Problem komiwojażera dla symulowanego wyżarzania

Kamil Król

244949

Referat w ramach kursu *algorytmy metaheurystyczne* na WPPT, PWr.

16.06.2020r.

Spis treści

1. Opis problemu komiwojażera

1.1. Opis

Problem komiwojażera jest klasycznym przykładem problemu optymalizacyjnego. Należy on do problemów NP-trudnych, a oznacza to między innymi, że nie jest dotąd znany algorytm rozwiązujący ten problem w czasie wielomianowym. Zatem rozwiązanie tego problemu poprzez sprawdzenie wszystkich możliwości jest bardzo *kosztowne*. Mówiąc kolokwialnie – jest to problem bardzo trudny. Stąd właśnie wynika fakt, że do rozwiązania tego problemu często są używane mniej *kosztowne* heurystyki.
Problem polega na znalezieniu cyklu Hamiltona w pewnym ważonym grafie o jak najmniejszej sumie wag krawędzi. Problem często jest przedstawiany jako podróżowanie między miastami.
Formalnie:

Dane: lista miast (wierzchołków), odległości (wagi krawędzi) między każdymi dwoma miastami (wierzchołkami)
Szukane: najkrótsza ścieżka odwiedzająca każde miasto dokładnie raz powracająca do miasta startowego (cykl Hamiltona o minimalnej sumie wag).
Konieczne jest założenie, że rozważany graf jest spójny – z każdego miasta da się dojechać do dowolnego innego. Jeśli graf jest niespójny to cykl Hamiltona nie istnieje.

1.2. Matematyczne sformułowanie problemu

Cykl Hamiltona (cykl między miastami spełniający warunki opisane wyżej) oznaczany będzie przez c, jest to ciąg wierzchołków (miast) oddający kolejność, w jakiej wierzchołki (miasta) są odwiedzane. Miasta oznaczane są liczbami ze zbioru $\{1, 2, \ldots, n\}$, gdzie n to łączna liczba miast.

$$\begin{gathered} c = (c_1, c_2, c_3, \ldots, c_n) \text{ i jeśli } i \neq j \text{ to } c_i \neq c_j \\ c_i \in \{1, 2, \ldots, n\} \end{gathered} \tag{1}$$

Zatem rozwiązanie problemu komiwojażera wiąże się z problemem minimalizacji funkcji:

$$F(c) = \sum_{i=1}^{n-1} (d_{c_i, c_{i+1}}) + d_{c_1, c_n} \tag{2}$$

gdzie d_{c_i, c_j} jest odległością między miastami i oraz j.

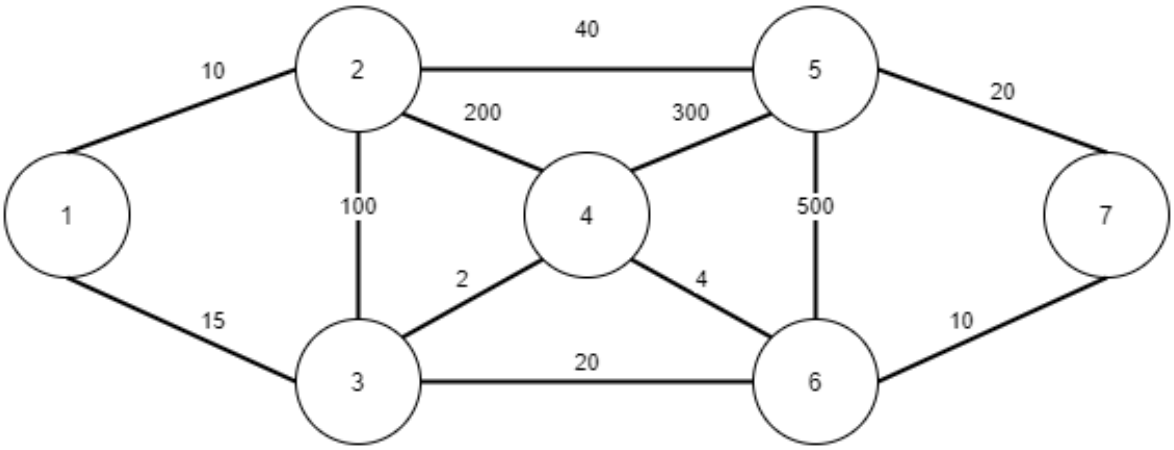
1.3. Przykład

Jest to mały przykład problemu komiwojażera. Na grafie poniżej (Rysunek 1.) można zauważyć kilka przykładowych cykli Hamiltona. Przez $c(x)$ oznaczmy sumę wag na cyklu x. Widać, że cykl τ_1 (Rysunek 2.) jest cyklem o najmniejszej sumie wag w tym grafie. Zatem można stwierdzić, że jest to optymalne rozwiązanie.

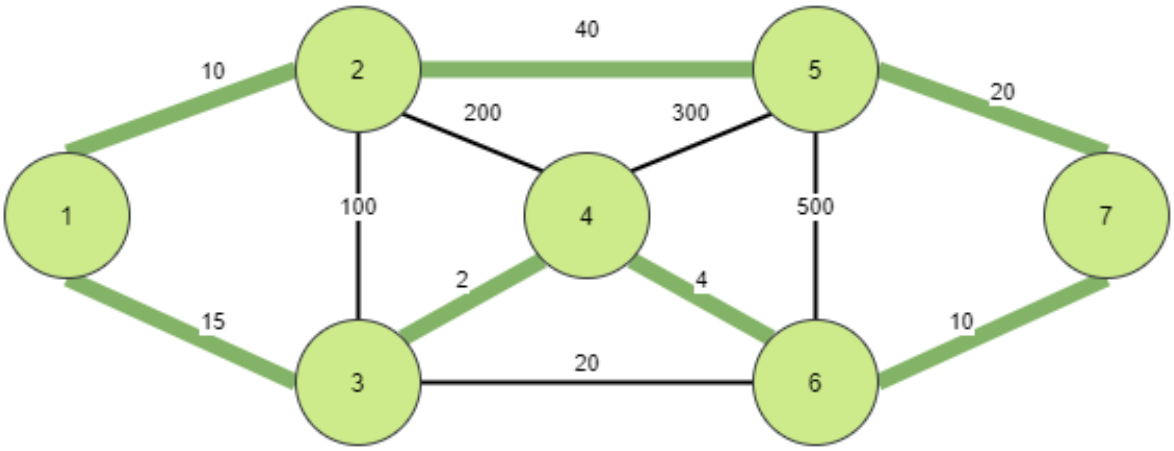
Przykładowe cykle Hamiltona w grafie na rysunku:

$$\tau_1 = 1 \to 2 \to 5 \to 7 \to 6 \to 4 \to 3 \to 1; \; c(\tau_1) = 101$$

$$\tau_2 = 1 \to 2 \to 4 \to 5 \to 7 \to 6 \to 3 \to 1; \; c(\tau_2) = 575$$



Rysunek 1: Przykładowy ważony graf nieskierowany



Rysunek 2: Graf z rysunku 1. z zaznaczonym cyklem τ_1. Na zielono oznaczono krawędzie oraz wierzchołki przez które przechodzi cykl.

1.4. Możliwe modyfikacje

Problem ten może być modyfikowany na wiele różnych sposobów. Np. skierowanie grafu (dopuszczenie dróg jednokierunkowych) nie zmienia istoty problemu ani jego klasy złożoności – problem nadal jest NP-trudny (o ile graf skierowany jest nadal spójny w sensie spójności w grafie skierowanym). Dodanie wymagania, aby graf był grafem pełnym nic nie zmienia, ponieważ TSP w formie przedstawionej wyżej (graf niekoniecznie pełny) można sprowadzić do grafu pełnego poprzez dodanie krawędzi o nieskończonej wadze – nie zmieni to optymalnego rozwiązania.
Co stanie się jeżeli zrezygnujemy z konieczności powrotu do miasta startowego? Problem ten nadal jest NP-trudny. Aczkolwiek nie jest to już problem komiwojażera.
Istnieją też pewne dość znane modyfikacje TSP.

- symetryczny TSP – jest to problem komiwojażera sformułowany dokładnie tak jak wyżej z dodatkowym założeniem, że dla dowolnych 2 miast a, b odległość z miasta a do miasta b jest taka sama jak odległość z miasta b do a.

- asymetryczny TSP – to samo co wyżej, ale dla dwóch dowolnych miast a, b odległość z a do b może być inna niż z b do a. Może być łatwo reprezentowane przez ważony graf skierowany.

- metryczny TSP – jest to taka instancja problemu komiwojażera, że odległości pomiędzy miastami spełniają nierówność trójkąta. Tak sformułowany TSP posiada 2-aproksymację.

- wielokrotny (multiple) TSP – jest to uogólnienie tradycyjnego problemu komiwojażera, w którym dopuszczone jest istnienie więcej niż jednego kupca (salesman). Celem jest zminimalizowanie kosztów podróży każdego kupca (minimalizacja kosztu każdej trasy), tak aby każde miasto zostało odwiedzone

dokładnie raz przez dokładnie jednego kupca. Wszyscy kupcy zaczynają w tym samym mieście i muszą do niego wrócić. Każdy z kupców ma za zadanie odwiedzić $k \leqslant n - m + 1$ miast, gdzie m to liczba kupców.

- rozszerzony (augmented) TSP [3] – Jest to modyfikacja tradycyjnego problemu komiwojażera polegająca na dodaniu nagród/zysków każdemu z n miast, które kupiec może odwiedzić. Kupiec musi odwiedzić $k \leqslant n$ miast i powrócić do miasta startowego. Jeśli kupiec przechodzi z miasta do miasta to ponosi koszty przebycia tej trasy, ale też otrzymuje pewien zysk związany z odwiedzeniem miasta, które odwiedził. Celem jest zmaksymalizowanie zysku z odwiedzonych miast.

2. Symulowane wyżarzanie

2.1. Przypomnienie

Symulowane wyżarzanie jest metaheurystyką aproksymującą globalne minimum/maksimum pewnej danej funkcji. Jej idea oraz nazwa pochodzi od metody wyżarzania w metalurgii, w której to rozgrzewa się dany metal oraz stopniowo się go ochładza kontrolując stale temperaturę. Zbyt szybkie jego ochłodzenie nie da pożądanego efektu.
Dane: funkcja f do zoptymalizowania, rozwiązanie początkowe x_0, definicja sąsiedztwa $\mathcal{N}$, temperatura początkowa T_0, sposób ochładzania δ (np. liniowy), współczynni ochładzania β (jeśli potrzebny), funkcja P obliczająca prawdopodobieństwo zaakceptowania (z ang. acceptance probability) danego rozwiązania x w zależności od aktualnej temperatury T, energii aktualnego rozwiązania oraz energii danego x. Przez energię jakiegoś rozwiązania rozumiemy tutaj wartość funkcji w pewnym punkcie.
Metoda:

1. Za aktualne rozwiązanie x_c podstaw x_0 – rozwiązanie początkowe: $x_c \leftarrow x_0$,
2. Za aktualną temperaturę T podstaw T_0: $T \leftarrow T_0$,
3. Dopóki $T > 0$ wykonuj kroki:
   i) Weź nowego sąsiada aktualnego rozwiązania.
   Tzn. wybierz losowy element ze zbioru sąsiadów x_c: $x_n \leftarrow randomFrom(\mathcal{N}(x_c))$
   Wybrany x_n jest teraz kandydatem na nowe aktualne rozwiązanie,
   ii) Oblicz $p \leftarrow P(x_n, x_c, T)$ – prawdopodobieństwo zaakceptowania kandydata x_n,
   iii) Jeśli $p \geqslant uniform(0, 1)$ to podstaw: $x_c \leftarrow x_n$. Jeśli nie, to nie rób nic.
   $uniform(0, 1)$ oznacza wylosowanie liczby z przedziału $[0, 1]$ zgodnie z rozkładem jednostajnym ciągłym,
   iv) Zaktualizuj aktualną temperaturę: $T \leftarrow \delta(T)$.
4. Zwróć aktualne rozwiązanie x_c.

2.2. Przykłady

Z racji tego, że w powyższym opisie symulowanego wyżarzania pojawia się dużo funkcji/operatorów, przedstawione tu zostaną proste ich przykłady.

generowanie rozwiązania początkowego x_0 – losowe rozwiązanie dopuszczalne. Dla TSP może być to losowy cykl Hamiltona w grafie.

definicja sąsiedztwa $\mathcal{N}(x)$ – Dla pewnej funkcji $h : \mathbb{R} \to \mathbb{R}$ za sąsiedztwo x-a można przyjąć punkty oddalone od niego o pewnien ϵ. Dla TSP musi być to zbiór rozwiązań dopuszczalnych innych od x. Zatem mogą być to cykle Hamiltona w tym samym grafie będące np. permutacją x-a lub x-em z dwoma miastami zamienionymi kolejnością (1 inwersja). Do znanych operatorów sąsiedztwa dla TSP należą 2-opt i 3-opt.

sposób ochładzania δ – Może być to przykładowo jedna z funkcji (dla wszystkich funkcji β oznacza współczynnik ochładzania, i jest numerem aktualnie wykonywanej iteracji):

$$\begin{gathered} \delta_1(T, i) = T - i\beta, \text{(liniowe)} \\ \delta_2(T) = (1 - \beta) \cdot T \\ \delta_3(i) = \frac{c}{\log(1 + i)}, \text{ gdzie c jest pewną odpowiednio dobraną do problemu stałą.} \end{gathered} \tag{3}$$

acceptance probability P – Może to być dowolna funkcja prawdopodobieństwa wpisująca się w ideę SA. Tzn. wraz ze spadkiem temperatury spada prawdopodobieństwo zaakceptowania gorszych rozwiązań. Przykładowa funkcja P to np. $P(x) = exp(\frac{f(x_c)-f(x)}{T})$ (T jest aktualną temperaturą).

3. Symulowane wyżarzanie z zachłannym przeszukiwaniem

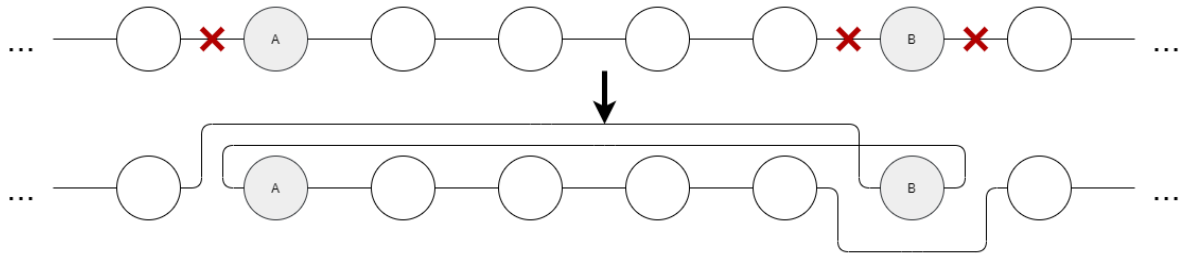
3.1. Wprowadzenie

Jest to pierwsze proponowane rozwiązanie TSP. Algorytm symulowanego wyżarzania z zachłannym szukaniem (z ang. adaptive simulated annealing algorithm with greedy search (ASA-GS)) polega na modyfikacji standardowego algorytmu symulowanego wyżarzania poprzez dodanie poszukiwania zachłannego na pewnym jego etapie. [1]. Problem sformułowany jest dokładnie tak jak w punkcie 1.1 i 1.2. Rozważany tu jest graf nieskierowany, oznacza to, że TSP jest symetryczny, czyli $d_{i,j} = d_{j,i}$, gdzie $d_{i,j}$ jest odległością z miasta i do miasta j. Dane takie jak definicja sąsiedztwa, acceptance probability (P) czy parametry związane z temperaturą lub jej zmianą zostaną podane niżej w odpowiednich miejscach.

3.2. Definicja sąsiedztwa

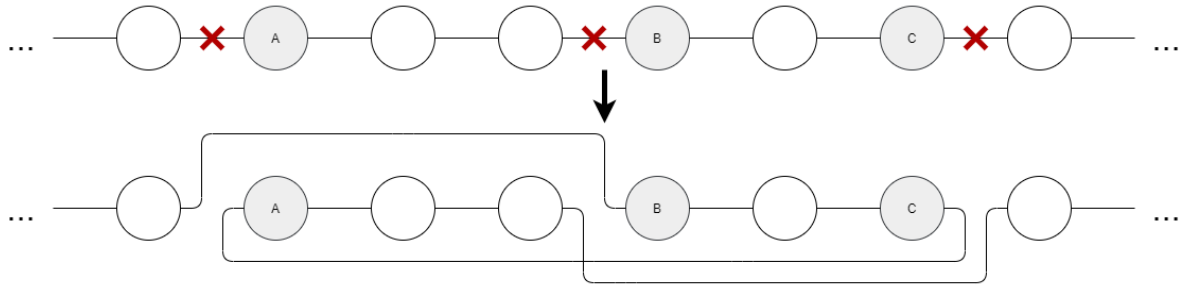
W tej metodzie sądziedztwo jest zdefiniowane w nastepujący sposób: $\hat{c}$ jest sąsiadem c, jeśli $\hat{c}$ jest mutacją c. Mutacja to zdefiniowana w pewnien sposób operacja na ciągu c (np. zamiana kolejności kilku miast). W tej metodzie mutacje będą 3 [1].

VI: Wstawienie wierzchołka (vertex insert mutation) (Rysunek poniżej):



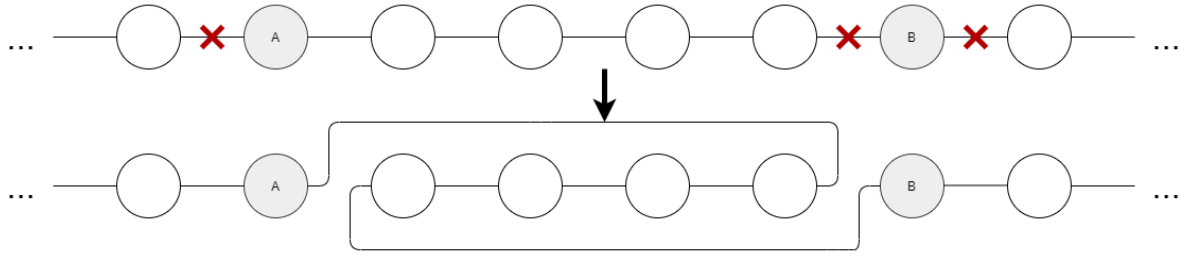
Rysunek 3: Mutacja - wstawienie wierzchołka

BI: Wstawienie bloku (block insert mutation) (Rysunek poniżej):



Rysunek 4: Mutacja - wstawienie bloku

BR: Odwrócenie bloku (block reverse mutation) (Rysunek ponizej):



Rysunek 5: Mutacja - odwrócenie bloku

Tutaj [1] (Table 1.) można znaleźć tabelę porównującą zużycie procesora przez każdą z tych metod oraz ich kombinacje. Najszybszą metodą jest BR, daje ona też najlepsze rozwiązanie. Najwolniejszą metodą, która też jest najgorsza pod względem jakości rozwiązania jest BI. Najlepsze rezulataty jeśli chodzi o jakość rozwiązania jak i szybkość uzyskano łącząc wszystkie 3 mutacje. Połączenie ich polega na braniu konkretnej mutacji z pewnym prawdopodobieństwem. Proponowane prawdopodobieństwa wynoszą: 10% dla VI, 1% dla BI, 89% dla BR.
Zapisując formalniej definicję sąsiedztwa otrzymujemy:

$$\mathcal{N}(x) = \{x' : x' \text{ jest mutacją } x, \text{według dowolnej spośród mutacji: VI, BI, BR}\} \tag{4}$$

3.3. Opis algorytmu

Standardowy algorytm symulowanego wyżarzania potrafi znaleźć lepsze rozwiązanie (o ile istnieje) kosztem czasu. Ideą proponowanej metody jest przyspieszenie jego zbieżności poprzez dodanie wyszukiwania zachłannego. Koncept zachłannego wyszukiwania polega tu na wprowadzeniu dodatkowych współczynników algorymtu: t_{greedy} – mówiący ile maksymalnie prób polepszenia rozwiązania (sprawdzania sąsiadów) powinno być wykonanych przed zmianą temperatury i zachłannym wyborem najlepszego z tych t_{greedy} sąsiadów, i podjęciu decyzji o zaakceptowaniu gorszego rozwiązania. Jaśniej jest to opisane poniżej.

Stare rozwiązanie c_{old} jest zastępowane przez swojego sąsiada $c_{neighbour1}$ jeśli $F(c_{neighbour1}) < F(c_{old})$, gdzie funkcja F jest optymalizowaną funkcją (patrz punkt 1.2 – matematyczne sformułowanie TSP). Po tej operacji wykonywany jest następny krok algorytmu. W przeciwnym wypadku, czyli jeśli warunek $F(c_{neighbour1}) < F(c_{old})$ okazał się nieprawdziwy, to generowany jest kolejny sąsiad $c_{neighbour2}$ oraz sprawdzany jest analogiczny warunek do tego co wyżej tj. $F(c_{neighbour2}) < F(c_{old})$. Jeśli jest on prawdziwy, to c_{old} jest zastępowane przez swojego sąsiada $c_{neighbour2}$ i następuje przejście do kolejnego kroku algorytmu, jeśli nie to generowany jest kolejny sąsiąd i procedura się powtarza. W momencie kiedy zostanie wygenerowany t_{greedy}-ty sąsiad (wygenerowano w sumie t_{greedy} sąsiadów i żaden z nich nie okazał się lepszy od c_{old}), stare rozwiązanie c_{old} jest zastępowane przez najlepsze rozwiązanie t_{best} spośród t_{greedy} wygenerowanych przed chwilą sąsiadów z prawdopodobieństwem P.

$$c_{best} : F(c_{best}) = min\{F(c_{neighbour1}), F(c_{neighbour2}), \ldots, F(c_{neighbour\_t_{greedy}})\} \tag{5}$$

Proponowany wzór na prawdopodobiestwo P poniżej.

$$P(c_{best}) = \exp\left(-\frac{F(c_{best}) - F(c_{old})}{t_{current}} \cdot \frac{10n}{Opt}\right) \tag{6}$$

Parametr $t_{current}$ to aktualna temperatura, n to liczba miast, Opt to optymalna długość cyklu.
Metoda zmiany temperatury jest dana wzorem poniżej.

$$\delta(t_{current}) = t_{current} \cdot \beta \tag{7}$$

Gdzie β to współczynnik ochładzania. **Metoda:**

1. Za aktualne rozwiązanie x_c podstaw x_0 – losowo wygenerowane rozwiązanie początkowe: $x_c \leftarrow x_0$,
2. Za aktualną temperaturę T podstaw T_0: $T \leftarrow T_0$,
3. Za zmienną G podstaw 0: $G \leftarrow 0$, będzie to zmienna służąca do liczenia ile już sąsiadów zostało sprawdzonych.
4. Dopóki $T > 0$ wykonuj kroki:
   i) Wybierz mutację $\mathcal{M}$ spośród: VI, BI, BR zgodnie z prawdopodobieństwami ich użycia.
   ii) Weź nowego sąsiada aktualnego rozwiązania. $x_{n_1} \leftarrow \mathcal{M}(x_c)$
   Wybrany x_{n_1} jest teraz kandydatem na nowe aktualne rozwiązanie,
   iii) Sprawdź czy $\triangle F = F(x_{n_1}) - F(x_c) \leqslant 0$. Jeśli tak to zaakceptuj kandydata x_{n_1}: $x_c \leftarrow x_{n_1}$, następnie przejdź do punktu **vii)**. Jeśli warunek nie był spełniony, to: $G \leftarrow G + 1$.
   iv) Jeśli że $G < t_{greedy}$, to przejdź do **i)** (nie zmieniając temperatury). Jeśli ten warunek jest fałszywy, to wybierz c_{best} zgodnie z formułą **5**, $x' \leftarrow c_{best}$.
   v) Oblicz $p \leftarrow P(x', x_c, T)$ – prawdopodobieństwo zaakceptowania kandydata x',
   P jest dane formułą 6,
   vi) Jeśli $p \geqslant uniform(0, 1)$ to podstaw: $x_c \leftarrow x'$. Jeśli nie, to nie rób nic.
   vii) Zaktualizuj aktualną temperaturę: $T \leftarrow \delta(T)$ oraz $G \leftarrow 0$. δ jest zadana formuła 7.
5. Zwróć aktualne rozwiązanie x_c.

3.4. Dobór parametrów

Parametry takie jak: t_{greedy}, t_{start} (temperatura początkowa), β, P będą zależały od *skali problemu*. Szybkie ochładzanie będzie działać dobrze, dla małej ilości miast. Dla instancji TSP, gdzie długość optymalnej ścieżki będzie długa odpowiednie będą wysokie temperatury początkowe. Proponowane współczynniki (8) są dla problemu komiwojażera gdzie liczba miast waha się od 51 do 85900.

$$
\begin{gathered}
P(c_{best}) = \exp(-\frac{F(c_{best}) - F(c_{old})}{t_{current}} \cdot \frac{10n}{Opt}) \\
\beta = \frac{(\alpha \cdot n^{0.5} - 1.0)}{\alpha \cdot n^{0.5}} \\
t_{greedy} = \beta \cdot n
\end{gathered}
\tag{8}
$$

4. Hybrydowe symulowane wyżarzanie z adaptywnym ochładzaniem

4.1. Wprowadzenie

Jest to drugie proponowane rozwiązanie dla problemu komiwojażera. W dalszej części nazywane będzie TS-SA. Cała ta sekcja bazuje głównie na pracy naukowej pt. „Hybrid Simulated Annealing Algorithm Based on Adaptive Cooling Schedule for TSP"[2]. Metoda jest nazwana hybrydową, ponieważ jest w pewnym stopniu połączeniem z Tabu Searchem. Lista tabu jest mechanizmem, który pozwala na ucieczkę z lokalnego minimum, ale ma tę wadę, że nie jest w stanie zapobiec zapętlaniu się. Zapętlanie jest powracaniem do pewnego rozważonego już rozwiązania i w efekcie pozostawaniem w lokalnym minimum. Algorytm symulowanego wyżarzania nie posiada pamięci, w której mógłby pamiętać ostatnio odwiedzone rozwiązania. Ideą proponowanego rozwiązania jest dodanie do SA listy tabu (pewnej pamięci), która zapobiegnie powracaniu do niedawno odwiedzonych rozwiązań. Okres przez jaki dane rozwiązanie będzie *zakazane*, czyli to jak długo będzie obecne w liście tabu może zależeć od długości tej listy. Proponowany okres trwania karencji to: $\lfloor n \cdot 2.375 \rfloor$.
Rozważany w tej metodzie problem komiwojażera to standardowy symetryczny problem komiwojażera sformułowany w punkcie 1.2.

4.2. Prawdopodobieństwo akceptacji

Funkcja prawdopodobieństwa zaakceptowania rozwiązania P jest dana wzorem 9.

$$P(x_{neighbour}) = \exp\left(-\frac{F(x_{neighbour}) - F(x_{current})}{T_i}\right) \tag{9}$$

Gdzie $x_{neighbour}$ jest rozwiązaniem, dla którego liczone jest prawdopodobieństwo akceptacji, F jest optymalizowaną funkcją (patrz sformułowanie TSP w punkcie 1.2), $x_{current}$ jest aktualnym rozwiązaniem, i jest numerem aktualnej iteracji, a T_i jest temperaturą w aktualnej iteracji i. Sposób liczenia temperatury dla danej iteracji zostanie przedstawiony niżej.

4.3. Metoda ochładzania (cooling schedule)

4.3.1. Wprowadzenie

Jedną z głównych modyfikacji proponowanych w tym rozwiązaniu jest dodanie adaptywnego sposobu ochładzania. Metoda ochładzania ma główny wpływ na aktualną temperaturę, a ta z kolei jest kluczowym parametrem do obliczenia prawdopodobieństwa akceptacji $P(x)$ dla danego rozwiązania x. Zatem temperatura jest jednym z czynników decydujących czy dane rozwiązanie zostanie zaakceptowane. Rozwiązanie, które okazuje się lepsze niż poprzednie jest w prezentowanej metodzie zawsze akceptowane (prawdopodobieństwo 100%), o ile nie jest w liście tabu.
Przypominjmy jaką rolę jaką pełni temperatura w SA. Poszukiwanie rozpoczyna z wysoką temperaturą początkową, co zwiększa szanse na zaakceptowanie gorszego rozwiązania. Jest to działania pożądane, ponieważ na początku ważna jest ekspansja. Wraz ze spadniem temperatury ekspansja maleje, a parcie na poprawienie aktualnego rozwiązania rośnie.
Zatem podczas pracy algorytmu temperatura stale maleje. Taki mechanizm pozwala na ucieczkę z lokalnego minimum jeśli znajduje się ono stosunkowo niedaleko od punktu startowego. Pozostaje pytanie: co stanie się jeśli algorytm napotka lokalne minimum przy względnie niskiej temperaturze? Wtedy szanse na wyjście z tego minimum są już mniejsze, ponieważ temperatura jest mniejsza. **Celem adaptywnego ochładzania** jest właśnie rozwiązanie tego problemu.

4.3.2. Sformułowanie metody

Zaproponowana tutaj metoda ochłoadzania (CS) będzie miała za zadanie dynamicznie dostosowywać aktualną temperaturę na podstawie ostatnich przeszukiwań przestrzeni rozwiązań. Możliwe będzie również **ocieplanie** (reheating).
Temperatura jest kontrolowana przez pewną funkcję δ, która utrzymuje temperaturę powyżej pewnego minimum T_{min}. Proces ogrzewania zaczyna się jeśli zaakceptowane zostanie jakieś gorsze rozwiązanie (*ruch w przeciwną stronę niż minimum*) i jest stopniowo kontynuowany. Ochładzanie jest przywracane natychmiast po znalezieniu lepszego rozwiązania niż aktualne (*ruch w stronę minimum*). Funkcja obliczająca temperaturę jest zadana wzorem 10.

$$\delta(i) = T_{min} + \lambda \ln(1 + r_i) \tag{10}$$

Gdzie i jest numerem iteracji, $T_{min} > 0$ jest minimalną wartością jaką może przyjąć temperatura, λ jest współczynnikiem kontrolującym wzrost temperatury, r_i jest liczbą nastepujących bezpośrednio po sobie akceptacji gorszych rozwiązań w iteracji i. Wartość początkowa r_i to 0.

Zauważmy, że $\delta(0) = T_{min}$. Zatem T_{min} jest też temperaturą początkową: $T_0 = \delta(0) = T_{min}$ Minimalna temperatura T_{min} musi byc też różna od zera, ponieważ w sytuacji kiedy r_i jest zerem to $\delta(i) = T_{min}$, a to nie może być zerem, ponieważ występuje w mianowniku funkcji obliczającej prawdopodobieństwo akceptacji P (Wzór 9).
Parametr λ jest odpowiedzialny za kontrolowanie wzrostu temperatury. Kiedy wartość λ jest duża, algorytm spędza mniej czasu na poszukiwaniu lepszego rozwiązania w swoim sąsiedztwie (większa ekspansja). Kiedy λ jest mała, algorytm więcej czasu poświęca na szukanie innych rozwiązań w swoim sąsiedztwie (mniejsza ekspansja). Wartość parametru λ zależy od rozmiaru i stopnia skomplikowania problemu i na tej podstawie powinien zostać dobrany.
Parametr r_i pozostaje niezmieniony jeśli nowe rozwiązanie ma taki sam koszt jak aktualne. Jeśli następuje polepszenie aktualnego rozwiązania to r_i jest równe 0.

4.4. Definicja sąsiedztwa

W tej metodzie sądziedztwo jest zdefiniowane w nastepujący sposób: $\hat{c}$ jest sąsiadem c, jeśli $\hat{c}$ jest mutacją c. Mutacja to zdefiniowana w pewnien sposób operacja na ciągu c. W tej metodzie będą dwie mutacje: $\mathcal{M}_1$ oraz $\mathcal{M}_6$ opisane poniżej. Definicja sąsiedztwa formalniej:

$$\hat{x} \in \mathcal{N}(x) \equiv \hat{x} = \mathcal{M}_1(x) \vee \hat{x} = \mathcal{M}_6(x) \tag{11}$$

Operator mutacji $\mathcal{M}_6$ polega na odwróceniu i przeniesieniu pewnej permutacji podciągu danego rozwiązania. Przykład w [4]. Operator ten daje duże szanse na przekierowanie procesu szukania lepszego rozwiązania w inny obszar przestrzeni rozwiązań. $\mathcal{M}_6$ przeszukuje więc dość szeroki obszar przez co jest nieefektywny przy szukaniu rozwiązań w pewnym konkretnym sąsiedztwie np. obszarze bliskim optimum. Zatem jako alternatywa dla $\mathcal{M}_6$ proponowany jest operator $\mathcal{M}_1$, który ma mniejszy wpływ na rozwiązanie (jest *słabszy*). Sprawdzi się on dobrze przy przeszukiwaniu konkretnego sąsiedztwa.
Operator mutacji $\mathcal{M}_1$ polega na zamienieniu miejscami dwóch nastepujących po sobie miast. Jest to nieznaczna zmiana, stąd operator ten sprawdza się dobrze w przeszukiwaniu konkretnego sąsiedztwa. Przykład mutacji według tego operatora poniżej.

$$c = 1 \to 2 \to 5 \to 7 \to 6 \to 4 \to 3 \to 1$$

$$\hat{c} = 1 \to 2 \to 5 \to 7 \to \mathbf{4} \to \mathbf{6} \to 3 \to 1$$

Operator $\mathcal{M}_6$ będzie używany na początku procesu szukania, a operator $\mathcal{M}_1$ wtedy kiedy algorytm skieruje się ku obszarowi bliskiemu optimum. Operator $\mathcal{M}_1$ zintensyfikuje wtedy lokalne przeszukiwanie.

4.5. Opis algorytmu (TS-SA)

Podczas działania algorytmu liczony będzie czas jego wykonywania i będzie on dostępny pod zmienną *time*. W zmiennej *maxTime* znajduje się oczekiwany, maksymalny czas wykonywania algorytmu. W zmiennej i przechowywany jest aktualny numer iteracji pętli opisanej w kroku 3.

1. Za aktualne rozwiązanie x_c podstaw x_0 – losowo wygenerowane rozwiązanie początkowe: $x_c \leftarrow x_0$,
2. Za początkową temperaturę T_0 podstaw 1.0 (proponowana wartość): $T_0 = 1.0$.
   Zainicjalizuj listę tabu TL jako pustą listę: $TL = [\,]$.
   Ustal współczynnik k. Proponowana wartość to 0.75: $k = 0.75$,
3. Dopóki $time < maxTime$ (czas działania algorytmu nie przekracza zadanego czasu pracy) wykonuj kroki:
   i) Jeśli $time < k \cdot maxTime$, to sąsiad x_{new} jest liczony według: $x_{new} = \mathcal{M}_6(x_c)$, w przeciwnym wypadku według: $x_{new} = \mathcal{M}_1(x_c)$,
   ii) Oblicz $\triangle F = F(x_{new}) - F(x_c)$
   iii) Zaktualizuj r_i (na podstawie $\triangle F$) oraz dodaj x_c do TL.
   iv) Oblicz $T_i = \delta(i)$.
   v) Sprawdź czy $\triangle F < 0$. Jeśli tak to przejdź do **ix)**. Jeśli warunek nie był spełniony, to przejdź do punktu **x)**.
   vi) Oblicz $p \leftarrow P(x_{new})$ (prawdopodobieństwo zaakceptowania sąsiada x_{new}), P jest dane formułą 9,
   vii) Jeśli $p \geqslant uniform(0, 1)$ to przejdź do **viii)**. Jeśli nie, to przejdź do **x)**.
   viii) Jeśli $x_{new} \in TL$, to idź do kroku **3.** Jeśli nie, to do **ix)**.
   ix) Zaakceptuj nowe rozwiązanie: $x_c \leftarrow x_{new}$.
   x) Sprawdź czy $time < maxTime$. Jesli tak to wykonaj ponownie krok 3. Jeśli nie, to przejdź do kroku 4.
4. Zwróć aktualne rozwiązanie x_c.

5. Porównanie i wnioski

Zaproponowane zostały dwie modyfikacje standardowego algorytmu symulowanego wyżarzania. Pierwszy to adaptywny algorytm symulowanego wyżarzania z zachłannych przeszukiwaniem (ASA-GS) (punkt 3). Drugi to hybrydowe symulowane wyżarzanie z adaptywnym ochładzaniem (TS-SA) (punkt 4).

5.1. Porównanie ASA-GS z TS-SA

Każda z zaproponowanych modyfikacji ma swoje *słabe punkty* jak i zalety. Zarówno pierwsza, jak i druga zaproponowana metoda nie gwarantują tego, że pod koniec poszukiwania proces skieruje się w obszar, w którym znajduje się globalne optimum, a zatem mogą dać rozwiązanie stosunkowo odległe od globalnego optimum. Jest to ogólna cecha algorytmów metaheurystycznych. Zauważmy też, że obie zaproponowane metody są stosunkowo łatwe w implementacji.
Pierwsza metoda (ASA-GS) posiada stały sposób generowania sąsiadów (losowanie spośród 3 mutacji w pewnym prawdopodobieństwem) tzn. nie zmienia się on w czasie działania algorytmu. Tym co przyspiesza zbieżność tego algorytmu do globalnego optimum jest dodanie zachłannego przeszukiwania (greedy search). Mechanizm zachłannego przeszukiwania powoduje, że algorytm *upewnia się*, że aktualne lokalne sąsiedztwo zostało stosunkowo dobrze przeszukane (reguluje to parametr t_{greedy}), przed przejściem do gorszego rozwiązania (zatem do potencjalnie innego sąsiedztwa/obszaru). Modyfikacja te znacznie poprawiła zbieżność standardowego SA. Spadek temperatury jest realizowany w sposób standardowy dla SA.
Nietypową cechą tego algorytmu jest obecność parametru *Opt* we wzorze na prawdopodobieństwo (6). Jest to wartość optymalnego rozwiązania. Czyni to ten algorytm mniej praktycznym, ponieważ nie zawsze wartość optymalnej ścieżki jest znana. W sytuacjach, w których przeprowadzane są testy na takich instancjach TSP, gdzie znana jest wartość minimalnej ścieżki, ASA-GS sprawdza się lepiej niż standardowy SA. W przeciwnym wypadku, tzn. gdy nie jest znana wartość optimum, konieczne jest szacowanie tej wartości co może być trudne i czasochłonne. Zatem algorytm ten sprawdzi się głównie w zastosowaniach teoretycznych.
Druga metoda (TS-SA) różni się od poprzedniej między innymi tym, że sposób generowania sąsiadów zależy od czasu pracy algorytmu (oraz parametru k). Konkretniej – metoda generowania sąsiadów zmienia się po przekroczeniu pewnego czasu pracy, co poprawia zarówno ekspansję w początkowej fazie algorytmu jak i eksplorację konkretnego sąsiedztwa w końcowej części algorytmu. Ideą samego SA jest właśnie wyznaczenie kompromisu pomiędzy eksploracją, a ekspansją – stąd właśnie w typowym SA prawdopodobieństwo P zależy od stopniowo zmniejszanej temperatury. W tej metodzie zmiana sposobu generowania sąsiadów w czasie pracy jest dodatkowym mechanizmem regulującym ten *trade-off* pomiędzy ekspansją, a eksploracją. Zatem ta modyfikacja daje lepsze możliwości dostosowania algorytmu do danej instancji problemu komiwojażera.
Kolejną modyfikacją, która polepsza działanie TS-SA jest dodanie listy tabu. Jest to mechanizm pomagający w ucieczce z lokalnego minimum w niektórych przypadkach. ASA-GS nie posiada takiego mechanizmu co czyni go potencjalnie bardziej podatnym na utknięcia w lokalnych minimach.
Obecność adaptywnego ochładzania w TS-SA jest dość niestandardowa modyfikacją, ponieważ dopuszcza ogrzewanie. Ten mechanizm omija problem standardowego SA opisany w punkcie 4.3.1. Wcześniej wspomniany ASA-GS nie posiada żadnego mechanizmu rozwiązującego ten problem.

5.2. Dobór parametrów

ASA-GS

W tej metodzie dodanie nowych mechanizmów wiąże się też z dodaniem nowych parametrów, które muszą być dobrane w sposób eksperymentalny. problemem może być dobór parametru *Opt*, co jest opisane wyżej.

TS-SA

Jeśli chodzi o problem doboru parametrów, to ta metoda ma znaczną przewagę nad poprzednią. Opisane jest to pod koniec punktu 2.1 w pracy [2]. Adaptywne ochładzanie oraz dynamiczny sposób generowania sąsiadów sprawiają, że ten algorytm wymaga mniej eksperymentów w celu dobrania parametrów dla niego. *„(...) less pre-search analysis may be required."* [2]. Zatem TS-SA posiada więcej parametrów, które w pewnym stopniu potrafią się same dostosować do problemu.

5.3. Wnioski z porównania dwóch metod

Na podstawie obu przedstawionych w tym referacie algorytmów/metod można zauważyć, że niekiedy nieznaczne modyfikacje dla symulowanego wyżarzania (zapewne dotyczy to również innych metaheurystyk) potrafiły znacznie poprawić jakość oraz szybkość jego działania. Co więcej, na uwagę zasługuje fakt, że standardowe SA nie należy do najszybszych metaheurystyk (Introduction w [1]). Jednak po dodaniu pewnych modyfikacji znacznie zyskało na szybkości (znaczna poprawa zbieżności). Przedstawione metody przy pewnych testach okazały się szybsze niż niektóre inne znane algorytmy rozwiązujące TSP w tym te oparte na sieciach neuronowych. [1, 2]

Rzeczą, która znacznie wpływa na osiągi (performance) jest dobór parametrów. Niekiedy ich dobranie może być czasochłonne i zwyczajnie trudne. Dlatego, modyfikacje, które pozwalają algorytmowi na lepsze dostosowanie się do problemu (takie jak np. cooling schedule w TS-SA) mogą być bardzo pomocne oraz mogą znacznie zwiększyć praktyczną użyteczność algorytmu. (Algorytm, który działa rewelacyjnie przy pewnych parametrach, których prawidłowe dobranie graniczy z cudem nie jest zbyt praktyczny).

Parametrem, który w przypadku obu przedstawionych metod miał niezwykle duży wpływ na osiągi była definicja sąsiedztwa. Zatem widać, że jest to parametr, który zasługuje na szczególną uwagę. Niekiedy użycie pewnego operatora do generowania sąsiadów może znacznie spowolnić lub przyspieszyć działanie algorytmu. Przykładem może być mutacja BI (w ASA-GS), która okazała się bardzo wolna, stąd też proponowane prawdopodobieństwo użycia tej mutacji wynosi tylko 1%.

Sąsiedztwo okazało się również niezwykle ważnym czynnikiem wpływającym na trade-off pomiędzy eksploracja, a ekspansją. W TS-SA sam sposób w jaki zdefiniowane sąsiedztwo poprawiał eksplorację ($\mathcal{M}_1$) lub poprawiał ekspansję ($\mathcal{M}_6$). (Patrz: 11) Warto dodać, że wszystkie użyte w obu proponowanych metodach operatory mutacji są stosunkowo proste w implementacji.

Dobór temperatury oraz funkcji prawdopodobieństwa akceptacji wymaga dobrego zrozumienia idei algorytmu symulowanego wyżarzania. Dzięki rozumieniu roli tych parametrów zaproponowane mogły być sensowne modyfikacje takie jak cooling schedule oraz konkretne funkcje prawdopodobieństwa adekwatne do metody ochładzania. Dlatego też, w tym referacie znajduje się dużo odniesień do ogólnej idei symulowanego wyżarzania oraz roli/idei jego parametrów. Zatem widać, że dobre zrozumienie używanych metaheurystyk jest kluczowe przy ich modyfikowaniu.

Możliwość ponownego ogrzewania w TS-SA jest dobrym przykładem tego, że pewna nietypowa zmiana potrafi wiele poprawić. Proces wyżarzania w metalurgii (będący inspiracją dla SA) nie zakłada ponownego rozgrzewania metalu, stąd można pomyśleć, że w SA również nie powinien mieć miejsca. Wniosek z tego jest taki,że przy modyfikowaniu metaheurystyk, odstępstwo od rdzennej idei algorytmu może być pomocne.

Metoda TS-SA jest świetnym przykładem hybrydy dwóch metaheurystyk. Jest to połączenie symulowanego wyżarzania z tabu seach'em. Widać, że połączenie stosunkowo wolnej metody jaką jest SA, z Tabu Search'em dało świetny rezultat. Otrzymana hybryda okazała się szybsza i skuteczniejsza niż standardowe SA czy TS. Płynie stąd jeden z najważniejszych wniosków w tym referacie – warto eksperymentować z łączeniem różnych heurystyk ze sobą. Jednak, aby robić to mądrze, wymagana jest znajomość i dobre zrozumienie używanych heurystyk, oraz znajomość ich wad i zalet. Każda z metaheurystyk ma swoje wady i zalety, ale poprzez ich łączenie możliwe jest w pewnym (lub nawet całkowitym) stopniu ominięcie ich wad i wyeksponowanie zalet. Inną niezwykle ważną rzeczą jest zrozumienie rozwiązywanego problemu. Pozwala to na dostosowanie heurystyk do konkretnego zagadnienia – w przypadku zaprezentowanych metod tym zagadnieniem był problem komiwojażera. Czynnością, która znacznie może pomóc w zrozumieniu problemu jest jego matematyczne sformułowanie.

Bibliografia

[1] Xiutang Geng, Zhihua Chen, Wei Yang, Deqian Shi, Kai Zhao. *Solving the traveling salesman problem based on an adaptive simulated annealing algorithm with greedy search*, 2011

[2] Yi Liu, Shengwu Xiong, Hongbing Liu. *Hybrid Simulated Annealing Algorithm Based on Adaptive Cooling Schedule for TSP*, 2009

[3] Chi-Hwa Song, Kyunghee Lee and Won Don Lee. *Extended Simulated Annealing for Augmented TSP and Multisalsemen TSP*, 2003

[4] Walid Mahdi, Seyyid Ahmed Medjahed, Mohammed Ouali. *Performance Analysis of Simulated Annealing Cooling Schedules in the Context of Dense Image Matching*, 2017
`http://www.scielo.org.mx/pdf/cys/v21n3/1405-5546-cys-21-03-00493.pdf`

[5] Artykuł *Travelling Salesman Problem* na Wikipedii,
`https://en.wikipedia.org/wiki/Travelling_salesman_problem`