ALGORYTM RÓWNOLEGŁY DLA PROBLEMU MARSZRUTYZACJI

Szymon JAGIEŁŁO, Dominik ŻELAZNY

Streszczenie: Transport odgrywa znaczącą rolę zarówno w produkcji jak i usługach przemysłowych. W dzisiejszych czasach, znaczenie dostarczania dóbr na czas do magazynów, jednostek produkcji oraz, ostatecznie, do klientów nie umknęło kompaniom logistycznym. Modele problemu marszrutyzacji (*Vehicle Routin Problem* – VRP) ewoluowały do bardziej zaawansowanych form, co z kolei doprowadziło do zwiększenia złożoności obliczeniowej podczas ewaluacji takowych. Szczególnie w przypadku metod wielokryterialnych, optymalizacja tras transportowych dla firm oznacza wykonywanie skomplikowanych obliczeń w jak najkrótszym możliwym czasie. Karty graficzne (*Graphics Processing Units* – GPU), będące wydajnymi równoległym koprocesorami, dostarczają obecnie ogromnej mocy obliczeniowej, jeśli operacje zostały poprawnie zaprogramowane jako działające równolegle. Obecnie GPU wyposażone są w łatwiejsze do programowania architektury jak CUDA (*Compute Unified Device Architecture*) czy OpenCL, więc stosowanie algorytmów równoległych nie jest ograniczone wyłącznie do stacji roboczych i wyspecjalizowanych komputerów. Praca to ma na celu wykazanie zysku płynącego z zastosowania równoległego algorytmu przeszukiwania z zabronieniami (*Tabu Search* – TS) oraz jego przewagi nad sekwencyjnym algorytmem TS. Zastosowany algorytm posłużył do rozwiązania problemu marszrutyzacji z ograniczoną długością trasy (*Distance-constrained Vehicle Routing Problem*) w ujęciu wielokryterialnym.

Słowa kluczowe: optymalizacja dyskretna, obliczenia równoległe, OpenCL, Tabu Search.

1. Wprowadzenie

Problem marszrutyzacji (VRP), będący rozwinięciem klasycznego problemu komiwojażera (*Traveling Salesman Problem* - TSP), jest problemem optymalizacji dyskretnej. Przekłada się na przypisanie pewnej liczby tras przy jednoczesnym zachowaniu ograniczeń wynikających z modelu. Pierwszy model zaproponowany został na przełomie lat pięćdziesiątych i sześćdziesiątych ubiegłego wieku, kiedy Dantzig i Ramser [1] opracowali matematyczną formułę problemu i algorytm dostarczania paliwa do stacji benzynowych. Od tamtej pory problem marszrutyzacji ewoluował i jest obecnie jednym z najczęściej badanych problemów optymalizacji dyskretnej. Z definicji, problem składa się z m liczby pojazdów oczekujących w punkcie startowym (magazynie lub zajezdni) oraz pewnej dyskretnej liczby dobór oczekujących na dostarczenie do n liczby klientów, z których każdy odwiedzany jest dokładnie raz. Określenie optymalnych tras definiuje problem marszrutyzacji. Głównym celem oryginalnego problemu była minimalizacja długości tras pojazdów dostawczych. Rozwiązaniem klasycznego problemu VRP jest zbiór dozwolonych, spełniających ograniczenia, tras zaczynających się i kończących w punkcie startowym. Pierwotnie, długość trasy przekładana była na koszt transportu. Zastosowania rzeczywiste zwykle wymagają bardziej rozbudowanych modeli, bazujących na dodatkowych wymaganiach i ograniczeniach, takich jak pojemność pojazdów, maksymalna

długość trasy lub pracy kierowcy, okna czasowe dostawy i wiele innych, które powstały na przestrzeni lat w związku z zapotrzebowaniem przemysłu. Między innymi powstały modele odbioru i dostarczania (*pickup and delivery*), które uwzględniają dostawę oraz pobieranie towarów od klientów (np. kurier dostarczający paczki i przyjmujący wysyłki w trasie). Ważna jest ładowność (pojemność) pojazdów oraz to jak zmienia się załadunek w trakcie pokonywania kolejnych, sąsiadujących, węzłów trasy (zmniejsza przy dostarczaniu, zwiększa przy odbiorze). Przez ponad połowę stulecia modele ewoluowały w relacji do zapotrzebowania przemysłu oraz spółek logistycznych, w związku z czym zostało sformułowanych wiele różnych typów problemów VRP, jak problemy z wieloma punktami startowymi (magazynami), periodyczne problemy marszrutyzacji, problemy z dzielonymi dostawami, zwrotami i wiele innych.

Problem marszrutyzacji uznawany jest za NP-trudny problem optymalizacji kombinatorycznej. Znalezienie rozwiązania optymalnego możliwe jest wyłącznie dla instancji o małych rozmiarach [2]. Algorytmy heurystyczne nie gwarantują znalezienia rozwiązania optymalnego, jednak w praktyce są najczęściej używane, ze względu na ich wysoką wydajność. Poniższa praca opisuje równoległy algorytm poszukiwania za zabronieniami (TS), który pozwala na znalezienie dobrych rozwiązań w pewnym, skończonym, czasie obliczeniowym.

2. Przegląd literatury dotyczącej problemów VRP i optymalizacji wielokryterialnej

Literatury na temat wielokryterialnej optymalizacji jest pod dostatkiem, aczkolwiek wielokryterialne problemy transportowe nie doczekały się tak wielu pozycji. Szczególnie w porównani do liczby prac na temat jednokryterialnych problemów tego typu. Prace dotyczące wielokryterialnych problemów VRP bazują najczęściej na algorytmach ewolucyjnych, podczas gdy tylko niektóre zajmują się metodami przeszukiwania lokalnego, takimi jak symulowane wyżarzanie (*Simulated Annealing* – SA) czy wspomniane wcześniej przeszukiwanie z zabronieniami (TS).

2.1. Algorytmy wielokryterialne

Powstało wiele podejść do optymalizacji wielokryterialnej. Jednym z pierwszych użytych było metakryterium, zarówno jako suma znormalizowanych wartości funkcji kryterialnych czy też jako ważona suma kryteriów. Niektórzy badacze próbowali posługiwać się odległością od punktu idealnego, jako wskaźnikiem jakości rozwiązania wielokryterialnego. Inni proponowali wybranie jednego, głównego, kryterium i przekształcenie pozostałych do ograniczeń (metoda ε-ograniczeń). Próbowano również zastosować metodę leksykograficzną, w której tworzona jest hierarchia kryteriów i jedno rozwiązanie jest uważane za lepsze od drugiego, jeśli wcześniej (w hierarchii kryteriów) ma lepszą wartość funkcji kryterialnej.

Algorytm VEGA (*Vector Evaluated Genetic Algorithm*) [3] był jednym z pierwszych zaproponowanych podejść do optymalizacji wielokryterialnej. Główną ideą był podział populacji na k subpopulacji, gdzie k – liczba kryteriów, oraz poszukiwanie rozwiązań najlepszych dla jednego, z góry ustalonego, kryterium w każdej subpopulacji podczas działania algorytmu. Ze względu na specyfikę tego podejścia, w trakcie działania algorytmu pomijane są często rozwiązania o wynika nieco gorszych dla danego kryterium w subpopulacji, ale o wiele bardziej zadowalających dla pozostałych kryteriów. Takie pośrednie rozwiązania ulegały eliminacji podczas operacji selekcji, mimo iż były ogólnie

lepsze od rozwiązań dla najlepszych wartości wybranych funkcji kryterialnych.

Algorytm SPEA (*Strength Pareto Evolutionary Algorithm*) jest elitarnym wielokryterialnym algorytmem ewolucyjnym, w którym podjęto koncepcję niezdominowanych rozwiązań. Zitzler i Thiele [4] zaproponowali utrzymywanie zewnętrznej populacji i zachowywanie, w każdej iteracji, wszystkich znalezionych rozwiązań niezdominowanych. Populacja ta bierze udział we wszystkich operacjach genetycznych. Wszystkie rozwiązania niezdominowane mają przypisaną wartość dopasowania zgodną z liczbą rozwiązań przez nie zdominowanych, podczas gdy rozwiązaniom zdominowanym przypisywana jest wartość dopasowania gorsza niż najgorsza z wartości rozwiązań niezdominowanych, tak że poszukiwanie ukierunkowane jest na rozwiązania Pareto optymalne. Dodatkowo, by utrzymać różnorodność pomiędzy rozwiązaniami niezdominowanymi, zastosowana jest technika klasteryzacji rozwiązań.

W przypadku algorytmu PAES (*Pareto-archived Evolutionary Strategy*) [5], potomkowie porównywani są z rodzicami. Jeśli potomek dominuje rozwiązanie pierwotne, to zostaje ono zastąpione przez tego potomka. Jeśli potomek jest zdominowany przez rodzica, wtedy zostaje odrzucony i wygenerowany zostaje nowy potomek. Jednakże, jeśli potomek i rodzic nie dominują się nawzajem, rozwiązanie potomne porównywane jest ze zarchiwizowanymi rozwiązaniami zdominowanymi, w celu sprawdzenia czy nie dominuje któregoś z nich. W przypadku gdy tak się dzieje, potomek obejmuje rolę rodzica w kolejnej iteracji, a rozwiązania zdominowane zostają usunięte z archiwum. W przeciwnym wypadku potomek i rodzic zostają sprawdzeni pod względem bliskości do rozwiązań z archiwum i osobnik znajdujący się w mniej zatłoczonym regionie przestrzeni kryteriów jest akceptowany jako rodzic i dodany do archiwum.

W swojej pracy, Rudolph [6] zasugerował prostą elitarną wielokryterialną strategię ewolucyjną opartą na systematycznym porównaniu osobników z populacji rodziców i potomków. Niezdominowane rozwiązania z obu populacji są porównane, by uformować nowy zbiór rozwiązań niezdominowanych, który staje się populacją rodziców w kolejnej iteracji. Jeśli rozmiar populacji jest mniejszy niż pożądany, wtedy inne rozwiązania zostają włączone do wyżej wymienionej. Niestety, w algorytmie tym brak mechanizmu utrzymywania różnorodności rozwiązań Pareto optymalnych.

Deb i inni [7] zaproponowali algorytm NSGA-II (*Elitist Non-dominated Sorting Genetic Algorithm*), będący ulepszoną wersję algorytmu NSGA, krytykowanego za wysoką złożoność obliczeniową sortowania rozwiązań niezdominowanych, potrzebę określenia parametru udziału i brak elitarność. Nowy algorytm pozbył się wyżej wymienionych dolegliwości poprzez dodanie metody szybkiego sortowania rozwiązań Pareto, estymator gęstości i operator porównania zatłoczenia. Pozwoliło to na zmniejszenie złożoności obliczeniowej i sterowanie procesem selekcji w kierunku jednorodnie rozłożonej aproksymacji frontu Pareto. Badania wskazują, że elitarność pozwala na przyśpieszenie procesu znajdywania rozwiązań niezdominowanych, a co za tym idzie przyśpiesza działanie algorytmu w znaczący sposób. Pozwala również zachować dobre rozwiązania, gdy już zostały one znalezione.

2.2. Wielokryterialna optymalizacja w transporcie

Jak wspomniano wcześnie, przez ostatnie lata zaprezentowano wiele technik dla problemów optymalizacji wielokryterialnej, w tym dyskretnej. Składają się one z metod skalarnych (np. metakryterium), metod Pareto (np. algorytmy NSGA-II lub PAES) oraz takie, których nie można zakwalifikować do żadnej z powyższych (np. VEGA). Niektórzy

badacze podchodzili do problemu wielokryterialnego transportu. Kilka z takowych podejść będzie tutaj wymienione.

Jedną z obiecujących technik skalarnych zaproponował Bowerman i inni [8]. Korzysta ona z pięciu różnych zestawów wag, wybranych przez decydenta (*decision maker*). Ich heurystyka wpierw grupuje węzły w klastry, które mogą być obsłużone przez pojazd, a następnie określa marszrutę dla każdego z klastrów. Jest to strategia *allocation-routing-location*. Panowie Lee i Ueng zaproponowali w pracy [9] algorytm oparty o technikę wstawień. W każdej iteracji dodaje on jeden węzeł do pojazdu z najkrótszą trasą używając do tego kryterium zachowywania. Kolejna heurystyka oparta o wstawienia została zaproponowana przez Zografosa i Androutsopoulosa w pracy [10], chociaż jej idea wywodzi się z metody zaproponowanej przez Solomona. Różni się ona w selekcji węzłów do wstawienia, pozwalając na wstawienie zarówno już uszeregowanym jak i nie uszeregowanym klientom. Kolejne z podejść skalarnych posługuje się metodą ε–ograniczeń. W tej strategii, tylko jedno z kryteriów jest optymalizowane, podczas gdy pozostałe traktowane są jako ograniczenia, wyrażone jako $f_i \leq \varepsilon_i$. W pracy [11] Pacheco i Marti optymalizują kryterium najdłuższej ścieżki dla każdej możliwej wartości drugiego kryterium, a następnie używają algorytmu przeszukiwania z zabronieniami do rozwiązania każdego z uzyskanych problemów. Podobnej strategii użył Corberan i inni [12], ale w miejsce algorytmu TS użyli oni podejścia rozproszonego poszukiwania.

W wielokryterialnych problemach marszrutyzacji, koncepcja Pareto optymalności jest najczęściej używana w zestawieniu z rodziną algorytmów ewolucyjnych. Jedna z prac korzysta z algorytmu zaproponowanego przez Żelaznego w [13]. Użyty w niej algorytm genetyczny w każdej iteracji posługuje się metodą poszukiwania lokalnego na niezdominowanych rozwiązaniach uzyskanych w ramach operacji genetycznych, w celu jeszcze lepszej aproksymacji frontu Pareto. Pojęcie dominacji zostało również użyte przez Ulungu i innych w technice symulowanego wyżarzania zwanej MOSA (*Multi-Objective Simulated Annealing*) [14]. Również Paquete i inni [15] odwołali się do technik przeszukiwania lokalnego Pareto (*Pareto Local Search*). Bazują one na zasadzie, iż obecne rozwiązanie wybierane jest z niezdominowanych rozwiązań w sąsiedztwie.

Niektóre badania wykorzystują do rozwiązywania problemów wielokryterialnej marszrutyzacji metody, które nie są ani skalarne ani oparte o Pareto optymalność. Te nieskalarne i nie-Pareto metody bazują między innymi na strategiach leksykograficznych oraz specyficznych heurystykach. Wcześniej wspomniany algorytm VEGA jest jedną z takich specyficznych heurystyk. Natomiast strategia leksykograficzna użyta była w pracach Kellera i Goodchilda [16] [17]. Ich podeście dotyczyło przypisania kryteriom pewnych wartości priorytetów, a problem rozwiązywano w kolejności malejących priorytetów. Kiedy jedno z kryteriów zostało już zoptymalizowane, jego wartość nie mogła być zmieniona i pozostawała uwzględniana jako nowe ograniczenie problemu.

3. Opis problemu

Typowo, problem VRP opisuje się następująco. Dana jest flota pojazdów (zazwyczaj identycznych) $V = \{1, \ldots, v\}$, zbiór klientów/lokacji reprezentowanych przez węzły $N = \{1, \ldots, n\}$, węzeł startowy zwany magazynem (nazywany typowo węzłem zerowym) oraz sieć połączeń pomiędzy magazynem a klientami/lokacjami. Dla każdej pary węzłów (i, j), gdzie $i, j \in N$ oraz $i \neq j$, przypisana jest ścieżka długości $d_{i,j}$ lub czas przejazdu $t_{i,j}$. W przypadku symetrycznych problemów marszrutyzacji (graf nieskierowany) długość ścieżki (czas przejazdu) są identyczne niezależnie od kierunku jazdy, więc $d_{i,j} = d_{j,i}$

(odpowiednio $t_{i,j} = t_{j,i}$) [18]. Kolizje, drogi jednokierunkowe lub różne trasy między węzłami, zależnie od kierunku ruchu, mogą doprowadzić do załamania tej symetrii. W tym wypadku rozważana jest nowa klasa problemów, asymetryczne VRP (graf skierowany), w której pomiędzy dwoma węzłami mogą występować np. ścieżki jednokierunkowe lub różne czasy podróży (wynikające z długości lub charakterystyki dróg) [19].

Poniższa praca opisuje problem DVRP (*Disctance-constrained* VRP), w przypadku którego każdy z kierowców (pojazdów) jest ograniczony maksymalną długością trasy, którą może przebyć. Oznacza to, że każdy pojazd posiada maksymalną trasę a_k, a dozwolone rozwiązanie to takie, w którym żaden z pojazdów we flocie nie przekracza swojej maksymalnej trasy a_k. W przypadku floty homogenicznej wszystkie pojazdy posiadają taką samą maksymalną trasę $a_k = a$, dla $k \in V$.

3.1. Kryteria optymalizacji i ograniczenia modelu

W naszej pracy posłużyliśmy się dwoma kryteriami: a) średnią długością trasy oraz b) maksymalną trasą dla pojazdu. Oparliśmy naszą pracę o wcześniej opisany model VRP, więcej ograniczenie maksymalnej długości trasy zostało dodane do każdego z pojazdów.

Kryterium maksymalnej długości trasy dla pojazdu reprezentuje wzór:

$$\max_{1\le k\le m} \sum_{i=0}^{n} \sum_{j=0}^{n} d_{ij} x_{ij}^{k} , \tag{1}$$

natomiast drugie z kryteriów, średnia długość trasy, przedstawia się następująco:

$$\frac{1}{m} \sum_{i=0}^{n} \sum_{j=0}^{n} \sum_{k=1}^{m} d_{ij} x_{ij}^{k} . \tag{2}$$

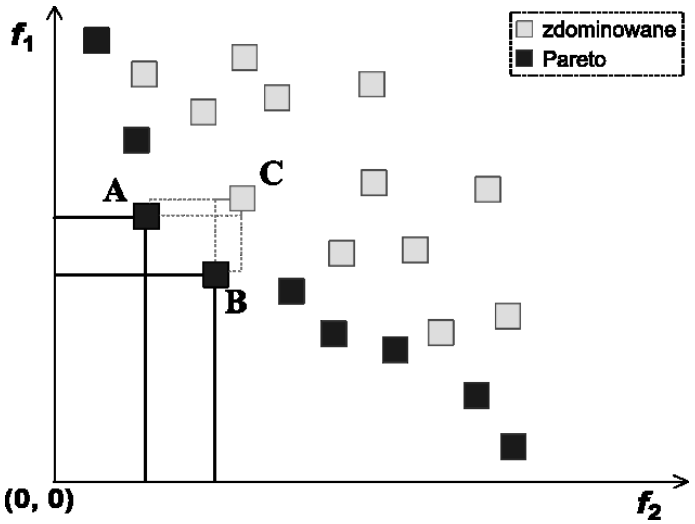
gdzie: x_{ij}^{k} - jest równe 1, jeśli j-ty węzeł następuje na trasie po i-tym, 0 w przeciwnym wypadku;

d_{ij} - długość ścieżki z i-tego do j-tego węzła.

Jako że nasz algorytm pracuje na dynamicznej liczbie pojazdów (zmienia się w trakcie przeszukiwania sąsiedztwa, można uznać za trzecie kryterium oceny), podjęliśmy się jednoczesnej minimalizacji średniej długości trasy oraz maksymalnej długości dla pojazdu. W ten sposób, rozwiązania powinny dostarczyć stabilnego rozkładu pracy dla wszystkich kierowców, szczególnie jeśli średnia długość trasy zbliżona będzie do maksymalnej długości dla pojazdu.

3.2. Pareto-optymalność

Rozwiązaniem problemu wielokryterialnego jest zbiór rozwiązań niezdominowanych zwany frontem Pareto, gdzie dominacja zdefiniowana jest jak następuje. Rozwiązanie $y = (y_1, y_2, ..., y_n)$ dominuje (w przypadku minimalizacji wszystkich kryteriów) rozwiązanie $z = (z_1, z_2, ..., z_n)$ wtedy i tylko wtedy, gdy $\forall_i \in \{1, ..., n\}$, $y_i \le z_i$ oraz $\exists_i \in \{1, ..., n\}, y_i \le z_i$.



Rys. 1. Rozwiązania Pareto oraz zdominowane

Rysunek 1 przedstawia sytuacje w której rozwiązania *A* i *B* dominują rozwiązanie *C*, aczkolwiek nie zachodzi (w żadną stronę) dominacja między powyższymi punktami i są one częścią aproksymacji frontu Pareto. Jest to klasyczny przykład dla minimalizacji dwóch funkcji kryterialnych.

4. Zastosowany algorytm

W naszej pracy zdecydowaliśmy się na skorzystanie ze zmodyfikowanego algorytmu przeszukiwania z zabronieniami, który dodatkowo wyposażyliśmy w zewnętrzne archiwum rozwiązań niezdominowanych. Nasz równoległy Pareto zarchiwizowany algorytm TS (*Parallel Pareto Archived Tabu Search* - PPATS) został zaprojektowany do przeszukiwania sąsiedztwa rozwiązań Pareto optymalnych. Jako że optymalizacja wielokryterialna z podejściem Pareto optymalności oznacza znajdywanie aproksymacji frontu Pareto, użycie metody równoległego przeszukiwania lokalnego (sąsiedztwa) pozwoliło nam na zwiększenie dokładności obszaru przeszukiwania, co było kłopotliwe w algorytmach sekwencyjnych, ze względu na dużą złożoność obliczeniową takowego.

Algorytm przeszukiwania z zabronieniami, zaproponowany przez Glovera [20], jest metaheurystycznym algorytmem przeszukiwania lokalnego używanym do rozwiązywania problemów optymalizacji kombinatorycznej, takich jak szeregowanie zadań produkcyjnych lub zagadnienia transportowe, w naszym wypadku problem marszrutyzacji. Korzysta on z procedury przeszukiwania sąsiedztwa, aby przejść od potencjalnego rozwiązania x do ulepszonego rozwiązania x', znajdującego się w sąsiedztwie rozwiązania x. W celu przeszukania przestrzeni rozwiązań, która zostałaby niezbadana przez inne procedury lokalnego przeszukiwania. Algorytm TS ostrożnie eksploruje sąsiedztwo każdego z rozwiązań podczas procesu przeszukiwania. Rozwiązania przyjęte do nowego sąsiedztwa, $N(x)$, są określone poprzez użycie listy tabu, tzn. zbioru reguł oraz zabronionych rozwiązań używanego do oceny, które z rozwiązań zostanie dodane do sąsiedztwa $N(x)$ i zbadane w procesie przeszukiwania. Użyliśmy najprostszej formy listy tabu, tzn.

krótkoterminowego zbioru rozwiązań już odwiedzonych w niedalekiej przeszłości i omijanych w trakcie dalszej eksploracji.

W każdej iteracji, algorytm wykonuje pewną określoną liczbę równoległych operacji przeszukania sąsiedztw z różnymi punktami startowymi, które wybrane zostały z niezdominowanych rozwiązań znalezionych przez algorytm w poprzedniej iteracji. Jednocześnie, algorytm rozpoczyna pierwszą iterację z jednym, wybranym losowo, punktem startowym. Podczas przeszukiwania, rozwiązania porównywane są ze swoimi poprzednikami. Jeśli nowe rozwiązanie z sąsiedztwa dominuje stare, wtedy zajmuje jego miejsce w kolejnej iteracji przeszukiwania. Jeśli natomiast jest zdominowane przez poprzednika, zostaje wtedy odrzucone i przeszukiwanie kontynuowane jest w sąsiedztwie niezmienionego poprzednika. W przeciwnym wypadku, gdy żadne z rozwiązań nie dominuje pozostałego, nowe rozwiązanie dodane zostaje do obecnie utworzonej listy Pareto, a w kolejnej iteracji do przeszukiwania sąsiedztwa przypisane zostaje rozwiązanie o mniejszej znormalizowanej sumie wartości funkcji kryterialnych. Po każdej iteracji głównej algorytmu równoległego, uzyskane w trakcie przeszukiwania sąsiedztwa rozwiązania zostają sprawdzone przez funkcję selekcji, która wyłania z nich wyłącznie rozwiązania niezdominowane, a następnie porównane z zarchiwizowaną aproksymacją frontu Pareto oraz, jeśli nie duplikuj już istniejących rozwiązań, dodane do archiwum. Dodatkowo, jeśli nowe rozwiązania dominują któreś z już istniejących w archiwum, usuwamy zdominowane rozwiązania z naszego zbioru zewnętrznego, tak że zawiera wyłącznie rozwiązania niezdominowane. Niepowtarzalne rozwiązania Pareto z obecnej iteracji zostają wykorzystane jako rozwiązania początkowe w kolejnej iteracji równolegle uruchomionych funkcji przeszukiwania sąsiedztw. W ten sposób, liczba punktów startowych na rozwiązanie (w przeszukiwaniu sąsiedztwa) uzależniona jest od liczby w/w unikalnych rozwiązań z poprzedniej iteracji algorytmu. Co więcej, jako że początkowe rozwiązanie algorytmu PPATS jest wybierane losowo, sam algorytm wykonuje pewną liczbę wstępnych kroków adaptacyjnych, podczas których wartość ograniczenia maksymalnej trasy pojazdu jest znacznie większa niż ustalona i w trakcie działania funkcji adaptacji jest zmniejszana z każdą iteracją, aż osiągnie pożądany poziom, ustalony jako ograniczenie dla instancji. W takcie działania w/w funkcji, tylko rozwiązania niezdominowane spełniające ustalone dla instancji ograniczenia są dodawane do zewnętrznego archiwum Pareto.

5. Wyniki badań

W celu zestawienia zaimplementowanych algorytmów TS i PPATS, postanowiliśmy użyć dwóch typów porównań. Pierwsze bierze pod uwagę szybkość działania algorytmu oraz liczbę przeszukanych w czasie działania sąsiedztw, natomiast drugie porównuje wyniki uzyskane z pomocą obu powyższych algorytmów.

Czasy wykonywania oraz liczbę przeszukanych sąsiedztw zaprezentowano w Tab. 1. Jako że algorytm TS pracuje sekwencyjnie, został ograniczony do przeszukiwania 100 sąsiedztw na uruchomienie. Jak widać, już dla instancji o 40 węzłach czas wykonania (mimo ograniczenia liczby przeszukań) algorytmu TS jest blisko ponad dwukrotnie dłuższy niż algorytmu PPATS. Dla instancji o rozmiarze 500 węzłów, algorytm sekwencyjny wykonywał się ponad 14-krotnie dłużej niż równoległy, w dodatku przeszukał ponad 16-krotnie mniej sąsiedztw od algorytmu PPATS.

Tab. 1. Podsumowanie czasów wykonania oraz przeszukanych sąsiedztw

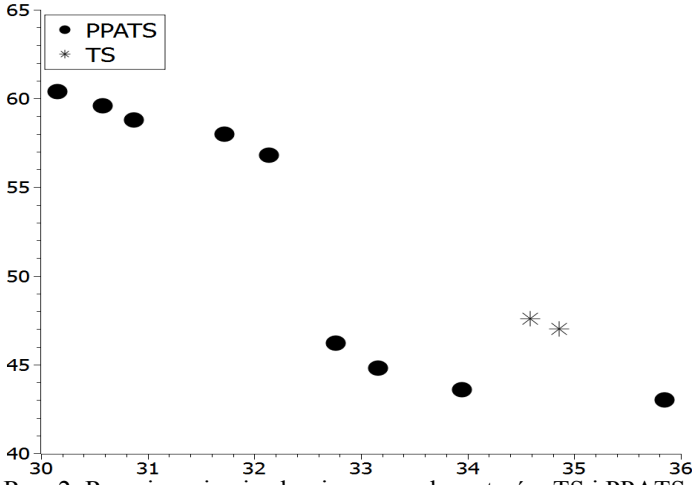
Rozmiar instancji	Sekwencyjny TS		PPATS	
	Czas [8]	Przeszukane sąsiedztwa	Czas [8]	Przeszukane sąsiedztwa
20	0,337	100	7,767	285
30	1,082	100	3,919	155
40	10,56	100	3,925	272
50	12,676	100	8,719	328
100	39,673	100	8,703	280
200	263,57	100	40,949	987
300	942,05	100	327,743	1162
400	2136,73	100	476,716	1891
500	4331,78	100	298,316	1628

Dla każdego rozmiaru testowanej instancji zabraliśmy zbiory rozwiązań Pareto-optymalnych P^{A}, A $\in$ {TS, PPATS}. Następnie ustaliliśmy zbiór P^{*} zawierający rozwiązania niezdominowane z sumy zbiorów należących do poszczególnych algorytmów. Ostatecznie, dla każdego algorytmu A ustaliliśmy liczbę rozwiązań d(A) z P^{A}, które znalazły się w zbiorze P*. Liczba niezdominowanych rozwiązań, które znalazły się w P^{*} oraz całkowita liczba rozwiązań niezdominowanych $|P^{*}|$ zostały zaprezentowane w Tab. 2.

Tab. 2. Sumaryczne porównanie wyników Pareto.

Instancja	d(TS)	d(PPATS)	$\|P^{*}\|$
20	1	4	5
30	0	2	2
40	0	3	3
50	0	5	5
100	1	2	2
200	0	9	9
300	0	8	8
400	0	30	30
500	0	25	25

Przykładowy rozkład rozwiązań niezdominowanych znalezionych przez oba algorytmy, TS i PPATS, został przedstawiony na Rys. 2. Wyższa wydajność algorytmu równoległego PPATS nad sekwencyjnym TS jest tutaj wyraźnie widoczna, szczególnie iż wszystkie rozwiązania znalezione przez algorytm TS zostały zdominowane przez aproksymację Pareto, której dostarczył algorytm PPATS. Jest to tylko jeden z przykładów testowych, aczkolwiek wyższość algorytmu równoległego, zarówno pod względem czasu działania jak i jakości odnalezionych rozwiązań, jest widoczna dla wszystkich przebadanych instancji problemu marszrutyzacji DVRP.



Rys. 2. Rozwiązania niezdominowane algorytmów TS i PPATS.

5.1. Platforma testowa

Wszystkich obliczeń oraz badań, które opisano w niniejszej pracy, dokonaliśmy korzystając z maszyny testowej wyposażonej w następujące komponenty:

- 12 x procesor Intel® Core ™ i7 X980@3,33GHz
- 24 GB RAM
- 1 x nVidia GeForce® GTX 480 – wyposażona w 480 jąder oraz 1536MB VRAM
- 1 x nVidia Tesla S2050 – posiadająca 4 GPU, każde wyposażone w 448 jąder, oraz 3072MB VRAM

Algorytm sekwencyjny testowany był zarówno na karcie GTX 480 jak i Tesla S2050.

6. Wnioski

Zgodnie z naszymi oczekiwaniami, zaproponowany algorytm równoległy PPATS w znaczącym stopniu przewyższył klasyczny algorytm TS, zarówno pod względem szybkości obliczeń (mimo całkowitej wyższej złożoności obliczeniowej), jak i w kwestii wygenerowanych przez niego rozwiązań niezdominowanych. Algorytm PPATS zdominował niemal wszystkie rozwiązania znalezione przez wspomniany wcześniej algorytm sekwencyjny TS, mimo iż zastosowane w jego konstrukcji przeszukiwanie sąsiedztwa było wykonane w jednej z najprostszych postaci i dalsze badania mogłyby przynieść jeszcze lepsze efekty w działaniu algorytmu. Na przyśpieszenie procesu znajdywania dobrych rozwiązań mogłaby wpłynąć zmiana metody generacji rozwiązania, bądź rozwiązań, początkowego. Również użycie rozbudowanej formy listy tabu powinno zwiększyć jakość oraz różnorodność generowanych rozwiązań.

Literatura

1. Dantzig G.B., Ramser J. H.: The Truck Dispatching Problem, Management Science 6, 1959, 80–91
2. Garey M.R., Johnson D.S.: A Guide to the Theory of NP-Completeness, CA: Freeman, San Francisco, 1979.

3. Schafer J.: Multiple objective optimization with vector evaluated genetic algorithm, in: Proc. of the Int. Conf. on Genetic Algorithm and their Applications, 1985.
4. Zitzler E., Thiele L.: Multiobjective optimization using evolutionary algorithms - a comparative case study., Springer.
5. Suresh R., Mohanasundaram K.: Pareto archived simulated annealing for permutation flowshop scheduling with multiple objectives., in: IEEE Conference on Cybernetics and Intelligent Systems (CIS), 2004.
6. Rudolph G.: Evolutionary search under partially ordered sets. Technical Report No. CI-67/99, Dortmund: Department of Computer Science/LS11, University of Dortmund, Germany, 1999.
7. Deb K., Pratap A., Agarwal S., Meyarivan T.: A fast and elitist multi-objective genetic algorithm NSGA-II. IEEE Transaction on Evolutionary Computation, 6(2), 2002, 181-197.
8. Bowerman R., Hall B., Calamai P.: A multi-objective optimization approach to urban school bus routing: Formulation and solution method, Transportation Research.
9. Lee T.-R., Ueng J.-H.: A study of vehicle routing problem with load balancing, International Journal of Physical Distribution and Logistics Management.
10. Zografos K., Androustsopoulos K.: A heuristic algorithm for solving hazardous material distribution problems, European Journal of Operational Research.
11. Pacheco J., Marti R.: Tabu search for a multi-objective routing problem, Journal of Operational Research Society.
12. Corberan A., Fernandez E., Laguna M., Marti R.: Heuristic solutions to the problem of routing school buses with multiple objectives, Journal of Operational Research Society.
13. Żelazny D.: Optymalizacja wielokryterialna w problemie marszrutyzacji, Krajowa Konferencja Automatyzacji Procesów Dyskretnych, 2012.
14. Ulungu E., Teghem J., Fortemps P., Tuyttens D.: MOSA method: A tool for solving moco problems, Journal of Multi-Criteria Decision Analysis.
15. Paquete T. S. L., Chiarandini L.: Pareto local optimum sets in the bi-objective traveling salesman problem: An experimental study, in: Lecture Notes in Economics and Mathematical Systems, 2004.
16. Keller C.: Multiobjective routing through space and time: The mvp and tdvrp problems, Ph.D. thesis, University of Western Ontario (1985).
17. Keller C., Goodchild M.: The multiobjective vending problem: A generalization of the traveling salesman problem, Environment and Planning B: Planning and Design.
18. Józefowicz N., Semet F., Talbi E.-G.: Multi-objective vehicle routing problems, European Journal of Operational Research.
19. Caric T., Gold H.: Vehicle Routing Problem, Published by In-Teh, Croatia, 2008.
20. Glover F., McMillan C.: The general employee scheduling problem: an integration of ms and ai, Computers and Operations Research.

Mgr inż. Szymon JAGIEŁŁO
Mgr inż. Dominik ŻELAZNY
Instytut Informatyki Automatyki i Robotyki
Politechnika Wrocławska
50-370 Wrocław, ul. Janiszewskiego 11/17
Tel: (71) 320 27 45 / 321 26 77
e-mail: szymon.jagiello@pwr.wroc.pl
dominik.zelazny@pwr.wroc.pl