się systemach. W celu znalezienia najlepszej trasy mrówki komunikują się ze sobą zostawiając w otoczeniu feromony. Taki rodzaj komunikacji nazywa się stygmergią. Mrówki mają skłonność do poruszania się za śladem feromonowym. Mrówka furażując, pozostawia ślad feromonowy na swej drodze. Gdy znajdzie źródło pożywienia, wraca do mrowiska wzmacniając drogę powrotu śladem feromonowym. Ślady te z czasem zanikają.

Algorytm mrówkowy dla problemu komiwojażera działa w następujący sposób:

1. Uruchamia się przez określoną liczbę iteracji lub do spełnienia warunku zatrzymania.
2. W każdej iteracji określona liczba mrówek poszukuje cyklu Hamiltona, podejmując probabilistyczne decyzje dotyczące wyboru kolejnych wierzchołków tworzących cykl. Decyzje te bazują głównie na ilości feromonów na ścieżce(Początkowa liczba feromonów na każdej ścieżce jest taka sama).
3. Po zakończeniu każdej iteracji aktualizuje się ilość feromonów.

Liczba wierzchołków jest równa liczbie mrówek. Każda mrówka zaczyna w innym wierzchołku.

Początkowa ilość feromonów na ścieżkach:

$$\tau_0 = \frac{m}{C^{\mathrm{nn}}}$$

m - liczba mrówek
Cnn - szacowana długość trasy. Obliczana na podstawie średniej z 1000 losowanych tras.

Mrówka podejmuje decyzje o wybraniu następnego wierzchołka na podstawie prawdopodobieństwa podanego wzorem:

$$p(c_{ij}|s^p) = \frac{\tau_{ij}^{\alpha} \cdot \eta_{ij}^{\beta}}{\sum_{c_{il} \in N(s^p)} \tau_{il}^{\alpha} \cdot \eta_{il}^{\beta}}, \bigvee c_{ij} \in N(s^p),$$

gdzie:

$C = \{c_{ij}\}, i + 1, \dots, n, j = 1, \dots, |D_i|$ - zbiór komponentów rozwiązania,

s^p – rozwiązanie częściowe uzupełniane o dopuszczalne komponenty z sąsiedztwa $N(s^p) \subset C$,

τ_{ij} i η_{ij} – wartości określające ilość feromonu i lokalnej funkcji wyboru dla komponentu (krawędzi) c_{ij},

α i β – współczynniki (dodatnie) określające wpływ (ważność) feromonu i heurystyki

W implementacji przyjęto wartości podane na wykładzie:
α = 1
β = 2,5 (na wykładzie 2 do 5)

Wostatecznej implementacji jako schematu rozkładu feromonów użyłem CAS(cykliczny), który polega na aktualizacji feromonów na ścieżkach po wykonaniu iteracji tzn. każda mrówka znalazła trasę.

W tym schemacie stała ilość feromonu $Q3$ dzielona jest przez długość drogi $L\ k$ przebytej przez k-tą mrówkę i rozkładana na wszystkich krawędziach tej drogi.

$$\Delta\tau_{ij}^{k}(t, t+n) = Q_3/L^k$$

Po każdej iteracji feromony aktualizowane są w sposób

$$\tau_{ij}(t+n) = \rho \cdot \tau_{ij}(t) + \Delta\tau_{ij}(t, t+n),$$

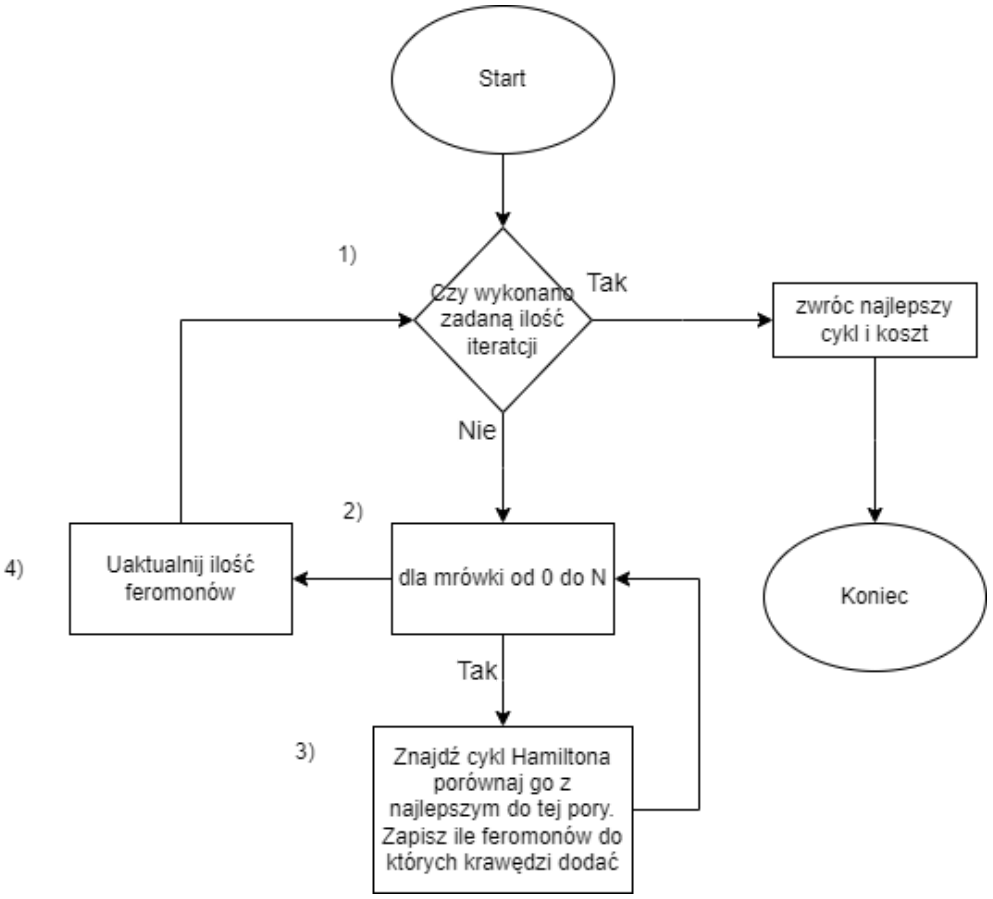
gdzie:

$$\Delta\tau_{ij}(t, t+n) = \sum_{k=1}^{m} \Delta\tau_{ij}^{k}(t, t+n)$$

ρ informuje jaka procentowa liczba feromonów wyparowała. W implementacji przyjęte zostało zgodnie z informacją podana na wykładzie ρ = 0.5

Oprócz niego badałem również schemat DAS(gęstościowy), który polega na aktualizowaniu ilości feromonów z każdym razem gdy mrówka przejdzie przez krawędź o stałą wartość feromonu Q3.

Algorytm



Rysunek 1: Schemat blokowy algorytmu

1) *Liczba iteracji dobierana na podstawie przedziałów dla błędów. Dla różnych przedziałów inna liczba iteracji*
2) *Pętla wykonująca tworzenie ścieżki dla N mrówek*
3) *Szukanie Cyklu Odbywa się przez pętle, która dobiera nowy wierzchołek probabilistycznie z prawdopodobieństwem obliczonym zgodnie z wyżej podanym wzorem na prawdopodobieństwo. Następnie następuje porównanie długości otrzymanej ścieżki z dotychczasowym najlepszym wynikiem, który w przypadku gdy jest to pierwsza iteracja wynosi -1, jeśli jest mniejszy to zostaje on nowym najlepszym wynikiem. Następnie obliczana jest wartość feromonu dla ścieżki zgodnie ze schematem CAS i jest ona zapisywana, by można jej było później użyć do aktualizowania feromonów*
4) *Po przejściu grafu przez wszystkie mrówki aktualizowana jest ilość feromonu na jego krawędziach. Najpierw wartości feromonu, przechowywane w tablicy, są mnożone przez współczynnik parowania a następnie dodawane są do nich wartości obliczone w kroku 3.*

Dane testowe

Do sprawdzenia poprawności działania algorytmu wybrano następujący zestaw instancji:
(Nazwa pliku)(optymalna ścieżka)

- Dostrajanie/gr17.txt_converted.csv 10 2085
- Dostrajanie/gr21.txt_converted.csv 10 2707
- Dostrajanie/gr24.txt_converted.csv 10 1272

Dla powyższych instancji używając algorytmu ACO udało się otrzymać rozwiązania optymalne. Jednak częściej zwracają rozwiązania przybliżone.

Do sprawdzenia jak zmiana parametrów α i β wpływa na działanie algorytmu wybrano następujący zestaw instancji:

- Dostrajanie/ft53.txt_converted.csv 10 6905
- Dostrajanie/gr202.txt_converted.csv 5 40160
- Dostrajanie/ch150.txt_converted.csv 5 6528

Do wykonania ostatecznych badań wybrano następujący zestaw instancji:
Instancje symetryczne

- Dostrajanie/gr17.txt_converted.csv 10 2085
- Dostrajanie/gr21.txt_converted.csv 10 2707
- Dostrajanie/gr24.txt_converted.csv 10 1272
- Dostrajanie/ftv70.txt_converted.csv 10 1950
- Dostrajanie/ft53.txt_converted.csv 10 6905
- Dostrajanie/gr202.txt_converted.csv 5 40160
- Dostrajanie/ch150.txt_converted.csv 5 6528
- Dostrajanie/linhp318.txt_converted.csv 5 41345
- Dostrajanie/rd400.txt_converted.csv 5 15281
- Dostrajanie/gr666.txt_converted.csv 5 294358

Są to przekonwertowane instancje problemu z instrukcji. pr1002.tsp i pr2392.tsp miały zbyt duże czasy wykonywania (ponad 10 minut) dlatego pozwoliłem je sobie pominąć w badaniach. Instancje rbg323.atsp i rgb 443.atsp zawierają trasy o wadze 0 (zablokowane), których moja implementacjaalgorytmu nie obsługuje. Zamiast nich użyłem instancji linhp318 i rd400

Procedura badawcza
Zbadałem czas rozwiązania problemu oraz błędu względem rozwiązania optymalnego w zależności od wielkości instancji.

Procedura badawcza polegała na podaniu pliku inicjalizującego z listą instancji, ręcznym zmienianiu wartości parametrów, schematu rozkładu feromonów i wykonaniu badania. Wyniki(czas i błąd) zapisywane były w pliku output.csv, które były zbierane do jednego folderu.

format pliku inicjalizującego:
nazwa_instancji liczba_wykonań rozwiązanie_optymalne

Przykładowy plik inicjalizujący:

Dostrajanie/bayg29.txt_converted.csv 5 1610
Dostrajanie/gr17.txt_converted.csv 5 2085
Dostrajanie/gr21.txt_converted.csv 5 2707
Dostrajanie/gr24.txt_converted.csv 5 1272
Dostrajanie/gr48..txt_converted.csv 5 5046
Dostrajanie/ftv33.txt_converted.csv 5 1286
Dostrajanie/ftv38.txt_converted.csv 5 1530
Dostrajanie/ftv44.txt_converted.csv 5 1613
Dostrajanie/ftv47.txt_converted.csv 5 1776

...

Na standardowe wyjście dla każdej instancji problemu zwracana była średnia wartość błędu oraz ścieżka.

Platforma sprzętowa:
• procesor: AMD Ryzen 2500U 4 x 2.00GHz
• pamięć operacyjna: 14.9 GB
• system operacyjny: Windows 10

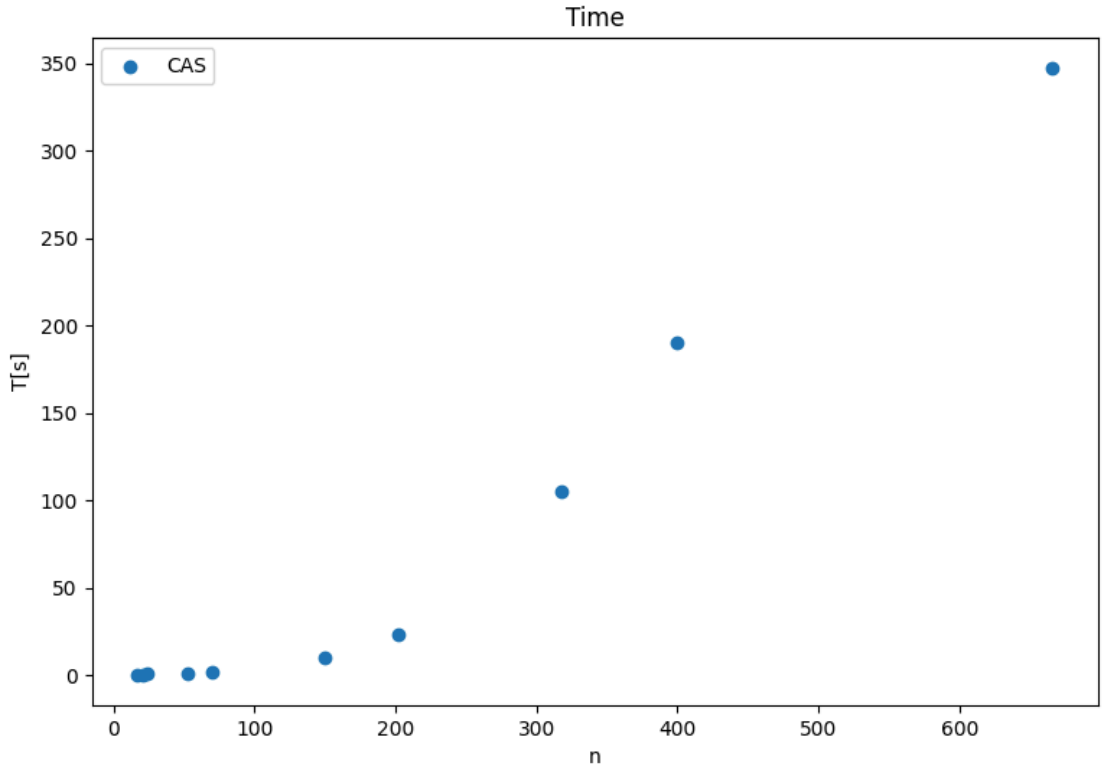
Sposób pomiaru czasu:
Do pomiaru czasu używam biblioteki tiemit i znajdującego się tam timera.

```
timer_start = timer()

sol_temp = aco_calculator.calculate(n[0], matrix)
sol, sol_len = sol_temp[:-1], sol_temp[-1]

timer_end = timer()
```

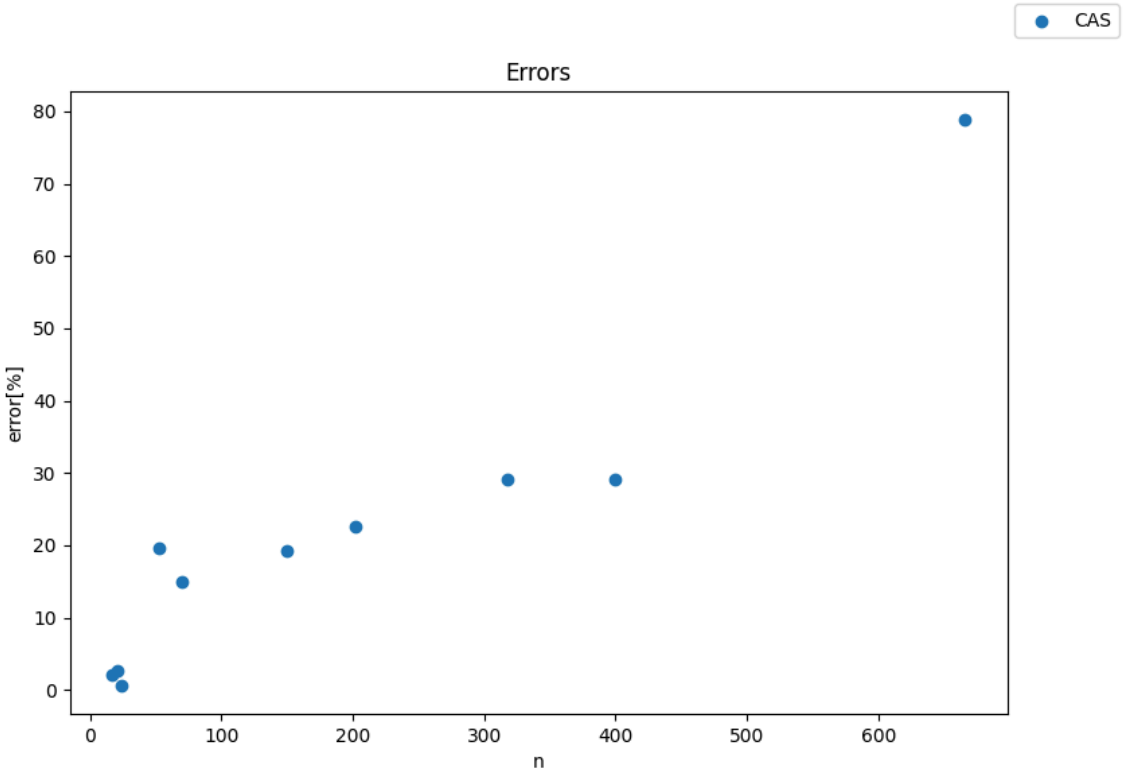
Rysunek 2: pomiar czasu

Wyniki

CAS

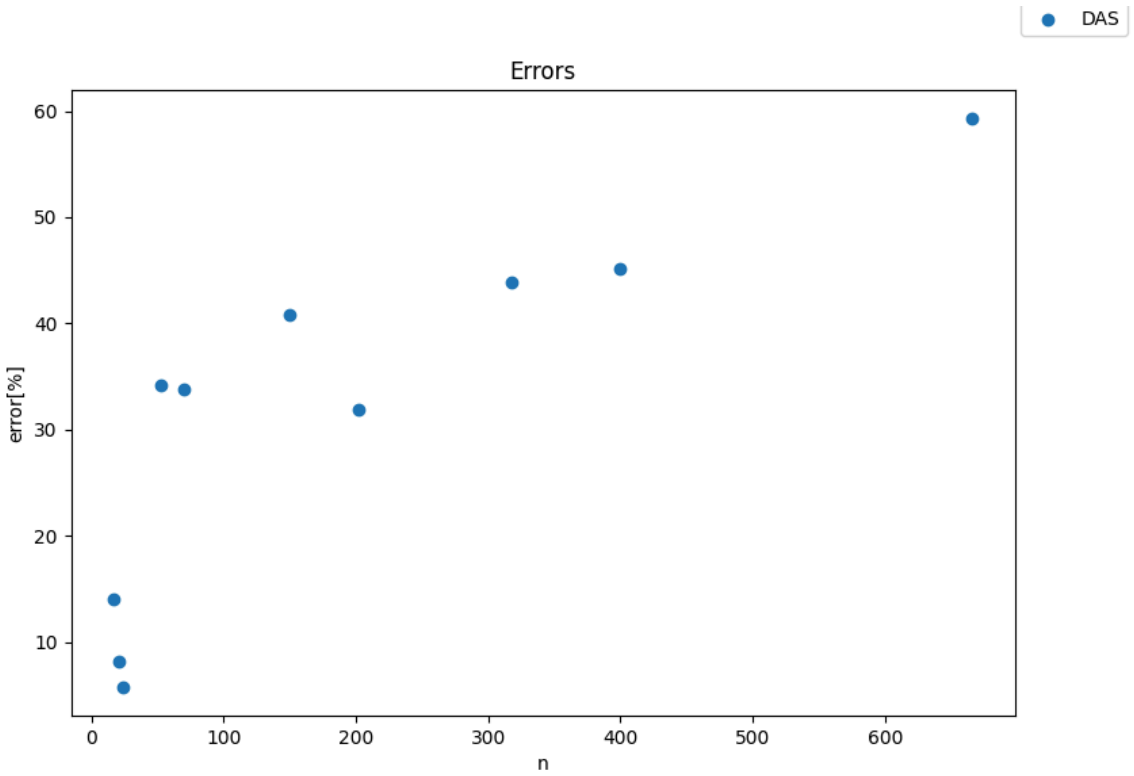


Rysunek 3: Zależność czasu od wielkości instancji n w schemacie CAS

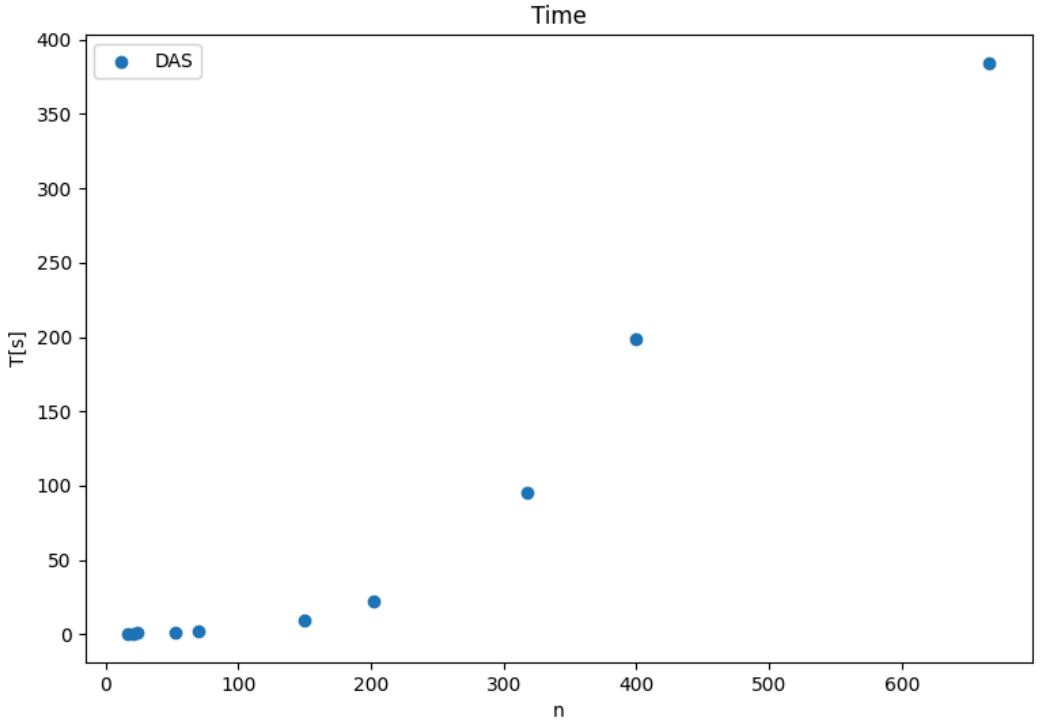


Rysunek 4: Zależność błędu od wielkości instancji n dla schematu CAS

DAS

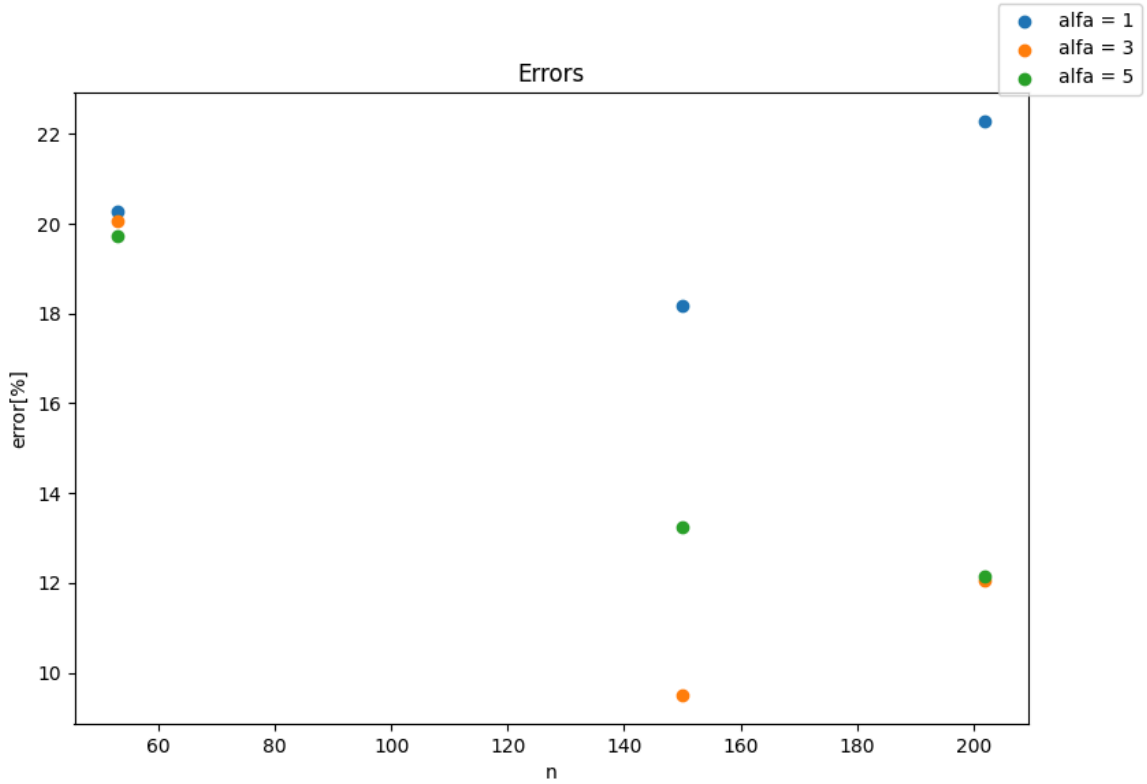


Rysunek 5: Zależność błędu od wielkości instancji n DAS

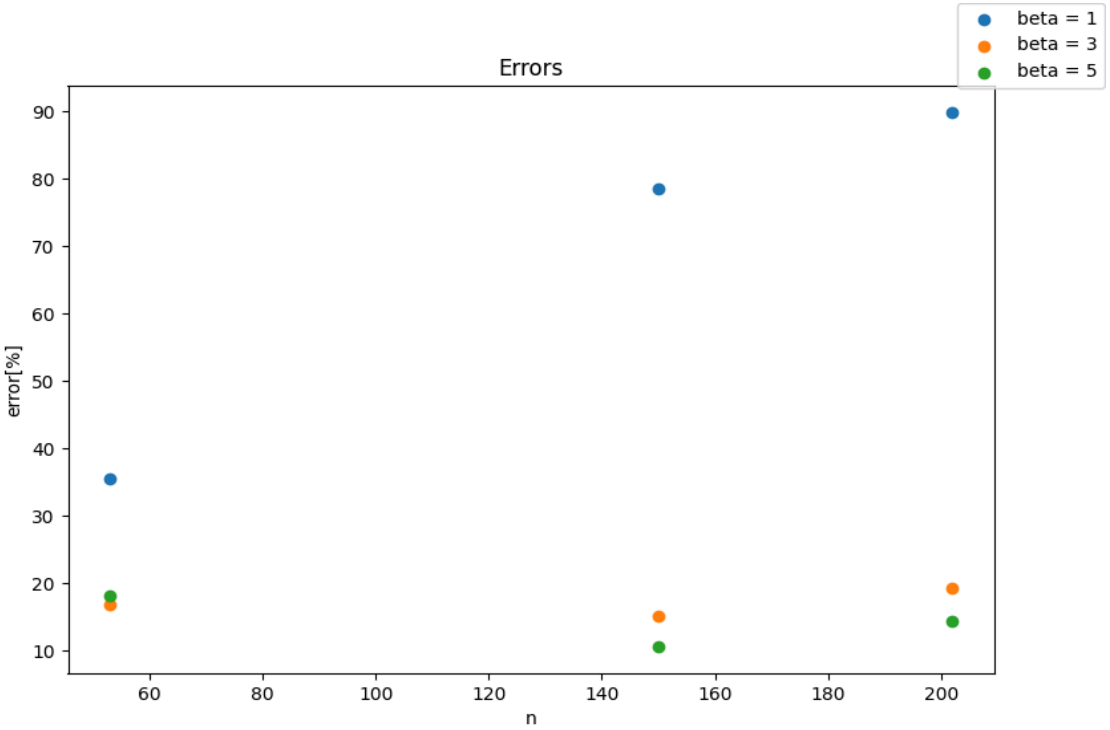


Rysunek 6: Zależność czasu od wielkości instancji n dla schematu DAS

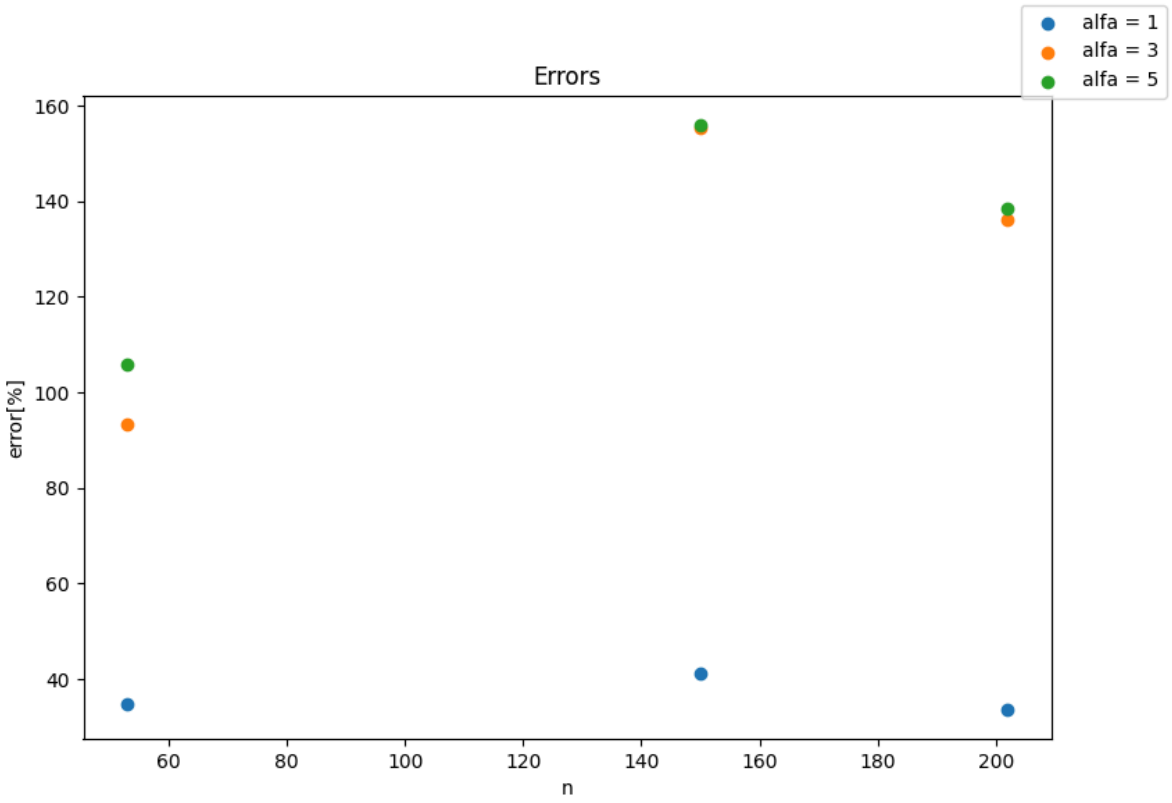
Badanie wpływu parametrów na błąd rozwiązania



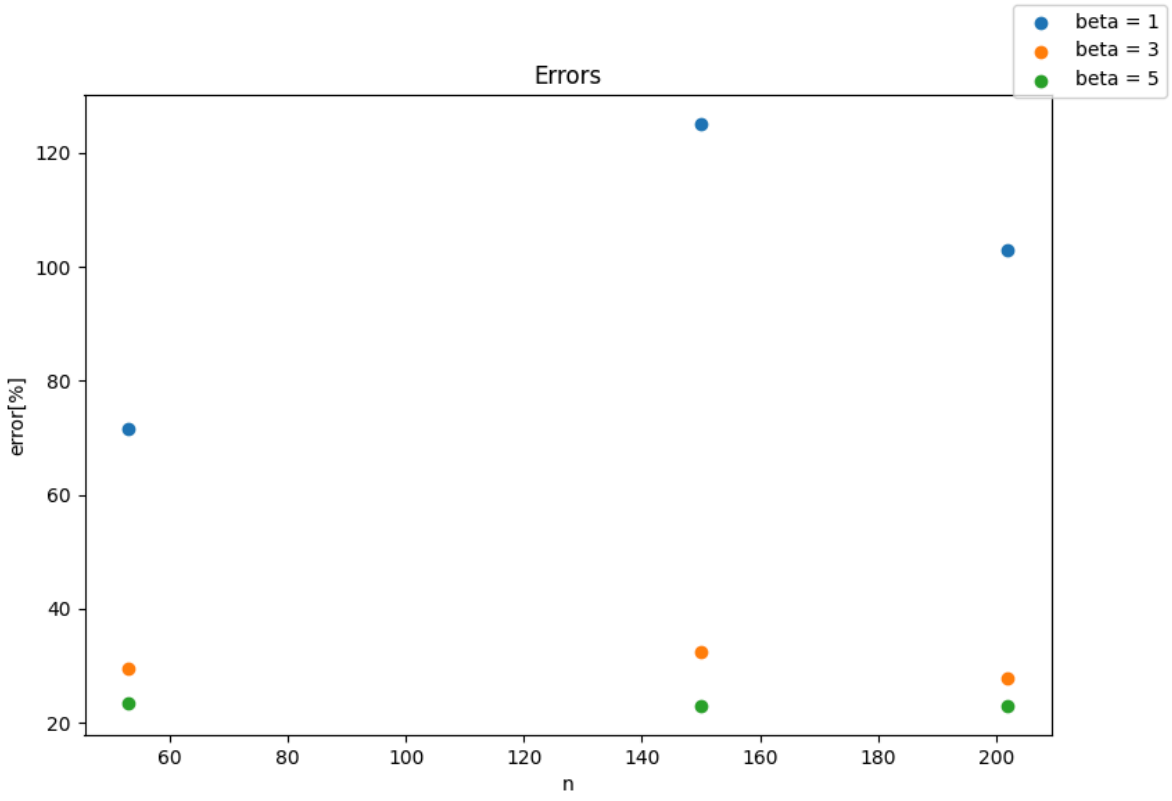
Rysunek 7: Wpływ parametru α na błędy w CAS



rysunek 8: Wpływ parametru β na błędy w CAS

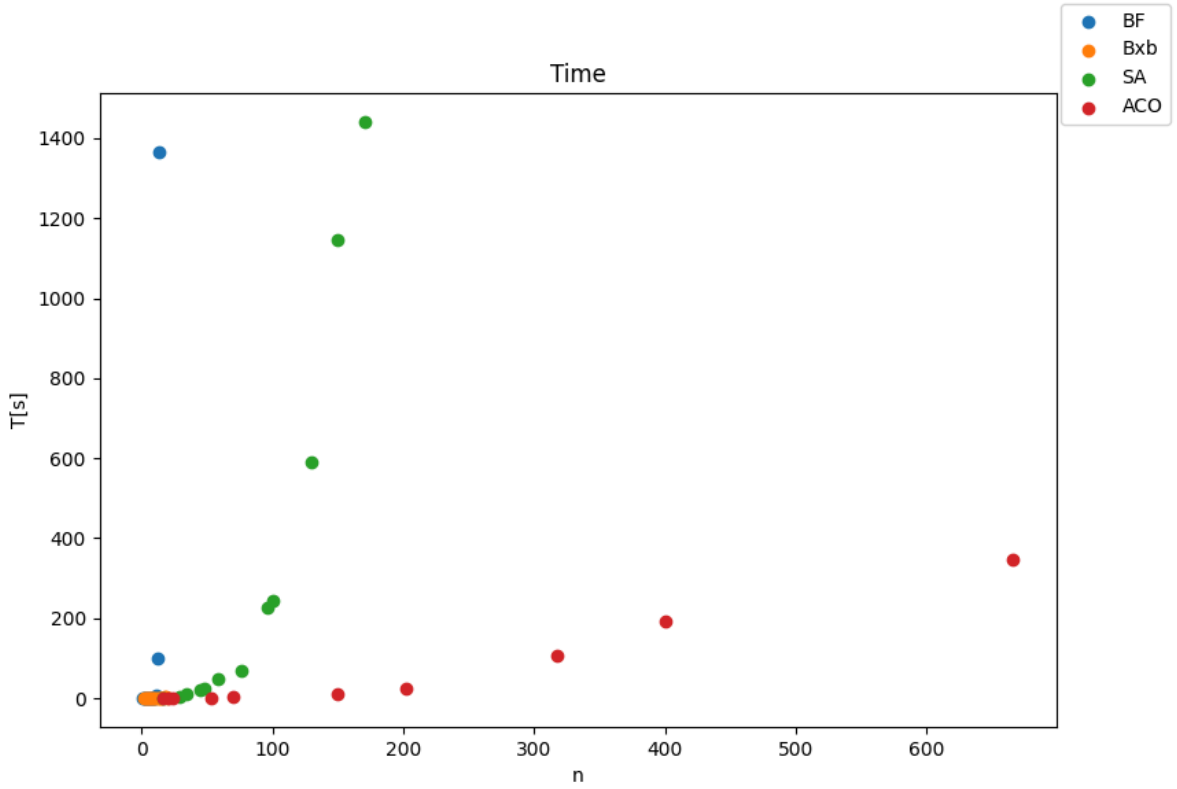


Rysunek 9: Wpływ parametru α na błędy w DAS



rysunek 10: Wpływ parametru β na błędy w DAS

Porównanie z innymi algorytmami



Rysunek 11: Porównanie czasu działania algorytmów

Jak widać na powyższym wykresie algorytm ACO pozwala na obliczenie ścieżki, w rozsądnym czasie, dla o wiele większych instancji problemu niż inne zaimplementowane algorytmy. Jednak należy pamiętać, że nie zwraca on rozwiązania optymalnego, a przybliżone. Z algorytmów zaimplementowanych przeze mnie na projekcie jest najbardziej wydajny (stosunek czas-wielkość błędu).

Wnioski

Udało się zaimplementować algorytm mrówkowy, który w rozsądnym czasie(do 10 min) rozwiązywał instancje o wielkości kilkuset wierzchołków(dla 666 ok. 400s) z założonymi w instrukcji błędami:

Wielkość błędu bezwzględnego, w zależności od wielkości instancji, nie może przekraczać:

- dla n < 24, 0%, (dla problemów tej wielkości algorytm zwraca błędy w ok 2%)
- dla 25 < n < 74, 50%,
- dla 75 < n < 449, 100%
- dla 450 < n < 2500, 150%.

Został zbadany również wpływ parametrów alfa i beta zarówno pod względem czasu jak i błędu.

Jak wynika z rysunków 9 i 10. Najlepsze jakościowo rozwiązania dla schematu gęstościowego dała najniższa z badanychwartość α(1) oraz najwyższa wartość β(5). Z badanych instancji wynika, że dla tego schematu wartość błędu rośnie wraz ze wzrostem α i maleje ze wzrostem β.

Schemat cykliczny zachowuje się podobnie dla parametru β (rysunek 8). Jednak dla parametru α(rysunek 7) ciężko jednoznacznie stwierdzić jak zachowuje się błąd.

Na rysunku 7 widać, że dla parametru α = 3 błąd dla instancji o 150 wierzchołkach jest znacznie mniejszy od pozostałych wartości. Interpretuje to w taki sposób, że zachowanie błędu przy zmianie tego parametru jest związane z budową konkretnej instancji. Czyli możliwym byłoby dobieranie parametrów specjalnie dla konkretnych instancji problemów w celu minimalizowania błędów.