Politechnika Śląska
Wydział Automatyki, Elektroniki i Informatyki
Instytut Informatyki

Autoreferat rozprawy doktorskiej

Równoległy algorytm memetyczny w rozwiązywaniu problemu trasowania pojazdów z oknami czasowymi

mgr Mirosław Błocho

Promotor: Prof. dr hab. inż. Zbigniew J. Czech

Gliwice, 2013

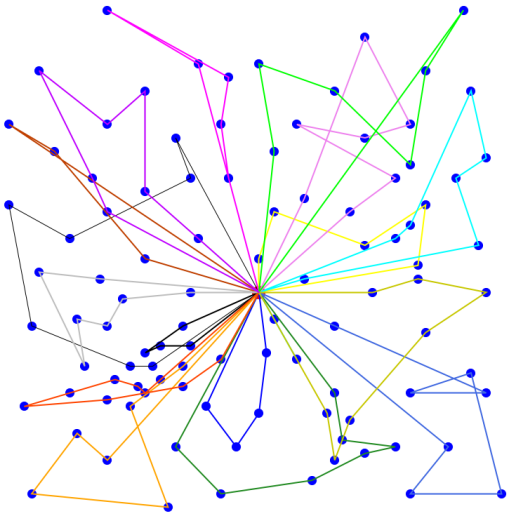
1. Wprowadzenie

Transport jest jednym z głównych działów gospodarki polegającym na przemieszczaniu ładunków, dóbr lub osób przy użyciu odpowiednich środków. Pozostałe działy gospodarki, takie jak usługi i przemysł są ściśle związane z transportem. Rosnące koszty transportu związane są bezpośrednio m.in. ze wzrostem wydatków na paliwo, kosztami zakupu i eksploatacji pojazdów oraz kosztem zatrudnienia kierowców. Z tego powodu coraz większego znaczenia nabiera stosowanie specjalistycznych programów komputerowych wspomagających optymalizację różnorodnych problemów transportowych i logistycznych.

Jednym z najważniejszych problemów optymalizacji dyskretnej dotyczących zagadnień związanych z transportem jest problem trasowania pojazdów (ang. *vehicle routing problem*, VRP), dotyczący dystrybucji dóbr z magazynu do klientów. Problem ten stanowi rozszerzenie problemu komiwojażera, w którym komiwojażer ma za zadanie wyruszyć z magazynu, odwiedzić zadaną liczbę miast w ramach jednej trasy oraz powrócić do miejsca skąd rozpoczął podróż. Celem jest wyznaczenie trasy o minimalnym koszcie, np. o minimalnej długości. Problem komiwojażera jest NP-trudny, co oznacza, że algorytm o wielomianowej złożoności obliczeniowej rozwiązujący ten problem prawodpodobnie nie istnieje. W problemie VRP każdy klient musi zostać odwiedzony jednokrotnie, a każdy pojazd musi wrócić z powrotem do magazynu. Rozwiązaniem problemu VRP jest znalezienie minimalnej liczby użytych pojazdów (tras) oraz minimalnej sumarycznej długości tras.

Rozprawa doktorska została poświęcona problemowi trasowania pojazdów z oknami czasowymi (ang. *vehicle routing problem with time windows*, VRPTW), który stanowi jeden z wielu wariantów problemu VRP. W problemie VRPTW wprowadzone zostały dodatkowe ograniczenia związane z maksymalną ładownością pojazdów oraz z wymogiem dostarczenia towaru w ściśle określonych ramach czasowych, tzw. oknach czasowych. Na rys. 1 przedstawiono rozwiązanie dla przykładowego testu R102 z repozytorium Solomona (100 klientów).

Problemy trasowania pojazdów znajdują wiele zastosowań praktycznych w planowaniu dystrybucji towarów w transporcie drogowym, lotniczym, kolejowym oraz morskim. Optymalizacja tras wiąże się bezpośrednio ze zmniejszeniem kosztów, a tym samym ze zwiększeniem konkurencyjności na rynku. Przynależność problemu trasowania pojazdów do klasy problemów NP-trudnych powoduje jednak, że istniejące algorytmy dokładne wymagają zbyt dużego czasu do znalezienia optymalnych rozwiązań. Z tego powodu coraz większym zainteresowaniem cieszą się algorytmy heurystyczne, które pomimo braku gwarancji uzyskania optymalnych wyników umożliwiają wyznaczenie bardzo dobrych jakościowo rozwiązań w rozsądnym czasie. Dodatkowe warunki, takie jak np. wymóg dostarczenia towaru w ściśle określonych ramach czasowych powodują, że czas uzyskania dostatecznie dobrego rozwiązania staje się nie do zaakceptowania nawet w przypadku użycia metod heurystycznych. Z pomocą przychodzą tutaj możliwości zrównoleglenia



Rysunek 1: Rozwiązanie problemu trasowania pojazdów z oknami czasowymi dla testu R102.

algorytmów ze względu na coraz powszechniejsze oraz coraz tańsze komputery wielordzeniowe. Algorytmy sekwencyjne działające w komputerach z procesorami wielordzeniowymi nie wykorzystują pełnej mocy obliczeniowej i dlatego tak istotne stają się korzyści wynikające z zastosowania algorytmów równoległych do rozwiązywania problemów optymalizacyjnych.

Większość badań nad algorytmami rozwiązującymi problem VRPTW koncentrowała się na rozwoju metaheurystyk, które polegają na akceptowaniu gorszych lub niedopuszczalnych rozwiązań w trakcie procesu przeszukiwania, celem znalezienia jeszcze lepszych rezultatów po opuszczeniu minimów lokalnych. W ostatnich latach spore zainteresowanie w rozwiązywaniu problemu VRPTW zyskały również algorytmy memetyczne, będące połączeniem algorytmu ewolucyjnego (np. algorytmu genetycznego) z algorytmami ulepszania rozwiązań (np. z algorytmem lokalnych poszukiwań). Algorytm ewolucyjny służy do eksploracji przestrzeni rozwiązań, a algorytm ulepszania do eksploatacji rozwiązań już znalezionych. Problem trasowania pojazdów z oknami czasowymi jest przedmiotem badań od wielu lat i aby móc porównywać jakość rozwiązań poszczególnych algorytmów zostało opracowanych kilka repozytoriów z danymi testowymi. Do najbardziej popularnych należą zbiory danych autorstwa Solomona (S) [8] oraz Gehringa i Hombergera (GH) [5] zawierające odpowiednio 56 oraz 300 egzemplarzy problemu VRPTW.

W rozprawie został zaproponowany oraz zaimplementowany oryginalny, równoległy algorytm memetyczny do rozwiązywania problemu VRPTW. Punktem wyjścia stał się algorytm opracowany przez Nagatę, Bräysy'ego oraz Dullaerta [7]. Zaproponowany równoległy algorytm został podzielony na dwa etapy. W pierw-

szym etapie liczba tras jest minimalizowana za pomocą heurystyki opartej na puli usuniętych klientów oraz strategii lokalnych poszukiwań z przewodnikiem. W drugim etapie, poświeconym minimalizacji sumarycznej długości tras, generowana jest populacja osobników (rozwiązań), w której iteracyjnie konstruowane są kolejne generacje zawierające skrzyżowane osobniki. Do krzyżowania osobników wykorzystywany jest operator EAX (ang. *edge assembly crossover*) [7]. W każdej generacji skonstruowane rozwiązania potomne ulepszane są przez operacje lokalnych poszukiwań. W celu zwiększenia jakości rozwiązań oraz zredukowania czasu działania algorytmu zaproponowano dwa schematy kooperacji między procesami oparte na strukturze pierścienia — podstawowy oraz randomizowany. W obu schematach procesy cyklicznie komunikują się ze sobą przesyłając najlepsze rozwiązania. W schemacie podstawowym najlepsze rozwiązanie przesyłane jest zgodnie z ustaloną, stałą sekwencją procesów. Oryginalny schemat randomizowany polega na generowaniu w każdej fazie kooperacji, losowej sekwencji procesów, między którymi przesyłane jest najlepsze rozwiązanie. Dodatkowo w schemacie tym kooperacja między procesami polega na generowaniu rozwiązań potomnych przez krzyżowanie operatorem EAX bieżącego rozwiązania procesu z nowo otrzymanym rozwiązaniem od innego procesu.

Zastosowanie równoległości jest jednym z najlepszych sposobów na skrócenie czasu obliczeń oraz zwiększenie jakości otrzymywanych rozwiązań. Ze względu na upowszechnienie się komputerów z procesorami wielordzeniowymi, jak również łatwiejszy dostęp do komputerów wieloprocesorowych, takie badania są w pełni uzasadnione. Rozprawa jest poświęcona równoległej implementacji memetycznego algorytmu, którego krótką charakterystykę przedstawiono powyżej. Ze względu na dwuetapowość badanego algorytmu memetycznego możliwe jest skonstruowanie dwóch oddzielnych algorytmów równoległych dla etapów minimalizacji liczby tras oraz minimalizacji sumarycznej długości tras. Zasadniczą kwestią związaną ze zrównoleglaniem już istniejących algorytmów jest sposób, w jaki poszczególne procesy powinny ze sobą współpracować. Ma to bezpośredni związek z wyborem schematu kooperacji oraz częstotliwości eksploatacji najlepszych rozwiązań w równoległym algorytmie memetycznym. Kooperacja procesów jest operacją stosunkowo kosztowną i należy znaleźć odpowiednią równowagę między jakością otrzymywanych rozwiązań, a częstotliwością kooperacji.

2. Teza rozprawy

Główną tezą rozprawy było stwierdzenie, że randomizowany schemat kooperacji z krzyżowaniem rozwiązań wymienianych przez procesy umożliwia uzyskanie wyników o lepszej jakości dla problemu VRPTW w równoległym algorytmie memetycznym.

3. Cele rozprawy

Głównymi celami rozprawy były:

1. Zaprojektowanie i zaimplementowanie równoległego algorytmu memetycznego przeznaczonego do rozwiązywania problemu VRPTW.

2. Opracowanie efektywnych schematów kooperacji procesów w zaproponowanym algorytmie równoległym.

3. Dokonanie weryfikacji poprawności zaproponowanego równoległego algorytmu memetycznego oraz przeprowadzenie analizy teoretycznej i eksperymentalnej wielkości charakterystycznych algorytmu, tj. złożoności czasowej i pamięciowej, przyspieszeń, kosztu obliczeniowego oraz efektywności wykorzystania procesorów.

4. Znalezienie za pomocą równoległego algorytmu z opracowanymi schematami kooperacji procesów, rozwiązań o możliwie najlepszej jakości dla danych testowych S oraz GH dla problemu VRPTW.

Dodatkowym celem rozprawy było:

Dokonanie wszechstronnych badań eksperymentalnych zaproponowanego równoległego algorytmu memetycznego, w szczególności przeprowadzenie:

a) analizy jakości uzyskiwanych rozwiązań dla danych testowych VRPTW ze zbiorów S oraz GH,

b) badań wpływu liczby równolegle pracujących procesorów na jakość uzyskiwanych rozwiązań,

c) badań wpływu częstotliwości kooperacji między procesami na jakość uzyskiwanych rozwiązań,

d) badań wpływu wielkości początkowej populacji rozwiązań na jakość uzyskiwanych rozwiązań,

e) badań wpływu liczby rozwiązań potomnych na jakość uzyskiwanych rozwiązań.

4. Zawartość pracy

Rozprawa doktorska składa się z dziewięciu rozdziałów oraz trzech dodatków. Rozdział pierwszy obejmuje wprowadzenie do problematyki rozprawy oraz wyszczególnienie tezy i celów rozprawy. W drugim rozdziale sformułowano problem VRPTW wraz z krótkim opisem innych wariantów problemu trasowania pojazdów. Rozdział trzeci zawiera przegląd literaturowy algorytmów dokładnych oraz heurystycznych rozwiązujacych problem VRPTW. W rozdziale czwartym omówiono standardowy schemat algorytmów memetycznych. Rozdział piąty został poświęcony sekwencyjnemu algorytmowi minimalizacji liczby tras oraz sekwencyjnemu algorytmowi memetycznemu dla problemu VRPTW. W rozdziale szóstym przedstawiono ulepszenia sekwencyjnego algorytmu memetycznego. Rozdział siódmy został poświęcony równoległemu algorytmowi minimalizacji liczby tras, rów-

noległemu algorytmowi memetycznemu dla problemu VRPTW oraz opracowanym schematom kooperacji między procesami. Druga część rozdziału poświęcona została weryfikacji poprawności oraz złożoności obu równoległych algorytmów. W rozdziale ósmym przedstawiono opis przeprowadzonych eksperymentów. Dalsza część rozdziału poświęcona została analizie wpływu liczby równolegle wykonywanych procesów, wpływu częstotliwości komunikacji, wpływu wielkości populacji początkowej oraz wpływu liczby rozwiązań potomnych na jakość otrzymywanych wyników. Oryginalny dorobek rozprawy zawarty został w rozdziałach 6-8. W ostatnim rozdziale przedstawiono podsumowanie rozprawy oraz oceną realizacji wszystkich celów szczegółowych wraz z uzasadnieniem postawionej tezy. Do pracy dołączono trzy dodatki zawierające szczegółowe zestawienie wyników równoległego algorytmu memetycznego oraz aktualne zestawienie najlepszych rozwiązań pobranych ze strony ogranizacji SINTEF.

5. Algorytm memetyczny dla problemu VRPTW

Algorytm memetyczny wykorzystujący krzyżowanie rozwiązań za pomocą operatora EAX (ang. *edge assembly crossover*) dla problemu trasowania pojazdów z oknami czasowymi został opracowany w 2010 r. przez Nagatę i in. [7]. Metaheurystyka ta podzielona została na dwa etapy. W pierwszym etapie minimalizowana była liczba tras za pomocą heurystyki Nagaty i Bräysy'ego [6] opartej na puli usuniętych klientów oraz strategii lokalnego poszukiwania z przewodnikiem. W drugim etapie minimalizacji ulegała sumaryczna długość tras. Opracowany algorytm memetyczny oparty został na krzyżowaniu osobników z bieżącej populacji za pomocą operatora EAX. Wygenerowane rozwiązania potomne ulepszane są następnie przez operacje lokalnych poszukiwań.

Algorytm Nagaty i Bräysy'ego [6] dla minimalizacji liczby tras w problemie trasowania pojazdów z oknami czasowymi został oparty w głównej mierze na idei puli usuniętych klientów oraz strategii lokalnych poszukiwań z przewodnikiem. Algorytm minimalizacji sumarycznej długości tras dla problemu VRPTW opracowany przez Nagatę i in. [7] składa się z dwóch etapów. W pierwszym etapie konstruowana jest początkowa populacja rozwiązań o liczebności N_{pop} za pomocą algorytmu minimalizacji liczby tras (przedstawionego powyżej). W drugim etapie przeprowadzana jest minimalizacja sumarycznej długości tras za pomocą algorytmu wykorzystującego operacje EAX oraz metody lokalnych poszukiwań.

6. Ulepszenia sekwencyjnej wersji algorytmu memetycznego

Do algorytmów Nagaty i in [6, 7] opracowano ulepszenia mające na celu zwiększenie szybkości działania. Wstawianie klientów z usuwanej trasy w kolejności

losowej do puli *EP*. Umożliwiło to większe zróżnicowanie działania algorytmu w poszczególnych próbach usunięcia losowej trasy oraz wyeliminowało zjawisko podobnego wyniku, jak w poprzedniej próbie usunięcia tej samej trasy. Ograniczono liczbę usuwanych klientów przy próbie usuwania trasy, ponieważ zauważono, że gdy liczba klientów w EP jest zbyt duża, to prawdopodobieństwo ich ponownego wstawienia do tras jest małe. Wprowadzono listę zabronionych klientów, których nie można usunąć po wstawieniu ich do tras. Bez tego ulepszenia występowało cykliczne wstawianie i usuwanie tych samych klientów. Sprawdzanie najlepszej kombinacji wstawiania-usuwania ze stopniowym usuwaniem coraz większej liczby klientów rozpoczynając od 1, a kończąc na k_{max} klientów. Zauważono bowiem, że operacje usunięcia dwóch lub więcej klientów ($k_{max} \geq 2$) zdarzają się stosunkowo rzadko. Zaproponowany sposób umożliwił nawet kilkukrotne przyspieszenie działania tej części algorytmu. Początkowe rozwiązania populacji poddawane są dodatkowo serii lokalnych poszukiwań w celu poprawy ich jakości przed przystąpieniem do generowania kolejnych generacji rozwiązań potomnych. Ulepszenia te w sposób istotny zmniejszyły złożoność czasową algorytmu co umożliwiło wykonanie większej liczby iteracji (usuwanie tras, generowanie kolejnych populacji), przy ograniczeniach na czas działania. Przeprowadzone badania eksperymentalne ulepszonej wersji algorytmu na zbiorze danych testowych GH umożliwiły znalezienie dwóch nowych rozwiązań w rankingu światowym ze względu na minimalną liczbę tras [1].

7. Równoległy algorytm memetyczny

Ulepszony sekwencyjny algorytm stał się podstawą opracowania równoległego, dwuetapowego algorytmu memetycznego. W pierwszym etapie, minimalizacji poddawana jest liczba tras za pomocą heurystyki opartej na puli usuniętych klientów oraz metodzie lokalnych poszukiwań z przewodnikiem. W drugim etapie, poświęconemu minimalizacji sumarycznej długości tras, na wygenerowanej populacji rozwiązań konstruowane są za pomocą algorytmu memetycznego kolejne generacje osobników przy użyciu operatora krzyżowania EAX. Ze względu na możliwość rozdzielenia etapów, opracowano dwa oddzielne algorytmy równoległe. Pseudokod równoległego algorytmu minimalizacji liczby tras został przedstawiony na rys. 2, a równoległego algorytmu memetycznego na rys. 3.

W obu równoległych algorytmach kooperacja między procesami $P_1, P_2, \ldots, P_p$ odbywa się okresowo przez wymianę najlepszych znalezionych rozwiązań. Kooperacja między procesami wykonywana jest w celu zwiększenia jakości rozwiązań oraz zredukowania czasu działania algorytmu dzięki lepszej zbieżności obliczeń. Celem zmniejszenia kosztu fazy kooperacji między procesami zarówno dla równoległego algorytmu minimalizacji liczby tras, jak i równoległego algorytmu memetycznego, wprowadzone zostały asynchroniczne operacje dla funkcji wysyłania

1: **Dane wejściowe:** m - wymagana liczba tras (opcjonalnie), T_{max_1} - limit czasowy, δ_1 - liczba iteracji między kolejnymi fazami kooperacji, lz_{max} - pojemność listy zabronionych klientów, k_{max}, c_d
2: **Dane wyjściowe:** σ_{najl} - najlepsze rozwiązanie ze wszystkich procesów (gdy $m \neq 0$) lub rozwiązanie z minimalną liczbą tras (gdy $m = 0$)
3: **function** *RównoległyAlgMinLiczbyTras*$(m, T_{max_1}, \delta_1, lz_{max}, k_{max}, c_d)$
4: **for** $P_i \leftarrow P_1$ **to** P_p **do in parallel**
5: $\sigma_t :=$ rozwiązanie początkowe σ_{pocz} ▷ σ_t - tymczasowe rozwiązanie
6: $koniec :=$ **false**
7: **while not** *koniec* **do**
8: $\sigma_{najl} := \sigma_t$ ▷ σ_{najl} - najlepsze rozwiązanie
9: zainicjuj losowo pulę EP klientami z trasy r usuniętej z σ_t
10: $c_m := R(EP) + c_d$ ▷ $R(EP)$ - rozmiar puli EP
11: inicjuj liczniki kar dla wszystkich klientów $e[v_{i\in\{1,\ldots,n\}}] := 1$
12: **while** $(EP \neq \emptyset)$ **and** $(R(EP) \leq c_m)$ **and** (**not** *koniec*) **do**
13: $it := it + 1$ ▷ it - licznik iteracji
14: wybierz i usuń klienta v_w z puli EP $(v_w \notin LZ)$
15: **if** $S_w^a(v_w, \sigma_t) \neq \emptyset$ **then**
16: $\sigma_t :=$ losowe rozwiązanie $\sigma' \in S_w^a(v_w, \sigma_t)$
17: dodaj v_w do listy zabronionych klientów LZ
18: **else**
19: $\sigma_t := Zaciskanie(v_w, \sigma_t)$
20: **end if**
21: **if** $v_w \notin \sigma_t$ **then**
22: $e[v_w] := e[v_w] + 1$
23: $\sigma_t :=$ znajdź $\sigma' \in S_u^a(v_w, \sigma_t)$ z $\min P_{sum} = \sum_{j=1}^{k} e[v_u^{(j)}]$
24: dodaj usuniętych klientów $\{v_u^{(1)}, \ldots, v_u^{(k)}\}$ do puli EP
25: dodaj v_w do listy zabronionych klientów LZ
26: $\sigma_t := Zaburzanie(\sigma_t)$
27: **end if**
28: **if** it **mod** $\delta_1 = 0$ **then**
29: $koniec := Kooperacja1\,(\sigma_{najl}, koniec)$ ▷ kooperacja
30: **end if**
31: **end while**
32: **if** $EP \neq \emptyset$ **then**
33: $\sigma_t := \sigma_{najl}$
34: **else**
35: $\sigma_{najl} := \sigma_t$
36: **end if**
37: **if** $(m > 0)$ **and** $(\mathrm{R}(\sigma_t) = m)$ **then** ▷ $R(\sigma_t)$ - liczba tras σ_t
38: **break**
39: **end if**
40: **end while**
41: **while not** *koniec* **do**
42: $koniec := Kooperacja1\,(\sigma_{najl}, koniec)$
43: **end while**
44: **end for**
45: **return** najlepsze rozwiązanie σ_{najl} ze wszystkich procesów ▷ zwraca P_1
46: **end function**

Rysunek 2: Równoległy algorytm minimalizacji liczby tras.

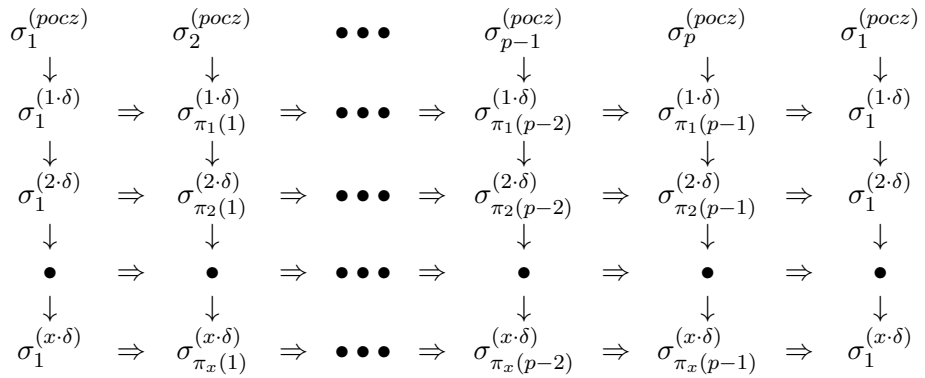
oraz odbierania rozwiązania. Schematy kooperacji w obu algorytmach są identyczne, a jedyne różnice występują w danych, które są wysyłane oraz operacjach wykonywanych podczas oczekiwania na potwierdzenie odebrania danych. Proces P_1 jest wyróżniony i traktowany jako proces główny, gdyż decyduje o zakończeniu działania algorytmu oraz zwraca najlepsze rozwiązanie uzyskane od wszystkich procesów. Aby zapobiec zakleszczeniu, operacje wysyłania i odbierania danych wykonywane są w procesie głównym P_1 w odwrotnej kolejności niż w pozostałych procesach.

W zaproponowanym równoległym algorytmie memetycznym zastosowano podstawowy schemat kooperacji między procesami oparty na strukturze pierścienia oraz oryginalny, randomizowany schemat kooperacji między procesami z możliwością krzyżowania rozwiązań przy użyciu operatora EAX. Jedną z zalet drugiego schematu jest mała złożoność czasowa – $O(pn)$ dla wersji bez zastosowania operatora EAX oraz $O(pn^2)$ przy użyciu operatora EAX, gdzie p jest liczbą procesów, a n liczbą klientów. Randomizowany schemat polega na generowaniu w każdej fazie kooperacji, losowej sekwencji numerów procesów, między którymi przesyłane są najlepsze rozwiązania. Stosowanie operatora EAX podczas wymiany rozwiązań między procesami jest korzystne, gdyż zawsze jedno z rozwiązań (argumentów operatora) jest najlepszym rozwiązaniem w danym procesie, a powstałe po krzyżowaniu rozwiązania mogą okazać się jeszcze lepsze. Kolejną zaletą schematu jest asynchroniczna komunikacja z nakładaniem obliczeń na operacje przesyłania danych, co umożliwia dobre wykorzystanie czasu procesów, zmniejszając okresy ich bezczynności. Zastosowanie dwuetapowego sposobu kooperacji między procesami umożliwia z kolei zmniejszenie jej złożoności. Niska złożoność czasowa faz kooperacji jest istotna w przypadku nałożonych limitów czasów wykonania obu etapów równoległego algorytmu memetycznego, gdyż umożliwia wykonanie większej liczby prób usunięcia trasy w pierwszym etapie oraz utworzenia większej liczby generacji w drugim. Randomizowany schemat kooperacji między procesami został przedstawiony na rys. 4.

W rozprawie doktorskiej przeprowadzono analizę teoretyczną zaproponowanego równoległego algorytmu memetycznego. W pierwszej kolejności wykazano jego poprawność częściową oraz własność dochodzenia obliczeń do punktu końcowego, tzw. własność stopu. Ze względu na to, że zaproponowany algorytm jest algorytmem równoległym udowodniono spełnienie własności bezpieczeństwa oraz żywotności. W dalszej kolejności przeprowadzono analizę złożoności czasowych oraz pamięciowych, pesymistycznych i średnich algorytmu. Analizy wykazały, że algorytm jest w pełni poprawny i charakteryzuje się pesymistyczną złożonością czasową $T_{pes}(p,n) = O(n^{k_{max}+1} + pn^2)$, średnią złożonością czasową $T_{sr}(p,n) = O(n^{2,7} + pn^2)$ oraz pesymistyczną złożonością pamięciową $S_{pes}(p,n) = O(n^{k_{max}+2} + pn^3)$, gdzie k_{max} jest maksymalną liczbą klientów możliwych do usunięcia z rozwiązania w trakcie próby wstawienia klienta z puli EP.

```
1: Dane wejściowe: wartości zmiennych N_pop, N_dz, T_max1, T_max2, T_GenPop, δ1,
   δ2, k_max, lz_max, c_d
2: Dane wyjściowe: σ_najl - najlepsze rozwiązanie w całej populacji
3: function RównoległyAlgMemetyczny(N_pop, N_dz, T_max1, T_max2, T_GenPop, δ1,
   δ2, k_max, lz_max, c_d)
4:     // Etap pierwszy - generowanie początkowej populacji rozwiązań pop
5:     σ1 := RównoległyAlgMinLiczbyTras(0, T_max1, δ1, lz_max, k_max, c_d)
6:     m := R(σ1)                                   ▷ m - minimalna liczba tras
7:     σ1 := LokalnePoszukiwania(σ1)
8:     for j ← 2 to N_pop do
9:         σ_j := RównoległyAlgMinLiczbyTras(m, T_max1, δ1, lz_max, k_max, c_d)
10:        σ_j := LokalnePoszukiwania(σ_j)
11:    end for
12:    // Etap drugi - minimalizowanie sumarycznej długości tras
13:    for P_i ← P_1 to P_p do in parallel
14:        koniec := false
15:        g := 0                                   ▷ g - licznik generacji
16:        while not koniec do
17:            wygeneruj losową permutację π(j) ∈ {1, 2, ..., N_pop}
18:            for j ← 1 to N_pop do
19:                σ_A := σ_π(j)
20:                σ_B := σ_π((j+1)%N_pop)
21:                if σ_A ≠ σ_B then
22:                    σ_t := σ_A
23:                    for k ← 1 to N_dz do
24:                        σ_dz := EAX(σ_A, σ_B)
25:                        σ_dz := Napraw(σ_dz)
26:                        σ_dz := LokalnePoszukiwania(σ_dz)
27:                        if (σ_dz jest prawidłowe) and (F_sd(σ_dz) < F_sd(σ_t)) then
28:                            σ_t := σ_dz
29:                        end if
30:                    end for
31:                    σ_π(j) := σ_t
32:                else
33:                    σ_π(j) := Zaburzanie(σ_π(j))
34:                end if
35:            end for
36:            g := g + 1
37:            if g mod δ2 = 0 then
38:                koniec := Kooperacja2(σ_najl, koniec)   ▷ faza kooperacji
39:            end if
40:        end while
41:    end for
42:    return najlepsze rozwiązanie σ_najl w całej populacji ▷ zwracane przez P1
43: end function
```

Rysunek 3: Równoległy algorytm memetyczny dla problemu VRPTW.



Rysunek 4: Randomizowany schemat kooperacji p procesów (σ_1 - rozwiązanie procesu głównego P_1; $\sigma_i^{(pocz)}$ - rozwiązanie początkowe procesu P_i; $\sigma_{\pi_j(i)}^{(j\cdot\delta)}$ - rozwiązanie procesu $P_{\pi_j(i)}$ w j-tej fazie kooperacji; $\pi_j(i)$ - i-ty element losowej permutacji $\pi_j \in \{2, 3, \ldots, p\}$ w j-tej fazie kooperacji; x - liczba faz kooperacji; δ - liczba generowanych populacji lub prób usunięcia trasy między kolejnymi fazami kooperacji; $\Rightarrow$ - oznacza transfer danych; $\downarrow$ - oznacza postęp działania algorytmu).

8. Opis przeprowadzonych eksperymentów

Dodatkowym celem rozprawy była analiza wpływu współpracy równolegle wykonywanych procesów memetycznego algorytmu na szybkość i jakość uzyskiwanych rozwiązań.

Analiza wpływu współpracy równoległych procesów na jakość otrzymywanych rozwiązań dla równoległego algorytmu minimalizacji liczby tras wykazała, że randomizowany schemat kooperacji z krzyżowaniem rozwiązań umożliwił uzyskanie lepszych wartości CVN w stosunku do schematu podstawowego. Wynik ten był najbardziej zauważalny dla grup z dużą liczbą klientów (≥ 600) z klientami rozmieszczonymi w klastrach (C1 i C2). Biorąc pod uwagę wszystkie 356 testy ze zbiorów S oraz GH, równoległy algorytm z randomizowanym schematem pozwolił obniżyć wartość CVN o 9 pojazdów (z 10709 do 10700). Ze względu na minimalną liczbę tras uzyskano łącznie 21 nowych światowych rozwiązań, z czego 13 za pomocą równoległego algorytmu z podstawowym schematem kooperacji i 8 dodatkowych z randomizowanym schematem z krzyżowaniem rozwiązań [2, 3]. Warto podkreślić, że dla 50% nowych światowych rozwiązań, średni czas działania równoległego algorytmu minimalizacji liczby tras był nie dłuższy niż 5 minut (dla prób z użyciem 64 procesorów). Stanowi to potwierdzenie tezy rozprawy, mówiącej o tym, że randomizowany schemat kooperacji między procesami z krzyżowaniem rozwiązań w równoległym algorytmie memetycznym umożliwia uzyskanie wyników o lepszej jakości dla problemu VRPTW.

Przeprowadzona analiza wpływu kooperacji procesów dla równoległego al-

Tabela 1.: Zestawienie najlepszych wyników równoległego algorytmu memetycznego dla zbioru GH o 1000 klientach (gwiazdka ($\star$) oznacza wynik gorszy, font zwykły – wynik równy światowemu rozwiązaniu, font pogrubiony – wynik lepszy).

1000 klientów			
	C1	R1	RC1
1	100/42478.95	**100/53762.92**	**90/46743.83**
2	**90/42509.62**	91/49819.61*	90/44720.62*
3	**90/40356.18**	91/46136.08*	90/42886.30*
4	90/39641.46*	91/43402.31*	90/41841.87*
5	100/42469.18	91/53971.76*	**90/45949.78**
6	**99/44108.34**	91/48615.92*	90/45799.43*
7	**97/44806.73**	91/45383.56*	90/45361.67*
8	94/41853.36*	91/43072.25*	90/44791.07*
9	**90/41006.16**	91/51093.89*	**90/44882.70**
10	90/40229.20*	91/49423.01*	**90/44391.00**
	C2	R2	RC2
1	30/16879.24	**19/42219.21**	**20/30278.50**
2	**29/17126.39**	**19/33586.49**	**18/26677.87**
3	**28/16884.08**	**19/25309.46**	**18/20177.96**
4	**28/15743.88**	**19/18182.09**	**18/15820.37**
5	**30/16561.57**	**19/36335.72**	**18/27269.46**
6	**29/16920.33**	**19/30247.98**	**18/26965.51**
7	**29/17882.42**	**19/23381.36**	**18/25317.29**
8	**28/18343.01**	**19/17598.63**	**18/24010.00**
9	**29/16442.47**	**19/33131.99**	**18/23341.21**
10	**28/15988.21**	**19/30656.00**	**18/22372.41**

gorytmu memetycznego potwierdziła jej wysoką skuteczność. W rezultacie wykonanych eksperymentów uzyskano łącznie 171 nowych rozwiązań (stan w dn. 5.04.2013) w światowym rankingu organizacji SINTEF najlepszych wyników dla zbiorów danych GH [4]. Przykładowe zestawienie najlepszych wyników równoległego algorytmu memetycznego dla zbioru danych GH o 1000 klientach zostało przedstawione na rys. 1.. Porównując uzyskane rozwiązania z najlepszymi wynikami opublikowanymi na stronie organizacji SINTEF można zauważyć, że dla wszystkich podgrup C1, C2, R1, R2, RC1 oraz RC2 ze zbioru danych GH udało się uzyskać lepszy rezultat. W zbiorze wszystkich testów uzyskano ulepszenie rzędu 3,05%, co również potwierdza tezę rozprawy doktorskiej.

Analiza wpływu liczby równoległych procesów na szybkość i jakość otrzymywanych rozwiązań wykazała, że równoległy algorytm minimalizacji liczby tras z randomizowanym schematem kooperacji z krzyżowaniem rozwiązań uzyskuje lepsze wartości przyspieszeń w stosunku do wyników algorytmu z podstawowym schematem kooperacji. Większą jakość dla pierwszego z nich uzyskano dla wszystkich sześciu wybranych testów, a najbardziej zauważalne różnice uzyskano dla prób

przeprowadzonych przy dużej liczbie procesów (≥ 16). Ponadto, randomizowany schemat kooperacji przyczynił się do lepszego rezultatu osiągniętego dla testów z grupy C niż z RC, w odróżnieniu od podstawowego schematu kooperacji, dla którego lepszy rezultat uzyskano z kolei dla testów z grupy RC. Ta obserwacja może wskazywać na to, że korzyści płynące z randomizowanego schematu i krzyżowania rozwiązań podczas kooperacji przewyższają koszt komunikacji między procesami oraz ich synchronizacji. Lepsze wyniki prób z randomizowanym schematem kooperacji wynikają ze specyficznych właściwości operatora EAX, który generuje rozwiązania potomne bez wprowadzania długich krawędzi między klientami, co jest bardzo istotne w przypadku testów z klientami umieszczonymi w klastrach.

Analiza wpływu częstotliwości kooperacji między procesami na jakość otrzymywanych rozwiązań wykazała, że wartość okresu kooperacji δ_2 ma duży wpływ na jakość uzyskiwanych wyników. Okazało się, że zwiększanie liczby konstruowanych generacji (równej okresowi kooperacji δ_2) między kolejnymi kooperacjami może mieć negatywny lub pozytywny wpływ na jakość rozwiązań w zależności od typu badanej grupy testów (C lub RC). Dla testów z klientami umieszczonymi w klastrach, większa wartość okresu kooperacji δ_2, a tym samym mniejsza częstotliwość kooperacji, umożliwiła otrzymanie rozwiązań o wiele lepszej jakości. Dla testów z klientami rozmieszczonymi w sposób losowy, to mniejsza wartość okresu kooperacji δ_2, a tym samym większa częstotliwość kooperacji, umożliwiła uzyskanie lepszych wyników. Zauważalny, istotny wpływ częstotliwości kooperacji na jakość otrzymanych rezultatów wynika ze średniej liczby generacji, które mogły zostać skonstruowane w zadanym limicie czasowym. Tak więc wyniki badań wpływu częstotliwości kooperacji na jakość otrzymywanych rozwiązań są niejednoznaczne.

Analiza wpływu wielkości populacji początkowych rozwiązań wykazała, że biorąc pod uwagę tą samą liczbę konstruowanych generacji, najlepsze wyniki uzyskano dla największej badanej populacji początkowej ($N_{pop} = 200$). Kolejne generacje konstruowane są najszybciej dla prób z małą populacją rozwiązań, jednakże ze względu na małe zróżnicowanie rozwiązań bardzo szybko następuje stagnacja i następne generacje nie pozwalają na otrzymanie lepszych jakościowo wyników. Duże populacje ($N_{pop} = 100, 200$) umożliwiają najwyższe tempo zmian sumarycznych długości tras najlepszych rozwiązań w trakcie konstruowania kolejnych generacji.

Analiza wpływu liczby konstruowanych rozwiązań potomnych wykazała, że rozwiązania o najlepszej jakości uzyskano dla prób z największą liczbą generowanych rozwiązań-dzieci. Im większa liczba generowanych rozwiązań potomnych, tym większe szanse na znalezienie najlepszego rozwiązania potomnego zawierającego najlepsze cechy krzyżowanych rozwiązań. Analiza wyników wykazała jednak, że różnice między wartościami $N_{dz} = 50$, a $N_{dz} = 100$ są niewielkie, co może oznaczać, że dalsze zwiększanie liczby konstruowanych rozwiązań potomnych nie pociągnie za sobą znacznej poprawy jakości otrzymywanych wyników, a jedynie

wydłuży czas obliczeń.

9. Podsumowanie

Na podstawie analizy złożoności równoległego algorytmu memetycznego można stwierdzić, że pesymistyczna złożoność czasowa tego algorytmu jest $O(n^4 + pn^2)$, a średnia złożoność czasowa $O(n^{2,7} + pn^2)$. Pesymistyczna złożoność pamięciowa tego algorytmu okazała się nieco wyższa i wyniosła $O(n^5 + pn^2)$. Analiza wpływu liczby równolegle pracujących procesów na jakość otrzymywanych rozwiązań wykazała, że wykorzystanie randomizowanego schematu kooperacji między procesami umożliwia uzyskiwanie lepszych wartości przyspieszeń niż podstawowy schemat kooperacji. Efektywność wykorzystania procesów okazała się najwyższa dla prób z użyciem 8–24 procesów i wyniosła średnio 70%. Dla prób z użyciem liczby procesów od 25 do 64 efektywność była nieco niższa i wyniosła średnio 64%. Równoległy algorytm uzyskał najlepsze wyniki dla testów z klientami umieszczonymi w klastrach (C1, C2) oraz dla testów z losowymi klientami przy założeniu szerokich okien czasowych (R2, RC2) bez wględu na rozmiar problemu. Wpływ na to miał prawdopodobnie również randomizowany schemat kooperacji, w którym dokonuje się operacji krzyżowania z użyciem tego operatora. Przeprowadzona analiza wyników wykazała, że randomizowany schemat kooperacji między procesami z rekombinacją rozwiązań w równoległym algorytmie memetycznym umożliwia uzyskanie wyników o lepszej jakości. Stanowi to jednocześnie potwierdzenie tezy rozprawy. Skonstruowany równoległy algorytm memetyczny dotyczy tylko problemu VRPTW, jednak opracowane schematy kooperacji między procesami mogą znaleźć zastosowanie zarówno w innych wariantach problemu trasowania pojazdów, jak i innych problemach optymalizacji dyskretnej.

Analiza wyników eksperymentalnych oraz przedstawione wnioski i obserwacje stanowią dobrą podstawę do dalszego rozwijania równoległego algorytmu memetycznego. W szczególności, można rozważyć ulepszenia algorytmu pod kątem testów z losowymi klientami o wąskich oknach czasowych (R1, RC1), dla których uzyskane wyniki okazały się nieco gorsze od testów z pozostałych grup. Najprawdopodobniej będzie to związane z modyfikacją operatora krzyżowania EAX. Dalsze badania mogą być również związane ze zrównolegleniem algorytmów dla innych wariantów problemu trasowania pojazdów ze szczególnym uwzględnieniem randomizowanego schematu kooperacji z krzyżowaniem rozwiązań.

Bibliografia

[1] Błocho M., Czech Z.J. *An improved route minimization algorithm for the vehicle routing problem with time windows.* Studia Informatica 2011, Vol.32, No. 3B(99), Gliwice, s. 5–19.

[2] Błocho M., Czech Z.J. *A parallel algorithm for minimizing the number of routes in the vehicle routing problem with time windows.* PPAM 2011, Part I, LNCS 7203, s. 255–265. Springer, Heidelberg, 2012.

[3] Błocho M., Czech Z.J. *A parallel EAX-based algorithm for minimizing the number of routes in the vehicle routing problem with time windows.* In: Proceedings of the 2012 IEEE 14th International Conference on High Performance Computing and Communications, Liverpool, 2012, s. 1239-1246.

[4] Błocho M., Czech Z.J. *A parallel memetic algorithm for the vehicle routing problem with time windows.* 3PGCIC - 8th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing, Compiegne, France, 2013.

[5] Gehring H., Homberger J. *A parallel hybrid evolutionary metaheuristic for the vehicle routing problem with time windows.* In: Proceedings of EUROGEN99–Short Course on Evolutionary Algorithms in Engineering and Computer Science. 1999, s. 57–64.

[6] Nagata Y., Bräysy O. *A powerful route minimization heuristic for the vehicle routing problem with time windows.* Operations Research Letters, 2009, 37(5), s. 333-338.

[7] Nagata Y., Bräysy O., Dullaert W. *A penalty-based edge assembly memetic algorithm for the vehicle routing problem with time windows.* Computers & Operations Research, 2010, 37(4), s. 724–737.

[8] Solomon M.M. *Algorithms for the vehicle routing and scheduling problems with time window constraints.* Operations Research, 1987, 35(2), s. 254–265.