Adam GŁUSZEK, Filip RUDZIŃSKI

ZASTOSOWANIE ALGORYTMÓW EWOLUCYJNYCH W PROBLEMIE MARSZRUTYZACJI Z OKNAMI CZASOWYMI

Streszczenie

Niniejszy artykuł prezentuje wyniki zastosowania wybranych algorytmów ewolucyjnych do problemu marszrutyzacji z oknami czasowymi. Problem marszrutyzacji stanowi zagadnienie należące do zadań optymalizacji kombinatorycznej, a w szerszym zakresie – do badań operacyjnych. Ze względu na jego duże znaczenie praktyczne, zwłaszcza w obszarze zarządzania transportem, wciąż trwają intensywne badania w zakresie poszukiwania nowych i udoskonalania już istniejących algorytmów, umożliwiających jego efektywne rozwiązywanie. W rozdziale pierwszym niniejszego artykułu przedstawiono formalnie zadanie marszrutyzacji z oknami czasowymi. Rozdział drugi prezentuje krótko algorytmy ewolucyjne zastosowane w rozważanym problemie planowania optymalnego zestawu tras dla zespołu pojazdów. Proponowane podejście obejmowało wykorzystanie klasycznego algorytmu genetycznego, strategii ewolucyjnej i algorytmu przeszukiwania rozproszonego. Rozdział trzeci przedstawia zestaw problemów testowych wykorzystywanych w niniejszej pracy oraz wyniki przeprowadzonych eksperymentów numerycznych. Rezultaty działania algorytmów ewolucyjnych porównano dodatkowo z wynikami uzyskanymi przy zastosowaniu zaawansowanego dwufazowego algorytmu heurystycznego, wykorzystującego zmodyfikowany algorytm wspinaczkowy.

WSTĘP

Problem marszrutyzacji (*ang. VRP – Vehicle Routing Problem*) stanowi zagadnienie optymalizacyjne zaliczane do obszaru badań operacyjnych i jest ściśle związany z tematyką zarządzania transportem. Problem decyzyjny polega na wyznaczeniu optymalnych tras obsługi klientów znajdujących się w różnych lokalizacjach, za pomocą określonej liczby środków transportu i z uwzględnieniem narzuconych ograniczeń. Z punktu widzenia teorii optymalizacji jest to rozwinięcie znanego problemu komiwojażera i podobnie jak on zaliczany jest do klasy zagadnień *NP*-trudnych. Chociaż zagadnienie to po raz pierwszy zostało zaprezentowane w 1959 roku [4], do dnia dzisiejszego stanowi duże wyzwanie, ponieważ żaden dokładny algorytm nie gwarantuje jego rozwiązania w rozsądnym czasie, gdy liczba obsługiwanych klientów jest duża. Prawie wszystkie istniejące rozwiązania opierają się na wykorzystaniu metod heurystycznych, obejmując także zastosowanie metod z obszaru sztucznej inteligencji takich jak symulowane wyżarzanie (*ang. simmulated annealing*), algorytmy ewolucyjne (*ang. evolutionary algorithms*), czy przeszukiwanie tabu (*ang. tabu search*). Ze względu na bardzo duże znaczenie problemu marszrutyzacji, zwłaszcza w obszarze zarządzania transportem, dystrybucji i ogólnie pojętej logistyki, ciągle prowadzone są intensywne badania związane z ulepszaniem algorytmów umożliwiających jego skuteczne rozwiązywanie.

Niniejszy artykuł prezentuje wyniki zastosowania wybranych algorytmów ewolucyjnych do standardowego, znanego z literatury [3, 17], zestawu zadań reprezentujących problem marszrutyzacji z oknami czasowymi (*ang. VRPTW – Vehicle Routing Problem with Time Windows*). Zastosowane metody to: klasyczny algorytm genetyczny, strategia ewolucyjna oraz przeszukiwanie rozproszone. Uzyskane wyniki porównano z rezultatami działania zaawansowanego dwufazowego algorytmu heurystycznego, wykorzystującego zmodyfikowany algorytm wspinaczkowy (*ang. LAHC – Late Acceptance Hill Climbing*) [2].

W rozdziale pierwszym przedstawiono formalnie zadanie marszrutyzacji z oknami czasowymi *VRPTW*. Rozdział drugi prezentuje krótko zastosowane algorytmy ewolucyjne, natomiast rozdział trzeci przedstawia zestaw problemów testowych, wykorzystywanych w niniejszej pracy, a także rezultaty przeprowadzonych eksperymentów numerycznych, w tym wyniki porównania skuteczności stosowanych algorytmów.

1. PROBLEM MARSZRUTYZACJI Z OKNAMI CZASOWYMI

Problem marszrutyzacji może być traktowany jako rozwinięcie klasycznego problemu komiwojażera. Zadanie polega na zaplanowaniu optymalnej trasy dla pewnej liczby pojazdów, które mają wyruszyć z punktu startowego (bazy), odwiedzić jednokrotnie wszystkich rozmieszczonych w różnych lokalizacjach klientów (punkty pośrednie), a następnie powrócić do punktu początkowego. Podstawowe warunki ograniczające to konieczność rozpoczynania i kończenia trasy w bazie oraz przypisanie każdemu klientowi tylko jednego pojazdu. Od pierwotnej prezentacji w 1959 roku [4], problem marszrutyzacji doczekał się wielu rozwinięć, zwiększających jego złożoność, ale jednocześnie rozszerzających możliwości jego zastosowania w modelowaniu i optymalizacji rzeczywistych zagadnień z obszaru zarządzania transportem i logistyki. W niniejszym artykule rozważany jest problem marszrutyzacji z oknami czasowymi oraz ograniczeniem pojemności środków transportu.

Problem marszrutyzacji w klasycznej postaci sformułowany jest w postaci grafu nieskierowanego $G=(V,A)$, gdzie $V=\{v_0,v_1,\ldots,v_n\}$ oznacza zbiór n wierzchołków grafu (reprezentujących lokalizację bazy i obsługiwanych klientów), natomiast A to zbiór jego krawędzi z przypisanymi kosztami przejazdu. Mogą to być po prostu odległości, ale również czas, zużyte paliwo lub innego rodzaju koszty przewozu. W procesie optymalizacji minimalizowana jest funkcja kosztu postaci:

$$F=\sum_{r\in R}\sum_{f\in V}\sum_{g\in V} c_{fg}\,x_{fgr} \qquad (1)$$

gdzie r to pojazd należący do zbioru jednakowych pojazdów R, przy czym liczność tego zbioru wynosi k, f oraz g to wierzchołki pomiędzy którymi odbywa się przewóz, c_{fg} to koszt przewozu pomiędzy

wierzchołkami *f* i *g*, natomiast x_{fgr} – zmienna binarna określająca czy pomiędzy wierzchołkami *f* i *g* pojazd *r* wykonuje przewóz. Podstawowe warunki ograniczające to:

- Wierzchołek v_0 stanowi bazę, gdzie każdy pojazd zaczyna i kończy swoją trasę.
- Każdy wierzchołek oprócz v_0 jest odwiedzany dokładnie raz przez dokładnie jeden pojazd.

Rozwinięcia klasycznego problemu marszrutyzacji polegają na wprowadzeniu dodatkowych parametrów i różnorodnych ograniczeń. Rozważany w niniejszej pracy problem marszrutyzacji uwzględnia ograniczenie pojemności pojazdów, polegające na przypisaniu każdemu wierzchołkowi grafu *G* (oprócz v_0) nieujemnej wagi d_f, reprezentującej zapotrzebowanie poszczególnych punktów dostaw/odbioru, a poszczególnym pojazdom maksymalnej pojemności m_r. Zapotrzebowanie zsumowane po całej trasie danego pojazdu nie może przekroczyć jego pojemności, co formalnie wyraża zależność:

$$\forall r \in R \quad \sum_{f \in V} d_f \sum_{g \in V} x_{fgr} \le m_r \tag{2}$$

Drugim dodatkowym ograniczeniem uwzględnianym w ramach niniejszej pracy jest występowanie tzw. okien czasowych. Oznacza to, że dla każdego wierzchołka v_i grafu *G* określony jest przedział czasu $[a_i, b_i]$, w którym punkt ten musi zostać obsłużony. Dodatkowo dla każdego punktu (oprócz bazy) określa się czas obsługi t_i, który pojazd musi poświęcić na zrealizowanie dostawy/odbioru. W przypadku przybycia pojazdu przed rozpoczęciem okna czasowego danego punktu, musi on odpowiednio długo oczekiwać na start obsługi. Przyjazd po czasie b_i jest niedopuszczalny. Formalnie konieczność dostosowania się pojazdów do okien czasowych obsługiwanych klientów można zapisać jako:

$$\forall r \in R, v_i \in V \qquad a_i \le t_{ir} \le b_i \tag{3}$$

gdzie t_{ir} oznacza czas przybycia pojazdu *r* do wierzchołka v_i. W przypadku zadania marszrutyzacji z oknami czasowymi oprócz kosztów przewozu trzeba także znać czasy przejazdu pomiędzy poszczególnymi wierzchołkami grafu *G*. W najprostszym przypadku czas przejazdu można utożsamić z kosztem przewozu, co umożliwia optymalizację problemu mając określone jedynie granice okien czasowych i czasy obsługi dla kolejnych punktów. Właśnie takie podejście zastosowano w eksperymentach numerycznych przeprowadzonych na potrzeby niniejszej pracy.

Istnieje wiele innych rozszerzeń klasycznego problemu marszrutyzacji. Do najczęściej wymienianych w literaturze [17] należą m.in.:

- problemy z ograniczeniem długości trasy poszczególnych pojazdów,
- problemy z niejednakowymi pojazdami (np. o różnych pojemnościach lub jednostkowych kosztach podróży),
- problemy z niesymetrycznym kosztem przewozu pomiędzy wierzchołkami,
- problemy uwzględniające możliwość obsługi jednego klienta przez kilka pojazdów (tzw. dzielenie dostaw),
- problemy z możliwością wyboru kilku baz (*ang. Multiple Depot VRP*),
- problemy ze zdefiniowaną kolejnością odwiedzania poszczególnych klientów,
- problemy uwzględniające możliwości odbioru i wysyłki towarów przez klientów (tzw. problem rozwózkowo-zwózkowy – *ang. VRP with Pick-Up and Delivering*).

Wszystkie te warianty znajdują zastosowanie w modelowaniu różnorodnych systemów dystrybucyjnych, umożliwiając uwzględnienie ich rzeczywistych właściwości. Ważnym zagadnieniem pokrewnym jest tzw. problem *ARP* (*ang. Arc Routing Problem*), gdzie zapotrzebowanie powiązane jest nie z wierzchołkami, lecz z krawędziami grafu *G*. Takie podejście pozwala na rozwiązywanie problemów decyzyjnych związanych np. z odbiorem i dostarczaniem poczty, zbieraniem odpadków, utrzymaniem różnego rodzaju sieci, czy odśnieżaniem ulic.

2. ALGORYTMY EWOLUCYJNE ZASTOSOWANE DO OPTYMALIZACJI PROBLEMU MARSZRUTYZACJI Z OKNAMI CZASOWYMI

Algorytmy ewolucyjne (ang. *evolutionary alghoritms*) stanowią część szerszej dziedziny zwanej obliczeniami ewolucyjnymi (ang. *evolutionary computations*), należącej do obszaru sztucznej inteligencji, a ściślej do jej części zwanej inteligencją obliczeniową (ang. *computational intelligence*) [13]. Techniki z tego obszaru opierają się na obserwacji zasad leżących u podstaw ewolucji biologicznej, takich jak dobór naturalny oraz dziedziczenie, co sprawia, że należą one do wyjątkowo elastycznych i skutecznych algorytmów przeszukiwania przestrzeni rozwiązań, znajdujących zastosowanie w bardzo szerokim zakresie zagadnień, zwłaszcza dotyczących różnorodnych problemów optymalizacji.

Algorytmy ewolucyjne posiadają szereg unikalnych cech, odróżniających je wyraźnie od innych heurystycznych metod optymalizacji, a co ważniejsze zapewniających im dużą uniwersalność i skuteczność. Do cech tego typu zaliczamy:

- przetwarzanie parametrów zadania w postaci zakodowanej, a nie bezpośrednio, co daje możliwość zastosowania w szerokiej klasie problemów,
- prowadzenie poszukiwań przy użyciu zbioru potencjalnych rozwiązań (często zwanego populacją), a nie pojedynczego punktu (ang. *search point*), co zapewnia większe prawdopodobieństwo odnalezienia globalnego maksimum funkcji celu,
- wykorzystywanie jedynie wartości funkcji celu (zwanej zazwyczaj funkcją przystosowania), nie zaś jej pochodnych lub innych dodatkowych dotyczących jej informacji, co powoduje znaczne uproszczenie złożoności obliczeń oraz daje możliwość zastosowania do nieróżniczkowalnych funkcji celu,
- probabilistyczny charakter działania, co daje m.in. możliwość powtórzenia procesu poszukiwania, bez zmiany parametrów algorytmu, w celu znalezienia innych rozwiązań analizowanego problemu.

W szybko rozwijającym się obszarze algorytmów ewolucyjnych pojawiają się wciąż nowe metodologie, ale wśród najważniejszych i najczęściej stosowanych obecnie podejść można wymienić szczególnie:

- algorytmy genetyczne,
- strategie ewolucyjne,
- programowanie ewolucyjne,
- programowanie genetyczne,
- przeszukiwanie rozproszone.

W ramach eksperymentów numerycznych przeprowadzonych na potrzeby niniejszego artykułu wykorzystano klasyczny algorytm genetyczny, strategię ewolucyjną oraz przeszukiwanie rozproszone. Na temat tych metod można znaleźć bardzo wiele informacji w literaturze (np. [1, 5, 7, 10, 11, 13, 14]), zostaną więc one tutaj krótko przedstawione, jedynie dla pokazania najważniejszych występujących pomiędzy nimi różnic.

Działanie algorytmu genetycznego [7, 11, 13] składa się z następujących kroków:

1. Inicjalizacja, czyli wybór początkowego zbioru (populacji) proponowanych rozwiązań (osobników).
2. Ocena jakości (przystosowania) wybranych osobników w populacji.
3. Sprawdzenie warunku zatrzymania.
4. Wybór (selekcja) osobników do następnej populacji.
5. Zastosowanie operatorów genetycznych (krzyżowanie i mutacja).
6. Utworzenie nowej populacji.
7. Zwrócenie jako rezultat najlepszego osobnika.

W pierwszym kroku dokonuje się losowo wyboru określonej liczby osobników do zbioru proponowanych rozwiązań rozważanego zagadnienia. Sposób reprezentacji osobników (najczęściej stanowią one po prostu zestaw optymalizowanych parametrów) jest przy tym wybierany indywidualnie i silnie zależy dla charakteru realizowanego zadania przeszukiwania. Ważne jest aby taka startowa populacja była możliwie różnorodna i w miarę możliwości pokrywała cały obszar prowadzonych poszukiwań. Po tym wstępnym etapie następuje szereg iteracji, zwanych przez analogię do systemów biologicznych generacjami lub pokoleniami, w których wykonywana jest główna pętla algorytmu. Za każdym razem dokonywana jest ocena jakości poszczególnych osobników – im wyższa wartość funkcji przystosowania tym dany osobnik reprezentuje lepsze rozwiązanie. Następnie wybierane są osobniki, będące podstawą do utworzenia następnego pokolenia, przy czym prawdopodobieństwo wyboru jest tym większe im osobnik ma wyższą wartość funkcji przystosowania. Wybrane osobniki, określane jako „rodzice", poddawane są działaniu operatorów genetycznych, co powoduje utworzenie nowej populacji, czyli „dzieci" i algorytm wraca do kroku drugiego. Operatory genetyczne, niezależnie od sposobu realizacji, mają za zadanie umożliwić rekombinację reprezentowanych przez poszczególne osobniki rozwiązań (krzyżowanie) oraz wprowadzić element losowego przeszukiwania przestrzeni rozwiązań i/lub utrzymać różnorodność następujących po sobie populacji (mutacja). Zatrzymanie działania algorytmu następuje po osiągnięciu, z założoną dokładnością, maksymalnej wartości funkcji przystosowania. Algorytm może też być oczywiście zatrzymany po upływie określonego czasu, po wykonaniu odpowiedniej liczby iteracji albo gdy jego działanie w kolejnych pokoleniach nie przynosi znaczącej poprawy uzyskanego dotychczas najlepszego rozwiązania. Jako rezultat działania całego algorytmu wykorzystywany jest osobnik o największej wartości funkcji celu (przystosowania).

Pomimo ogólnego podobieństwa istnieje kilka zasadniczych różnic pomiędzy strategią ewolucyjną a algorytmem genetycznym. Szczególnie istotne są dwie kwestie:

- W strategii ewolucyjnej operatory genetyczne (punkt 5) stosowane są przed procesem selekcji (punkt 4).
- Wybór osobników do następnej populacji (selekcja) następuje w sposób deterministyczny. Osobniki potomne wybierane są na zasadzie rankingu z tymczasowej znacznie większej populacji rodzicielskiej z ewentualnym dodatkowym uwzględnieniem oryginalnej populacji z poprzedniego pokolenia. Tymczasowa populacja rodzicielska powstaje przy tym przez wielokrotne losowanie (z powtórzeniami) spośród osobników stanowiących poprzednie pokolenie. Po selekcji liczebność populacji powraca do wartości pierwotnej.

Dodatkowo, w przypadku strategii ewolucyjnej, inaczej niż w algorytmie genetycznym, dominującym operatorem genetycznym jest mutacja, przy czym bardzo często ma ona charakter samoadaptacyjny tzn. dostosowuje swój zakres do aktualnych postępów procesu poszukiwania rozwiązania optymalnego. Początkowo, w klasycznym ujęciu, mutacja była jedynym operatorem używanym przez strategie ewolucyjne, ale z czasem okazało się, że w wielu przypadkach zastosowanie krzyżowania jest bardzo korzystne i mechanizm ten został trwale „zapożyczony" od innych, korzystających z niego, algorytmów ewolucyjnych.

Przeszukiwanie rozproszone [5, 6, 10, 11] również zaliczane jest do obszaru algorytmów ewolucyjnych, jest to jednak metoda o nieco innym charakterze. Działanie algorytmu opiera się na utworzeniu niewielkiego zbioru możliwie dobrych i zróżnicowanych rozwiązań, tworzących tzw. zbiór bazowy (*ang. reference set*), a następnie doskonaleniu ich na drodze rekombinacji oraz optymalizacji metodami lokalnymi.

Pierwszym etapem działania algorytmu przeszukiwania rozproszonego jest utworzenie startowej populacji rozwiązań, reprezentującej punkty w przestrzeni poszukiwań dla rozważanego problemu optymalizacji. W następnym kroku, rozwiązania z populacji startowej poddawane są wstępnej optymalizacji z wykorzystaniem wybranej metody o charakterze lokalnym (np. którejś z odmian algorytmu wspinaczkowego, symulowanego wyżarzania itp.). Z tak ulepszonej populacji wybierany jest zbiór bazowy, stanowiący podstawę dalszego działania algorytmu. Jego liczność jest niewielka i zazwyczaj nie przekracza 20 elementów. Na tym kończy się pierwsza, wstępna faza działania algorytmu.

W drugiej, iteracyjnej części algorytmu przeszukiwania rozproszonego, zbiór bazowy dzielony jest na podzbiory, stanowiące podstawę do utworzenia drogą rekombinacji ulepszonych rozwiązań, nazywanych potomnymi. Podzbiory te mają jednakową liczbę elementów (najczęściej po dwa) i wybierane są deterministycznie według ustalonego schematu. Rozwiązania w ramach każdego podzbioru podlegają rekombinacji, co stanowi, odpowiednik operacji krzyżowania występującej w innych algorytmach ewolucyjnych. W algorytmie przeszukiwania rozproszonego zamiast tradycyjnej rekombinacji bardzo często używana jest metoda tzw. ścieżek łączących (*ang. path relinking*) [5, 11].

W kolejnym etapie działania algorytmu przeszukiwania rozproszonego, uzyskane w drodze rekombinacji rozwiązania potomne, podlegają optymalizacji lokalnej. Jest to operacja analogiczna do tej, która w fazie wstępnej dotyczy populacji startowej. W następnym kroku, udoskonalone rozwiązania potomne uczestniczą w procesie aktualizacji zbioru bazowego. Jest to bardzo istotny etap, gdyż dla skutecznego działania algorytmu zbiór bazowy musi zachować różnorodność zawartych w nim rozwiązań, nie tracąc jednocześnie najlepszych pod względem jakości ich reprezentantów. Typowym sposobem postępowania jest przyjęcie do aktualizowanego zbioru bazowego pewnej liczby rozwiązań najlepszych (tzw. elitarnych) i uzupełnienie go przypadkami możliwie od nich różnymi, czyli inaczej mówiąc, jak najbardziej odległymi w rozważanej przestrzeni poszukiwań.

Iteracyjna część algorytmu jest wykonywana, aż do osiągnięcia założonego kryterium zatrzymania. W najprostszej wersji algorytm kończy swoje działanie, gdy w szeregu kolejnych iteracji do zbioru bazowego nie trafiają nowe, lepsze od dotychczasowych, rozwiązania. Gdy sytuacja taka następuje zbyt szybko, jest to zazwyczaj niepożądane ze względu na niemożność uzyskania przez algorytm wystarczająco dobrego rozwiązania końcowego. Odpowiedzią na taki problem może być ponowne wygenerowanie populacji startowej i uaktualnienie z jej udziałem zbioru bazowego, a następnie powrót algorytmu do głównej fazy iteracyjnej. Tego typu zabieg zastosowano w badaniach przeprowadzonych na potrzeby niniejszej pracy, dopuszczając do jednokrotnego „restartu" populacji początkowej.

3. OPTYMALIZACJA TRAS POJAZDÓW W PROBLEMIE MARSZRUTYZACJI Z OKNAMI CZASOWYMI ZA POMOCĄ WYBRANYCH ALGORYTMÓW EWOLUCYJNYCH

Wykorzystywany w eksperymentach numerycznych, przeprowadzonych na potrzeby niniejszego opracowania, zestaw dziesięciu zadań marszrutyzacji, stanowi podzbiór znanego z literatury (np. [3, 17]) zbioru problemów testowych, stosowanych często do porównywania skuteczności działania różnych algorytmów optymalizacji problemu marszrutyzacji z oknami czasowymi *VRPTW*. Oryginalne dane można znaleźć w sieci np. pod adresem http://neo.lcc.uma.es/vrp/vrp-instances/capacitated-vrp-with-time-windows-instances/ [17].

Do eksperymentów wybrano dziesięć przykładów obejmujących zadania wymagające obsługi 100 klientów, przy wykorzystaniu pojazdów o jednakowej pojemności. Liczba pojazdów nie była z góry określona i wynikała tylko z minimalizacji sumarycznej długości trwania przebytych przez nie tras. Każdy zbiór danych zawiera współrzędne lokalizacji bazy oraz obsługiwanych klientów. Poszczególni klienci maja przypisane wartości stanowiące zapotrzebowanie na dobra dostarczane przez obsługujące je pojazdy oraz okna czasowe, w których musi nastąpić przybycie pojazdu. Wszystkie punkty mają również zdefiniowany dodatni czas obsługi, w którym pojazd musi pozostać w danym punkcie. W ogólnym przypadku dla zadania marszrutyzacji z oknami czasowymi oprócz kosztów przewozu trzeba także znać czasy przejazdu pomiędzy poszczególnymi, reprezentującymi obsługiwanych klientów, punktami. Najprostszym rozwiązaniem, zastosowanym także w ramach niniejszej pracy, jest przyjęcie, że czas przejazdu jest jednocześnie jego kosztem. Takie podejście umożliwia znacznie prostsze opisanie rozważanych problemów marszrutyzacji.

W ramach badań przeprowadzono optymalizację wybranych problemów testowych z wykorzystaniem klasycznego algorytmu genetycznego, strategii ewolucyjnej, algorytmu przeszukiwania rozproszonego oraz dla porównania dwufazowego algorytmu heurystycznego, wykorzystującego zmodyfikowany algorytm wspinaczkowy [2]. Algorytm genetyczny operował na populacji składającej się ze 100 osobników, a jego działanie trwało przez 1000 pokoleń. Zastosowano, często wykorzystywane dla problemu *VRPTW*, operatory mutacji i krzyżowania opisane w opracowaniu [12]. Prawdopodobieństwo mutacji ustalono na 0,02, a krzyżowanie, ze względu na charakter problemu, przeprowadzane było dla wszystkich osobników. Najlepszy rezultat z puli rodzicielskiej był obowiązkowo włączany do kolejnego pokolenia (tzw. strategia elitarna).

Zastosowana strategia ewolucyjna należała do kategorii $(\mu+\lambda)$, co oznacza że selekcja dokonywana była z połączonej populacji rodzicielskiej μ (liczącej 20 osobników) i potomnej λ (liczącej 100 osobników). Wykorzystano operatory mutacji i krzyżowania identyczne jak dla algorytmu genetycznego. Maksymalna liczba iteracji (pokoleń) strategii ewolucyjnej została ustalona na 2000.

Algorytm przeszukiwania rozproszonego wykonywał dwie iteracje, obejmujące generację populacji startowej (patrz rozdział 2) liczącej 20 rozwiązań. Rozmiar zbioru bazowego ustalono na 10 elementów, z czego połowa stanowiła rozwiązania elitarne, a połowa – dywersyfikujące. Do rekombinacji elementów zbioru bazowego zastosowano mechanizm ścieżek łączących, gdyż przy wykorzystaniu tradycyjnego krzyżowania (według schematu przedstawionego np. w [12]) algorytm osiągał bardzo słabe wyniki.

Przeszukiwanie rozproszone, algorytm genetyczny oraz strategia ewolucyjna zaliczane są do algorytmów ewolucyjnych i w związku z tym mają, w mniejszym bądź większym stopniu, charakter probabilistyczny. Z tego względu eksperymenty dla wszystkich problemów testowych powtórzono po 10 razy, a wyniki zostały uśrednione. Obliczenia zostały wykonane za pomocą pakietu oprogramowania HeuristicLab w wersji 3.3.8, dostępnego na zasadach wolnego oprogramowania (licencja GNU GPL) [15, 16].

Jako alternatywę dla metod ewolucyjnych, zastosowano dwufazowy algorytm optymalizacji o charakterze lokalnym, realizowany za pomocą aplikacji *optaplanner 6.1.0* (dostępnej pod adresem [18] jako wolne oprogramowanie). W pierwszej fazie działania tego algorytmu znajdowane jest heurystycznie rozwiązanie problemu zbliżone do poprawnego. Takie rozwiązanie, spełniające przynajmniej pewną część narzuconych ograniczeń, stanowi punkt startu dla drugiej fazy optymalizacji, obejmującej zastosowanie zmodyfikowanego algorytmu wspinaczkowego z tzw. późną akceptacją (*ang. LAHC – late acceptance hill climbing*) [2]. W zastosowanej konkretnej implementacji obydwie fazy algorytmu mają charakter deterministyczny, w związku z czym dla każdego zadania testowego uruchamiany był on tylko jednokrotnie.

Tabela 1 przedstawia rezultaty uzyskane przez poszczególne metody w odniesieniu do rozważanego zestawu problemów testowych. Rodzaj problemu został oznaczony według konwencji przyjętej we wszystkich opisujących rozważane zagadnienie pracach (patrz np. [3, 12, 17]). Czas działania porównywanych algorytmów wyrażono w sekundach, a obliczenia przeprowadzono z wykorzystaniem komputera wyposażonego w dwurdzeniowy procesor AMD Turion 64X2 2,00 GHz z 2 GB pamięci RAM.

Wyniki zebrane w tabeli 1 pokazują, że dla rozważanego zestawu problemów *VRPTW* algorytmy ewolucyjne zdecydowanie przewyższają heurystyczną metodę opartą o zmodyfikowany algorytm wspinaczkowy. Efekt taki należy zapewne w dużej mierze przypisać globalnemu charakterowi tych metod podczas gdy algo-

Tab.1. *Wyniki działania porównywanych algorytmów dla zestawu testowych problemów VRPTW*

Rodzaj problemu	Algorytm genetyczny			Strategia ewolucyjna			Przeszukiwanie rozproszone			Algorytm wspinaczkowy LAHC		
	liczba poj.	dł. trasy	czas dz.	liczba poj.	dł. trasy	czas dz.	liczba poj.	dł. trasy	czas dz.	liczba poj.	dł. trasy	czas dz.
C102	10	831,20	83,43	10	911,60	184,09	10	828,94	78,03	13	1040,41	240,50
C103	10	828,87	81,44	10	936,86	193,84	10	828,06	71,70	13	1174,55	261,79
C104	10	846,83	71,10	10	898,41	244,43	10	852,21	84,35	11	1059,14	269,47
C202	3	591,55	82,61	3	601,13	121,26	3	591,55	82,06	4	745,00	295,61
C203	3	601,77	79,53	3	599,25	128,21	3	591,17	86,81	4	750,52	233,67
C204	3	599,52	84,79	3	625,93	137,40	3	594,66	73,48	5	874,64	273,31
R108	10,3	999,62	89,00	11	1041,35	252,53	10,7	971,17	90,82	13	1280,58	247,12
R204	4	745,33	126,70	5	774,58	483,62	5	758,65	105,28	4	982,36	149,84
RC104	11	1198,39	82,81	11	1310,27	221,75	10,6	1194,24	84,86	12	1390,20	275,36
RC204	4	808,79	145,25	5	832,64	477,96	5	807,32	92,20	5	994,54	294,71

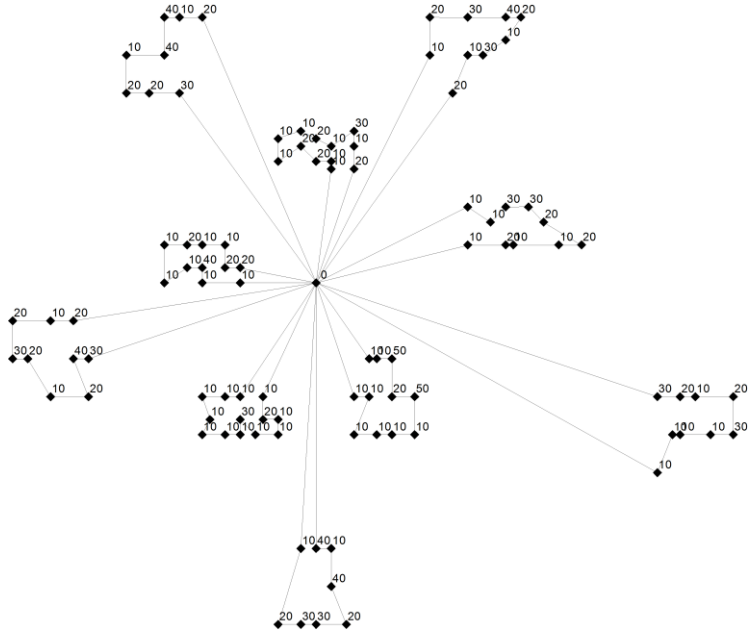
rytm wspinaczkowy, niezależnie od szczegółów realizacji, pozostaje metodą lokalną. Stosowane przez algorytmy ewolucyjne przeszukiwanie za pomocą zbioru (populacji) punktów (ang. *search points*) jest szczególnie wydajne w przypadku problemów o dużej złożoności, do których niewątpliwie zaliczamy *NP*-trudny problem *VRPTW*.

Ogólnie za najskuteczniejszą z porównywanych metod uznać można algorytm przeszukiwania rozproszonego. W siedmiu rozważanych przypadkach uzyskał on najmniejszą średnią długość zaplanowanych tras. Dwukrotnie najlepszy okazał się algorytm genetyczny, a w problemie C202 obydwa algorytmy znalazły jednakowe najlepsze rozwiązanie z trzema pojazdami. Wyniki osiągane przez algorytm genetyczny były jednak tylko nieznacznie gorsze, a ponadto w trzech przykładach algorytm genetyczny znalazł rozwiązanie z mniejszą wymaganą liczbą pojazdów.

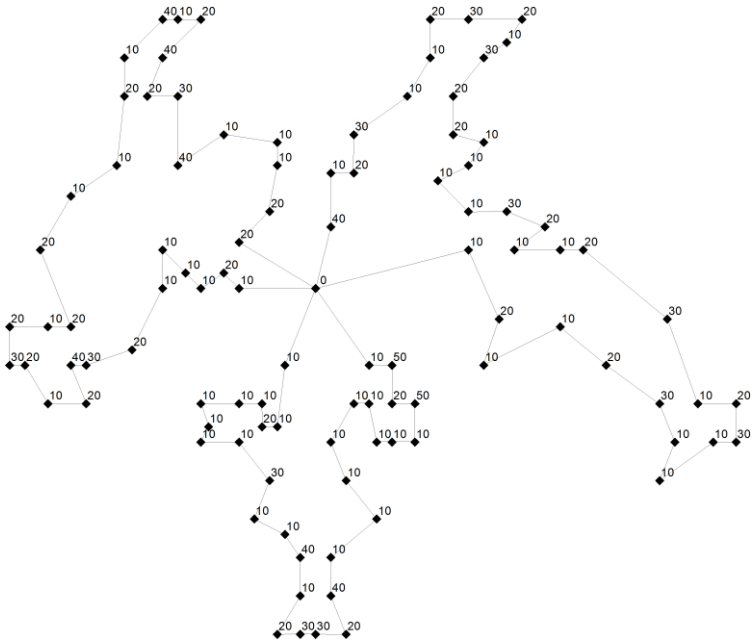
Trzeba przy tym pamiętać, że wszystkie zastosowane podejścia minimalizowały jedynie sumaryczny czas pokonania planowanych tras, mając jako ograniczenia maksymalną dostępną liczbę pojazdów, ich pojemność oraz okna czasowe określone dla poszczególnych klientów, bez uwzględniania czasów oczekiwania w punktach, do których pojazd dotarł przed rozpoczęciem ich okien czasowych oraz czasów przeznaczonych na obsługę kolejnych klientów. W praktyce najczęściej większa liczba pojazdów odpowiada dłużej trwającej trasie całkowitej, stąd w większości przypadków stosowane algorytmy preferowały zestawy tras z mniejszą liczbą użytych pojazdów. Optymalizowanie całkowitego czasu przejazdu skutkowałoby zapewne znalezieniem nieco innych rozwiązań.

Najkrótsze i zbliżone czasy działania uzyskały algorytm genetyczny i przeszukiwanie rozproszone (z lekką przewagą tego drugiego). Strategia ewolucyjna wymagała ponad dwukrotnie większych czasów na obliczenia, ale musiała wykonać aż 2000 iteracji, a algorytm *LAHC* był jeszcze wolniejszy i swoje najlepsze rezultaty osiągał w większości przypadków po ponad trzykrotnie dłuższych czasach działania.

Dla przykładu rysunek 1 prezentuje najlepsze znalezione rozwiązanie z dziesięcioma pojazdami dla problemu C102 o sumarycznej długości trwania trasy równej 828,94. Rozwiązanie to było bez-



Rys.1. *Najlepsze rozwiązanie problemu C102 uzyskane przez algorytm przeszukiwania rozproszonego (10 pojazdów; suma długości trwania tras 828,94)*



Rys.2. *Najlepsze rozwiązanie problemu C202 uzyskane przez algorytm genetyczny i przeszukiwanie rozproszone (3 pojazdy; suma długości trwania tras 591,55)*

błędnie znajdywane przez algorytm przeszukiwania rozproszonego we wszystkich przeprowadzonych próbach. Punkty reprezentują obsługiwanych przez poszczególne pojazdy klientów, a opisujące je liczby to wartości zgłaszanego w nich zapotrzebowania. Wszystkie pojazdy były jednakowe i miały maksymalną pojemność równą 200.

Rysunek 2 prezentuje najlepsze znalezione rozwiązanie z trzema pojazdami dla problemu C202 o sumarycznej długości trwania trasy równej 591,55. Rozwiązanie to było bezbłędnie znajdywane przez algorytm genetyczny i przeszukiwanie rozproszone we wszystkich przeprowadzonych próbach. Wszystkie pojazdy były jednakowe i miały maksymalną pojemność równą 700.

PODSUMOWANIE

Niniejszy artykuł prezentuje wyniki zastosowania wybranych algorytmów ewolucyjnych do problemu marszrutyzacji z oknami czasowymi. Zastosowane metody to: klasyczny algorytm genetyczny, strategia ewolucyjna oraz przeszukiwanie rozproszone. Uzyskane wyniki porównano z rezultatami działania zaawansowanego dwufazowego algorytmu heurystycznego, wykorzystującego zmodyfikowany algorytm wspinaczkowy. Problem marszrutyzacji stanowi zagadnienie należące do zadań optymalizacji dyskretnej o dużym znaczeniu praktycznym, zwłaszcza w obszarze zarządzania transportem, stąd wciąż istnieje duże zainteresowanie algorytmami umożliwiającymi jego efektywne rozwiązywanie.

W rozdziale pierwszym przedstawiono formalnie zadanie marszrutyzacji z oknami czasowymi *VRPTW*. Rozdział drugi prezentuje krótko zastosowane algorytmy ewolucyjne, natomiast rozdział trzeci przedstawia zestaw problemów testowych, wykorzystywanych w niniejszej pracy, a także rezultaty przeprowadzonych eksperymentów numerycznych, w tym wyniki porównania skuteczności stosowanych algorytmów.

Otrzymane wyniki pokazują, że dla rozważanego zestawu problemów, algorytm przeszukiwania rozproszonego okazał się najskuteczniejszy ze wszystkich porównywanych metod. W większości przypadków osiągnął on najlepsze rezultaty optymalizacji, charakteryzując się przy tym najkrótszym czasem działania. Klasyczny algorytm genetyczny ustępował mu tylko nieznacznie, zarówno w zakresie uzyskanych rezultatów, jak i czasów działania. Pozostałe podejścia wymagały znacznie dłuższych czasów obliczeń, a zaplanowane z ich użyciem trasy były przeważnie dużo dłuższe.

W ramach dalszych prac przewiduje się wykorzystanie do problemu marszrutyzacji innych narzędzi z obszaru inteligencji obliczeniowej (*ang. computational intelligence*), zwłaszcza związanych z systemami rozmytymi i neuronowo-rozmytymi [8, 9].

BIBLIOGRAFIA

1. Beyer H.-G., *The Theory of Evolution Strategies*. Springer, Berlin, 2001.
2. Burke E.K., Bykov Y., *The Late Acceptance Hill-Climbing Heuristic*. Technical Report CSM-192, Department of Computing Science and Mathematics, University of Stirling, 2012.
3. Cordeau J.-F., Desaulniers G., Desrosiers, J., Solomon, M.M., Soumis, F., *VRP with Time Windows*. W: Toth P. and Vigo D. (eds.), *The Vehicle Routing Problem*, SIAM Monographs on Discrete Mathematics and Applications, vol. 9, Philadelphia, PA, str. 157-193, 2002.
4. Dantzig G.B., Ramser J. H., *The Truck Dispatching Problem*. Management Science, Vol. 6, str. 80-91, 1959.
5. Glover F., Laguna M., *Fundamentals of Scatter Search and Path Relinking*. Control and Cybernetics, Vol. 29, Nr 3, str. 653-684, 2000.
6. Głuszek A., Rudziński F., *Zastosowanie algorytmu przeszukiwania rozproszonego do problemu marszrutyzacji z ograniczeniem pojemności środków transportu*, Logistyka, str. 3417-3425, nr 4/2015.
7. Goldberg D.E., *Algorytmy genetyczne i ich zastosowania*. WNT, Warszawa, 1998.
8. Gorzałczany M.B., *On some idea of a neuro-fuzzy controller*. Information Sciences, nr 120, str. 69-87, 1999.
9. Gorzałczany M.B., Piasta Z., *Neuro-fuzzy approach versus rough-set inspired methodology for intelligent decision support*. Information Sciences, nr 120, str. 45-68, 1999.
10. Marti R., Laguna M., Glover F., *Principles of scatter search*. European Journal of Operational Research, Tom 169, North-Holland, str. 359-372, 2006.
11. Talbi E., *Metaheuristics: from design to implementation*. John Wiley & Sons, Hoboken, NJ, 2009.
12. Potvin J.-Y. Bengio S., *The Vehicle Routing Problem with Time Windows - Part II: Genetic Search*. INFORMS Journal of Computing, Vol. 8, str.165-172, 1996.
13. Rutkowski L., *Metody i techniki sztucznej inteligencji*. PWN, Warszawa, 2005.
14. Schwefel H.-P.: Evolution and Optimum Seeking. J.Wiley&Sons, 1995.
15. Wagner S. i in., *Architecture and Design of the HeuristicLab Optimization Environment*. W: Advanced Methods and Applications in Computational Intelligence, Topics in Intelligent Engineering and Informatics Series, Springer, str. 197-261, 2014.
16. http://dev.heuristiclab.com/
17. http://neo.lcc.uma.es/vrp/
18. http://www.optaplanner.org/

APPLICATION OF EVOLUTIONARY ALGORITHMS IN VEHICLE ROUTING PROBLEM WITH TIME WINDOWS

Abstract

The paper presents application of evolutionary algorithms to capacitated vehicle routing problem with time windows. Vehicle routing problem is an important combinatorial optimization task and it is related to operations research. It has great practical relevance, especially in the fields of transport management, distribution and logistics. Development of the algorithms for efficient solving of the vehicle routing problem is still very intensive. In the first section of the paper, capacitated vehicle routing problem with time windows is formally presented. Next section describes in outline evolutionary algorithms applied to considered problem of designing the optimal set of routes for team of vehicles. In our approach we use genetic algorithm, evolution strategy and scatter search algorithm. The third section presents a set of test examples, used in this work and the results of performed numerical experiments. The comparison of results obtained by evolutionary algorithms and advanced two-phase heuristic method, based on modified hill climbing algorithm, is also provided.

Autorzy:
dr inż. **Adam Głuszek** – Politechnika Świętokrzyska, Wydział Elektrotechniki, Automatyki i Informatyki, Al. 1000-lecia P.P. 7, 25-314 Kielce. Tel: +48 41 34-24-190,E-mail: a.gluszek@tu.kielce.pl
dr inż. **Filip Rudziński** – Politechnika Świętokrzyska, Wydział Elektrotechniki, Automatyki i Informatyki, Al. 1000-lecia P.P. 7, 25-314 Kielce. Tel: +48 41 34-24-190,E-mail: f.rudzinski@tu.kielce.pl