AUTOMATYKA • 2011 • Tom 15 • Zeszyt 2

Jarosław Pempera*

Równoległy algorytm tabu z elementami inspirowanymi naturą dla problemu planowania tras

1. Wprowadzenie

Minimalizacja kosztów transportu w systemach transportowych jest obiektem badań naukowych już od ponad 40 lat. Nowoczesne urządzenia elektroniczne oraz systemy informatyczne pozwalają na znaczne ułatwienie pracy kierowców ciężarówek, w szczególności wspomagając nawigację oraz przyspieszając identyfikację przesyłek i generowanie niezbędnej dokumentacji. Urządzenia elektroniczne rejestrują również przebieg jazdy pojazdów umożliwiając służbom kontrolnym weryfikację przestrzegania czasów pracy kierowców.

W planowaniu przydziału zadań oraz wyznaczeniu harmonogramu realizacji zadań transportowych należy uwzględnić wiele ograniczeń występujących w rzeczywistości. Jednymi z najczęściej rozważanych jest ograniczona pojemność (ładowność) pojazdu oraz przedziały czasowe, w których przesyłka może być odebrana przez klienta. W literaturze można spotkać wiele wariantów problemu marszrutyzacji, w których brana jest pod uwagę tylko niewielka liczba ograniczeń jednocześnie. Do najczęściej rozważanych problemów należą:

- CVRP (*Capacitated VRP*) – VRP z ograniczeniem pojemności pojazdów,
- VRPTW (*VRP with Time Windows*) – VRP z oknami czasowymi,
- VRPSTW (*Vehicle Routing Problem with Soft Time Windows*) – VRP z miękkimi oknami czasowymi,
- CVRPTW – VRP z oknami czasowymi i ograniczoną ładownością pojazdów,
- MDVRP (*Multiple Depot VRP*) – VRP z wieloma magazynami centralnymi,
- MDVRPTW (*Multiple Depot VRP with Time Windows*) – MDVRP z oknami czasowymi,
- SDVRP (*Split Delivery VRP*) – dopuszczalny jest podział przesyłek i realizacja przewozów wieloma pojazdami,
- VRPPD (*VRP with Pick-Up and Deliveries*) – VRP z odbieraniem przesyłek od klientów.

* Politechnika Wrocławska, Instytut Informatyki Automatyki i Robotyki

Ze względu NP-trudny charakter nawet najprostszych problemów marszrutyzacji, do ich rozwiązania stosuje się algorytmy metaheurystyczne oparte na takich metodach jak: symulowane wyżarzanie, przeszukiwanie z zabronieniami, przeszukiwanie genetyczne, przeszukiwanie mrówkowe, sieci neuronowe [1–9].

W przytoczonych typach problemu marszrutyzacji praktycznie nie uwzględnia się czasów postoju pojazdów (przerw w pracy na odpoczynek). Biorąc pod uwagę fakt, że przerwy te mogą być stosunkowo długie, należy się liczyć z istotnymi zakłóceniami realizacji harmonogramu wyznaczonego bez ich uwzględnienia. W przypadku problemów z oknami czasowymi może to skutkować niezawinionymi przekroczeniami terminów dostawy. Ustawy dotyczące czasu pracy kierowców precyzyjnie określają jak długo kierowca może prowadzić pojazd bez odpoczynku, czas odpoczynku, jak długo może pracować w ciągu dnia, minimalny okres odpoczynku na sen, maksymalny czas pracy w ciągu tygodnia itd. Ustawy precyzują również okresy pracy i odpoczynku dla dwóch kierowców prowadzących pojazd.

Zadania transportowe możemy podzielić na dwa typy. W pierwszym typie zadań realizowane jest pojedyncze zamówienie wymagające przewozu przesyłek na duże odległości pomiędzy dużymi centrami logistycznymi. Zdecydowana większość zadań transportowych należy do drugiej grupy i wymaga przewozu towarów na stosunkowo niewielkie odległości. Dobrym przykładem tego typu zadań jest zaopatrywanie sklepów spożywczych w pieczywo lub nabiał, gdzie większość sklepów zaopatruje się u tego samego producenta regionalnego i najczęściej korzysta z jego oferty przewozowej. Zadania należące do drugiej grupy realizowane są w ciągu jednego dnia. Obowiązujące przepisy prawne, dla jednodniowego okresu pracy umożliwiają pracę przez okres maksymalnie 9 godzin. Czas pracy kierowcy, musi być podzielony na dwa okresy 4,5-godzinne, pomiędzy którymi kierowca musi odpoczywać co najmniej 45 minut. Okres przerwy może być podzielony na 3 przerwy po co najmniej 15 minut.

2. Model matematyczny

Firma transportowa dysponuje flotą składającą się z f identycznych pojazdów ze zbioru $F = \{1, \ldots, f\}$. W czasie jednego dnia pracy firmy należy wykonać zlecenia przewozowe dla n klientów. Każde zlecenie generuje zadanie transportowe, które polega na przewiezieniu towarów wyspecyfikowanych w zamówieniu z magazynu centralnego do miejsca wskazanego przez klienta zamawiającego towar. Dany jest czas przejazdu $d_{i,j} > 0$, $i, j = 0, 1, .., n$, pomiędzy wszystkimi miejscami załadunku i rozładunku przesyłek, przy czym 0 oznacza magazyn centralny, natomiast $1, \ldots, n$ miejsca rozładunku przesyłek klientów. Czas rozładunku przesyłki oraz załatwienia związanych z nim formalności wynosi $p_i > 0$, $i = 1, .., n$. Przyjmuje się, że pojemność każdego pojazdu jest nieograniczona tj. w dowolnej chwili można przewieźć dowolną liczbę towarów.

Każdemu z zadań transportowych ze zbioru $J = \{1, \ldots, n\}$ należy przyporządkować pojazd oraz wyznaczyć trasy przejazdu i harmonogram przejazdów każdego pojazdu nale-

żącego do floty F. Trasa każdego pojazdu musi rozpoczynać się i kończyć się w magazynie. Przydział zadań do pojazdów oraz trasy przejazdu pojazdów można jednoznacznie opisać przy pomocy zestawu permutacji $t = (t_1, \ldots, t_f)$, gdzie $t_l = (t_l(1), \ldots, t_l(n_l))$ jest permutacją określającą trasę l-tego pojazdu, natomiast n_l jest liczbą klientów obsługiwanych przez ten pojazd. Oczywiście dopuszczalny zestaw tras t musi zagwarantować obsługę wszystkich klientów, tj.

$$\bigcup_{i=1}^{f} T_i = J \tag{1}$$

gdzie $T_i = \{t_i(1), \ldots, t_i(n_i)\}$ oraz każdy z klientów musi być obsłużony przez jeden pojazd, tj.

$$T_i \cap T_j = \varnothing \quad i \neq j, i, j = 1, ..., f \tag{2}$$

Efektywny czas pracy kierowcy $l \in F$ realizującego zamówienia wynikające z trasy t_l jest równy

$$E(t_l) = d_{0,t_l(1)} + \sum_{s=2}^{n_l} d_{t_l(s-1),t_l(s)} + d_{t_l(n_l),0} + \sum_{s=1}^{n_l} p_{t_l(s)} \tag{3}$$

Jest on sumą czasów przejazdów oraz czasów obsługi klientów. Natomiast całkowity czas pracy uwzględniający niezbędną przerwę na odpoczynek wynosi:

$$C(t_l) = \begin{cases} E(t_l) & \text{dla} \quad E(t_l) \leq u \\ E(t_l) + \rho & \text{dla} \quad E(t_l) > u \end{cases}, \quad l = 1, ..., f \tag{4}$$

Jak wspomniano wyżej, w przypadku jednodniowego planowania trasy przejazdu, jeżeli efektywny czas pracy kierowcy jest dłuższy od u = 270 minut, to w harmonogramie pracy kierowcy należy zaplanować przerwę ρ = 45 minutową. Maksymalny z czasów pracy kierowców dla zestawu tras t definiujemy następująco:

$$C_{\max}(t) = \max_{1 \leq l \leq f} C(t_l) \tag{5}$$

Firma transportowa dysponuje flotą składającą się z f identycznych pojazdów ze zbioru $F = \{1, \ldots, f\}$. W czasie jednego dnia pracy firmy należy wykonać zlecenia przewozowe dla n klientów. Każde zlecenie generuje zadanie transportowe, które polega na przewiezieniu towarów wyspecyfikowanych w zamówieniu z magazynu centralnego do miejsca wskazanego przez klienta zamawiającego towar. Dany jest czas przejazdu $d_{i,j} > 0$, $i, j = 0, 1, .., n$, pomiędzy wszystkimi miejscami załadunku i rozładunku przesyłek, przy czym 0 oznacza magazyn centralny, natomiast 1, …, n miejsca rozładunku przesyłek klientów. Czas rozładunku przesyłki oraz załatwienia związanych z nim formalności wynosi $p_i > 0$, $i = 1, .., n$. Przyjmuje się, że pojemność każdego pojazdu jest nieograniczona, tj. w dowolnej chwili można przewieźć dowolną liczbę towarów.

Praca współczesnego kierowcy wspomagana jest różnego rodzaju urządzeniami elektronicznymi. Umożliwiają one sprawną nawigację, identyfikację przesyłek oraz rejestrowanie i generowanie dokumentacji. Fakt ten w połączeniu z zaufaniem do odbiorcy pozwala na przekazanie czynności spedycyjnych pracownikowi klienta i zaplanowanie niezbędnej przerwy w okresie obsługi tego klienta.

W celu spełnienia wymagań formalnych, przerwa powinna nastąpić nie później niż u = 4,5 h (270 minut) po rozpoczęciu jazdy kierowcy oraz nie wcześniej niż 4,5 h przed zakończeniem pracy kierowcy. Oznaczmy przez $a = u$ oraz $b = E(t_l) - u$ odpowiednio najpóźniejszy moment rozpoczęcia przerwy oraz najwcześniejszy moment zakończenia przerwy.

Dla $E(t_l)$ większego od $2u$ (9 h) $b > a$, czas pracy kierowcy można skrócić do dopuszczalnego lub krótszego, jeżeli okres (a, b) przypada w całości na okres obsługi jednego z klientów, w przeciwnym przypadku nie można utworzyć harmonogramu dopuszczalnego. W przypadku gdy $E(t_l) < 2u$, tj. $a \geq b$, początek przerwy można zaplanować na dowolny moment z tego przedziału, przy czym wybór najdłuższego czasu obsługi przypadającego na ten okres redukuje najwięcej czas pracy kierowcy $C(t_l)$.

Niech Ω będzie zbiorem wszystkich dopuszczalnych zestawów tras kierowców, tj.

$$\Omega = \{ t=(t_1,\ldots,t_f): t \text{ jest rozbiciem zbioru } J, t \text{ jest zestawem dopuszczalnych tras}\} \quad (6)$$

Należy znaleźć zestaw t^* taki, że

$$C_{\max}(t^*) = \min_{\tau \in \Omega} C_{\max}(\tau).$$

3. Algorytm tabu

Metoda przeszukiwania z zabronieniami (tabu) [10] jest jedną z najczęściej stosowanych metod konstrukcji algorytmów do rozwiązywania kombinatorycznych problemów optymalizacyjnych. Wysoka efektywność algorytmów konstruowanych w oparciu o tę metodę dla problemów VRP została potwierdzona między innymi w pracach [3, 6, 9].

W każdej iteracji tego typu algorytmów przeglądane jest otoczenie rozwiązania bazowego w celu wyboru najlepszego. Wybrane rozwiązanie zastępuje rozwiązanie bazowe w następnej iteracji. W celu wyeliminowania powrotu do rozwiązań bazowych wcześniej wygenerowanych wykorzystuje się mechanizm zabronień. Algorytm rozpoczyna działanie od rozwiązania początkowego wygenerowanego algorytmem konstrukcyjnym oraz kończy działanie po wykonaniu określonej liczby iteracji. Wysoką skuteczność tego typu algorytmów upatruje się właśnie w systematycznym przeglądaniu całego otoczenia. Domniema się, że w otoczeniu rozwiązania bazowego znajduje się nowe lepsze rozwiązanie lub niewiele gorsze. Intensywne przeszukiwanie otoczenia ma również niekorzystną cechę, tj. wymaga dużo czasu. Dla wielu problemów czas ten można znacząco zmniejszyć redukując liczność otoczenia (np. przez eliminację rozwiązań apriorycznie gorszych od bazowego) i/lub akceleracją obliczeń.

Podstawowym elementem algorytmu tabu jest reprezentacja rozwiązania problemu. W rozważanym problemie rozwiązane reprezentowane jest w postaci zestawu składającego się z f permutacji. Dla tego typu reprezentacji stosuje się zasadniczo dwa typy ruchów: zamień i wstaw. Z tych dwóch typów ruchami dającymi większe możliwości modyfikacji rozwiązań są ruchy typu wstaw. Czwórka $v = (l, a, k, b)$ opisuje ruch typu wstaw, który oznacza, że należy usunąć klienta stojącego na pozycji a w trasie kierowcy l i umieścić na pozycji b w trasie kierowcy k. Dla tego typów mechanizm zabezpieczający przed powrotem do rozwiązań wygenerowanych we wcześniejszych T iteracjach można zrealizować w postaci listy cyklicznej o długości T, na której zapisywane są następujące informacje pochodzące od rozwiązania bazowego t oraz ruchu $v = (l, a, k, b)$ generującego nowe rozwiązanie bazowe: (*i*) $(t_l(a), t_l(a+1), l)$, gdy $a < b$ oraz $l = k$, (*ii*) $(t_l(a-1), t_l(a), l)$, gdy $a < b$ oraz $l = k$, (*iii*) obie trójki dla $l \neq k$. Rozwiązanie sąsiednie t^v jest zabronione, jeżeli dla co najmniej jednej trójki $s = (a, b, l)$ znajdującej się na liście zabronień, klienci a raz b obsługiwani są przez kierowcę l oraz a obsługiwany jest przed b.

W zaawansowanych algorytmach stosuje się mechanizmy intensyfikacji i dywersyfikacji przeszukiwań. Pierwsze mają na celu intensywniejsze przeszukanie regionów, w których znaleziono najlepsze rozwiązania, natomiast drugie na przeniesieniu przeszukiwań do obszarów jeszcze nie przeszukiwanych.

4. Równoległy algorytm tabu

Obecnie, dzięki powszechnie dostępnym systemom wieloprocesorowym (wielordzeniowym) oraz wieloprocesowym systemom operacyjnym, obserwuje się intensywny rozwój algorytmów równoległych. Algorytmy równoległe możemy podzielić na jednościeżkowe oraz wielościeżkowe. W algorytmach jednościeżkowych najczęściej równolegle obliczana jest wartość funkcji celu dla wielu rozwiązań jednocześnie. Z tego punktu widzenia najłatwiej jest zrównoleglić algorytmy populacyjne lub przeglądające wiele rozwiązań w jednej iteracji. Algorytmy wielościeżkowe, podobnie jak wymienione wyżej, bazują na tradycyjnych algorytmach metaheurystycznych, w których dzięki zmianie parametrów algorytmów uzyskuje się inny przebieg (ścieżkę w przestrzeni rozwiązań) obliczeń. W szczególności tym parametrem może być rozwiązanie początkowe.

Pewną słabością algorytmów jednościeżkowych jest wysoka wrażliwość na parametry algorytmu. Strojenie tych parametrów podczas testów wstępnych algorytmu pozwala na generowanie dobrych wyników w grupie testowej w sensie średnim. Niemniej, w przypadku konkretnej instancji, rozwiązanie wygenerowane algorytmem z innymi parametrami może być znacznie lepsze. Ze swojej natury algorytmy wielościeżkowe w dużej mierze pozbawione są tej wady. Działanie algorytmów wielościeżkowych można kontrolować, zastępując przebiegi produkujące rozwiązania gorszej jakości nowymi przebiegami przeszukującymi obszary w pobliżu wcześniej znalezionych dobrych rozwiązań (intensyfikacja) lub przeszukującymi nowe obszary rozwiązań (dywersyfikacja). W literaturze można spotkać niewiele

propozycji sposobów zarządzania przebiegami w tego typu algorytmach równoległych [11]. W proponowanym równoległym algorytmie tabu proponuje się wykorzystanie wybranych mechanizmów ewolucyjnych do zarządzania przebiegami wielościeżkowego algorytmu tabu.

Algorytmy ewolucyjne w istocie swojego działania naśladują zachowania ewolucyjne obserwowane w przyrodzie. Proces ewolucyjny odbywa się na populacji osobników ocenianych na podstawie pewnej miary dopasowania. Pewne osobniki podlegają transformacji genetycznej, w wyniku której powstają nowe osobniki. Wyróżniamy dwa podstawowe operatory genetyczne: operator krzyżowania i operator mutacji. Zadaniem operatora krzyżowania jest przekazanie najlepszych cech osobników danej populacji następnej generacji w nadziei otrzymania populacji średnio lepiej dopasowanej. Natomiast zadaniem operatora mutacji jest wygenerowanie osobników różniących się od osobników występujących w danej populacji w nadziei przypadkowego odkrycia nowych korzystnych cech. Nowa populacja tworzona jest na podstawie bieżącej populacji rozszerzonej o osobniki wygenerowane operatorami genetycznymi poprzez wybór stałej liczby osobników najlepiej dopasowanych (selekcja naturalna).

W proponowanym mechanizmie zarządzania przebiegami, używając terminologii ewolucyjnej, przez osobnika będziemy rozumieli abstrakcyjny byt, jakim jest przebieg algorytmu. Zakłada się, że każdy przebieg może być w dowolnej chwili zatrzymany, wznowiony lub usunięty. W każdej iteracji procesu ewolucyjnego z populacji składającej się z X osobników wybierany podzbiór składający się z x przebiegów, z których każdy uruchamiany jest na zadaną liczbę iteracji *citer*. Podczas każdego uruchomienia, dla przebiegu l zapamiętywane jest najlepsze znalezione rozwiązanie t_l' oraz najlepsze rozwiązanie t_l^* znalezione podczas działania całego przebiegu. Dodatkowo w przypadku znalezienia nowego rozwiązania t_l^* zapamiętywany jest również zbiór wszystkich pozostałych rozwiązań sąsiednich oraz zawartość listy zabronień.

Rozmnażaniu podlegają tylko (osobnicy) przebiegi, w których znaleziono nowe najlepsze rozwiązanie. W każdym przebiegu algorytmu oprócz najlepszego rozwiązania zapamiętywane są wszystkie rozwiązania sąsiednie oraz zawartość listy zabronień. Z każdego rozwiązania sąsiedniego generowany jest nowy przebieg z zawartością listy tabu odtwarzaną z pamięci. W przypadku przebiegów, w których liczba uruchomień bez znalezienia nowego najlepszego rozwiązania przekroczy zadaną liczbę *live*, przebieg jest modyfikowany poprzez losową zmianę długości listy zabronień.

Pierwszy z przedstawionych mechanizmów pozwala na powrót do obszarów z dobrymi rozwiązaniami i znany jest z literatury [12] jako metoda skoku powrotnego, natomiast drugi z mechanizmów implementuje znaną metodę dywersyfikacyjną, tj. metodę dynamicznej długości listy zabronień [13]. Przebiegi oceniane są na podstawie wartości t_l'. W przypadku nowo wygenerowanych przebiegów wartością tą jest wartość funkcji celu dla odpowiedniego rozwiązania sąsiedniego.

5. Experyment komputerowy

Równoległy algorytm tabu PTS został zaimplementowany w języku C++ środowisku Visual Studio 2005 w dwóch wersjach : PTS–E z selekcją elitarną oraz PTS–R z selekcją rankingową. Testy przeprowadzono na komputerze z procesorem Intel i 7 2.4 GHz wyposażonym w 4 rdzenie rzeczywiste (8 rdzeni wirtualnych dzięki technologii multithreading). Testy algorytmów zostały przeprowadzone na zestawie 50 instancji wygenerowanych losowo. Instancje te zostały podzielone na 5 grup. Instancje z tej samej grupy miały identyczną liczbę klientów $n = \{10, 20, 50, 80, 100\}$. Liczba pojazdów (kierowców) dla wszystkich instancji z danej grupy była jednakowa i wynosiła odpowiednio dla grup 2, 2, 6, 9, 10. Parametry instancji wygenerowano generatorem jednostajnym, przyjmując następujące zakresy dla poszczególnych danych problemu: $p_j \in (1,100)$, $d_{ij} \in (10,120)$.

Jakość rozwiązań generowanych przez proponowany algorytm porównano z rozwiązaniami generowanymi przez sekwencyjne algorytmy oparte na metodzie przeszukiwania z zabronieniami TS oraz symulowanego wyżarzania SA, które zostały opisane w pracy [14]. Testy algorytmu PTS przeprowadzono dla populacji składającej się z 8 osobników. Taki wybór zapewnił prawie identyczny czas działania wersji równoległej i sekwencyjnej algorytmu tabu. Populacja początkowa składała się z identycznych osobników wygenerowanych podobnie jak w przypadku TS algorytmem opartym na metodzie wstawień. Poszczególne przebiegi algorytmów różniły się długością listy zabronień, która wynosiła w zależności od numeru przebiegu 5, 7, …, 19. Czynność zarządzania przebiegami następowała po wykonaniu 200 iteracji. Ustalono czas życia osobnika na 5 pokoleń.

Algorytm SA dla każdej instancji został uruchomiony 5-krotnie z rozwiązań wygenerowanych losowo. Algorytmy SA i TS zostały uruchomione na 1000 iteracji, natomiast algorytmy PTS na 50 pokoleń, tj. 10 000 iteracji. Dla każdego przebiegu algorytmów został wyznaczony czas działania oraz najlepsze rozwiązanie znalezione podczas przeszukiwań. Następnie na podstawie wszystkich rozwiązań wygenerowanych dla tej samej instancji wyznaczono najlepsze rozwiązane t^{ref} .

Dla każdego rozwiązania t wyznaczono błąd względny $B(t) = 100\% \ (C_{\max}(t) - C_{\max}(t^{ref}))/C_{\max}(t^{ref})$.

W tabeli 1 przedstawiono błąd względny algorytmów uśredniony względem instancji należących do tej samej grupy. W przypadku algorytmu SA podano średni błąd oraz minimalny błąd wyznaczony na podstawie 5 uruchomień. W tabeli 2 podano bezwzględny czas działania algorytmów oraz liczbę iteracji wykonywanych w czasie jednej sekundy.

Z rezultatów badań przedstawionych w tabelach wynika, że algorytmy równoległe są znacznie efektywniejsze od algorytmów sekwencyjnych. Porównując błąd względny algorytmów PTS, można zauważyć, że w przypadku algorytmów równoległych jest on istotnie mniejszy od błędów algorytmów sekwencyjnych już po wykonaniu 400 iteracji (200 w przypadku PTS–E). Wyjątkowo słabo w tym zestawieniu wypada algorytm SA. Średni błąd algorytmu wacha się w granicach od 1,5% do 27%.

Tabela 1
Błąd względny algorytmów

Liczba zadań	TS	SA		PTS–E		PTS–R	
		MIN	AVE	200 iter	10 000 iter	200 iter	10 000 iter
10	0,08	0,40	1,54	0,00	0,00	0,00	0,00
20	2,57	19,84	27,18	2,30	0,27	3,15	0,71
50	5,14	6,22	8,17	2,98	1,42	3,57	0,65
80	5,21	7,83	12,08	3,71	2,07	3,84	0,42
100	5,17	10,43	20,33	5,30	0,58	6,12	0,20
Średnio	**3,64**	**8,94**	**13,86**	**2,86**	**0,87**	**3,34**	**0,39**

Tabela 2
Średni czas działania algorytmów

Liczba zadań	TS (1000 iter)		SA (1000 iter)		PTS–E (10 000 iter)		PTS–R (10 000 iter)	
	CPU [s]	iter/s	CPU [s]	iter/s	CPU [s]	Iter/s	CPU [s]	iter/s
10	0,04	25 000	0,01	100 000	0,5	20000	0,60	16667
20	0,2	5 000	0,02	50 000	2,2	4545	2,20	4545
50	0,9	1 111	0,07	14 285	13,7	730	14,20	704
80	2,2	455	0,14	7 143	36,6	273	37,80	265
100	3,5	286	0,20	5 000	62,0	161	63,60	157

Tabela 3
Zależność błędu algorytmów równoległych od liczby iteracji

Algorytm	Liczba zadań	Liczba iteracji						
		200	400	800	1600	3200	6400	10000
PTS-E	10	0,00	0,00	0,00	0,00	0,00	0,00	0,00
	20	2,30	1,88	1,31	0,64	0,27	0,27	0,27
	50	2,98	2,35	1,72	1,50	1,42	1,42	1,42
	80	3,71	2,66	2,07	2,07	2,07	2,07	2,07
	100	5,30	2,77	1,59	0,62	0,58	0,58	0,58
	Średnio	**2,86**	**1,93**	**1,34**	**0,97**	**0,87**	**0,87**	**0,87**
PTS-R	10	0,00	0,00	0,00	0,00	0,00	0,00	0,00
	20	3,15	2,74	2,18	1,62	1,31	1,31	0,71
	50	3,57	2,72	1,96	0,67	0,67	0,67	0,65
	80	3,84	2,24	1,26	0,66	0,55	0,53	0,42
	100	6,12	3,57	1,44	0,55	0,37	0,31	0,20
	Średnio	**3,34**	**2,25**	**1,37**	**0,70**	**0,58**	**0,56**	**0,39**

Sekwencyjny algorytm TS znalazł praktycznie wszystkie rozwiązania najlepsze instancji z pierwszej grupy tj. grupy o najmniejszej liczbie zadań transportowych. W pozostałych grupach błąd algorytmu waha się w granicach od 2,5% do 5,2%. Porównując algorytmy równoległe łatwo można zauważyć (patrz tab. 3), że algorytm z selekcją elitarną generuje rozwiązania znacznie lepszej jakości dla małej liczby iteracji (do 1 000). Dla większej liczy iteracji rozwiązania lepsze, w sensie średnim, generuje algorytm z selekcją rankingową. Przewagę tego algorytmu szczególnie widać na instancjach z liczbą zadań, tj. 50 i większą.

6. Podsumowanie

W pracy zaproponowano nowy równoległy algorytm oparty na metodzie tabu dla problemu marszrutyzacji pojazdów. Zaproponowano w nim mechanizm zarządzania przebiegami, w którym zastosowano mechanizmy inspirowane zachowaniami ewolucyjnymi. Z przeprowadzonych eksperymentów komputerowych wynika, że algorytmy te generują rozwiązania lepsze od algorytmów sekwencyjnych w mniejszej liczbie iteracji. Zastosowanie mechanizmów ewolucyjnych pozwala na uzyskanie lepszych rozwiązań w perspektywie wykonania większej liczby iteracji. Ponadto wykonanie algorytmu równoległego z ilością przebiegów równą liczbie procesorów (rdzeni) pozwala na wykonanie prawie takiej samej liczby iteracji w podobnym czasie co algorytm sekwencyjny.

Literatura

[1] Alba E., Dorronsoro B., *Solving the Vehicle Routing Problem by Using Cellular Genetic Algorithms. In* Conference on Evolutionary Computation in Combinatorial Optimization, Portugal, 2005, 11–20.

[2] Bräysy O., Gendreau M., *Vehicle Routing Problem with Time Windows, Part I: Route Construction and Local Search Algorithms.* Transportation Science, 39, 1, 2005, 104–118.

[3] Cordeau J.-F., Laporte G., *A tabu search algorithm for the site dependent vehicle routing problem with time windows*. INFOR, 39(3), 2001, 292–298.

[4] Czech Z.J., Czarnas P., *Parallel simulated annealing for the vehicle routing problem with time windows, in 10th Euromicro Workshop on Parallel*. Distributed and Networkbased Processing, Spain, 2002, 376–383.

[5] Dorigo M., Maniezzo V., Coloni A., *Ant system: Optimization by a colony of cooperating agents.* IEEE Transactions on Systems, Man, and Cybernetics – Part B, 26, 1996, 29–41.

[6] Gendreau M., Laporte G., Seguin R., *A tabu search heuristic for the vehicle routing problem with stochastic demand and customers.* Operations Research, 44, 3, 1996, 469–477.

[7] Golden B.L., Magnanti T.L., Nguyan H.Q., *Implementing vehicle routing algo*rithms. Networks, 7, 2, (1972), 113–148.

[8] Osman I.H., Kelly J.P., *Meta-heuristics: An overview*. [w:] I.H. Osman & J. P. Kelly (Eds.), Meta-heuristics: Theory and applications Boston: Kluwer Academic Publishers, 1996, 1–21.

[9] Rego C., Roucairol C., *Using tabu search for solving a dynamic multiterminal truck dispatching problem.* European Journal of Operational Research, 83, 1995, 411–429.

[10] Glover F., Laguna M., *Tabu search*. Boston: Kluwer Academic Publishers, 1997.

[11] Bożejko W., Wodecki M., *Solving Permutational Routing Problems by Population-Based Metaheuristics*. Computers & Industrial Engineering, 57, 2009, 269–276.
[12] Nowicki E., Smutnicki C., *A fast taboo search algorithm for the job shop problem*. Management Science, 42, 6, 1996, 797–813.
[13] Grabowski J., Pempera J., *Some local search algorithms for no-wait flow-shop problem with makespan criterion.* Computers & Operations Research, 32, 8, 2005, 2197–2212.
[14] Pempera J., *Planowanie tras pojazdów z ograniczeniami czasu pracy kierowców. Logistyka i zarządzanie produkcją – nowe wyzwania, odległe granice.* Red. Marek Fertsch, Katarzyna Grzybowska, Agnieszka Stachowiak. Poznań, 2007, 156–163.