Symulacja w Badaniach i Rozwoju
Vol. 3, No. 3/2012

Paweł ZABIELSKI[1], Jolanta KOSZELEW[1]
Robert ZIMNOCH[2]
[1]Politechnika Białostocka, ul. Wiejska 45A 15-351 Białystok
[2]Transition Technologies, ul. Lipowa 19/21, 15-424 Białystok
E-mail: p.zabielski@pb.edu.pl, j.koszelew@pb.edu.pl,
robert.zimnoch@live.com

Problem komiwojażera z zyskami i oknami czasowymi dla sieci o wagach zmiennych w czasie i jego zastosowania w systemach typu e-tourism

1 Wstęp

W dzisiejszych czasach stały się popularne wyjazdy zorganizowane samodzielnie do mało znanych miejsc na całym świecie. Cele takich wypraw mogą być przeróżne i zależą od indywidualnych preferencji wyjeżdżających. Jedni wyjeżdżają, aby odpocząć, inni z chęcią odkrycia czegoś nowego, a jeszcze inni z nadzieją, że aktywnie spędzą wolny czas. Niezależnie od wymienionych wyżej preferencji uczestnik podróży ma na ogół potrzebę jej zaplanowania. Nie jest to przedsięwzięcie łatwe, wymaga bowiem czasu oraz dodatkowej wiedzy, którą trzeba zdobyć na podstawie przewodników czy informacji zamieszczonych na portalach podróżniczych. Pomocne w rozwiązywaniu wyżej wymienionych problemów stają się coraz bardziej popularne systemy internetowe typu e-tourism, które zawierają tzw. planery podróży [5]. Zaimplementowane i wykorzystane w nich algorytmy są w stanie wygenerować, w czasie rzeczywistym, trasy o możliwie najwyższej atrakcyjności turystycznej i spełniającej preferencje i ograniczenia narzucone przez użytkownika aplikacji.

Liczba użytkowników takich systemów w Polsce rośnie i wynosi około 10% rocznie w ciągu ostatnich dwóch lat [7]. Jak wynika z danych przygotowanych przez Instytut Turystyki, ciągle panuje trend, iż większość wyjeżdżających w zakresie własnym organizuje i planuje wyjazd. Najpopularniejsze w Polsce serwisy podróżnicze to: mktramping, odysei, globtroter i tubyliśmy. Jednak nie zawierają one planera podróży [5].

Niniejszy artykuł jest poświęcony algorytmowi TDILS (ang. *The Time Dependent Iterated Local Search*), rozwiązującemu słabo zbadaną wersję problemu generowania trasy spełniającej ograniczenia czasu podróży oraz godzin dostępności obiektów turystycznych, tzw. punktów POI (ang. *Points of Interest*), o najwyższej łącznej ocenie użytych w trasie obiektów. Bardzo istotne jest to, że rozważana wersja problemu nie zakłada ograniczeń na rodzaj środków transportu, którymi realizowana jest podróż. Wszystkie dostępne aktualnie planery zakładają podróż przy użyciu samochodu lub pieszo. Szczególnie w przypadku zwiedzania miast istotne jest natomiast użycie środków transportu publicznego, tj. autobusu, tramwaju czy metra. Możliwość wykorzystania środków transportu o zadanych rozkładach jazdy powoduje bardzo

istotne utrudnienie problemu, gdyż połączenia w rozważanej sieci obiektów są zmienne w czasie. Zaproponowany algorytm został przetestowany na realnej sieci obiektów Krakowa. Na podstawie przeprowadzonych testów dokonano oceny algorytmu ze względu na czas realizacji i jakość uzyskanych tras.

2 Definicja problemu

Artykuł został poświęcony rzadko dotąd rozważanej w literaturze odmianie problemu komiwojażera z zyskami i oknami czasowymi dla sieci o wagach zmiennych w czasie (ang. *The Time Dependent Orienteering Problem with Time Windows*, w skrócie TDOPTW). Standardowa wersja problemu komiwojażera z zyskami (ang. *Orienteering Problem*, w skrócie OP) pochodzi od nazwy gry sportowej - biegu na orientację [9]. Zadaniem zawodników jest wyruszenie z punktu startowego i zdobycie jak największej sumy ocen przypisanych odpowiednio do punktów kontrolnych w określonym przedziale czasowym. Problem OP można podobnie zdefiniować poprzez pewien zbiór wierzchołków z profitami i poprzez zadanie polegające na odnalezieniu trasy o zadanym punkcie początkowym i końcowym, maksymalizującej sumę profitów, oraz której czas trwania nie przekracza z góry zadanego limitu czasowego.

Jednym z rozwinięć problemu OP jest problem komiwojażera z zyskami i oknami czasowymi (ang. *Orienteering Problem with Time Windows*, w skrócie - OPTW) [1]. Zbiór danych wyposażony jest w tym przypadku w dodatkowe informacje odnośnie do okien czasowych w każdym z wierzchołków grafu: czas otwarcia i czas zamknięcia danego obiektu. Okna czasowe stanowią ograniczenie, które wymusza, aby określone miejsca mogły być odwiedzane tylko w określonych porach dnia. Należy zaznaczyć, iż dopuszczalne jest wcześniejsze dotarcie do obiektu i oczekiwanie na otwarcie lokalizacji w celu uzyskania profitu.

Problem komiwojażera z zyskami i oknami czasowymi dla sieci o wagach zmiennych w czasie (ang. *Time Dependent Orienteering Problem with Time Windows*, w skrócie: TDOPTW) to rozszerzenie problemu OPTW [1]. Uwzględniona w nim została zmienność wag sieci, tzn. czas podróży pomiędzy dwoma wierzchołkami nie jest stały, lecz zależy od czasu dotarcia do wierzchołka początkowego połączenia. Jest to kolejny czynnik, który należy uwzględnić, aby planowana trasa realniej wpasowana była w kryteria rzeczywistości. Różnorodność czasu połączeń może wynikać z wielu powodów, innych niż tylko użycie środków transportu publicznego o nieregularnych rozkładach jazdy. Powodem tej zmienności może być również dynamiczne natężenie ruchu w zależności od pory dnia, czy też robót drogowych. Problem OP jest problemem NP-trudnym [2]. Z tego wynika, iż rozszerzenie, jakim jest TDOPTW, również stanowi problem trudny do rozwiązania. Związany z tym jest fakt, iż zagadnienie TDOPTW praktycznie nie ma rozwiązania, które uwzględniałoby w pełni dynamikę wag połączeń. Przykładowo Garcia [1] przedstawia metodę, w której czasy połączeń są brane jako średnie czasy przejazdu autobusów. Jednak nie jest to do przyjęcia w sieciach, które mają nieregularne rozkłady jazdy, a jest to często spotykane zjawisko, np. w Polsce. Jin Li [6] omawia ideę uwzględniającą tylko różnorodność czasów trasy pomiędzy wierzchołkami, jednak nie bierze pod uwagę okien czasowych. Artykuł ten przedstawia koncepcję, która uwzględni czasy dostępności wierzchołków oraz rzeczywiste czasy połączeń.

3 Algorytm TDILS

Algorytm został oparty na modyfikacji metody iteracyjnych lokalnych poszukiwań (ang. *Iterated Local Search,* w skrócie ILS) dla problemu OPTW, uwzględniającej zmienność w czasie krawędzi grafu obiektów [8]. Metoda ILS polega na iteracyjnym przeszukiwaniu lokalnych rozwiązań (aktualnie wygenerowanej trasy) w celu wyznaczenia optimum lokalnego, a następnie określenia globalnego rozwiązania. Zaproponowany algorytm składa się z dwóch następujących kroków: wstawiania (ang. *insertion step*) oraz „potrząsania" (ang. *shake step*). Pierwszy krok jest odpowiedzialny za wybór i wstawienie nowych elementów do rozwiązania. Natomiast drugi krok dotyczy usuwania obiektów z trasy tak, aby rozwiązanie mogło wydostać się z optimum lokalnego.

Parametry S i R definiują liczbę obiektów do usunięcia oraz indeks, od którego są usuwane elementy trasy (w kroku *shake*). Przed rozpoczęciem działania algorytmu trasa jest inicjalizowana obiektem startowym oraz końcowym. Algorytm wykonuje się do czasu, aż liczba iteracji bez poprawy rozwiązania (profitu trasy) przekroczy określony parametr maksymalnej liczby bez poprawy wyniku. Wewnątrz głównej pętli wykonywane są kolejno kroki wstawiania, sprawdzenia czy jest to najlepsze rozwiązanie, oraz krok „potrząsania" z wyznaczeniem parametrów R i S. Parametr C ogranicza maksymalną liczbę obiektów do usunięcia w kroku „potrząsania".

Krok wstawiania polega na wyznaczeniu tablicy $Ratio[i,j]$, gdzie i oznacza pozycję do wstawienia oraz j – indeks obiektu z listy obiektów nie będących w trasie. Wyznaczenie tablicy *Ratio* składa się z następujących kroków:

- wyznaczenie czasu oczekiwania na rozpoczęcie zwiedzania / usługi ($Wait_{ij}$),
- obliczenie maksymalnego opóźnienia rozpoczęcia zwiedzania ($MaxShift_{ij}$),
- wyznaczenie całkowitego czasu poświęconego na obiekt ($Shift_{ij}$),
- sprawdzenie, czy istnieje możliwość wstawienia obiektu,
- obliczenie współczynnika atrakcyjności lokalizacji ($Ratio_{ij}$).

W pierwszym kroku wyliczany jest czas oczekiwania na rozpoczęcie zwiedzania danego obiektu. Wyliczana jest maksymalna wartość z 0 oraz różnicy czasu otwarcia punktu (O_j) i czasu przybycia do danego miejsca (a_{ij}). Jeżeli czas przybycia do lokalizacji zawiera się w oknie czasowym obiektu, to oznacza, że zwiedzanie pozycji rozpocznie się tuż po przybyciu do tego punktu (1):

$$Wait_{ij}=Max(0,O_j - a_{ij}) . \tag{1}$$

Następnym krokiem jest wyznaczenie maksymalnego opóźnienia, jakie może nastąpić przy zachowaniu warunku, że wszystkie kolejne elementy trasy będą dostępne do zwiedzania. Maksymalny czas opóźnienia zwiedzania obiektu jest minimalną wartością z pary wyników: sumy $Wait_{i+1}$i $MaxShift_{i+1}$ kolejnego elementu z trasy oraz różnicy czasów zamknięcia (C_j), rozpoczęcia (s_j), długości zwiedzania obiektu - T_j (2):

$$MaxShift_{ij} = Min(C_j - s_j - T_j, Wait_{i+1} + MaxShift_{i+1}) . \tag{2}$$

Trzecim krokiem potrzebnym do wyznaczenia $Ratio_{ij}$ jest obliczenie całkowitego czasu poświęconego na zwiedzanie obiektu. $Shift_{ij}$ jest sumą czasów przejazdu, z poprzedniego do kandydata do wstawienia (c_{ij}), $Wait_{ij}$, T_j i różnicy dwóch czasów

przejazdu kandydat – następny obiekt ($c_{j,i+1}$) oraz poprzedni (i-ty) – następny (i+1-wszy) obiekt ($c_{i,i+1}$):

$$Shift_{ij} = c_{ij} + Wait_{ij} + T_j + c_{j,i+1} - c_{i,i+1} \ . \tag{3}$$

Następnym krokiem jest sprawdzenie tego, czy wstawienie nowego obiektu nie spowoduje, że kolejne obiekty nie zostaną odwiedzone (4). W przypadku, gdy $Shift_{ij}$ nie spełnia następującego warunku, wtedy $Ratio_{ij}$ = -1:

$$Shift_{ij} \leq Wait_{i+1} + MaxShift_{i+1} \ . \tag{4}$$

W pozostałym przypadku, gdy został spełniony warunek, jest wyliczane $Ratio_{ij}$ na podstawie stosunku oceny obiektu (S_j) do całkowitego czasu poświęconego na obiekt. W zależności od wielkość potęgi i sposobu wyliczania wartości można wpływać na algorytm, np. tak, aby większą wagę przyznawał obiektom, które są najlepiej oceniane lub najmniej należy poświęcić czasu na zwiedzanie. Ważne jest to, aby znaleźć złoty środek, który uzyska najwyższą sumę wszystkich ocen obiektów zawierających się w trasie (5):

$$Ratio_{ij}=(S_j)^2/Shift_{ij} \ . \tag{5}$$

Po wyznaczeniu tablicy *Ratio* należy wybrać jeden obiekt, który osiągnął najwyższą wartość, a następnie wstawić go na pozycji i+1 do trasy turystycznej. Kolejnym elementem *insertion step* jest aktualizacja danych pozycji znajdujących się w planowej wycieczce. Począwszy od obiektu, który został wstawiony do obiektu końcowego, wyliczane są nowe czasy przybycia, rozpoczęcia zwiedzania oraz oczekiwania na otwarcie POI (6, 7, 8):

$$a_i = a_{i-1} + Wait_{i-1} + T_{i-1} + c_{i-1,i} \ , \tag{6}$$

$$Wait_i=Max(0,O_i - a_i) \ , \tag{7}$$

$$s_i = a_i + Wait_i \ . \tag{8}$$

Następnie dla wszystkich obiektów zmierzając od końca do początku aktualizowane są wartości *MaxShift*(3):

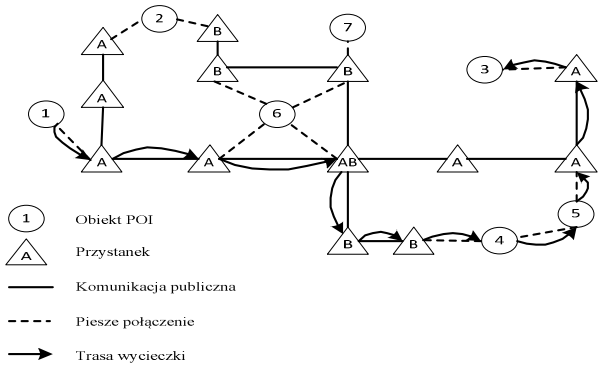
$$MaxShift_i = Min(C_i - s_i - T_i, \ Wait_{i+1} + MaxShift_{i+1}) \ . \tag{9}$$

Krok „potrząsania” rozpoczyna się od usunięcia obiektów z rozwiązania, począwszy od indeksu S, a kończąc na R kolejnym elemencie trasy. Jeżeli liczba usuwanych obiektów jest większa niż końcowy element, to usuwane są obiekty z początku trasy. Następnie są aktualizowane wartości *Arrive* (czas przybycia), *Start* (czas rozpoczęcia zwiedzania), *Wait, MaxShift* zgodnie ze wzorami z kroku wstawiania.

Istotnym elementem realizacji algorytmu jest wybór najlepszego połączenia między obiektami. Przygotowana została struktura danych w postaci tablicy dwuwymiarowej słowników (n na n obiektów POI), zawierająca dane o połączeniach pieszych, komunikacji miejskiej oraz samochodowej. W przypadku komunikacji miejskiej w ramach obliczeń wstępnych, tzn. przeprowadzonych jednorazowo przed wywołaniem algorytmu właściwego, zostały wyznaczone uśrednione czasy przejazdu dla każdej pary różnych punktów POI w poszczególnych godzinach doby. Do wyznaczenia tych czasów użyto algorytmu ewolucyjnego [4]. Wyznaczenie połączeń zrealizowano w sposób hybrydowy, tzn. do wyznaczenia tablicy *Ratio* została użyta opisana wyżej tablica połączeń z uśrednionymi czasami. Natomiast w procesie wstawienia nowego obiektu,

jak również w kroku „potrząsania" obliczenia bazują na rzeczywistych czasach połączeń komunikacji przechowywanych w bazie danych. Dzięki temu ostateczne trasy są trasami realnymi, tzn. wynikają z rozkładów jazdy, bez względu na porę dnia, w której trasa jest realizowana.

Rysunek 1 przedstawia schemat przykładowej sieci systemu dla omawianego problemu. Zaznaczone są na nim obiekty użyteczności publicznej oraz przystanki komunikacji publicznej. Każda para obiektów POI posiada jeden link pieszy (brak go na schemacie ze względu na czytelność rysunku) i ewentualne połączenie z komunikacją publiczną. Nie wszystkie obiekty POI muszą posiadać dostęp do przystanków (istnieje ograniczenie odległości między tymi typami do 0,5 km).



Rys. 1. Schemat sieci systemu

Fig. 1. Network diagram of the system

4 Wyniki testów

W celu przetestowania algorytmu przeprowadzone zostały testy badające jego efektywność pod względem czasu realizacji i jakości tras wynikowych. Do przeprowadzania testów przygotowana została baza danych realnych obiektów turystycznych i sieci transportu publicznego Krakowa. Baza punktów sieci składa się z 282 punktów POI. Sieć transportu publicznego zawiera 1385 przystanków oraz 157 linii transportu publicznego [10].

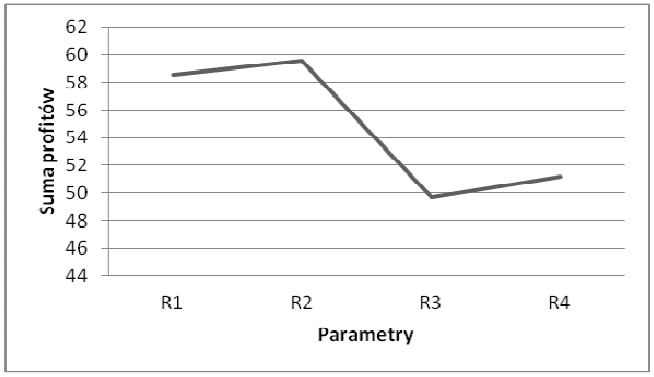
Para obiektów: początkowy (Parafia Miłosierdzia Bożego, współrzędne: [50,0149, 20,00471]) i końcowy (Wawel, współrzędne: [50,0538, 19,93504])). Punkty trasy, na których przetestowano algorytm, zostały dobrane w taki sposób, aby obiekt początkowy znajdował się na obrzeżach miasta, zaś obiekt końcowy w samym jego centrum, czyli miejscu o wysokiej gęstości POI.

Algorytm został uruchomiony z podanymi niżej parametrami:

- liczba iteracji algorytmu bez poprawy wyniku: [10, 50, 100, 150, 200],
- stopień potęgi *Ratio*: (*R*) [1, 2, 3, 4],
- czas rozpoczęcia: 10:00,

- czas zakończenia: 18:00,
- maksymalna długość połączenia pieszego między obiektami POI: 1 km.

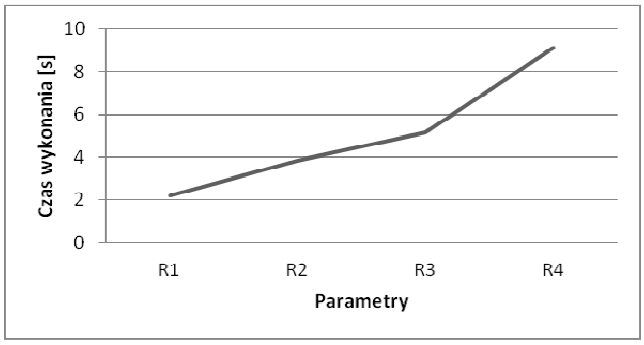
Rysunek 2 przedstawia zależność wyników algorytmu od parametru *R*. Można z niej odczytać silną zależność wyników od parametru *R*. Jakość trasy ulega widocznej zmianie. Najlepsze wyniki zostały uzyskane dla *R*=2 oraz *R*=1. Najlepszy wynik uzyskany podczas przeprowadzonych testów to: 59,53.



Rys. 2. Suma profitów w zależności od parametru R dla 10 iteracji bez poprawy wyniku

Fig. 2. Total profits depending on the parameter R for10 iterations without improvement in results

Rysunek 3 pokazuje zależność czasu wykonywania algorytmu od liczby iteracji bez poprawy wyniku. Czas realizacji jest stosunkowo krótki (maksymalnie 10 sekund) i stanowi wynik zadowalający. Należy zwrócić uwagę, iż liczba iteracji bez poprawy rezultatu nie ma znaczącego wpływu na wydłużenie czasu trwania obliczeń. Ustaloną przyczyną tego faktu jest zastosowanie cachowania zapytań do bazy danych.



Rys. 3. Średnia wartość czasu realizacji algorytmu w zależności od parametru R dla 200 iteracji bez poprawy wyniku

Fig. 3. The average value of the algorithm execution time, depending on the parameter R for 200 iterations without improvement in results

5 Podsumowanie

Czas wykonywania zaproponowanego w artykule algorytmu rozwiązującego problem TDOPTW jest stosunkowo krótki (kilka sekund). Pozwala to na wykorzystanie go w komercyjnych systemach internetowych typu e-tourism, zawierających tzw. planery podróży. Przeanalizowano wpływ różnych parametrów algorytmu na czas oraz jakość uzyskanych tras. W przyszłych pracach zaplanowane są testy opisanej metody na jeszcze większych sieciach, które będą obejmowały duże regiony turystyczne, czy nawet całe kraje. W przypadku trudności spowodowanych złożonością czasową, czy też jakością tras wynikowych, metoda TDILS zostanie zastąpiona rozwiązaniem opartym na algorytmie ewolucyjnym dla problemu OP, przedstawionym w [3].

Ponadto rozważane jest dodatkowe rozbudowanie problemu, np. poprzez planowanie tras wielodniowych, czy też wprowadzenie dodatkowego ograniczenia w postaci ograniczenia kosztów podróży związanych zarówno z koniecznością przemieszczania się, jak też wydatków związanych z wejściem do obiektów (biletów wstępu, czy też wynajęcia przewodnika).

Literatura

1. Garcia A., O., Vansteenwegen P., SouffriauW., Linaza M.T., *Hybrid Approach for the Public Transportation Time Dependent Orienteering Problem with Time Windows*, 151-158
2. Golden B., Levy L., Vohra R., The orienteering problem, *Naval Research Logistics*, vol. 34, pp. 307–18, 1987
3. Karbowska-Chilińska J., Koszelew J., Ostrowski K., Zabielski P.: *Genetic Algorithm Solving Orienteering Problem in Large Network,* KES 2012
4. Koszelew J., An Evolutionary Algorithm for The Urban Public Transportation, Springer-Verlag, *Computational and Collective Intelligence – Technologies and Applications, Lecture Notes in Computer Science*, vol. 6922, pp. 234-243
5. Koszelew J., Logistyka na usługach obieżyświatów – innowacyjny komponent oprogramowania typu e-tourism, *Logistyka*, 6/2010, pp. 54-56
6. Li J., Research on Team Orienteering Problem with Dynamic Travel Times, *Journal of Software*, vol. 7, no. 2, 2012
7. www.e-biznes.pl
8. Vansteenwegen P., Souffriau W., Berghe G.V., Oudheusden D.V., *Iterated local search for the team orienteering problem with time windows*, Gent, Leuven, 2009
9. Vansteenwegen P., Souffriau W., Oudheusden D.V., *The orienteering problem: A survey*, Gent, Leuven, 2010
10. Zimnoch R., *Efektywne rozwiązania dla problemu Time Dependent Orienteering Problem with Time Windows*, Białystok 2012

Streszczenie

W pracy przedstawiono problem komiwojażera z zyskami i oknami czasowymi dla sieci o wagach zmiennych w czasie. Jest to rozszerzenie standardowego problemu komiwojażera z zyskami. Przyczynia się to do bardziej praktycznego zastosowania go na przykład w systemach typu e-tourism. Dzięki dodatkowym ograniczeniom może stanowić prawdziwą pomoc podczas planowania ciekawych wycieczek, spełniających wszelkie preferencje użytkownika. W artykule zaproponowano algorytm wykorzystujący iteracyjne poszukiwanie lokalnych rozwiązań. Wykonano testy na realnych danych i przeanalizowano je pod względem czasu wykonywania oraz jakości otrzymanych wyników.

Słowa kluczowe: problem komiwojażera z zyskami i oknami czasowymi, iteracyjny algorytm lokalnego wyszukiwania, planer podróży, algorytm ewolucyjny

Time Dependent Orienteering Problem with Time Windows and its use in systems of e-tourism

Summary

This paper presents the Time Dependent Orienteering Problem with Time Windows. It is an extension of the standard traveling salesman problem with profits. This contributes to a more practical application of it, for example in systems of e-tourism. It can be a really helpful during planning trips and meet all your preferences and additional restrictions. The article proposes an algorithm that uses an iterative search for local solutions. This algorithm was tested on real data. After that we analyzed it in terms of execution time and quality results.

Keywords: salesman problem with profits and time windows, iterated local search, planner of travel, evolutionary algorithm