Symulacja w Badaniach i Rozwoju
Vol. 4, No. 2/2013

Joanna KARBOWSKA-CHILIŃSKA, Paweł ZABIELSKI
Politechnika Białostocka, ul. Wiejska 45A 15-351 Białystok
E-mail j.karbowska@pb.edu.pl, p.zabielski@pb.edu.pl

Zastosowanie problemu komiwojażera z profitami i oknami czasowymi do wyznaczania tras atrakcyjnych turystycznie okolic Białegostoku

1 Wstęp

W dzisiejszych czasach nie mamy zbyt dużo wolnego czasu na podróżowanie. Często wyjazdy są krótkie, organizowane na własną rękę. Wybierając się na wycieczkę, chcielibyśmy zobaczyć jak najwięcej ciekawych miejsc w przewidzianym nieprzekraczalnym czasie, po którym znów musimy wrócić do codziennych obowiązków. Problem, jaki tu napotykamy, to optymalne zaplanowanie trasy zwiedzania, żeby wykorzystać maksymalnie wolny czas, a jednocześnie zobaczyć obiekty najbardziej polecane przez innych turystów.

Problem komiwojażera z profitami i oknami czasowymi [8] (ang. *Orienteering Problem with Time Windows* – OPTW) jest dogodnym modelem dla problemu optymalnego planowania tras. OPTW należy do ogólnej klasy grafowych problemów optymalizacyjnych tzw. klasy *orienteering problem* (OP) [4]. W OP dane są obiekty i znane są odległości pomiędzy nimi. Każdy obiekt ma przypisany zysk (ang. *profit*) związany z jego odwiedzeniem. Rozwiązaniem OP jest taka trasa, która przechodzi przez te obiekty, których sumaryczny zysk jest największy i której długość nie przekracza ustalonego ograniczenia. W OPTW każdy obiekt ma dodatkowo przyporządkowany przedział czasu (tzw. okno czasowe), w którym może on zostać odwiedzony, oraz czas związany z przebywaniem w obiekcie.

Zarówno OP, jak i OPTW należą do klasy problemów NP-trudnych [2, 8]. W zastosowaniach praktycznych w takich dziedzinach, jak turystyka, logistyka, znaczenie mają metaheurystyki, które zwracają rozwiązanie w bardzo krótkim czasie. Do znanych metod metaheurystycznych rozwiązujących OPTW należą: metody lokalnych poszukiwań (ang. *local serach*), zmienne przeszukiwanie sąsiedztwa [2] (ang. *variable neighbour search*) czy algorytmy mrówkowe [6] (ang. *ant colony optimization*).

We wcześniejszych naszych badaniach został opracowany algorytm genetyczny dla OPTW [3], zwany dalej GA, który dawał dobre trasy pod względem ich jakości, jak również pod względem złożoności czasowej dla standardowych benchmarków znanych z literatury. W badaniach przedstawionych w tym artykule modyfikujemy poprzednie rozwiązanie genetyczne: zamiast krzyżowania wykonujemy wymianę obiektów na wybranych losowo trasach (ang. *path relinking* (PR)) [1, 7, 5]. W procesie PR wybierane są dwie losowe trasy X, Y i obiekty, które są obecne w trasie X, a nie są obecne w Y, są do niej włączone. Jeśli nie można włączyć obiektu do Y ze względu

na ograniczenia czasowe, usuwa się pewne obiekty z Y, żeby zrobić miejsce na nowe obiekty z trasy X. Po takiej wymianie zazwyczaj nowo wygenerowana trasa jest lepszej jakości i zastępuje gorszą trasę (X lub Y). Zaprezentowana w tym artykule metoda (w skrócie dalej nazywana GAPR) daje porównywalne wyniki w testach przeprowadzonych na realnych obiektach turystycznych okolic Białegostoku w stosunku do poprzedniej metody GA (z krzyżowaniem dwóch losowych osobników).

W dalszej części pracy, w rozdziale 2, przedstawimy definicję problemu OPTW. W rozdziale 3 opiszemy szczegóły zaprezentowanego algorytmu. Wyniki testów algorytmu GAPR w porównaniu z wersją algorytmu z krzyżowaniem na realnej sieci obiektów turystycznych Białegostoku zostaną przedstawione w rozdziale 4, natomiast wnioski i perspektywy dalszych badań w ostatnim rozdziale.

2 Definicja problemu

Rozważany problem komiwojażera z profitami i oknami czasowymi (ang. OPTW) formalnie jest definiowany jako optymalizacyjny problem grafowy [8]. Rozważamy graf G z n obiektami, gdzie każdy obiekt i ma przyporządkowany zysk p_i związany z jego odwiedzeniem, czas T_i związany z przebywaniem w obiekcie oraz okno czasowe $[O_i, C_i]$, gdzie O_i i C_i oznaczają czas otwarcia i czas jego zamknięcia. Jeśli obiekt zostanie odwiedzony przed czasem O_i, to możliwe jest czekanie w tym obiekcie do czasu jego otwarcia. Dane są punkt startowy s i końcowy e trasy oraz godziny w których trasa ma się zaczynać i kończyć (godziny te wyznaczają nieprzekraczalny czas, w którym obiekty będą odwiedzane). Rozwiązaniem OPTW jest trasa pomiędzy punktem s i e, która daje największy sumaryczny zysk odwiedzanych obiektów, a czas jej trwania nie przekracza ustalonego nieprzekraczalnego czasu t_{max}.

Jeśli graf G jest grafem obiektów turystycznych, to wartość p_i przypisana obiektowi jest interpretowana jako jego atrakcyjność. Okno czasowe obiektu i $[O_i, C_i]$ oznacza czas otwarcia i zamknięcia tego obiektu, a T_i wyznacza czas zwiedzania danego obiektu. Punkty s i e wyznaczają punkt początkowy i końcowy trasy wycieczki, a t_{max} oznacza długość (czas trwania) wycieczki. Przy takiej interpretacji rozwiązanie OPTW zwraca najbardziej atrakcyjną trasę wycieczki, której czas trwania nie przekracza ustalonego t_{max}.

3 Algorytm genetyczny

W pierwszym etapie generowana jest losowo pewna liczba tras P_{size} należących do populacji początkowej. Poszczególne trasy są zakodowane jako ciąg obiektów, a każdy obiekt dodawany do trasy musi spełniać ograniczenia czasowe nałożone przez okno czasowe i t_{max}. W kolejnych iteracjach algorytmu wygenerowane trasy podlegają ocenie za pomocą selekcji, losowe trasy podlegają wymianie obiektów (ang. *path relinking*-PR) oraz mutacji. Liczba iteracji algorytmu genetycznego, oznaczona jako Ng jest ustalonym parametrem. W stosowanej grupowej selekcji turniejowej cała populacja jest dzielona na grupy i selekcja jest przeprowadzana oddzielnie w każdej grupie. Podczas selekcji w każdej z grup wybierane są osobniki o największej wartości funkcji *fitness*= *sumaryczny profit trasy*^{3}*/całkowity czas trasy*. Po selekcji stosujemy mutację wstawiającą lub usuwającą (prawdopodobieństwo każdej metody zostało ustalone na 0.5). W procesie PR (który jest zastosowany zamiast krzyżowania) wybierane są dwie losowe trasy X i Y i obiekty, które są obecne w trasie X, a nie są obecne w Y są do niej

włączone. Jeśli nie można włączyć obiektu do Y ze względu na ograniczenia czasowe, usuwa się pewne obiekty z Y, żeby zrobić miejsce na nowe obiekty z trasy X. Po takiej wymianie zazwyczaj nowo wygenerowana trasa jest lepszej jakości i zastępuje gorszą trasę (X lub Y).

W poniższych podrozdziałach przedstawiamy szczegółowy opis kolejnych etapów algorytmu GAPR.

3.1 Generowanie populacji początkowej

Każda trasa jest inicjowana przez punkt początkowy *s* i końcowy *e,* którym są przyporządkowane następujące wartości: $arrive_i$ - czas przybycia do obiektu *i*, $wait_i$ - czas oczekiwania na obsługę w obiekcie *i* oraz $start_i$ i end_i czas rozpoczęcia i zakończenia obsługi obiektu *i*. Ponadto obiektowi jest przyporządkowana zmienna $MaxShift_i=Min(C_i-start_i-T_i,\ wait_{i+1}+MaxShift_{i+1})$, która określa, jakie może być maksymalne opóźnienie przy odwiedzeniu obiektu *i*, żeby obiekty stojące za obiektem mieściły się w swoich oknach czasowych i ograniczeniu t_{max}. Kolejne losowe wierzchołki są dołączone na koniec trasy przed wierzchołek końcowy *e*, jeśli spełniają ograniczenia czasowe związane z ich oknami czasowymi oraz po jego włączeniu trasa nie przekracza t_{max}. Wraz z włączeniem nowego obiektu *i* do trasy zmienne $arrive_i$, $wait_i$, $start_i$ i end_i są przyporządkowywane oraz zmienna $MaxShift_i$ jest uaktualniana dla obiektów, począwszy od końca do początku trasy. W populacji początkowej jest generowanych P_{size} tras.

3.2 Selekcja turniejowa z podziałem na grupy

Ten etap ma na celu wyselekcjonowanie osobników z największą wartością funkcji *fitness* do nowej populacji. Cała populacja jest podzielona na *k* grup, każda grupa zawiera P_{size}/k tras. Trasy są przydzielane do grup w kolejności ich występowania. Turniej jest przeprowadzany oddzielnie w każdej grupie. Z jednej grupy jest wybieranych t_{size} tras i następnie ich jakość jest oceniana za pomocą funkcji: *fitness= sumaryczny profit trasy*^{3}*/całkowity czas trasy.* Trasa z największą wartością *fitness* jest wybierana do nowo tworzonej populacji, a wybranych t_{size} tras jest zwracane do starej populacji. Proces selekcji w tej samej grupie jest powtarzany P_{size}/k razy.

3.3 Wymiana obiektów między wybranymi trasami

Ten etap algorytmu zastępuje krzyżowania. Przez $V_{R1\text{-}R2}$ oznaczmy zbiór wierzchołków, występujących w trasie R_1 i nie występujących w R_2. $V_{R2\text{-}R1}$ oznacza podobnie zbiór wierzchołków występujących w trasie R_2 i nie występujących w R_1. Podczas wymiany obiektów pomiędzy losowo wybranymi trasami R_2 i R_1 (symbolicznie oznaczamy przez PR(R_1, R_2)) obiekty ze zbioru $V_{R2\text{-}R1}$ są wstawiane w najbardziej korzystne miejsce w R_1 wyznaczane na podstawie współczynnika $InsertionRatio= (p_j)^2/ (Shift_j)$, gdzie obiekt *j* jest kandydatem do wstawienia. Uwzględniamy całkowity czas związany ze wstawieniem obiektu *j* pomiędzy *i* oraz *k*, jak we wzorze: $Shift_j=t_{ij}+wait_i+T_j+t_{jk}-t_{ik}$, oraz sprawdzamy, czy przesunięcie związane ze wstawieniem nowego obiektu nie przekroczy *MaxShift* dla wierzchołków znajdujących się za miejscem, gdzie wierzchołek możemy wstawić. Jeśli przesunięcie związane ze wstawieniem nowego wierzchołka nie mieści się w ograniczeniach czasowych, wtedy obiekty ze zbioru $V_{R1\text{-}R2}$ są usuwane z R_1, żeby przygotować miejsce dla nowych obiektów. Dla każdego

wierzchołka z R_1 (z wyjątkiem początkowego i końcowego) jest wyznaczany następujący współczynnik $RemovalRatio= (p_j)^2 / (end_j - arrive_j)$. Obiekt z najmniejszą wartością tego współczynnika jest usuwany. Usuwanie jest powtarzane, gdy nie ma jeszcze miejsca, żeby wstawić nowy obiekt. Po zwolnieniu miejsca nowy obiekt j wstawiamy w najbardziej korzystną pozycję w R_1, wyznaczoną na podstawie największej wartości współczynnika *InsertionRatio*. Po wstawieniu obiektu przyporządkowujemy mu zmienne $arrive_j$, $wait_j$, $start_j$ i end_j oraz zmienna $MaxShift_j$ jest uaktualniana dla obiektów począwszy od końca do początku trasy. Proces jest powtarzany, dopóki t_{max} nie jest przekroczony oraz zbiór V_{R2-R1} nie jest pusty. Następnie proces PR(R_2, R_1) jest wykonywany: wierzchołki ze zbioru V_{R1-R2} są włączane do R_2. Wynikiem PR(R_1, R_2) i PR(R_2, R_1) są dwie nowe trasy, które podlegają ocenie za pomocą funkcji *fitness*. Jeśli wartość funkcji *fitness* tych tras jest wyższa niż tras będących rodzicami, to zastępują one słabszego ze swoich rodziców.

3.4 Mutacja

Losowa trasa wybrana z całej populacji jest poddawana tej modyfikacji. Wykorzystywane są dwa rodzaje mutacji: wstawiająca i usuwająca obiekty. Przyjmujemy, że prawdopodobieństwo każdej z nich wynosi 0.5. Mutacja jest powtarzana w jednej trasie N_m razy, w zależności od wartości tego parametru.

Podczas każdej mutacji wstawiającej wyznaczane jest miejsce do jak najkorzystniejszego wstawienia obiektu przez wyznaczenie współczynnika *InsertionRatio*, tak jak w PR. Po wstawieniu obiektu zostają wyznaczone wartości zmiennych związanych z odwiedzeniem obiektu oraz zostaje uaktualniona wartość *MaxShift* od końca do początku mutowanej trasy.

W procesie usuwania obiektu wybierany jest losowy obiekt, oprócz pierwszego i ostatniego elementu trasy. Kiedy obiekt zostanie usunięty z trasy, wszystkie obiekty za tym elementem usuniętym są przesuwane w kierunku początku trasy i nowe wartości zmiennych *arrive*, *wait*, *start* i *end* oraz *MaxShift* są wyznaczane dla obiektów na tej trasie.

4 Testy i wyniki eksperymentów

Algorytm GAPR został zaimplementowany w C++. Testy zostały przeprowadzane na komputerze z procesorem Intel Core i7, 1.73 GHz CPU na rzeczywistej sieci 174 obiektów turystycznych Białegostoku i okolic. Każdy obiekt jest opisany za pomocą współrzędnych geograficznych. Dodatkowo każdy obiekt ma przyporządkowane okno czasowe i czas związany ze zwiedzaniem obiektu (na podstawie danych o obiekcie z Internetu). Profit każdego obiektu został wyznaczony na podstawie przeskalowanej liczby wyników wyszukiwania tego obiektu w Google (liczba wyników/1000). Odległości między obiektami c_1 i c_2 są wyznaczone na podstawie podanego wzoru uwzględniającego krzywiznę Ziemi:

$$distance_{c1c2}=6378137 \cdot arccos(sin(lat_{c1}) \cdot sin(lat_{c2}) + cos(lat_{c1}) \cdot cos(lat_{c2}) \cdot cos(long_{c1} - long_{c2})),$$

gdzie $long_{c1}$, $long_{c2}$, lat_{c1}, lat_{c2} oznaczają długość i szerokość geograficzną danych punktów c_1 i c_2. Przyjęto, że średnia prędkość poruszania się między obiektami wynosi 50 km/h.

Podczas wstępnych testów zostały ustalone najkorzystniejsze wartości parametrów

algorytmu genetycznego: P_{size}=150 to liczba tras w populacji początkowej, k=15 to liczba grup, na które populacja jest podzielona w selekcji turniejowej, t_{size} =3 to liczba tras wybranych do turnieju, N_m=15 to liczba mutacji wykonywanych na tej samej trasie podczas jednej iteracji algorytmu. Maksymalna liczba iteracji została ustalona na 500. Możliwe jest wcześniejsze zakończenie programu, jeśli wynikowe trasy przez 100 kolejnych iteracji nie mają lepszego profitu.

Podczas testów GAPR były generowane różne trasy z ustalonych punktów z centrum Białegostoku oraz okolic Białegostoku (np. Zabłudów). Rozpatrywane były różne godziny trwania wycieczki: 5.00-24.00, 8.00-18.00, 10.00-18.00. Dla porównania wyników z tych samych punktów startowych zostały wygenerowane trasy poprzednią wersją algorytmu genetycznego GA w wersji z krzyżowaniem [3]. W GA krzyżowaniu są poddawane dwie losowe trasy. Najpierw jest tworzony zbiór punktów z jednej i drugiej trasy, które mają podobne okna czasowe i podobne zmienne *start* i *end*. Następnie wśród tych punktów jest wybierany losowy punkt, według którego wymieniane są trasy od puntu przecięcia do końca tras.

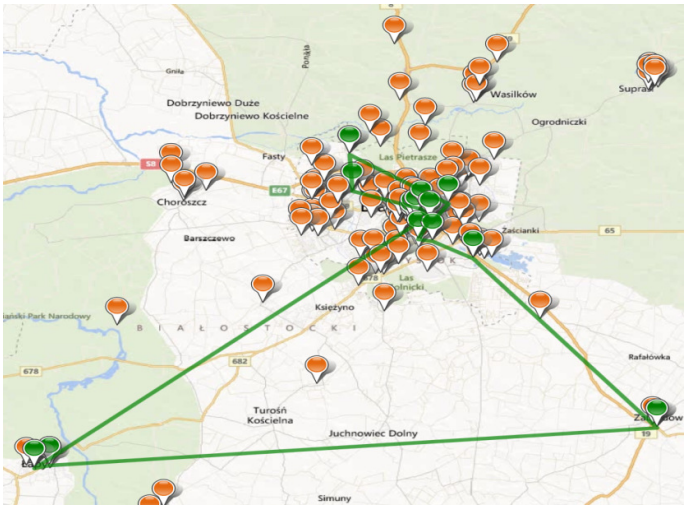
Dla porównania przedstawiamy przykładowe trasy wygenerowane przez GAPR i GA. Wynik uzyskanego profitu GAPR w poniższym przykładzie różni się około 0.5% w stosunku do GA. Czasy wygenerowania tras są porównywalne w jednym i drugim algorytmie i nie przekraczają 1 s.

Tab. 1. Wycieczka zaczynająca się i kończąca w Zabłudowiu w godzinach 10.00-18.00
Tab. 1. The trip staring and ending in Zabłudow (10:00-18:00)

GAPR	**GA (z krzyżowaniem)**
Zabłudów - start, godz. 10.00	Zabłudów - start, godz. 10.00
Kino Dom Kultury w Łapach	Kino Dom Kultury w Łapach
Muzeum Szkolne w Łapach	Muzeum Szkolne w Łapach
Uniwersytet Medyczny w Białymstoku	Cmentarz Wojenny w Białymstoku
Kino Forum	Kino Forum
Książnica Podlaska	Ratusz
Ratusz	Pałac Nowika
Pałac Nowika	Kościół pw. św. Krzysztofa
Kościół pw. św. Wojciecha	Kościół pw. Świętej Rodziny w Wasilkowie
Kościół pw. św. Krzysztofa	Centrum pielgrzymkowe św. Woda
Ceriew pw. Świętych Niewiast	Kościół pw. św. Wojciecha
Kościół pw. Świętej Rodziny w Białymstoku	Kościół pw. Świętej Rodziny w Białymstoku
Centrum pielgrzymkowe św. Woda	Uniwersytet w Białymstoku

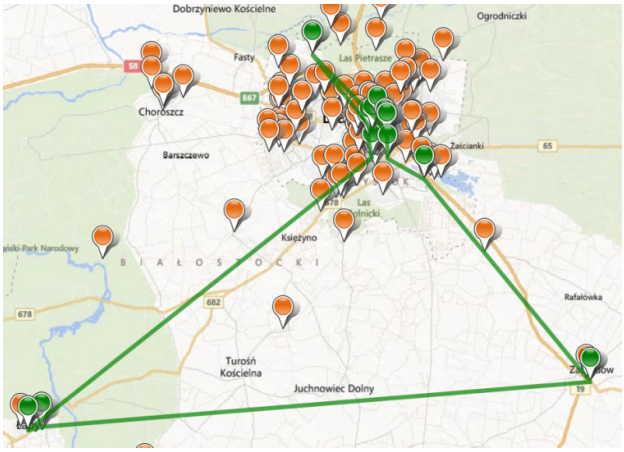
Kościół pw. Świętej Rodziny w Wasilkowie Pomnik Piłsudskiedgo	Powrót do punktu startowego w Zabłudowie – godz.17.57
Uniwersytet w Białymstoku Powrót do punktu startowego w Zabłudowie – godz. 17.58	
Maksymalny profit: 10288 **czas wygenerowania trasy**: 0.754 s	**Maksymalny profit:** 10230 **czas wygenerowania trasy:** 0.787 s

Na rysunku 1 i 2 powyższe trasy zostały wyrysowane na mapie.



Rys. 1. Trasa wycieczki z Zabłudowa wygenerowana algorytmem GA(z krzyżowaniem)
Fig. 1. The trip from Zabłudów obtained by GA (with the crossover)

Przeprowadzone eksperymenty (biorąc pod uwagę różnorodnie zlokalizowane punkty startowe trasy) wykazują, że atrakcyjność tras turystycznych wygenerowanych algorytmem GAPR jest około 0.5% wyższa niż algorytmem GA wykorzystującym krzyżowanie. Czas generowania trasy przy trasach najdłuższych (w godzinach 5.00-24.00) nie przekraczał 1.5 sekundy.



Rys. 2. Trasa wycieczki z Zabłudowa wygenerowana algorytmem GAPR
Fig. 2. The trip from Zabłudów obtained by GAPR

5 Podsumowanie

Problem komiwojażera z profitami i oknami czasowymi jest dogodnym modelem do wyznaczania tras atrakcyjnych turystycznie. Przedstawiony w pracy algorytm genetyczny został przetestowany na grafie obiektów turystycznych okolic Białegostoku. Algorytm, który zamiast krzyżowania używa wymiany obiektów pomiędzy wybranymi losowymi trasami, daje satysfakcjonujące rezultaty, jeśli chodzi o jakość generowanych tras i czas wykonania algorytmu.

W dalszych badaniach zaplanowane są testy na jeszcze większych grafach obiektów turystycznych (np. kilku regionów turystycznych). Kolejnym celem jest taka modyfikacja rozwiązania, żeby przy generowaniu trasy była uwzględniana możliwość generowania tras kilkudniowych, których trasa na każdy dzień ma ustalony nieprzekraczalny czas trwania.

Literatura

1. Campos, V., Marti R., Sanchez-Oro J., Duarte A.: Grasp with Path Relinking for the Orienteering Problem, *Journal of the Operational Research Society*, pp. 1-14, 2013
2. Labadie N., Mansini R., Melechovsky J., Wolfler Calvo, R.: The Team Orienteering Problem with Time Windows: An LP-based Granular Variable Neighborhood Search, *European Journal of Operational Research*, 220 (1), 15-27, 2012
3. Karbowska-Chilinska J., Zabielski P.: A Genetic Algorithm Solving Orienteering Problem with Time Windows, *Advances in Systems Science*, *Advances in Intelligent Systems and Computing*, Springer, 240, pp. 609-619, 2014

4. Karbowska-Chilinska J., Koszelew J., Ostrowski K., Zabielski P.: Genetic algorithm solving orienteering problem in large networks, *Frontiers in Artificial Intelligence and Applications*, 243, pp. 28-38, 2012
5. Karbowska-Chilinska J., Zabielski P.: Genetic Algorithm with Path Relinking for the Orienteerig Problem with Time Widows, *Proceedings of the 22nd International Workshop on Concurrency, Specification and Programming*, pp. 245-258, 2013
6. Montemanni R., Weyland D., Gambardella L. M.: An Enhanced Ant Colony System for the Team Orienteering Problem with Time Windows, *Proceedings of IEEE ISCCS*, 381-384, 2011
7. Souffriau W., Vansteenwegen P., Vanden Berghe G., Van Oudheusden D.: A path relinking approach for the team orienteering problem. *Computers & operations research*, 37 (11), 1853-1859, 2010
8. Vansteenwegen P., Souffriau W., Van Oudheusden D.: The Orienteering Problem: A survey, *European Journal of Operational Research*, 209 (1), pp. 1-10, 2011

Streszczenie

Problem komiwojażera z profitami i oknami czasowymi jest dogodnym modelem dla problemu optymalnego planowania tras atrakcyjnych turystycznie. W pracy przedstawiono rozwiązanie tej odmiany problemu komiwojażera za pomocą algorytmu genetycznego GAPR. W miejsce krzyżowania zaproponowano wymianę obiektów między losowo wybranymi trasami. Rozwiązanie przetestowano na rzeczywistej sieci obiektów turystycznych okolicy Białegostoku. W wyniku testów otrzymano trasy o porównywalnej atrakcyjności jak w algorytmie genetycznym GA z krzyżowaniem (różnica jest na korzyść GAPR około 0.5%) i czasem generowania trasy nie przekraczającym 1.5 sekundy. Algorytm może być zastosowany w planerach tras turystycznych.

Słowa kluczowe: problem komiwojażera, algorytm genetyczny, generowanie tras atrakcyjnych turystycznie

An application of the Orienteering Problem with Time Windows for tour planning problem in Bialystok region

Summary

Orienteering problem with time windows (OPTW) is a good model for the tour planning problem. In this article a genetic algorithm with path relinking (GAPR) is used for solving OPTW. The path relinking (PR) process is applied instead of a crossover. The solution has been tested on a real network of tourist points of interests in Bialystok region. Routes which are the test results are comparable with the routes generated by the previous GA with crossover (the GAPR exceeds profit result about 0.5% relative to GA, the execution time of GAPR does not exceed 1.5 s) The algorithm can be used in trip planners.

Keywords: orienteering problem, genetic algorithm, tour planning problem

Artykuł został sfinansowany z prac badawczych S/WI/1/2014 oraz W/WI/2/2013 realizowanych na Wydziale Informatyki Politechniki Białostockiej.