Akademia Górniczo-Hutnicza im. Stanisława Staszica

w Krakowie

Wydział Inżynierii Mechanicznej i Robotyki

Rozprawa doktorska

mgr inż. Marcin Woch

STEROWANIE DYSKRETNYMI PROCESAMI DYSTRYBUCJI W LOGISTYCE

Opiekun naukowy :

dr hab. inż. Piotr Łebkowski, prof AGH

Kraków, 2011

AGH

AKADEMIA GÓRNICZO-HUTNICZA
IM. STANISŁAWA STASZICA W KRAKOWIE

Promotorowi,
Panu prof. dr hab. inż. Piotrowi Łebkowskiemu, serdecznie dziekuję za cenne uwagi, udzieloną pomoc oraz poświęcony mi czas w trakcie realizacji niniejszej pracy.

Spis treści

Oznaczenia i skróty stosowane w pracy 5
Symbole stosowane we wzorach 7
1. Wstęp 8
2. Problemy dostaw oraz metody ich rozwiązywania 15
2.1. Problemy dostaw 15
2.1.1. Przegląd badań dotyczących problemów dostaw 20
2.2. Koncepcje badawcze i tezy rozprawy 29
2.2.1. Podsumowanie stanu badań w odniesieniu do specyfiki zadania 30
2.2.2. Tezy rozprawy 34
3. Rozwiązanie problemów dostaw 37
3.1. Rozwiązanie problemu dostaw za pomocą heurystyki z wykorzystaniem metody sympleksowej 37
3.2. Rozwiązanie problemu dostaw za pomocą heurystyki z wykorzystaniem programowania dynamicznego 47
3.3. Zastosowane heurystyki do rozwiązywania problemów dostaw 55
3.4. Opis zastosowanej hybrydy symulowanego wyżarzania oraz adaptacyjnego przeszukiwania dużego sąsiedztwa 62
3.5. Parametry ogólne algorytmu symulowanego wyżarzania 77
3.5.1. Temperatura początkowa 78
3.5.2. Funkcja redukcji temperatury 79
3.5.3. Długość epoki 79
3.5.4. Warunki zatrzymania 80
3.6. Funkcja kosztu 80
3.7. Weryfikacja algorytmu 81
3.8. Problem testowy oraz otrzymane wyniki 81
3.8.1. Zbiory testowe zastosowane do testowania problemów dostaw 81
3.8.2. Wyniki testów Solomona 87
3.8.3. Wyniki testów Cordeau 92
3.8.4. Wyniki testów opartych na danych rzeczywistych 94
4. Implementacja algorytmu w systemie Microsoft Dynamics NAV 2009 97

4.1. Implementacja algorytmu 98
4.1.1. Ustawienia modułu „Planowanie wysyłek” 98
4.1.2. Tworzenie listy wysyłek 101
4.1.3. Wyznaczenie oraz obliczanie tras 102
4.2. Budowa modułu „Planowanie wysyłek” 108
4.2.1. Budowa aplikacji 109
4.2.2. Szczegółowy opis zmian 112
5. Wnioski końcowe i kierunki dalszych badań 122
5.1. Podsumowanie badań i wnioski 122
5.2. Dalsze kierunki badań 127
Literatura 129
Załącznik 1 – Testy VRPTW M. Solomona 139
Załącznik 2 – Testy MDVRPTW Cordeau 197
Załącznik 3 – Transportation Optimization Portal 237
Załącznik 4 – Szczegółowe wyniki testów VRPTW Solomona 243
Załącznik 5 – Szczegółowe wyniki testow MDVRPTW 259

Oznaczenia i skróty stosowane w pracy

Skrót	Język angielski	Język polski
ACO	Ant Colony System	system mrówkowy
ALNS	Adaptive Large Neighbourhood Search	adaptacyjne przeszukiwanie dużego sąsiedztwa
C/AL	C/SIDE Application Language	język środowiska C/SIDE
CRM	Customer Relationship Management	zarządzanie relacjami z klientem
C/SIDE	Client - Server Integrated Development Environment	zintegrowane środowiko programistycze klient - serwer
CVRP	Capacity Vehicle Routing Problem	problem dostaw ze zdefiniowanymi nośnościami pojazdów
ERP	Enterprise Resource Planning	zintegrowany system zarządzania przedsiębiorstwem
GIS	Geographical Information System	system informacji przestrzennej, system informacji geograficznej
MDVRP	Multi Depot Vehicle Routing Problem	problem dostaw z wieloma magazynami
MDVRPTW	Multi Depot Vehicle Routing Problem with Time Windows	problem dostaw z wieloma magazynami z oknami czasowymi
MPS	Master Production Schedule	harmonogram produkcji
MRP 1	Material Requirement Planning	planowanie zapotrzebowanie materiałowego
MRP 2	Manufacturing Resource Planning	planowanie zasobów produkcyjnych
MTSP	Multiple Travelling Salesman Problem	problem wielu komiwojażerów
OVRP	Open Vehicle Routing Problem	otwarty problem dostaw
PK	Primary Key	klucz główny
PTSP	Periodic Travelling Salesman Problem	okresowy problem komiwojażera
PVRP	Periodic Vehicle Routing Problem	okresowy problem dostaw
RDPTW	Rich Delivery Problem with Time Windows	ogólny problem dostaw z oknami czasowymi
RTC	Role Tailored Client	klient zorientowany zadaniowo
SDVRP	Site-Dependent Vehicle Routing Problem	problem dostaw z różnymi nośnościami pojazdów
SVRP	Stochastic Vehicle Routing Problem	stochastyczny problem dostaw
VRPSD	Vehicle Routing Problem with Stochastic Demands	problemu dostaw ze stochastycznymi zapotrzebowaniami
TSP	Travelling Salesman Problem	problem komiwojażera
VLNS	Very Large Neighborhoods Search	szerokie przeszukiwanie sąsiedztwa
VNS	Variable Neighborhood Search	zmienne przeszukiwanie sąsiedztwa

WOK	Multi-Criterion Combinatorical Optimisation	wielokryterialna optymalizacja kombinatorycznej
VRP	Vehicle Routing Problem	problem dostaw
VRPTW	Vehicle Routing Problem with Time Windows	problem dostaw z oknami czasowymi

Symbole stosowane we wzorach

Symbol	Opis
$G(V,A)$	graf przedstawiający model ogólnego problemu dostaw
V	zbiór wierzchołków odwzorowujących magazyny
A	zbiór łuków odwzorowujących odległości
$v_1,...,v_m \in V$	wierzchołki grafu G odwzorowujące magazyny
$v_{m+1},...,v_n \in V$	wierzchołki grafu G odwzorowujące klientów
d_i	zapotrzebowanie wierzchołka i
s_i	czas obsługi wierzchołka i
$[e_i, l_i]$	okno czasowe wierzchołka i
e_i	najwcześniejszy możliwy czas obsługi wierzchołka i
l_i	najpóźniejszy możliwy czas obsługi wierzchołka i
$t_i \in K$	zbiór pojazdów
D_k	ładowność pojazdu k
X_k, Y_k, Z_k	wymiary pojazdu k
T_k	czas przejazdu całej trasy
$S_i \in R_0^+$	rozpoczęcie obsługi klienta i
$L_i \in R_0^+$	ilość towaru
$c_{\bar{r}}$	koszt przejazdu na trasie $\bar{r}$
$\bar{r}$	prawidłowa trasa
X_{ij}^k	zmienna równa 1, gdy pojazd k przemieszcza się z wierzchołka i do j, a 0 w przeciwnym wypadku
P	prawdopodobieństwo akceptacji gorszego stanu w algorytmie symulowanego wyżarzania
E	liczba Eulera, liczba Nepera
K	stała Boltzmana
T	temperatura
E	energia systemu
$X(k)$	zmienna oznaczająca wynik losowania k-tej próby
P_{ij}	prawdopodobieństwo warunkowe określające łańcuch Markova
$P=\|R\|*\|R$	macierz przejść w łancuchu Markova
$a_i(k)$	prawdopodobieństwo wyniku i w *k-tej* próbie
G_{ij}	prawdopodobieństwo wygenerowania stanu j ze stanu i
A_{ij}	prawdopodobieństwa akceptacji stanu j wygenerowanego ze stanu i
χ	funkcja charakterystyczna zbioru stanów systemu R
R	zbiór stanów systemu

1. Wstęp

Transport dóbr materialnych jest bardzo ważnym elementem aktywności człowieka. Prawidłowe zarządzanie siecią dystrybucyjną oraz parkiem pojazdów pozwala na znaczne obniżenie kosztów dostaw ponoszonych przez przedsiębiorstwa. Sterowanie dystrybucją sprowadza się najczęściej do rozwiązania pewnego problemu decyzyjnego i może być rozpatrywane w trzech płaszczyznach (Krawczyk S., 2001): strategicznej, taktycznej oraz operacyjnej. Decyzje strategiczne uwzględniają położenie i rozmieszczenie magazynów, firm lub/oraz fabryk. Decyzje taktyczne determinują strukturę sieci dystrybucyjnej oraz park pojazdów. Natomiast samo określenie tras oraz liczby pojazdów użytych do dostarczenia towarów do klientów jest decyzją operacyjną. Badania przedstawione w niniejszej pracy dotyczą metod sterowania procesem dystrybucji towarów do odbiorców, czyli decyzji operacyjnych.

Systemy informatyczne służące wspomaganiu zarządzaniem przedsiębiorstwem to tzw. systemy ERP (planowanie zasobów przedsiębiorstwa, ang. Enterprise Resource Planning). Na rynku tych systemów można zaobserwować zwiększone zapotrzebowanie na aplikacje wspomagające planowanie tras. Jest to związane z faktem, że nawet niewielka oszczędność na pojedynczych wysyłkach umożliwia znaczne obniżenie kosztów firmy rozpatrywanych w dłuższym okresie. Prawidłowe zarządzanie wysyłką pozwala na lepsze spełnianie potrzeb klientów. Sprzyja także poprawie wizerunku firmy.

Niniejsza tematyka rozprawy wynika z konkretnej praktycznej potrzeby. Firma z Wielkiej Brytanii produkująca rękawice dla potrzeb medycyny, wojska, przemysłu (rękawice ochronne w kopalniach, hutach) używa systemu Microsoft Dynamics NAV 2009. Firma sama wysyła towary do klientów, posiada własny park pojazdów, a także współpracuje z kilkoma firmami logistycznymi. Pojazdy transportujące towary do klientów wysyłane przez badaną firmę charakteryzują się różnymi wymiarami oraz różnymi nośnościami. Dziennie firma wysyła towary do kilkudziesięciu odbiorców. Zadaniem działu planowania jest ustalenie liczby wysłanych pojazdów, przejechanego przez nie dystansu oraz odpowiedni załadunek wysyłanych towarów do samochodów z uwzględnieniem ich wagi oraz rozmiarów. Obliczone w ten sposób trasy następnie są dodatkowo sprawdzane, czy za pomocą dostępnego parku pojazdów jest możliwe dostarczenie towarów do klientów. Jeśli dana trasa może być obsłużona przez więcej niż jeden pojazd, wtedy proponowany powinien być pojazd tańszy.

Problem badany w niniejszej pracy to problem wielokryterialny, w którym kryterium minimalizacji jest funkcją następujących zmiennych: liczby tras, przebytego sumarycznego dystansu oraz kosztu używania pojazdu. Ponadto wysyłki towarów do poszczególnych klientów uwzględniają rozmiary i wagi wysyłanych towarów.

W omawianym problemie klienci posiadają zdefiniowane zapotrzebowanie, a także godziny pracy magazynów, czyli tzw. okno czasowe. Każdy pojazd, dostarczający towary do końcowych użytkowników charakteryzuje się określoną nośnością, a zatem problem posiada cechy ogólnie znanych problemów dostaw VRPTW (Problem dostaw z oknami czasowymi, ang. Vehicle Routing Problem with Time Windows) oraz CVRP (Problem dostaw ze zdefiniowanymi nośnościami pojazdów, ang. Capatity Vehicle Routing Problem). W problemie VRPTW klienci posiadają określone zapotrzebowanie oraz okna czasowe, natomiast cechą CVRP są zdefiniowane nośności pojazdów.

W związku z faktem, że flota pojazdów badanej firmy nie jest homogeniczna (nośności oraz wymiary pojazdów mogą się rożnić), należało dokonać dalszego uogólnienia do problemu SDVRP (Problem dostaw z różnymi nośnościami pojazdów, ang. Site-Dependent Vehicle Routing Problem). Ponadto firma posiada kilka magazynów, dlatego badany problem posiada także cechy MDVRP (Problem dostaw z wieloma magazynami, ang. Multi Depot Vehicle Routing Problem). MDVRP jest problemem dostaw, w którym wysyłka towarów następuje z więcej niż jednego magazynu.

Najbardziej znanym problemem dostaw łączącym w sobie wszystkie cechy wymienionych zadań jest RDPTW (Ogólny problem dostaw z oknami czasowymi, ang. Rich Delivery Problem with Time Windows) (Ropke, Pisinger, 2005). W problemie RDPTW klienci posiadają zdefiniowane okna czasowe, towary charakteryzujące się wagą dostarczane są z jednego lub kilku magazynów, a flota pojazdów nie jest homogeniczna.

Analiza badań nad problemami dostaw oraz warianty problemów dostaw zostały szczegółowo opisane w rozdziale 2.

Problemy dostaw w związku z dużą złożonością obliczeniową należą do problemów NP-trudnych (Frederickson G, i in., 1978), dlatego ciągle trwają poszukiwania metody, która pozwoli na znalezienie rozwiązania bliskiego optymalnemu w czasie akceptowalnym w praktyce. Wiele problemów logistycznych ma charakter kombinatoryczny, tzn. polegają one na wybraniu optymalnego rozwiązania ze skończonego zbioru, zwanego zbiorem decyzyjnym. Z reguły zbiory decyzyjne charakteryzują się bardzo dużą licznością. W związku ze stosowaniem wielu kryteriów

przy podejmowaniu decyzji zagadnienia tego typu nazywane są problemami wielokryterialnej optymalizacji kombinatorycznej (WOK, ang. Multicriteria Combinatorial Optimization).

W literaturze światowej znaleźć można wiele doniesień o zastosowaniach zaawansowanych technik planowania wysyłek towarów. Narzędzia te jednak albo nie oferują możliwości integracji z systemi ERP, albo rozwiązania problemów dostaw dalekie są od optymalnych, bądź też w celu uzyskania optymalnych rozwiązań konieczny jest długi czas pracy algorytmu lub uruchomienie go równolegle na kilku procesorach.

Celem pracy jest zbudowanie metody zdolnej w zadowalającym czasie generować bliskie optymalnym plany wysyłek, ze względu na przyjęte kryteria.

W początkowych etapach swoich badań nad problemami dostaw autor niniejszej pracy użył własnych heurystyk wykorzystujących programowanie dynamiczne oraz metodę sympleksową. Wyniki przeprowadzonych testów pokazały, że uzyskane rozwiązania za pomocą tych metod są gorsze od obecnych najlepszych wyników światowych zarówno pod względem uzyskanej liczby tras i całkowitego dystansu, dlatego też dalszy rozwój tych heurystyk został przez autora niniejszej rozprawy zaprzestany. Uzyskane wyniki za pomocą tych heurystyk przedstawione zostały odpowiednio w rozdziałach 3.1 oraz 3.2.

Z tego powodu zdecydowano się zbudować algorytm w oparciu o ogólnie znane heurystyki. Przeanalizowano literaturę w celu znalezienia heurystyki charakteryzującej się rozwiązaniami na poziomie najlepszych światowych rozwiązań. Skoncentrowano się w szczególności na przeszukiwaniu tabu (ang. Tabu Search), algorytmach mrówkowych (ang. Ant Colony Systems), symulowanym wyżarzaniu (ang. Simulated Annealing) oraz różnych wariantach przeszukiwania sąsiedztwa lokalnego.

Najlepsze znane rozwiązania problemów testowych zamieszczone na stronie Transportation Optimization Portal (http://www.sintef.no/projectweb/top). Opublikowane algorytmy udowadniają, że symulowane wyżarzanie jest bardzo użyteczną metodą w rozwiązywaniu tego typu zagadnień. Sposób ten cechuje się wysoką efektywnością, czyli w stosunkowo krótkim czasie pozwala uzyskać rozwiązania bliskie optymalnym.

Ropke, Pisinger (Ropke, Pisinger, 2005) zastosowali algorytm oparty o ALNS (Adaptacyjne przeszukiwanie dużego sąsiedztwa, ang. Adaptive Large Neiborhood Search) do rozwiązania problemów SDVRPTW, MDVRPTW oraz ogólnego zadania RDPTW. Udowodniono wtedy efektywność ALNS przy optymalizowaniu tych problemów, gdyż

poprawiono wiele światowych rekordów (Ropke, Pisinger, 2005). Do dzisiaj wiele z tych wyników nie zostało polepszonych.

W związku z tym, że rozwiązywany problem nie sprowadza się tylko do VRPTW, ale także posiada cechy SDVRPTW oraz MDVRPTW, w niniejszej rozprawie, zdecydowano się użyć także algorytmu ALNS.

Jako główną koncepcję przyjęto, że podstawą poszukiwanego algorytmu będzie rozwiązanie hybrydowe na bazie dwóch heurystyk: symulowanego wyżarzania oraz adaptacyjnego przeszukiwania dużego sąsiedztwa. Do osiągnięcia tak postawionego celu usystematyzowano metody stosowane do wielokryterialnej optymalizacji planowania tras, przeprowadzono badania porównawcze metod optymalizacyjnych. Zbudowany algorytm postanowiono zweryfikować za pomocą trzech rodzajów danych testowych: zbiorów opracowanych przez M. Solomona, zbiorów Cordeau oraz danych rzeczywistych. Szczegółowy opis zbiorów testowych znajduje się w rozdziale 3.8.1.

Do osiągnięcia celu rozprawy należało zdefiniować nowy problem logistyczny będący rozszerzeniem ogólnego problemu dostaw RDPTW o następujące parametry:

- pojazdy oprócz nośności posiadają szerokość, długość oraz wysokość,
- dostarczane towary oprócz wagi posiadają szerokość, długość oraz wysokość,
- uwzględniane są koszty użytkowania pojazdów,
- każda trasa zaczyna się oraz kończy w tym samym centralnym magazynie.

Sformułowanie algorytmu będącego hybrydą symulowanego wyżarzania oraz adaptacyjnego przeszukiwania dużego sąsiedztwa, który pozwala znaleźć rozwiązania problemów dostaw w zadowalającym czasie oraz zbadanie czy tego typu hybryda pozwoli na rozwiązanie trójkryterialnego problemu dostaw są najważniejszymi elementami oryginalnymi rozprawy.

Z praktycznego punktu widzenia, istotną zaletą przyjętego podejścia jest możliwość stworzenia algorytmu, który może być z powodzeniem zaimplementowany w dowolnym systemie klasy ERP. Realizacja przedstawionego celu pracy winna przyczynić się do ułatwienia dostępu szerszej rzeszy przedsiębiorców do zaawansowanych technik planowania dostaw.

W niniejszej rozprawie do testów niniejszego algorytmu zdecydowano się użyć systemu ERP Microsoft Dynamics NAV 2009 produkowanego przez firmę Microsoft.

Dynamics NAV 2009 jest systemem składającym z tzw. modułów tworzących jedną całość. Moduły te pozwalają na pełną obsługę firm i są to:

- księgowość i finanse,

- zarządzanie sprzedażą, rozliczeniami z klientami,

- zarządzanie relacjami z klientami i kontaktami (moduł marketingowy) – CRM (Zarządzanie relacjami, ang. Customer Relationship Management),

- zakupy i rozliczenia z dostawcami,

- zarządzanie zasobami ludzkimi,

- zlecenia,

- zarządanie serwisem,

- produkcja z uwzględnieniem MPS (Harmonogram produkcji, ang. Master Production Schedule), MRP 1 (Planowanie zapotrzebowania materiałowego, ang. Material Requirement Planning) oraz MRP 2 (Planowanie zasobów produkcyjnych, ang. Manufacturing Resource Planning).

Wdrożenie systemu NAV polega na odpowiednim skonfigurowaniu systemu oraz na zaprojektowaniu i oprogramowaniu dodatkowych funkcjonalości wg potrzeb klienta. Aplikacja NAV została napisana przez Microsoft (historycznie przez duńską firmę Navision, zakupioną przez Microsoft w 2002 r.) w specjalnie do tego celu opracowanym języku programowania C/AL (Język aplikacji C/SIDE, ang. C/SIDE Application Language) w środowisku programistycznym C/SIDE (Zintegrowane środowisko programistyczne klient-serwer, ang. Client Server Integrated Development Environment).

Rozszerzając NAV o nowe moduły i funkcjonalności używa się języka C/AL w celu zachowania pełnej integralności systemu oraz zgodności z jego standardami. Aplikacja testująca niniejszy algorytm została napisana jako dodatkowy moduł systemu NAV. Rozszerzenie systemu NAV opisane w niniejszej rozprawie, będące aplikacją testującą badany algorytm zostało nazwane modułem „Planowania wysyłek".

Dokonano także integracji modułu „Planowania wysyłek" z systemem informacji geograficznej. Ze względu na uniwersalność zdecydowano się na integrację z aplikacją multimap.com. Multimap.com jest aplikacją, której interfejs użytkownika jest napisany pod przeglądarki internetowe dostępne także w urządzeniach mobilnych. Ponadto swoim zasięgiem mapy multimap.com obejmują praktycznie cały świat.

Praca została podzielona na 5 rozdziałów. Po wstępie, w rozdziale 2, omówiono stan wiedzy dotyczący badanego obszaru, opisano także problemy dostaw w szczególności ogólny problem dostaw z uwzględnieniem jego modelu matematycznego. Omówiono zastosowane w niniejszej rozprawie heurystyki. Przedstawiono także teoretyczny,

matematyczny model symulowanego wyżarzania, który jest oparty o łańcuchy Markova. Ponadto zaakcentowano problematykę badanego przedsiębiorstwa, gdzie skupiono się na aktualnym sposobie planowania wysyłek. Omówiono także szczegółowo badane zbiory testowe, sposoby przeprowadzenia tych testów, a także zamieszczono uzyskane rozwiązania (szczegółowe znalezione trasy znajdują się w załącznikach 4 oraz 5). Rozdział ten kończy podsumowanie literatury, prezentacja koncepcji badawczych oraz tezy pracy.

W rozdziale 3 scharakteryzowano metody zastosowane do rozwiązania ogólnego problemu dostaw, omówiono próby rozwiązania problemów dostaw za pomocą heurystyk wykorzystujęceych metodę sympleksową oraz programowanie dynamiczne, opisano parametry ogólne symulowanego wyżarzania, dostosowano symulowane wyżarzanie do badanego problemu oraz zdefiniowano funkcję kosztu. Przedstawiono eksperymenty z różnymi parametrami symulowanego wyżarzania, w szczególności sposobem redukcji temperatury, warunków zatrzymania algorytmu. Rozdział 3 zawiera także oryginalną koncepcję funkcji generującej rozwiązania początkowe oraz funkcji przejścia. Integralnym elementem funkcji przejścia w niniejszym algorytmie jest tzw. metoda zwiększających się promieni służąca do grupowania klientów w celu lepszego przeszukiwania przestrzeni rozwiązań.

Rozdział 4 zawiera opis rozszerzenia systemu Dynamics NAV 2009 o aplikację będącą implementacją zbudowanego algorytmu. Rozszerzenie to zostało nazwane: „Planowanie wysyłek”. W rozdziale 4 zawarta jest budowa aplikacji, w szczególności zmiany standardowych obiektów systemu NAV oraz opis nowych funkcji i procedur dodanych na potrzeby niniejszej pracy.

W rozdziale 5 podsumowano przeprowadzone badania, sformułowano najważniejsze wnioski oraz zarysowano dalsze kierunki planowanych prac autora związanych z problematyką rozprawy.

Rozprawę uzupełnia 5 załączników. Załącznik 1 zawiera listę wszystkich wykorzystanych 56 zbiorów testowych opracowanych przez M. Solomona. W załączniku 2 umieszczono listę zbiorów testowych Cordeau, służących do testowania algorytmów rozwiązujących wielomagazynowy problem dostaw. Załącznik 3 zawiera zrzut ekranowy strony internetowej Transportation Optimization Portal z listopada 2010. Strona ta nadzorowana przez organizację badawczą SINTEF zawiera najlepsze na dzień dzisiejszy wyniki wybranych problemów dostaw. Na zrzucie z listopada 2010 znajdują się poprawione najlepsze wyniki światowe dwóch testów Solomona przez autora niniejszej

pracy. W załączniku 4 umieszczono szczegółowe wyniki znalezionych rozwiązań testów Solomona w rozpisaniu na poszczególne trasy. Załącznik 5 zawiera – szczegółowe znalezione przez autora niniejszej rozprawy – wyniki testów Cordeau dotyczących problemu dostaw z wieloma magazynami.

2. Problemy dostaw oraz metody ich rozwiązywania

2.1. Problemy dostaw

Istotnym parametrem wpływającym na sukces marketingowy firmy jest sieć dystrybucji i jej efektywność (Coyle i in., 2002; Michlowicz, 2002). Jednym z możliwych sposobów analizy sieci transportowej jest, tak zwana, metoda przeglądu zupełnego. Przy małej liczbie klientów i przy zastosowaniu szybkich komputerów tego typu algorytm sprawdza się w praktyce. Jednak przy wzroście liczby odbiorców liczba iteracji i tym samym czas rośnie wykładniczo.

Duża liczba klientów oraz dodatkowe warunki ograniczające sieć transportową, np. z góry narzucony czas dostawy, pojemności pojazdów powodują, że algorytm bazujący na sprawdzaniu każdego połączenia nie jest możliwy w praktyce do wykonania. Z tego powodu algorytmy rozwiązujące różnego rodzaju problemy logistyczne odrzucają pewne połączenia między klientami oraz bazują na prawdopodobieństwie wyboru sąsiedztwa dla bieżącego rozwiązania. Podczas tworzenia sieci połączeń w omawianych problemach logistycznych dąży się do tego, żeby sumaryczny koszt połączeń był jak najmniejszy.

Ogólny problem dostaw rozpatrywany w niniejszej pracy jest rozbudowaną postacią problemu komiwojażera. Został on rozszerzony o warunki, które mają zastosowanie w rzeczywistych problemach transportowych spotykanych w praktyce.

Pierwotny problem komiwojażera TSP (ang. Travelling Salesman Problem) można sformułować następująco: dana jest określona liczba klientów, z których wszyscy są ze sobą połączeni, czyli istnieje droga łącząca dowolną parę klientów. Komiwojażer musi odwiedzić wszystkich klientów na trasie dokładnie jeden raz. Po odwiedzeniu wszystkich wraca do pierwszego klienta. Zadaniem tego problemu jest minimalizacja przebytej drogi przez wszystkie samochody.

Rozwinięciem tego zadania jest problem wielu komiwojażerów MTSP (ang. Multiple Travelling Salesman Problem). W tym zadaniu wielu komiwojażerów dostarcza towary do klientów. Każdy komiwojażer rozpoczyna swoją trasę i kończy w tym samym centralnym magazynie. W problemie MTSP także należy zminimalizować przebytą obległość.

Typowy problem dostaw VRP (ang. VRP – Vehicle Routing Problem) polega na minimalizacji kosztów dojazdów z jednego centralnego magazynu do dowolnej liczby

klientów. Problem ten różni sie od MTSP tym, że każdy pojazd posiada zdefiniowaną ładowność, a każdy klient określone zapotrzebowanie. Trasa musi być zaprojektowana w taki sposób, aby każdy klient został odwiedzony tylko raz przez dokładnie jeden pojazd, wszystkie kursy zaczynają się i kończą w centralnym magazynie. Całkowite zapotrzebowanie klientów na jednej trasie nie może przekraczać zasobów danego pojazdu. W przypadku tej odmiany problemu w pierwszej kolejności minimalizuje się liczbę pojazdów, a tym samym liczbę tras, a następnie całkowitą długość tras.

Istnieje wiele odmian problemów dostaw (Ropke, Pisinger, 2003). Najczęściej rozróżnia się wymienione w niniejszej rozprawie CVRP (Problem dostaw ze zdefiniowanymi nośnościami pojazdów, ang. Capacity Vehicle Routing Problem), MDVRP (Wielomagazynowy problem dostaw, ang. Multi Depot Vehicle Routing Problem), SDVRP (Stochastyczny problem dostaw, ang. Stochastic Vehicle Routing Problem), VRPTW (Problem dostaw z oknami czasowymi, ang. Vehicle Routing Problem with Time Windows) oraz OVRP (Otwarty problem dostaw, ang. Open Vehicle Routing Problem).

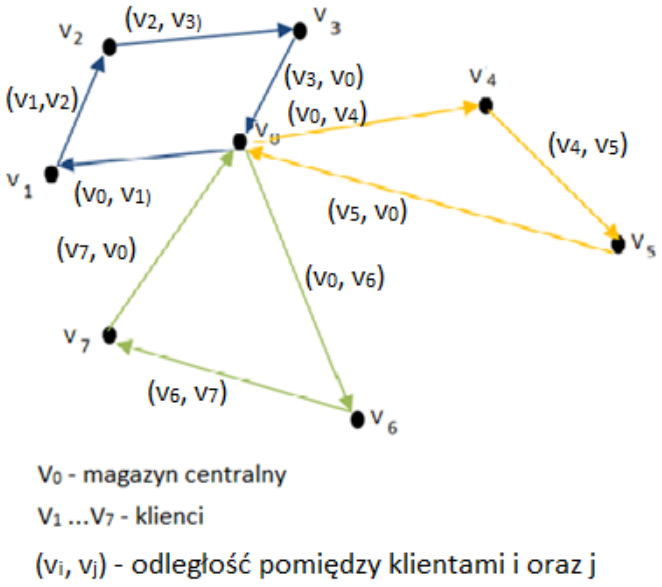
W wersji CVRP wszystkie pojazdy są identyczne, czyli posiadają jednakowe pojemności. Każda trasa rozpoczyna się i kończy w centralnym magazynie. W niektórych wersjach tego problemu pojazdy muszą także przestrzegać długości tras. Problem OVRP różni się od CVRP tylko tym, że trasa kończy się po obsłużeniu ostatniego klienta. Pojazd taki nie wraca później do centralnego magazynu.

VRPTW jest rozszerzeniem CVRP o okna czasowe każdego klienta. Okna czasowe definiują interwał czasowy, w którym klient może być obsługiwany. SDVRP jest kolejnym uogólnieniem CVRP, w którym pojazdy posiadają różne pojemności i jako takie nie mogą obsługiwać wszystkich klientów. W problemie MDVRP występuje wiele magazynów centralnych.

W pracy zbadano odmianę problemu nazwanego RDPTW (Ropke, Pisinger, 2003), będacego uogólnioną wersją wymienionych problemów. RDPTW definiuje wiele magazynów w ten sposób, że tworzony jest jeden wirtualny magazyn nadrzędny i wszystkie trasy zaczynają się z tego miejsca. Następnym punktem na trasie zawsze jest rzeczywisty magazyn firmy. Badany problem różni się od RDPTW tym, że pojazdy zawsze startują z magazynu rzeczywistego (nie jest tworzony magazyn wirtualny) i żaden pojazd nie może pobierać towarów z więcej niż jednego magazynu. Ponadto opracowany

algorytm uwzględnia wymiary wysyłanych towarów oraz wymiary pojazdu. Zarówno pojazdy, jak i dostarczane towar posiadają długość, szerokość oraz wysokość.

Model w swojej postaci obejmuje planowanie tras do wielu klientów z różnych magazynów. Rysunek 2.1 przedstawia graficzną reprezentację problemu RDPTW. V_0 oznacza magazyn centralny, V_1 do V_7 to wierzchołki oznaczające klientów. Na rysunku 2.1 pokazane są przykładowe trasy reprezentowane przez pętle rozpoczynające się zawsze w magazynie centralnym V_0, a następnie prowadzące do klientów V_1 ... V_7.



Rys. 2.1. Graficzna reprezentacja problemu RDPTW (źródło: opracowanie własne)

Model może być przedstawiony na przestrzeni płaskiej jako graf $G=(V,A)$ (rysunek 2.1), gdzie $V = \{v_1, ..., v_m, v_{m+1}, ..., v_n\}$ jest zbiorem wierzchołków odwzorowujących magazyny (wierzchołki $v_1, ..., v_m$) oraz klientów (wierzchołki $v_{m+1}, ..., v_n$), a $A=\{(v_i, v_j): v_i, v_j \in V, i \neq j\}$ to zbiór łuków odwzorowujących odległości (dostawy) pomiędzy klientami.

Każdy wierzchołek v_i posiada zestaw parametrów:

- zapotrzebowanie d_i,

- czas obsługi s_i,

- okno czasowe $[e_i, l_i]$, gdzie e to najwcześniejszy, a l to najpóźniejszy czas rozpoczęcia obsługi. W praktyce okno czasowe związane jest z godzinami otwarcia i zamknięcia magazynów.

Dla potrzeb niniejszej pracy przyjęto uproszczenie zgodne z omawianymi problemami, że czas załadunku jest zerowy, a zatem zapotrzebowanie oraz czas obsługi magazynów, czyli wierzchołków v_1 do v_m są równe zero (wzór 2.1):

$$\forall i \in \{1,...,m\} : d_i = s_i = 0 \tag{2.1}$$

Dla każdego łuku (v_i, v_j) definiuje się dodatni czas przejazdu oraz koszt z tym związany.

Każdy z m magazynów posiada zdefiniowany zbiór pojazdów $t_i \in K$. Każdy pojazd k posiada ładowność D_k, dodatnie wymiary X_k, Y_k, Z_k oraz czasy przejazdu całej trasy oznaczone przez T_k.

Jeden pojazd może nie być wystarczający do obsługi wszystkich klientów. To ograniczenie odzwierciedla fakt, że $\forall k \in K : V_k \subseteq V \wedge A_k \subseteq A$. W związku z faktem, że dana dostawa jest obsługiwana przez jeden pojazd, można przyjąć, że $i \in V_k \Leftrightarrow i + n \in A_k$.

Każdy pojazd k wyrusza z magazynu startowego V_i, a kończy w magazynie V_i'. Prawidłowa trasa $\bar{r}$ pomiędzy tymi magazynami wynosi (wzór 2.2):

$$\bar{r} = (V_i = v_1,...,v_h = V_i') \tag{2.2}$$

Trasa uwzględnia okna czasowe obsługiwanych klientów, nośności pojazdów oraz fakt, że pobranie towaru z magazynu musi nastąpić przed wyładunkiem u klienta.

Para załadunek-wyładunek może być obsługiwana tylko przez ten sam pojazd, a załadunek powinien następować przed wyładunkiem. To ograniczenie zapisane jest przy pomocy wzoru 2.3:

$$i \leq j \Rightarrow v_i \in V_k, v_j \in A_k \wedge v_j = v_i + n \tag{2.3}$$

Niech $S_i \in R_0^+$ oznacza rozpoczęcie obsługi klienta v_i. Dla upewnienia się, że obługa klientów w żadnym wypadku nie naruszy okien czasowych, spełnione muszą być warunki danem wzorami 2.4 oraz 2.5:

$$\forall i = 1,...,h : e_{v_i} \le s_i \le l_{v_i}$$
$$\forall i = 1,...,h-1 : s_{i+1} \ge s_i + t_{v_i,v_{i+1}} \quad (2.4)$$

$$e_{T_k} \le s_1 \le l_{T_k}$$
$$e_{T_k'} \le s_h \le l_{T_k'} \quad (2.5)$$

gdzie $[e_{Tk}, l_{Tk}]$ jest oknem czasowym magazynu T_k, a $[e_{T'k}, l_{T'k}]$ jest oknem czasowym magazynu T'_k.

Brak przekraczanie nośności pojazdów zapewnia wzór 2.6:

$$\forall i = 1,...,h : L_i \le C_i$$
$$\forall i = 1,...,h-1 : L_{i+1} = L_i + l_{i+1} \quad (2.6)$$
$$L_1 = 0$$
$$L_h = 0$$

gdzie $L_i \in R_0^+$ oznacza pozostałą ilość towaru znajdującego się w pojeździe obsługującym węzeł v_i po jego obsłużeniu.

Całkowity koszt przejazdu na trasie $\bar{r}$ wynosi (wzór 2.7):

$$c_{\bar{r}} = \sum_{i=1}^{h-1} d_{v_i,v_{i+1}} \quad (2.7)$$

Rozwiązanie ogólnego problemu dostaw sprowadza się do minimalizacji funkcji kosztu (wzór 2.8), która to funkcja stanowi rozwiązanie problemu dostaw, jest to tzw. funkcja kosztu:

$$\sum_{k \in V} \sum_{(i,j) \in A} c_{ij} X_{ij}^k \quad (2.8)$$

gdzie zmienna X_{ij}^k (zdefiniowana jako $\forall (i,j) \in A, \forall k \in K$) jest równa 1, jeżeli pojazd k przemieszcza się z wierzchołka i do j, a 0 w przeciwnym wypadku.

Problem rozpatrywany w niniejszej rozprawie sprowadza się do minimalizacji trzech kryteriów: liczby pojazdów, całkowitego dystansu przejechanego przez pojazdy oraz wskaźnika kosztów. Zadanie to jest uogólnioną wersją różnych problemów dostaw takich jak problem dostaw z oknami czasowymi, wielomagazynowy problem dostaw, problem dostaw z różnymi nośnościami pojazdów.

Podrozdział 2.1.1 poświęcony jest analizie literatury oraz badań dotyczących tych problemów.

2.1.1. Przegląd badań dotyczących problemów dostaw

Od momentu zdefiniowania problemu dostaw (Solomon, 1987) jest on celem bardzo intensywnych prac i praktycznie każdy typ metaheurystyki został przetestowany. Marius Solomon (Solomon, 1987) opracował 56 zestawów testów, które służą do sprawdzania badanych algorytmów. Szczegółowy opis zbiorów testowych Solomona znajduje się w rozdziale 3.8.1.

Od momentu opublikowania zestawów Solomona wyniki wszystkich testów zostały wielokrotnie ulepszone przy zastosowaniu coraz nowszych algorytmów. Pod koniec lat dziewięćdziesiątych ubiegłego wieku wyniki testów Solomona praktycznie przestały być już poprawiane. Na początku dwudziestego pierwszego wieku poszukiwania algorytmów rozwiązujących problem VRPTW zaczęły się koncentrować na przetwarzaniu równoległym, wieloprocesorowym. Algorytmy te polegają na zdefiniowaniu kilku procesów, każdy z tych procesów działa na osobnym procesorze (jednostka CPU). Następnie wyniki osiągnięte przez każdy z tych procesów są porównywane i wybierane jest jedno rozwiązanie. W przypadku przetwarzania równoległego główną zaletą jest zwiększenie całkowitej liczby iteracji badanych algorytmów w stosunku do algorytmów sekwencyjnych. Jeśli dany algorytm będzie w tym samym czasie działał np. na 5 różnych komputerach, to może on wykonać ok. 5 razy więcej kroków. Pomimo to ostanie ulepszenia testów Solomona nastąpiły w roku 2005. Od tego czasu aż do 2009 roku (Woch, Łebkowski, 2009) nie było żadnej znaczącej poprawy w dziedzinie badań nad algorytmami rozwiązującymi problemy dostaw w szczególności VRPTW.

W początkowych latach prac nad problemem dostaw bardzo efektywnym algorytmem okazała się procedura zbudowana przez Ora (Or, 1976). Procedura ta doczekała się wielu implementacji oraz rozwinięć. Na ten temat traktują między innymi

prace Russell (Russell, 1995, 1997), Baker i Schaffer (Baker, Schaffer, 1986). Inny algorytm bazujący na procedurze Or został stworzony przez Thompsona oraz Psaraftisa (Thompson, Psaraftis, 1993). Funkcja przejścia znajduje sąsiedztwo bieżącego rozwiązania analizując żądania poszczególnych klientów na trasach. Wymiana klientów następuje pomiędzy wylosowanymi podzbiorami tras.

Bardzo ciekawe są prace skupiające się na dużych zbiorach klientów, powyżej 100. Dla przykładu rozmiary testów opracowanych przez Kallehauge i in. (Kallehauge i in., 2001) wynoszą do 1000 klientów. Badania te skupiały się na znajdywaniu rozwiązania problemu dostaw za pomocą metody wielokrotności Lagrange'a (ang. Langrange Multipliers, Langrange Duality). Niemniej jednak przy większej liczbie ograniczeń jak np. przy rozwiązywaniu ogólnych problemów dostaw metoda ta jest dość czasochłonna.

Podstawowy problem dostaw VRP także jest szeroko omawiany w literaturze. Heurystyka przeszukiwania tabu została zaproponowana przez oraz Cordeau i in. (Cordeau i in., 1997), a następnie rozwijana przez Nowicki (Nowicki, 1999), Gendreau i in. (Gendreau i in., 2002) oraz Laporte i Semet (Laporte, Semet, 2002). Przeszukiwanie tabu polega na znajdywaniu rozwiązania optymalnego w przestrzeni stworzonej ze wszystkich możliwych rozwiązań. W sekwencji tej istnieją tzw. ruchy tabu, czyli niedozwolone. Algorytm także unika powtarzania wcześniej znalezionych rozwiązań dzięki przechowywaniu przez pewien czas informacji o nich w specjalnie do tego celu stworzonej liście tabu.

W 2004 r. Golden i in. (Golden i in., 1998), a także Li i in. (Li i in., 2004) zaproponowali zbiory testowe problemu CVRP (problem dostaw ze zdefiniowanymi nośnościami, ang. Capacity Vehicle Routing Problem) składające się z klientów w liczbie pomiędzy 240 i 1200. Testy te i znalezione przez nich rozwiązania spowodowały powrót zainteresowania problemem CVRP. Aż do tego czasu metody rozwiązywania CVRP były zdominowane przez algorytm branch-and-cut zaproponowany przez Cordeau i in. (Cordeau i in., 2004). Metoda ta polega na użyciu metody sympleksowej w początkowych krokach algorytmu. Gdy takie rozwiązanie zostaje znalezione, jest ono polepszane za pomocą algorytmu „cutting plane". W metodzie „cutting plane" kryteria optymalizacji oraz warunki ograniczające tworzą tzw. kolumny, które zostają obcinane, czyli znajduje się rozwiązanie przyjmując, że dana kolumna nie istnieje. Gdy już nie da się polepszyć bieżącego rozwiązania, algorytm przechodzi do kolejnego kroku zwanego „branch and bound", w którym problem dzieli się na mniejsze części w zależności od kolumn, które są

z kolei optymalizowane niezależnie od siebie, a następnie rozwiązania są scalane. Jednym z lepszych algorytmów z grupy branch-and-cut był sposób zaproponowany przez Lysgaarda i in. (Lysgaard i in., 2004). Dalszy rozwój tej metody między innymi przez Fukasawa i in. (Fukasawa i in., 2004) zaowocował stworzeniem algorytmu branch-and-cut-and-price. Algorytm ten oprócz kroku obcinania kolumn posiada kolejny, który polega na dodawaniu wcześniej obciętych kryteriów do zbioru rozwiązań. Najbardziej wydajne algorytmy branch-and-cut-and-price rozwiązują problemy do 135 klientów.

OVRP jest dużo rzadziej pojawiającą sie w literaturze odmianą problemu CVRP. W tym wariancie pojazdy zatrzymują się po obsłużeniu ostatniego klienta, a powrót do magazynu centralnego nie jest rozpatrywany. Zadanie to zostało pierwszy raz przedstawione przez Sariklis oraz Powell (Sariklis, Powell, 2000). Zaproponowali oni dwuetapowe rozwiązanie wg metody „najpierw klaster, potem trasa". Polega to na próbie zgrupowania klientów znajdujących się blisko siebie w tzw. grupy, a następnie znajdywaniu tras w obrębie tych grup. Zespól pod kierunkiem Fu (Fu i in., 2003), a także Brandão (Brandão, 2004) zastosowali przeszukiwanie tabu do rozwiązania problemu OVRP.

Cortes i Bullo (Cortes, Bullo, 2009) rozwinęli algorytm Sariklis i Powell (Sariklis, Powell, 2000) „najpierw klaster, potem trasa" i stworzyli algorytm grupujący klientów za pomocą optymalizacji geometrycznej (ang. Geometric Optimization). W metodzie tej wybierany jest jeden klient w ramach danej iteracji, a następnie szukani są inni klienci w bezpośrednim jego sąsiedztwie. Takie grupy klientów przenoszone są razem podczas przeszukiwania przestrzeni rozwiązań.

Problem MDVRP po raz pierwszy został zdefiniowany przez zespół badawczy Chao (Chao i in., 1993). Do rozwiązania tego problemu zastosowali oni metodę zwaną *record-to-record*. Metoda *record-to-record* wywodzi sie od heurystyki *Great Deluge* (wielka powódź) opracowanej przez Dueck (Dueck, 1990) i polega na tym, iż każde rozwiązanie może zostać zaakceptowane pod warunkiem, że nie jest ono „dużo gorsze" od poprzedniego rozwiązania (rekordu). Heurystyka *Great Deluge* natomiast symuluje ucieczkę osoby przed wzbierającą wodą. W przypadku ciągłego podnoszenia się poziomu wodu istnieje ryzyko utknięcia w optimum lokalnym. Metody record-to-record oraz *Great Deluge* są odmianami symulowanego wyżarzania.

Nag i in. (Nag i in., 1988) jako pierwsi przedstawili problem SDVRP. Zaproponowali oni kilka prostych heurytyk do jego rozwiązania. Chao i in. (Chao i in.,

1999) zastosowali bardziej zaawansowane techniki (m.in. użyli przeszukiwania tabu) i skonstruowali kilka nowych instancji testów. Cordeau oraz Laporte (Cordeau, Laporte, 2001) pokazali, że ten problem może być rozwiązany jako specjalny przypadek PVRP (okresowy problem dostaw, ang. Periodic Vehicle Routing Problem) i przedstawili rozwiązanie problemu SDVRP za pomocą swojej heurystyki przeszukiwania tabu służącej do rozwiązywania PVRP.

Zespół Renaud (Renaud i in., 1996) poprawił kilka testów Chao za pomocą przeszukiwania tabu. Cordeau i in. (Cordeau i in., 1997) ulepszyli ten algorytm i użyli do rozwiązania problemów PVRP oraz PTSP (okresowy problem komiwojażera, ang. Periodic Travelling Salesman Problem). PTSP oraz PVRP polegają na tym, że klienci mogą mieć zdefiniowane dostawy okresowe, a pojazd nie musi wrócić do magazynu centralnego w ten sam dzień. W swojej pracy Polacek i in. (Polacek i in., 2008) do poprawy testów Chao użyli metody adaptacyjnego przeszukiwania lokalnego.

Do rozwiązania problemów dostaw Thangiah i in. (Thangiah i in., 1991) stworzyli system GIDEON oparty o algorytmy genetyczne. W innej pracy (Thangiah, 1999) zastosował bardzo złożony algorytm będący hybrydą symulowanego wyżarzania, algorytmu genetycznego oraz przeszukiwania tabu. Podobna hybryda, również poprawiająca wiele wyników testów Solomona została użyta przez autorów biblioteki programistycznej GreenTrip przedstawionej na stronie SINTEF – Greentrip (http://www.math.sintef.no/opti/projects/greentrip.html).

Bräysy i in. (Bräysy i in., 2000, 2001) zastosowali ciekawą hybrydę algorytmów genetycznych oraz algorytmów ewolucyjnych. Zaproponowali oni dwukrokowe podejście do problemu dostaw. W pierwszej fazie rozwiązanie jest szukane przez algorytm genetyczny, a w drugiej znalezione wyniki są ulepszane przez algorytm ewolucyjny. W fazie ulepszania grupują oni losowo wszystkie trasy w pary i tworzą na tej bazie nowe trasy. Kroki te są powtarzane określoną liczbę razy. W celu zapewnienia szybkości działania w tej fazie wykorzystują przeszukiwanie lokalne.

Kilka algorytmów szukających optymalnego rozwiązania problemu dostaw zostały zaproponowane przez Mariusa Solomona (Solomon, 1987). Te algorytmy tworzą rozwiązanie przez wstawianie do bieżącej trasy w każdej iteracji jednego klienta, który w danej chwili nie znajduje się na żadnej trasie. Za ich pomocą próbowano minimalizować jedno z dwóch kryteriów: albo liczbę tras, albo przebyty dystans. Obecnie metody Solomona wykorzystywane są głównie przy tworzeniu rozwiązania początkowego.

W niniejszej pracy własny algorytm generowania rozwiązania początkowego jest rozwinięciem jednej z metod Solomona.

Wśród algorytmów godnych wzmianki są między innymi prace Potvina i in. (Potvin i in., 1996, 2006), które przedstawiają algorytmy równoległe dla VRPTW. Prace Potvina oraz Rousseau (Potvin, Rousseau, 1993) są równoległą implementacją sekwencyjnych algorytmów Solomona.

W roku 1988 Van Landeghem (Van Landeghem, 1988) zaproponował algorytm analizujący powiązania czasowe pomiędzy klientami. Jedną z możliwych implementacji tego algorytmu jest sortowanie klientów według okna czasowego. W niniejszej pracy w pierwszym kroku procedury generującej rozwiązanie początkowe klienci są sortowani według rosnącego okna czasowego .

Kontoravdis oraz Bard (Kontoravdis, Bard, 1995) opisują równoległą losową procedurę przeszukiwania sąsiedztwa. Następnie używają przeszukiwania lokalnego do ulepszenia tras. Kolejne rozwinięcia tego sposobu znalazły się w pracach Potvin i Rousseau (Potvin, Rousseau, 1993), Antes oraz Derigs (Antes, Derigs, 1995), Shaw (Shaw, 1997, 1998), Cordone i Wolfler-Calvo (Cordone, Wolfler-Calvo, 1998), a także Caseau i Laburthe (Caseau, Laburthe, 1999).

Rochat i Taillard (Rochat, Taillard, 1995) zaprezentowali probabilistyczną technikę, która używa pamięci adaptacyjnej, aby zachowywać najlepsze rozwiązania znalezione za pomocą przeszukiwania tabu. Metoda Rochata i Taillarda jako pierwsza znalazła rozwiązanie testów Solomona z grupy C. Poprawili oni także kilka testów z grupy R jak R102, R105. Także Taillard i in. (Taillard i in., 1997) używają pamięci adaptacyjnej, ale w inny sposób przeszukują sąsiedztwo. Bazują oni na wymianie podobnych klientów pomiędzy trasami. Oba te sposoby zwracają nie jedno najlepsze rozwiązanie, lecz kilka rozwiązań.

Russell (Russell, 1995) zaproponował rozwiązanie wywołujące okresowo procedurę ulepszania tras podczas standardowego algorytmu tworzącego trasy. W niniejszej rozprawie własna procedura ulepszania tras na bazie metody Russella wykorzystywana jest w ostatnim kroku funkcji przejścia.

Pod koniec lat dziewięćdziesiątych ubiegłego wieku powstało kilka prac wykorzystujących przeszukiwanie lokalne z przewodnikiem, które znacznie ulepszyły tą metodę. Przeszukiwanie z przewodnikiem bazuje na wyznaczaniu kar za wybór gorszego sąsiedztwa. Warto wspomnieć badania Voudouris (Voudouris, 1997) oraz Voudouris

i Tsang (Voudouris, Tsang, 1998), a także Kilby, Prosser and Shaw (Kilby i in., 1999). Schulze i Fahle (Schulze, Fahle, 1999) użyli specjalnej techniki sekwencyjnych zmian do wygenerowania sąsiedztwa. Technika ta jest rozwinięciem algorytmu zbudowanego przez Glover (Glover, 1991, 1992) i rozwijanego przez Grabowski i in. (Grabowski i in., 2003) w ramach przeszukiwania tabu.

De Backer i in. (De Backer i in., 2000) przetestowali cztery interaktywne techniki ulepszeń tras na bazie programowania deklaratywnego. Aplikacje wykorzystujące te algorytmy czekają na reakcję użytkownika i, w zależności od jego wyboru, wybierają dalszą drogę postępowania. Techniki te zostały pogrupowane w dwóch metaheurytykach: przeszukiwanie tabu oraz przeszukiwanie lokalne z przewodnikiem.

Brandão (Brandão, 1999) oraz Cordeau, Laporte i Mercier (Cordeau i in., 2000) wprowadzili proste przeszukiwanie tabu, które pozwala na uwzględnienie dużo gorszych sąsiedztw podczas procesu przeszukiwania przestrzeni rozwiązań. Inne udane implementacje przeszukiwania tabu można znaleźć w pracach Garcia, Potvin oraz Rousseau (Garcia, 1996), Barnes i Carlton (Barnes, Carlton, 1995), Carlton (Carlton, 1995) oraz Chiang i Russell (Chiang, Russell, 1997), którzy opisują reakcyjne przeszukiwanie tabu, które dynamicznie dostosowuje swoje parametry bazując na aktualnym rozwiązaniu.

Blanton i Wainwright (Blanton, Wainwright, 1993) zaproponowali algorytm genetyczny, który składa się z chromosomów reprezentujących sekwencje klientów na trasach. Przeszukiwanie przestrzeni rozwiązań jest skierowane w stronę odpowiedniego pobierania klientów z listy biorąc pod uwagę relacje nadrzędności (tymczasową, przestrzenną oraz mieszaną), a także stałą nadrzędność klientów zdefiniowaną przez okno czasowe. Thangiah (Thangiah, 1995) także używa algorytmów genetycznych do znalezienia dobrego klastra klientów przy użyciu strategii „najpierw klaster, potem trasa". Thangiah i in. (Thangiah i in., 1995) używają tego samego sposobu do rozwiązania problemu dostaw z oknami czasowymi. Metoda ta przy rozwiązaniu problemów Solomona z grupy R (losowo wybrani klienci) nie spełniła swoich oczekiwań.

Thangiah, Osman oraz Sun (Thangiah i in., 1994) opracowali hybrydową metodę, w której rozwiązanie początkowe jest stworzone przez algorytm genetyczny, a funkcja przejścia używa tzw. λ–wymiany. Funkcja ta składa się z kilku metaheurystyk, a symulowane wyżarzanie z niemonotonicznym harmonogramem chłodzenia jest używane jako przewodnik dla przeszukiwania lokalnego i w końcu przeszukiwania tabu.

W algorytmie genetycznym zaproponowanym przez Potvin i Bengio (Potvin, Bengio, 1998) podczas operacji krzyżowania, nowy potomek jest stworzony poprzez połączenie dwóch tras w jeden segment, a następnie odrzucani są klienci, którzy nie mogą wejść w skład trasy z powodu swoich parametrów. Następnie przeprowadzana jest mutacja, która służy do zredukowania liczby tras i do lokalnej optymalizacji rozwiązania. Podczas tego kroku odrzuceni klienci wstawiani są do innych tras lub, jeśli to jest niemożliwe, tworzone są nowe trasy.

Berger, Salois i Begin (Berger i in., 1998) zaproponowali hybrydowy algorytm genetyczny bazujący na usuwaniu losowo wybranych klientów z tras i próbie stworzenia z nich nowej trasy. Celem mutacji w tym wypadku była redukcja liczby tras przez wymianę niektórych klientów. Bräysy (Bräysy, 1999, 1999) kontynuował prace zapoczątkowane przez Berger i in. (Berger i in., 1998) i ulepszył jego algorytm przez stworzenie nowych operatorów krzyżowania i mutacji oraz przez testy ważności rozwiązania początkowego.

Homberger i Gehring (Homberger, Gehring, 1999) zaproponowali dwie metaheurystyki bazujące na algorytmach ewolucyjnych zwanych strategiami ewolucyjnymi oraz trzech popularnych technikach ulepszeń tras opracowanymi przez Or (Or, 1976), Osman (Osman, 1993) oraz Potvin i Rousseau (Potvin, Rousseau, 1995). W innej publikacji Gehring i Homberger (Gehring, Homberger, 1999) użyli podobnego podejścia, ale z implementacją równoległego przeszukiwania tabu. Bräysy, Berger i Barkaoui (Bräysy i in., 2000) opisali dwufazową hybrydę algorytmu genetycznego oraz algorytmu ewolucyjnego składającą się z kilku przeszukiwań lokalnych i heurystyk tworzenia tras. Użyty przez nich algorytm genetyczny bazuje na studiach przeprowadzonych przez Berger, Salois i Begin (Berger, 1998) oraz Bräysy (Bräysy, 1999).

Bachem, Hochstättler i Malich (Bachem i in., 1996) użyli koncepcji Simulated Trading, gdzie główną ideą jest użycie mechanizmu podmiany w przyporządkowaniach klientów na trasach. Badania przeprowadzone przez Potvin i Robillard (Potvin, Robillard, 1995) doprowadziły do użycia sieci neuronowej wykorzystywanej w ramach równoległej implementacji heurystyki tworzenia tras także opracowanej przez Potvin i Rousseau (Potvin, Rousseau, 1993). Potvin, Dube i Robillard (Potvin i in., 1996) użyli analogicznego podejścia, lecz zdeterminowali wartości parametów za pomocą algorytmu genetycznego.

Chiang i Russell (Chiang, Russell, 1996) użyli symulowanego wyżarzania jako przewodnika dla hybrydy opracowanej przez Russell (Russell, 1995). Liu oraz Shen (Liu, Shen, 1999) opracowali nową heurystykę nazwaną Route-Neighborhood, która konstruuje trasy w równoległy, zagnieżdżony sposób. Rousseau, Gendreau i Pesant (Rousseau, 2000) użyli zmiennego schematu Descent Neighborhood opracowanego przez Mladenovic i Hansen (Mladenovic, Hansen, 1997) oraz nowych, dużych operatorów sąsiedztwa z uwzględnieniem programowania deklaratywnego.

Czech (Czech, 2001) do rozwiązania problemu dostaw oraz Czech i Czarnas (Czech, Czarnas, 2002) do rozwiązania problemu dostaw z oknami czasowymi wykorzystują równoległy algortym symulowanego wyżarzania w celu lepszego przeszukiwania przestrzeni rozwiązań. W tym celu definiuje się kilka równoległych procesów, które próbują polepszyć rozwiązanie niezależnie od siebie. W pierwszej fazie pracy algorytmu po wygenerowaniu rozwiązania początkowego to rozwiązanie jest wysłane do każdego procesu. Procesy te niezależnie od siebie przeprowadzają obliczenia. Następnie po zdefiniowanej liczbie iteracji wyniki są porównywane między sobą i wybierane jest najlepsze rozwiązanie. Kroki te powtarzane są określoną liczbę razy.

Względnie nową heurystyką są algorytmy mrówkowe opisane w pracy Gambardella i in. (Gambardella i in., 1999). Algorytmy mrówkowe zostały pierwsze zaproponowane przez Dorigo i in. (Dorigo i in., 1996) jako metoda przybliżona rowiązywania trudnych problemów optymalizacji kombinatorycznej. Inspiracją do stworzenia algorytmów mrówkowych były obserwacje prawdziwych kolonii mrówek. Ważną i interesującą cechą kolonii mrówek jest zachowanie dotyczące możliwości znajdowania najkrótszej ścieżki pomiędzy źródłem pożywienia a ich mrowiskiem. Podczas spacerów od źródła pożywienia do gniazda i spowrotem, mrówki pozostawiają na ziemi substancje zwaną feromonem, formując w ten sposób feromonową ścieżkę. Mrówki wyczuwają feromony i kiedy wybierają swoją drogę, mają one tendencje do wyboru ścieżki o większej koncentracji feromonu. Feromony z czasem wyparowywują, więc dłuższe trasy w trakcie działania algorytmu mają mniej feromonów niż krótsze. W takiej sytuacji prawdopodobieństwo wyboru dłuższej trasy przez kolejną mrówkę jest zdecydowanie mniejsze. Największą skutecznością charakteryzuje się tzw. algorytm Ant Cycle, w którym mrówki uaktualniają ślad feromonowy po znalezieniu danego rozwiązania. Badania nad And Density oraz Ant Quantity porzucono z powodu zbyt wysokiej złożoności obliczeniowej (Dorigo i in., 1999).

Ai oraz Kachitvichyanukul (Ai, Kachitvichyanukul, 2009) zastosowali optymalizację roju cząsteczek (ang. Particle Swarm Optimization) do rozwiązania problemu dostaw z oknami czasowymi. Metoda ta symuluje zachowanie grupy ludzi, którzy próbują dojść do porozumienia poprzez rozmowę z innymi osobami grupy.

Figliozzi w swoich badaniach (Figliozzi, 2009) zastosował kilka ogólnie znanych algorytmów do planowania krótkich tras dla małych grup klientów. Gribkovskaia i in. (Gribkovskaia i in., 2008) zdefiniowali nowy problem transportowy, który polega na optymalizacji trasy tylko jednego pojazdu. Pojazd odwiedza klientów dostarczając im towar i zabiera materiały lub zepsute elementy od klientów do magazynu centralnego.

Prace nad heurystyką adaptacyjnego przeszukiwania dużego sąsiedztwa ALNS skupiły się głownie na liczbie klientów powyżej 100. Godnymi polecenia są studia Mester oraz Bräysy nad VRPTW (Mester, Bräysy, 2004), gdzie zaproponowali oni algorytm do rozwiązywania problemów charakteryzujących się dużą liczbą klientów od 200 do 1000.

Cordeau i. in. (Cordeau i. in., 2004) zaproponowali użycie algorytmu ALNS do rozwiązania ogólnego problemu RDPTW. RDPTW jest uogólnieniem problemów VRP, CVRP, OVRP, VRPTW, MDVRP oraz SDVRP. ALNS pierwszy raz został zaprezentowany przez Ropke i Pisinger (Ropke, Pisinger, 2005) jako rozwinięcie metody Large Neighborhood Search opracowanej przez Shaw (Shaw, 1998).

ALNS jest pewną strukturą opartą o przeszukiwanie lokalne, w której kilka prostych, niezależnych algorytmów konkuruje ze sobą w celu znalezienia najlepszego rozwiązania. W każdej iteracji bieżące rozwiązanie jest psute, a następnie naprawiane, a parametry heurystyki są dostosowywane do otrzymanych wyników na bieżąco podczas wykonywania kolejnych iteracji algorytmu. Nowe rozwiązanie jest zaakceptowane, jeśli spełnione są odpowiednie kryteria. W każdej iteracji bieżące rozwiązanie jest częściowo niszczone, a następnie naprawiane.

ALNS wykazuje także podobieństwo do metody VLNS (ang. Very Large Neighborhoods Search) zaprezentowanej przez Ahuja i in. (Ahuja i in., 2002). W metodzie VLNS wszystkie algorytmy operują na bardzo dużych sąsiedztwach wybieranych w taki sposób, żeby ciągle mogły być one efektywnie przeszukiwane.

Inna struktura zaprezentowana przez Hansena i Mladenovica (Hansena, Mladenovica, 1997) nazwna zmiennym przeszukiwaniem sąsiedztwa VNS (ang. Variable Neighborhood Search) używa pewnej sparametryzowanej rodziny sąsiedztw, uzyskanych przez użycie zmiennego parametru, tzw. głębokości. Kiedy algorytm osiąga minimum

lokalne, jest w stanie z niego wyjść i przeszukiwać inne gorsze rozwiązania różniące się od bieżącego minimum o tę głębokość. VNS jest tak sparametryzowany, że po wyjściu z minimum lokalnego kolejne przeszukiwane sąsiedztwa są zawsze coraz mniejsze.

Duże zainteresowanie usprawnianiem algorytmów rozwiązujących problemy dostaw powoduje, że badania związane z tym zagadnieniem prowadzone są w wielu ośrodkach. Do roku 2005 można zaobserwować dość częste poprawy zestawów testowych opracowywanych dla różnych wariantów problemów dostaw. Niemniej jednak za wyjątkiem badań Polacek i in. (Polacek i in., 2008) oraz autora niniejszej pracy (Woch, Łebkowski, 2009) po roku 2005 nie ma znaczących ulepszeń najlepszych wyników światowych. Polacek i in. opracowali algorytm bazujący na heurystyce adaptacyjnego przeszukiwania sąsiedztwa do znalezienia optymalnych rozwiązań testów Cordeau do wielomagazynowego problemu dostaw. Woch, Łebkowski (Woch, Łebkowski, 2009) za pomocą algorytmu opisanego w niniejszej rozprawie poprawili dwa testy Solomona oraz wyrównali 42 najlepsze światowe wyniki.

W ostatnich latach powstało także kilka interesujących wariantów problemów dostaw z oknami czasowymi. Potwin i in. (Potwin i in., 2006) zdefiniowali dynamiczne okna czasowe, czyli takie, które mogą się zmieniać w trakcie trwania obliczeń algorytmu. Tang i in. (Tang i in., 2009) stworzyli wariant problemu z elastycznymi (rozmytymi, ang. fuzzy) oknami czasowymi.

Klienci opisywanej firmy posiadają dokładnie zdefiniowane godziny pracy magazynów. W związku z tym badania opisywane w niniejszej pracy nie uwzględniają dynamicznych okien czasowych. Rozprawa skupia się na problemach, gdzie klienci posiadają statyczne, dokładnie określone okna czasowe. Z tego powodu prace analizujące dynamiczne okna czasowe nie zostały tutaj szerzej opisane.

2.2. Koncepcje badawcze i tezy rozprawy

2.2.1. Podsumowanie stanu badań w odniesieniu do specyfiki zadania

Istniejące systemy klasy ERP dla małych i średnich firm nie posiadają modułu planowania wysyłek. Praktycznie każdy system generuje tylko zestawienie klientów, do których należy wysłać transport oraz listę zamówionych przez nich towarów.

Planowaniem tras i rezerwowaniem odpowiedniej liczby pojazdów zajmują sie wyspecjalizowani logistycy. Przy złożonych problemach planowania wysyłek i ich strukturze proces optymalizacji transportu staje się na tyle skomplikowany, że logistycy przyjmują parametry powodujące, że wygenerowane trasy dalekie są od optymalnych.

Obecnie prowadzonych jest wiele badań nad problemami dostaw, jednak większość z nich skupia się nad uproszczonymi oraz czysto teoretycznymi problemami. Wystarczy wspomnieć problem dostaw z oknami czasowymi VRPTW, który zakłada jednakowe parametry pojazdów oraz jeden centralny magazyn.

Niniejsza praca jest efektem potrzeby stworzenia modułu wspomagającego planowanie tras i załadunek pojazdów, który powinien być zintegrowany z systemem Microsoft Dynamics NAV 2009. Aplikacja ta jest zamknięta i dlatego autor niniejszej rozprawy podjął decyzję o implementacji algorytmu bezpośrednio w języku C/AL, który jest wbudowanym językiem programowania w system NAV.

Istnieje potrzeba zbadania możliwości zastosowania do planowania tras algorytmów optymalizacyjnych, bazujących na rozwiązaniach z zakresu sztucznej inteligencji. Są to między innymi: algorytmy ewolucyjne, symulowane wyżarzanie, lub przeszukiwanie tabu. Najbardziej uniwersalną techniką sposród wymienionych są algorytmy ewolucyjne. Przeszukiwanie tabu oraz symulowane wyżarzanie są algorytmami dużo prostszymi w implementacji, które swoją siłę opierają przede wszystkim na budowie odpowiedniego sąsiedztwa, wykorzystując dokładną charakterystykę badanego problemu. Przeszukiwanie sąsiedztwa wymaga prostego i dobrego algorytmu optymalizacji lokalnej, dedykowanego również dla ściśle określonego problemu. W przypadku takich problemów jak problemy dostaw algorytmy ewolucyjne wypadają trochę gorzej niż symulowane wyżarzanie i przeszukiwanie tabu, jeżeli chodzi o prędkość działania czy jakość generowanych rozwiązań. Algorytmy ewolucyjne natomiast sprawdzają się lepiej przy problemach, w których parametry zmieniane są w czasie pracy algorytmu, np. harmonogramowanie linii produkcyjnej (Duda, 2003).

Analizując wyniki rozwiązań testów można powiedzieć, że bardzo dobrymi rezultatami odznacza się symulowane wyżarzanie. Metody zaproponowane w pracach Czecha i in. (Czech, 2001; Czech, Czarnas 2002) polepszyły wcześniejsze rozwiązania wielu testów Solomona, m. in. RC202, RC203. Dla zbiorów Solomona oprócz symulowanego wyżarzania bardzo dobre wyniki daje algorytm Large Neighborhood Search zaproponowany przez Bent i Van Hentenryck (Bent, Van Hentenryck, 2001), który

poprawił testy R204, R211, RC207 oraz hybrydowy algorytm genetyczny zaproponowany przez Hombergera i Gehringa (Homberger, Gehring, 1999), a następnie rozwijany przez Bräysy i in. (Bräysy i in., 2001). Algorytmy te charakteryzują się najlepszymi wynikami na świecie dla testów R101, R108, R109, R201, R209, RC105, RC206.

Najnowsze światowe rozwiązania problemu dostaw z oknami czasowymi VRPTW wydają się sugerować, że symulowane wyżarzanie jest skuteczniejszą techniką niż przeszukiwanie tabu (Ropke, Pisinger, 2005).

Badania (Cordeau i in. 2004; Ropke, Pisinger 2005; Potvin J.-Y. i in. 2006) wskazują, że bardzo efektywną heurystyką do rozwiązywania problemu RDPTW z dużą liczbą klientów (powyżej 100) jest ALNS. ALNS to metoda oparta na przeszukiwaniu lokalnym, w której kilka prostych algorytmów konkuruje ze sobą w celu znalezienia najlepszego rozwiązania. W każdej iteracji bieżące rozwiązanie jest psute, a następnie naprawiane. Nowe rozwiązanie jest zaakceptowane, jeśli spełnione są odpowiednie kryteria. W każdej iteracji bieżące rozwiązanie jest częściowo niszczone, a następnie naprawiane.

Z wyżej wymienionych powodów w niniejszej pracy zdecydowano na zastosowanie nowatorskiej metody będącej hybrydą ALNS oraz symulowanego wyżarzania.

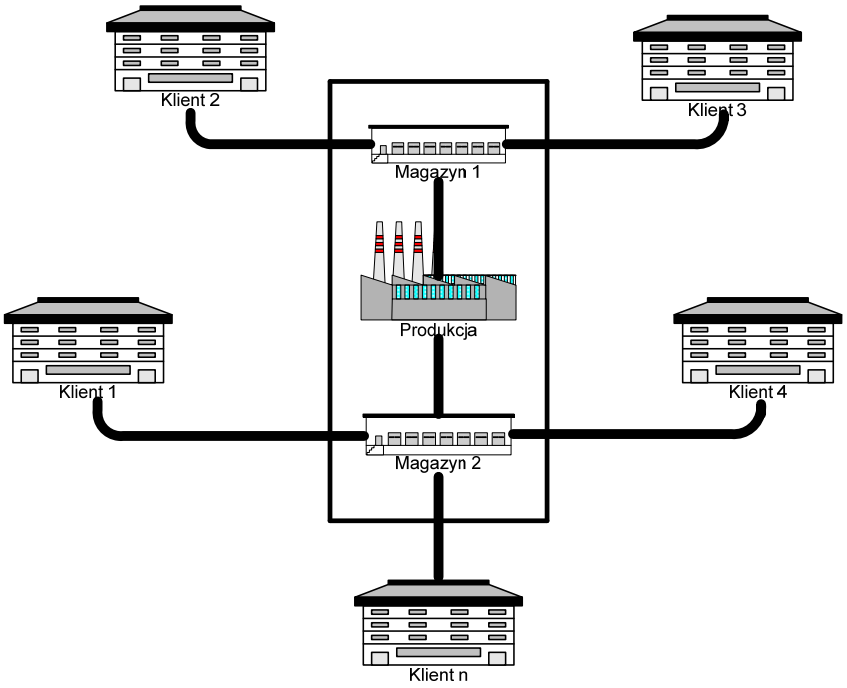
Do podjęcia tematu pracy skłoniły zatem autora następujące przesłanki:

- potrzeba wskazania skutecznych metod wspomagających optymalizację planowania tras w warunkach różnorokich ograniczeń,

- przykłady skutecznego stosowania ALNS oraz symulowanego wyżarzania w rozwiązywaniu uproszczonych problemów zachęcają do wykorzystania ich do planowania rzeczywistych wysyłek.

Przedmiotem badań jest firma z Wielkiej Brytanii produkująca rękawice na potrzeby przemysłu, medycyny oraz do użytku domowego. Spółka ta posiada własne centralne magazyny wysyłkowe, ktore zlokalizowane są w niewielkiej odległości od siebie. Codziennie wysyłają oni swój towar do 50–150 klientów. Firma nie posiada własnych hurtowni. Towar wysyłany jest z dwóch centralnych magazynów do końcowych odbiorców.

Przedsiębiorstwo dysponuje własnym parkiem pojazdów oraz współpracuje z kilkoma firmami logistycznymi i korzysta z ich floty w sytuacji, gdy własne pojazdy nie są w stanie obsłużyć wszystkich zamówień.



Rys. 2.2. Schemat sieci dystrybucji w omawianym przedsiębiorstwie (źródło: opracowanie własne)

Zadanie harmonogramowania i optymalizacji wysyłek, będące przedmiotem niniejszych badań stanowi rozszerzenie ogólnego problemu dostaw RDPTW. Celem jest opracowanie planu dziennych wysyłek towarów do wszystkich klientów tak, aby zminimalizować koszty transportu. Koszty transportu są wypadkową trzech zmiennych:

- liczby pojazdów,
- całkowitej długości przejechanych tras,
- wskaźników kosztów użytkowania pojazdów, co przekłada się na minimalizację całkowitych kosztów użytkowania pojazdów.

Minimalizacja liczby pojazdów sprowadza się do minimalizacji liczby przebytych tras, czyli należy doprowadzić do sytuacji, w której dany pojazd obsłuży maksymalną liczę klientów uwzględniając ich okna czasowe, zapotrzebowanie w stosunku do nośności pojazdów oraz wymiary zamówionych towarów w stosunku do wymiarów samochodu. Należy także tak opracować poszczególne trasy każdego pojazu, aby przebyty dystans był możliwie jak najmniejszy. Trzecie kryterium zależy od kosztu użytkowania danego pojazdu. Może się w praktyce okazać, że koszty przewozu towarów dwoma własnymi samochodami będą niższe niż wynajęcie jednego pojazdu od zewnętrznego dostawcy.

Klasyczny problem RDPTW nie uwzględnia kosztów użytkowania pojazdów, także wymiary pojazdów oraz transportowanych towarów nie są brane pod uwagę.

Obecnie trasy planowane przez operatorów oraz plany pakowania towarów są dalekie od optymalnych. Planista musi brać pod uwagę jednocześnie kilka kryteriów, zapewniając z jednej strony wykorzystanie zasobów na minimalnym możliwym poziomie, a z drugiej strony dotrzymanie terminów dostaw.

Działanie prezentowanych w literaturze algorytmów bazuje na przetwarzaniu danych pochodzących ze źródeł specjalnie do tego celu przygotowanych (Homberger, Gehring, 1999; Golden i in., 1998; Bräysy i in., 2001; Li i in., 2004). W wielu przypadkach obiektem badań są dane symulowane, dla których istnieje potencjalna możliwość spadku skuteczności działania w przypadku implementacji dla nieznanego, nowego zestawu danych rzeczywistych.

Prace opisane w literaturze głównie skupiają się na jednym specyficznym problemie dostaw (Renaud i in., 1996; Czech, 2001; Czech, Czarnas 2002; Cordeau i in., 2002; Ai, Kachitvichyanukul, 2009; Tang i in., 2009). Ograniczone zastosowanie wielu algorytmów wynika często z właściwości danych testowych. Część prezentowanych w literaturze algorytmów doskonale sprawdza się dla kilku testów, jednocześnie generując zdecydowanie gorsze rozwiązania dla innych zestawów.

Można także zauważyć, że przedmiotem badań często jest udoskonalenie metod wcześniej opisywanych (Van Landeghem, 1988; Cordeau i in., 1997; Thangiah, 1999; Figliozzi, 2009; Tang i in., 2009), bądź zastosowanie ich dla innych uwarunkowań. Porównanie prezentowanych wyników uwidacznia zazwyczaj nieznaczne usprawnienia w odniesieniu do wcześniej opisywanych metod.

W literaturze praktycznie nie spotyka się rozwiązań problemów dostaw za pomocą metod programowania matematycznego. Algorytmy te sprawdzają się z powodzeniem przy niewielkich zestawach danych. Jednak przy złożonych problemach z dużym zestawem danych ich wydajność spada, co wyklucza ich praktyczne zastosowanie. Metoda sympleksowa oraz programowanie dynamiczne zostało zastosowane w niektórych pracach jako element polepszający dane rozwiązanie w ramach danej heurystyki. Zespól prof. Cordeau (Cordeau, 1999; Cordeau i in. 2001) zastosował algorytm sympleksowy do ulepszania pojedynczych tras znalezionych za pomocą algorytmu zbudowanego na bazie heurystyki przeszukiwania tabu. Lau i in. (2002) użyli metody sympleksowej do tworzenia rozwiązania z klientów, którzy zostali usunięci z bieżącego rozwiązania w funkcji

przejścia w ramach algorytmu symulowanego wyżarzania. Ceselli i in. (Ceselli i in, 2004) zastosowali programowanie dynamiczne do generowania kolumn w metodzie „cutting plane".

Obecnie istnieje rosnąca tendencja do stosowania algorytmów równoległych, czyli takich, które mogą być uruchamiane na raz na kilku różnych procesorach (Taillard, 1993; Czech, 2001; Czech, Czarnas 2002; Potvin i in., 1996, 2006). Metody te poprzez zwiększenie liczby iteracji pozwalają na dokładniejsze przeszukiwanie przestrzeni rozwiązań, niemniej jednak ich zastosowanie w praktyce, w szczególności w systemach ERP jest dość niewielkie.

2.2.2. Tezy rozprawy

Badany w niniejszej rozprawie problem ma na celu optymalizację kosztów całkowitych dostaw do różnych klientów z rozproszonych geograficznie magazynów firmy. Optymalizacja tych kosztów sprowadza się do minimalizacji następujących kryteriów: liczby pojazdów do obsługi klientów w ramach bieżącej wysyłki, całkowitego dystansu przejechanego przez pojazdy oraz sumarycznego współczynnika kosztu przypisanych do tras pojazdów.

Z naukowego punktu widzenia interesujące jest poszukiwanie nowych metod rozwiązań tak sformułowanego problemu.

Olbrzymie zainteresowanie usprawnianiem algorytmów rozwiązujących problemy dostaw powoduje, że badania związane z tym zagadnieniem prowadzone są w wielu ośrodkach. Brakuje jednak na rynku rozwiązań łatwych w implementacji, w integracji z istniejącymi systemami ERP, a jednocześnie efektywnych.

W związku z powyższym, **celem niniejszej rozprawy jest zbudowanie algorytmu zdolnego wygenerować plany wysyłki towarów, które są bliskie rozwiązaniu optymalnemu w zadowalającym czasie**. Proponowany w rozprawie algorytm jest przewidziany do przetwarzania danych zawierających dowolną liczbę klientów. Zaimplementowany przez autora niniejszej rozprawy sposób przenoszenia całych grup klientów pomiędzy trasami powoduje, że przeszukiwanie przestrzeni rozwiązań jest bardzo efektywne i głębokie bez zwiększonego nakładu czasu obliczeń. Podejście to umożliwi zastosowanie proponowanego algorytmu z wykorzystaniem ogólnodostępnych mocy obliczeniowych.

Powyższe przesłanki pozwalają na sformułowanie głównej tezy rozprawy oraz tez roboczych.

TEZA GŁÓWNA:

Algorytm zbudowany na bazie heurystyki symulowanego wyżarzania w połączeniu z adaptacyjnym przeszukiwaniem dużego sąsiedztwa pozwala na wygenerowanie planu wysyłek bliskiego optymalnemu ze względu na przyjęte kryteria w czasie akceptowalnym w praktyce.

TEZY ROBOCZE:

- Zastosowanie proponowanych rozwiązań w procesie planowania wysyłek istotnie pomaga w zarządzaniu dostawami i parkiem pojazdów, co pozwala na znaczne obniżenie kosztów transportu.

- Symulowane wyżarzanie może być wykorzystywane w rozwiązywaniu trójkryterialnych problemów decyzyjnych.

W celu wykazania powyższych tez przeprowadzono następujące badania:

a. Zdefiniowano nowy problem logistyczny będący rozszerzeniem ogólnego problemu dostaw.
b. Stworzono algorytm do rozwiązywania dwu i trójkryterialnych problemów dostaw.
c. Zaimplementowano opisywany algorytm w aplikacji zbudowanej w systemie Microsoft Dynamics NAV 2009.
d. Przeprowadzono szereg testów porównując uzyskane wyniki z najlepszymi obecnie wynikami światowymi.
e. Stworzono oryginalną metodę grupowania klientów wzraz ze wzrostem odległości od centralnego magazynu.
f. Stworzono oryginalną funkcję generującą rozwiązanie początkowe.

Konstrukcja algorytmu oraz jego implementacja, a także generowanie rozwiązania początkowego i metoda zwiększających się promieni są oryginalnymi osiągnięciami autora rozprawy.

Niektóre uzyskane wcześniej wyniki opisywanych w niniejszej pracy badań zostały opublikowane m. in. w artykułach:

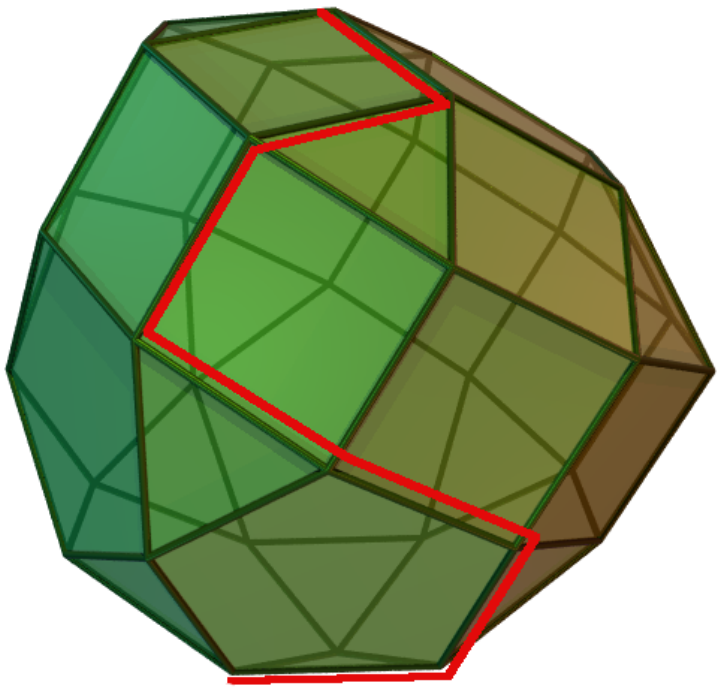
- Woch M., Łebkowski P.: Zastosowanie algorytmu symulowanego wyżarzania do rozwiązania problemu dostaw z oknami czasowymi. XIII Konferencja Logistyki Stosowanej. Zakopane, 2009.
- Woch M., Łebkowski P.: Sequential Simulated Annealing for the Vehicle Routing Problem with Time Windows. Decision Making in Manufacturing and Services, vol. 3, no. 1-2., s. 87-100, 2009.
- Woch M., Łebkowski P.: Symulowane wyżarzanie w rozwiązywaniu dwukryterialnych problemów optymalizacyjnych. Współczesne problemy zarządzania przedsiębiorstwem — Konferencja Międzynarodowa, Szczyrk, 10-12.06.2010.
- Woch M., Łebkowski P.: Zastosowanie algorytmu symulowanego wyżarzania do rozwiązania problemu dostaw z oknami czasowymi. Logistyka, ISSN 1231-5478, nr 2 s. 1-9, Kraków, 2010.
- Woch M., Łebkowski P.: Algorytm dwukryterialnej optymalizacji w rozwiązywaniu problemów dostaw. Zarządzanie produkcją – planowanie, wytwarzanie, optymalizacja i kontrola = Management of production – planning, production, optimization and control : monografia, red. nauk. Ryszard Barcik, Marek Dudek, Wiesław Waszkielewicz, Wydawnictwo Akademii Techniczno-Humanistycznej, ISBN 978-83-62292-76-9, s. 151–164, Bielsko-Biała, 2010.

3. Rozwiązanie problemów dostaw

W kolejnych podrozdziałach opisane są metody użyte przez autora niniejszej pracy do rozwiązania problemów dostaw. Skupiono się na dwóch algorytmach wykorzystujących metodę sympleksową oraz programowanie dynamiczne. Następnie autor niniejszej rozprawy stworzył ostateczną wersję algorytmu, który jest rozwiązaniem hybrydowym łączącym dwie heurystyki: symulowane wyżarzanie oraz adaptacyjne dużego przeszukiwanie sąsiedztwa. Metoda ALNS została użyta do określenia funkcji przejścia w heurystyce symulowanego wyżarzania.

3.1. Rozwiązanie problemu dostaw za pomocą heurystyki z wykorzystaniem metody sympleksowej

Opracowany w 1947 roku przez George Dantzig algorytm sympleksowy tzw. metoda sympleksów jest iteracyjną metodą do rozwiązywania zadań programowania liniowego za pomocą kolejnego polepszania bieżącego rozwiązania. Nazwa tej metody pochodzi od sympleksu, czyli zbioru wypukłego będącego uogólnieniem trójkąta i czworościanu na dowolną przestrzeń liniową (Wiens, 2004).



Rys. 3.1. Wielościan algorytmu sympleksowego (źródło: Borgwart K., 1987)

Zadanie programowania liniowego z dowolną liczbą zmiennych rozwiązuje się wyznaczając wszystkie wierzchołki wielościanu (sympleksu). Następnie porównuje się wartości funkcji celu w punktach wierzchołkowych. W związku z wielością punktów powstaje problem wyznaczenia wartości funkcji celu i znalezienie optymalnego wierzchołka, który spełniłby warunek zadania programowania liniowego, czyli doprowadzenie do tego, aby wszystkie warunki ograniczające oraz funkcja celu miały postacie liniowe.

Istota metody sympleksów sprowadza się do tego, że jeżeli jest znany jakikolwiek wierzchołkowy punkt i wartość w tym punkcie funkcji celu, to wtedy wszystkie wierzchołkowe punkty, w których funkcja celu przyjmuje gorsze wartości, są odrzucane. Kolejny krok polega na tym, że przechodzi się do następnego wierzchołka, znajdującego się na jednej krawędzi z odnalezionym już punktem, w którym funkcja celu osiąga lepsze wartości. Działanie algorytmu kończy się, gdy kolejny przeglądany punkt wierzchołkowy jest najlepszy lub osiągnięto z góry określowy warunek stopu np. zdefiniowaną liczbę iteracji.

W literaturze nie ma wielu przykładów wykorzystania metody sympleksowej do rozwiązania problemów dostaw. Lau, Sim oraz Meng Teo (Lau i in., 2002) zastosowali algorytm sympleksowy w kroku odpowiedzialnym za wstawianie klientów do tras w algorytmie zbudowanym na bazie przeszukiwania tabu. Podczas każdej iteracji, w celu przeszukiwania sąsiedztwa usuwają oni pewną liczbę klientów z tras, a następnie próbują ich wstawić do innych tras. Używają oni do tego celu metody sympleksowej w kroku nazwanym przez siebie „zagęszczanie na siłę” (ang. force dence). W tym celu znajdują oni najkrótszą odległość do przejechania pomiędzy klientami usuniętymi z bieżących tras.

Lau i in. zastosowali wariant problemu dostaw z miękkimi oknami czasowymi, czyli pozwalają oni na przekroczenie okna czasowego z uwzględnieniem kary. Wyniki algorytmu Lau dla sztywnych okien czasowych przyniosły jednak dość słabe rozwiązania w porównaniu z ówczesnymi światowymi wynikami problemów Solomona.

Zespół badawczy prof. Cordeau (Cordeau i in., 2000) zastosował metodę symplesową jako jeden z kroków algorytmu szukającego najlepszego rozwiązania metodą uśredniania (ang. Aproximation Algorithm). Metoda ta w każdym kroku znajduje tzw. granice górne i dolne dla danej iteracji. Granicą górną jest rozwiązanie znalezione w poprzedniej iteracji lub rozwiązanie początkowe, a granicą dolną jest wartość funkcji

celu znaleziona za pomocą metody sympleksowej dla złagodzonych kryteriów – autorzy pozwalają na przekroczenie kryteriów poprzez wprowadzenie funkcji kary.

Autor niniejszej rozprawy opracował własną heurystykę, która wykorzystuje metodę sympleksową. Algorytm sympleksowy jest wykorzystany w kroku 2 algorytmu przedstawionego na rysunku 3.2 i służy do znalezienia minimum funkcji celu dla bieżącej iteracji. Funkcja celu wynika bezpośrednio z modelu opisanego wzorem 2.8. Dla problemu dwukryterialnego funkcja celu opisana wzorem 3.1 przyjmuje postać:

$$f(x_1, x_2) = ax_1 + bx_2 \tag{3.1}$$

Algorytm szukający rozwiązania problemu dostaw jest przedstawiony na rysunku 3.2:

```
Krok 1. Stwórz rozwiązanie początkowe poprzez stworzenie tylu tras ile
jest klientów.
Krok 2. Za pomocą metody sympleksowej znajdź minimalną wartość funkcji
danej wzorem:
      F = Liczba_tras * x1 + Całkowita_długość * x2
Krok 3. Z każdej trasy usuń losowo 1 klienta
Krok 4. Jeśli dana trasa zawiera tylko 1 klienta usuń ją
Krok 5. Powtórz kroki 3 oraz 4, dopóki wartość funkcji celu bieżących
tras jest większa od F
Krok 6. Dla każdego usuniętego klienta znajdź najlepsze miejsce do
wstawienia na trasach
Krok 7. Oblicz funkcję celu wg wzoru a*Liczba_tras + b*Całkowita_długość
Krok 8. Jeśli nowe rozwiązanie jest lepsze od poprzedniego to zapamiętaj
to rozwiązanie
Krok 9. Powtarzaj kroki począwszy od kroku 2 dopóki udaje się ulepszyć
rozwiązanie
```

Rys. 3.2. Algorytm do rozwiązania problemu dostaw z oknami czasowymi z wykorzystaniem metody sympleksowej (źródło: opracowanie własne)

Algorytm opisany na rysunku 3.2 w kroku 2 dynamicznie dopasowuje wagi funkcji kosztu, w celu dokładniejszego przeszukiwania przestrzeni rozwiązań. Funkcja ta dopuszcza gorsze rozwiązania w stosunku do liczby tras po to, aby uniknąć utknięcia w minimum lokalnym.

W kroku 6 wykorzystana jest własna funkcja szukająca najlepszego miejsca na trasach dla danego klienta. Funkcja wstawia rozważanego klienta za każdym odbiorcą na każdej trasie sprawdzając, czy nie zostały przekroczone okna czasowe oraz dopuszczalna ładowność pojazdu. Jeżeli nowe trasy są poprawne to następuje porównanie kosztu rozwiązania z poprzednim. Funkcja ta została szczegółowo opisana na rysunkach 3.13 oraz 3.14. W krokach 7 oraz 8 znalezione rozwiązanie porównywane jest z rozwiązaniem poprzednim jako podstawowe kryterium minimalizacji przyjmując liczbę tras.

Algorytm szuka minimum funkcji kosztu dla danej iteracji za pomocą metody sympleksowej. Dla potrzeb metody sympleksowej przyjęto następujące ograniczenia:

$$\begin{aligned} x_1 &\geq 0 \\ x_2 &\geq 0 \\ x_1 &\leq L \\ x_2 &\leq D, \end{aligned} \tag{3.2}$$

gdzie L to liczba tras poprzedniego rozwiązania, a D to długość tras przejechanych przez wszystkie pojazdy tegoż rozwiązania. Parametry L oraz D znajdywane są dla każdej iteracji. Zmienne x_1 oraz x_2 przyjmują wartości równe zero w sytuacji, gdy trasa jest niemożliwa do skomponowania.

Zagadnienie sprowadzono do postaci standardowej, wprowadzono dodatkowe zmienne x_3 i x_4. Zastąpiono nierówności 3.2 następującym układem równości i nierówności elementarnych:

$$\begin{aligned} x_1 + x_3 &= L \\ x_2 + x_4 &= D \\ x_1 &\geq 0 \\ x_2 &\geq 0 \end{aligned} \tag{3.3}$$

Inaczej układ ten można zapisać:

$$Ax = b; x \geq 0 \; gdzie:$$
$$A = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}$$
$$X = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \tag{3.4}$$
$$B = \begin{bmatrix} L \\ D \end{bmatrix}$$

Następnie tworzona jest tablica sympleksowa, składająca się z macierzy bazowego układu równiań zadania programowania liniowego. Tablica ta jest następnie rozszerzona o o tzw. wiersz funkcji celu, który reprezentuje równanie

$$z = f(x_1, x_2, x_3, x_4) \tag{3.5}$$

gdzie *z* oznacza dodatkową zmienna. Tablica sympleksowa została rozszerzona o kolumnę kolumnę reprezentującą wybór zmiennych układu bazowego.

Początkowym dopuszczalnym wektorem bazowym jest $(x_1, x_2, x_3, x_4) = (0, 0, L, D)$.

Zmienna Bazowa	Z	x_1	x_2	x_3	x_4	k
x_3	0	1	0	1	0	L
x_4	0	0	1	0	1	D
wiersz f	1	a	b	0	0	0

Tabela 3.1. Tabela sympleksowa (źródło: opracowanie własne)

Dla rozpatrywanej funkcji kosztu $a >> b$, dlatego w kolejnym kroku próbuje się zmniejszyć funkcję celu przechodząc do nowego dopuszczalnego wektora bazowego, przyjmując, że x_1 zostanie nową zmienną bazową. Wynika to z faktu, że w jej kolumnie w wierszu *f* stoi największa liczba.

Ponadto spośród zmiennych x_3, x_4 wybierana jest zmienna, która opuści zbiór zmiennych bazowych. W tym celu obliczane są ilorazy wyrazów stałych z kolumny *k*

przez odpowiadające im elementy w kolumnie nowej zmiennej bazowej x_1, pod warunkiem, że są to elementy dodatnie.

W przykładzie z tabeli 3.1 zmienna x_3 opuszcza zmienne bazowe.

Następnie budowana jest nowa tabela sympleksowa (tabela 3.2) za pomocą operacji 3.6:

$$W_3 - W_1$$
$$W_f - L*W_1 \qquad (3.6)$$

gdzie symbole W oznaczają wiersze tabeli.

Zmienna Bazowa	Z	x_1	x_2	x_3	x_4	K
x_1	0	1	0	1	0	L
x_4	0	0	1	0	1	D
wiersz f	1	0	b	-a	0	a-La

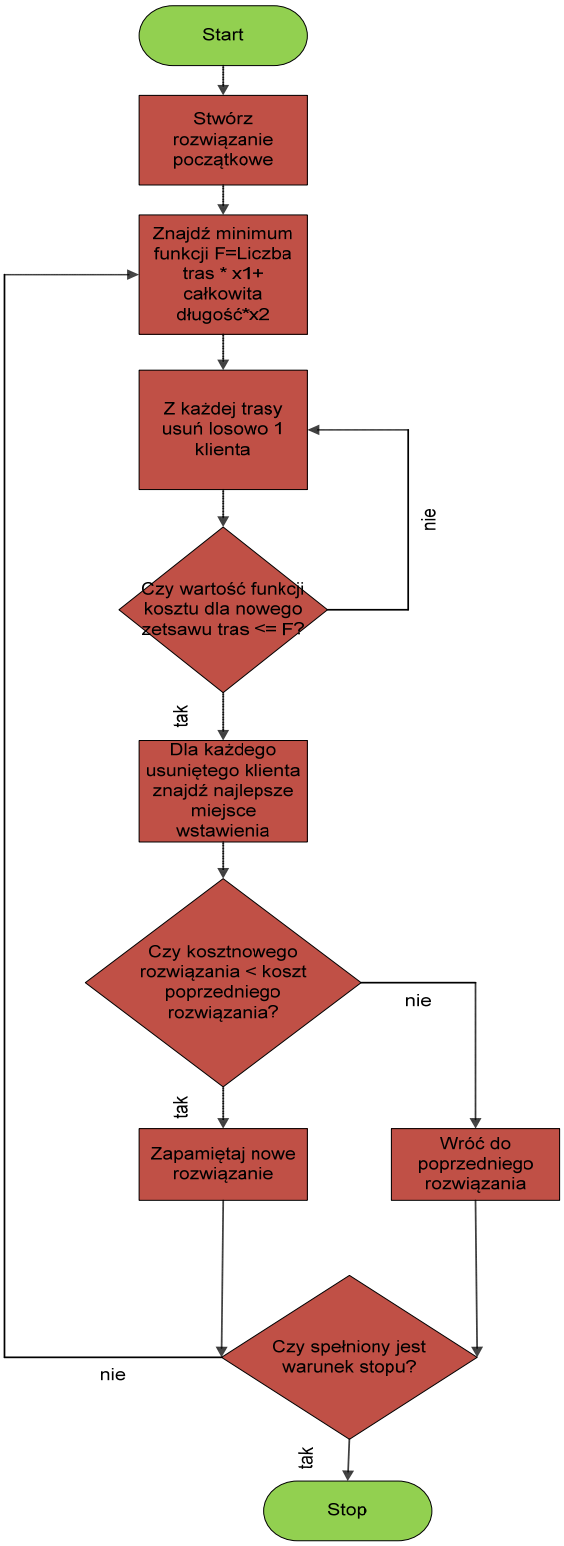
Tabela 3.2. Tabela sympleksowa – kolejny krok (źródło: opracowanie własne)

Dalej przyjmuje się, że x_2 jest nową zmienną bazową, gdyż w jej kolumnie występuje liczba dodatnia b. Tworzona jest nowa tabela sympleksowa wg zasady opisanej w niniejszym przykładzie.

Dla w ten sposób znalezionej funkcji dopasowywane jest rozwiązanie w krokach 3, 4 oraz 5 algorytmu z rysunku 3.2. Algorytm usuwa losowo klientów z tras dopóki wartość bieżącej funkcji celu jest większa od znalezionej wartości minimalnej za pomocą metody sympleksowej. Następnie usunięci klienci wstawiani są w najlepsze miejsce na trasach i tworzone jest nowe rozwiązanie.

Kroki te są powtarzane, dopóki funkcji nie da się już poprawić.

Rysunek 3.3 przedstawia schemat blokowy algorytmu przedstawionego na rysunku 3.2:



Rys. 3.2. Szczemat blokowy algorytmu do rozwiązania problemu dostaw z oknami czasowymi z wykorzystaniem metody sympleksowej (źródło: opracowanie własne)

Autor niniejszej rozprawy skupił się na poszukiwaniu rozwiązań dwukryterialnego problemu dostaw z oknami czasowymi (VRPTW). Algorytm został sprawdzony za

pomocą zbiorów testowych Mariusa Solomona. Przy rozwiązywaniu opisywanego w niniejszej rozprawie problemu dostaw szukane jest minimum funkcji celu opisanej wzorem 2.8.

Aplikacja implementująca algorytm opisywany w niniejszym rozdziale została napisana w języku C++ i skompilowana za pomocą darmowego kompilatora Borland C++ 5.5 w środowisku Windows XP. Ze względu na szybkość działania programów napisanych w języku C++ aplikacja ta miała być prototypem pod przyszłą implementację z języku C/AL. Jednak z powodu słabych wyników otrzymanych za pomocą tego algorytmu autor niniejszej rozprawy zdecydował nie dokonywać implementacji algorytmu w języku C/AL.

W tabelach 3.3–3.8 przedstawione zostały aktualne najlepsze wyniki światowe oraz uzyskane rezultaty za pomocą algorytmu na bazie heurystyki, która wykorzystywała metodę sympleksową. Wyniki uzyskane za pomocą tej heurystyki są gorsze od obecnie znanych wyników światowych, dlatego testowanie algorytmu na trudniejszych testach Cordeau lub testach rzeczywistych zostało pominięte.

W pierwszej kolumnie tabel znajduje się nazwa każdego testu, w drugiej kolumnie przedstawione są najlepsze wyniki światowe, a w trzeciej kolumnie wyniki testów niniejszego algorytmu.

- w pierwszej z nich są inicjały autora lub autorów najlepszego światowego wyniku. Lista tych autorów oraz prace, gdzie te wyniki zostały opublikowane, znajdują się w tabeli 3.21.

- W drugiej i trzeciej kolumnie znajdują się najlepsze rezultaty danych testów z podziałem na liczbę tras oraz całkowitą długość tras.

Kolumna „Uzyskane wyniki” z wynikami algorytmu przedstawionego na rysunkach 3.2 oraz 3.3 także została podzielona na dwie podkolumny, w których znajdują się rezultaty testów z podziałem na liczbę tras oraz całkowitą długość tras.

Nazwa testu	Wyniki światowe			Uzyskane wyniki	
	Autor	Liczba tras	Dług. Tras	Liczba tras	Dług. Tras
C101	RT	10	828,94	10	828,94
C102	RT	10	828,94	11	831,04
C103	RT	10	828,06	10	1000,15
C104	RT	10	824,78	10	824,78

C105	RT	10	828,94	10	828,94
C106	RT	10	828,94	10	891,80
C107	RT	10	828,94	10	828,94
C108	RT	10	828,94	10	830,01
C109	RT	10	828,94	11	881,67

Tabela 3.3. Wyniki testów dla grupy C10x (źródło: opracowanie własne)

Nazwa testu	Wyniki światowe			Uzyskane wyniki	
	Autor	Liczba tras	Dług. Tras	Liczba tras	Dług. Tras
C201	RT	3	591,56	3	591,56
C202	RT	3	591,56	3	596,67
C203	RT	3	591,17	4	601,12
C204	RT	3	590,60	3	590,60
C205	RT	3	588,88	3	588,88
C206	RT	3	588,49	3	604,09
C207	RT	3	588,29	3	588,29
C208	RT	3	588,32	3	588,32

Tabela 3.4. Wyniki testów dla grupy C20x (źródło: opracowanie własne)

Nazwa testu	Wyniki światowe			Uzyskane wyniki	
	Autor	Liczba tras	Dług. Tras	Liczba tras	Dług. tras
R101	H	19	1645,79	19	1651,08
R102	RT	17	1486,12	18	1516,10
R103	LLH	13	1292,68	13	1301,23
R104	M	9	1007,24	10	1010,34
R105	RT	14	1377,11	14	1379,01
R106	M	12	1251,98	12	1262,18
R107	S97	10	1104.66	10	1200,56
R108	BBB	9	960.88	10	1001,34
R109	HG	11	1194.73	11	1231,10
R110	M	10	1118,59	11	1145,87

R111	RGP	10	1096,72	10	1145,02
R112	GTA	9	982,14	9	956,56

Tabela 3.5. Wyniki testów dla grupy R1x (źródło: opracowanie własne)

Nazwa testu	Wyniki światowe			Uzyskane wyniki	
	Autor	Liczba tras	Dług. Tras	Liczba tras	Dług. Tras
R201	HG	4	1252,37	5	1301,34
R202	RGP	3	1191,70	3	1298,23
R203	M	3	939,54	3	998,17
R204	BVH	2	825,52	3	903,28
R205	RGP	3	994,42	3	1002,34
R206	SSSD	3	906,14	3	909,23
R207	BVH	2	890,61	2	906,63
R208	M	2	726,75	2	812,34
R209	H	3	909,16	3	1004,25
R210	M	3	939,34	3	1103,23
R211	BVH	2	892,71	3	941,09

Tabela 3.6. Wyniki testów dla grupy R2x (źródło: opracowanie własne)

Nazwa testu	Wyniki światowe			Uzyskane wyniki	
	Autor	Liczba tras	Dług. Tras	Liczba tras	Dług. tras
RC101	TBGGP	14	1696,94	14	1731,00
RC102	TBGGP	12	1554,75	12	1561,01
RC103	S98	11	1261,67	12	1402,89
RC104	CLM	10	1135,48	10	1200,67
RC105	BBB	13	1629,44	13	1734,56
RC106	BBB	11	1424,73	11	1429,07
RC107	S97	11	1230,48	11	1299,03
RC108	TBGGP	10	1139,82	10	1340,94

Tabela 3.7. Wyniki testów dla grupy RC10x (źródło: opracowanie własne)

Nazwa testu	Wyniki światowe			Uzyskane wyniki	
	Autor	Liczba tras	Dług. Tras	Liczba tras	Dług. tras
RC201	M	4	1406,91	4	1459,76
RC202	CC	3	1365,65	4	1403,19
RC203	CC	3	1049,62	4	1108,27
RC204	M	3	798,41	3	817,23
RC205	M	4	1297,19	4	1382,29
RC206	H	3	1146,32	3	1303,03
RC207	BVH	3	1061,14	3	1068,32
RC208	IKMUY	3	828,14	3	909,24

Tabela 3.8. Wyniki testów dla grupy RC20x (źródło: opracowanie własne)

Dla 56 przeprowadzonych testów Solomona tylko w 8 przypadkach udało się wyrównać najlepsze wyniki światowe. W pozostałych przypadkach wyniki różnią się liczbą tras oraz ich długością.

Słabe wyniki osiągnięte za pomocą tej metody spowodowały, że autor niniejszej rozprawy zaprzestał badań w tym kierunku.

3.2. Rozwiązanie problemu dostaw za pomocą heurystyki z wykorzystaniem programowania dynamicznego

Programowanie dynamiczne zostało opracowane przez R. Bellmana (Bellman, 1952). Algorytm ten został opracowany jako alternatywa dla algorytmów zachłannych do rozwiązywania problemów optymalizacyjnych. Algorytm zachłanny zawsze akceptuje lepsze rozwiązanie od bieżącego, przykładem jest przeszukiwanie lokalne.

W metodzie programowania dynamicznego problem jest dzielony na mniejsze podproblemy ze względu na kilka parametrów. Problemy, które mogą być rozwiązywane za pomocą programowania dynamicznego, cechują się następującymi parametrami (Bellman, 1952):

- podproblemy posiadają tzw. własność optymalnej podstruktury, czyli rozwiązanie optymalne jest funkcją optymalnych rozwiązań podproblemów,

- zastosowanie tzw. metody siłowej do rozwiązania problemu prowadzi do ponadwielomianowej liczby rozwiązań podproblemów, nawet w sytuacji, gdy liczba podproblemów jest wielowymianow. Metoda siłowa polega na sprawdzeniu wszystkich możliwych kombinacji w poszukiwaniu rozwiązania.

Podstawą zaprojektowania algorytmu na bazie programowania dynamicznego jest znalezienie odpowiedniej funkcji rekurencyjnej, która opisując funkcje celu dla podproblemów opisze także optymalną wartość funkcji celu dla całego problemu. Z powodu użytej rekurencji algorytm programowania dynamicznego jest z reguły dość pamięciochłonny, dlatego bardzo istotna okazuje się odpowiednia definicja podproblemów, a tym samym parametrów opisujących te podproblemy. W praktyce ze względu na duże zużycie pamięci stosowanie większej ilości parametrów niż 4 jest niespotykane.

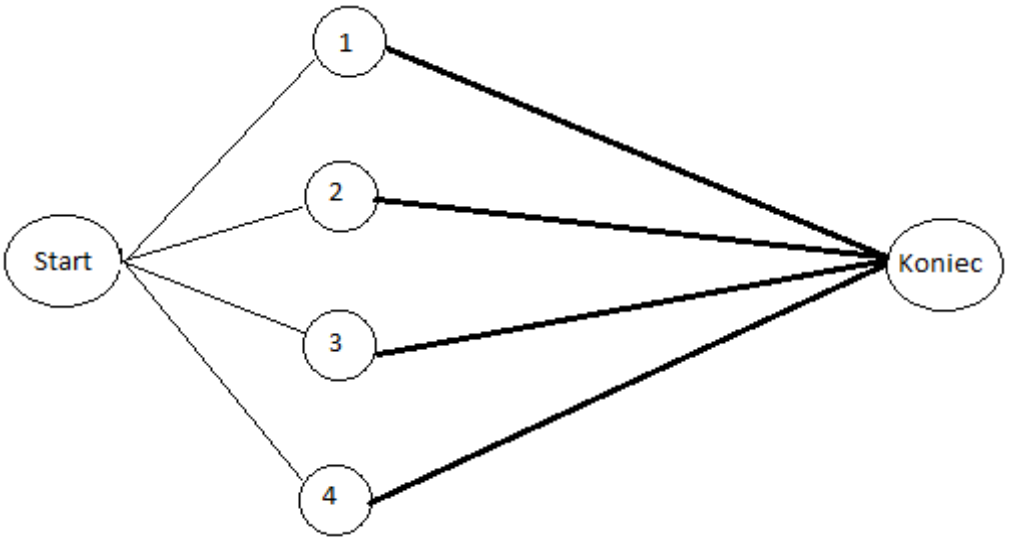
Programowanie dynamiczne znajduje optymalną wartość funkcji celu dla całego zagadnienia rozwiązując podproblemy od najmniejszego do największego i zapisując optymalne wartości w tablicy. Zapamiętywanie kolejnych rozwiązań gwarantuje, że dany problem jest rozwiązywany tylko raz, a także w niektórych przypadkach pozwala zastąpić wywołanie rekurencyjne odwołaniami do odpowiednich komórek tablicy.

Przykład z rysunku 3.4 przedstawia rozwiązanie problemu komiwojażera metodą programowania dynamicznego. Najpierw tworzone są trasy pomiędzy punktem startowym, a miastami 1, 2, 3 oraz 4, a następnie odpowiednie trasy z miast 1, 2, 3 i 4 do punktu końcowego rozwiązywane są analogicznie jako mniejsze podproblemy.

Podczas tworzenia algorytmów programowania dynamicznego stosuje się jedno z dwóch podejść:

- góra–dół (ang. Top-down) – przy tym podejściu każde rozwiązanie podproblemu jest zapamiętywane. Podczas rozwiązywania nowego podproblemu najpierw sprawdza się, czy takie rozwiązanie już nie istnieje. Jeśli nie, to nowy podproblem jest rozwiązywany, a następnie zapamiętany.

- dół–góra (ang. Bottom-Up) – przy tym podejściu rozwiązanie podproblemu służy do rozwiązania kolejnego bardziej ogólnego podproblemu.



Rys. 3.4. Przykład grafu służącego do rozwiązania problemu komiwojażera (źródło: opracowanie własne)

Z reguły oba podejścia charakteryzują się taką samą złożonością algorytmu, ale podejście dół-góra zajmuje mniej pamięci.

W literaturze nie spotyka się zbyt wielu implementacji programowania dynamicznego w rozwiązaniu problemów dostaw. Zespół Ceselli (Ceselli i in. 2004) zastosował programowanie dynamiczne do generowania kolumn w metodzie „cutting plane". Liczba kolumn, które są stworzone z kryteriów optymalizacji oraz warunków ograniczających, jest zmniejszana (obcinana) i szuka się rozwiązania dla uproszczonego problemu. W ostatecznym kroku rozwiązania uproszczonych problemów są grupowane i tworzone jest rozwiązanie dla ogólnego problemu.

Novoa i Storer (Novoa, Storer, 2008) zastosowali programowanie dynamiczne do rozwiązania stochastycznego problemu dostaw (ang. Stochastic Vehicle Routing Problem) oraz problemu dostaw ze stochastycznymi zapotrzebowaniami (ang. Vehicle Routing Problem with Stochastic Demands). W problemie SVRP jeden lub więcej elementów opisujących problem nie jest określony, tylko zdefiniowane jest prawdopodobieństwo wystąpienia danego elementu. Tymi zmiennymi elementami może być liczba klientów, ich zapotrzebowanie, czas podróży, odległości pomiędzy klientami. Problem VRPSD posiada niezdefiniowane zapotrzebowanie do klientów wszystkie pozostałe parametry są z góry określone. Novoa i Storer przyjęli podejście, w którym odwiedza się wszystkich klientów nawet tych, którzy w danym dniu nie mają żadnej dostawy.

Salani (Salani, 2006) zastosował programowanie dynamiczne do rozwiązania problemu dostaw z oknami czasowymi i swoje testy przeprowadził za pomocą zbiorów

Solomona. Niestety otrzymane wyniki były zdecydowanie gorsze od ówczesnych wyników. Przykładowo rozwiązania problemów R101 oraz R102 zaproponowane przez Salaniego zawierały odpowiednio 20 oraz 18 tras, natomiast najlepsze wyniki zawierały 19 i 17 tras.

Algorytm z wykorzystaniem programowania dynamicznego zastosowany w niniejszej pracy został przedstawiony na rysunkach 3.5 oraz 3.6.

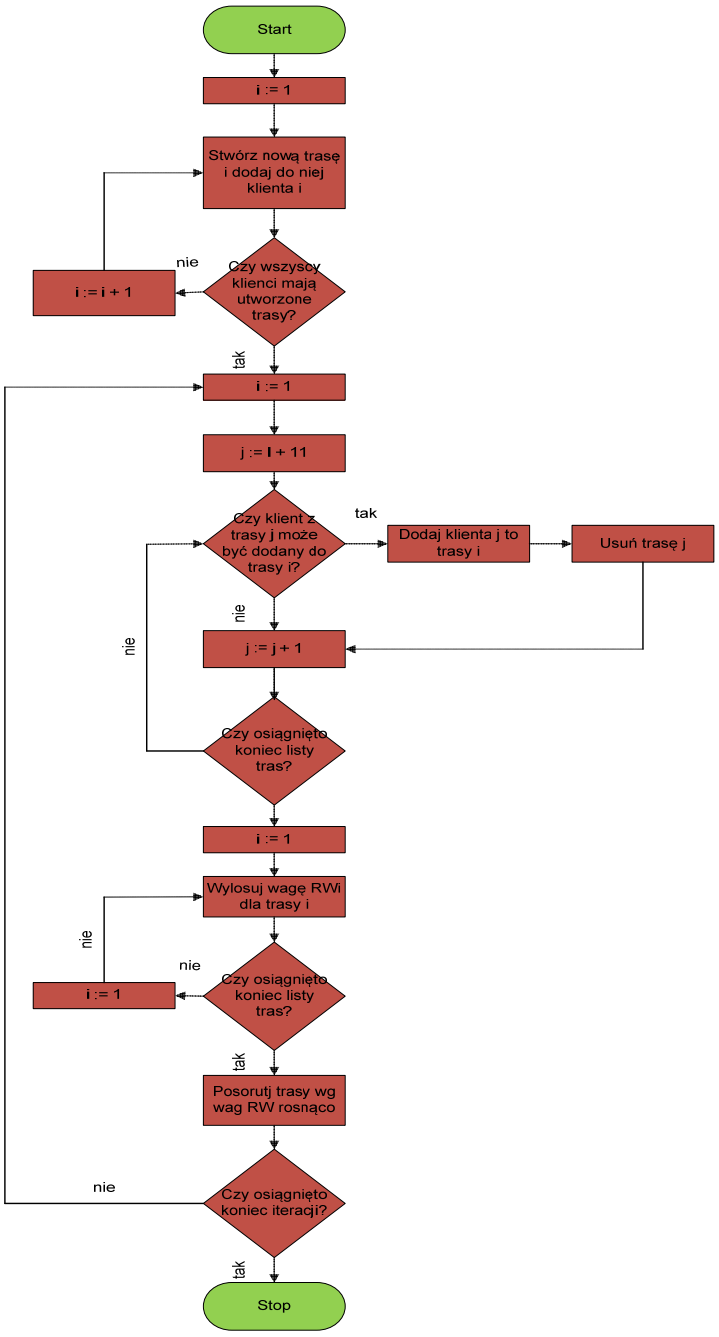
```
Krok 1. Stwórz rozwiązanie początkowe poprzez stworzenie tylu tras ile
jest klientów.
Krok 2. Dla każdej stworzonej trasy i
      Krok 2.1. Sprawdź czy klient z kolejnej trasy i+1 może zostać
      dodany do bieżącej trasy
              Krok 2.1.1. Jeśli tak, to dodaj klienta z trasy i+1 do
              trasy i
                        Krok 2.1.1.1 Usuń trasę i+1
              Krok 2.1.2. Jeśli nie, to sprawdzaj kolejne trasy aż do
              ostatniej lub do momentu dodanie klienta do trasy i
Krok 3. Dla każdej trasy wylosuj wagę RW
Krok 4. Posortuj trasy wg parametru RW rosnąco
Krok 5. Powtarzaj krok 2, aż nie do momentu, gdy nie da się już ulepszyć
rozwiązania lub wykonanu z góry zdefiniowaną liczbę kroków
Krok 6. Zwróć rozwiązanie
```

Rys. 3.5. Algorytm programowania dynamicznego do rozwiązania problemu dostaw z oknami czasowymi (źródło: opracowanie własne)

Algorytm przedstawiony na rysunkach 3.5 oraz 3.6 wykorzystuje wersję góra-dół programowania dynamicznego. Sortowanie tras w kroku 3 jest elementem dwukierunkowego modelu zaproponowanego przez Salani (Salani, 2006). Algorytm w kroku 1 tworzy zestaw tras, a następnieje redukuje ich liczbę oraz całkowitą przejechaną długość poprzez usuwanie klientów z jednej trasy i dodawanie do kolejnej. Sprawdzanie, czy nowe rozwiązanie spełnia warunki ograniczające tj. okna czasowe klientów, nieprzekraczanie pojemności pojazdów, itp., jest realizowane w kroku 2.1. W kroku 5 sprawdza się, czy nowe rozwiązanie jest lepsze od poprzedniego. Jeśli tak, to nowe rozwiązanie jest akceptowane, jeśli nie, to zwracane jest poprzednie rozwiązanie.

W niniejszej wersji algorytmu algorytm kończył działanie, gdy nie udało się polepszyć rozwiązania przez 50 kroków. Wartość 50 jest dobrana eksperymentalnie.

Podczas prób działania algorytmu zdarzało się, że po 40 niezmienionych iteracjach nagle udało się polepszyć bieżące rozwiązanie, natomiast jeśli rozwiązanie nie zostało polepszone przez 50 iteracji, to nie udało się już go polepszyć bez względu na to jak długo działał program implementujący niniejszy algorytm.



Rys. 3.6. Schemat blokowy algorytmu programowania dynamicznego do rozwiązania problemu dostaw z oknami czasowymi (źródło: opracowanie własne)

Aplikacja implementująca niniejszy algorytm została napisana w języku C++ i skompilowana za pomocą darmowego kompilatora Borland C++ 5.5 w środowisku Windows XP. Wg zamysłu autora niniejszej rozprawy ze względu na szybkość działania programów napisanych w języku C++ aplikacja ta miała być prototypem pod przyszłą implementację z języku C/AL.

W tabelach 3.9 – 3.14 przedstawione zostały aktualne najlepsze wyniki światowe oraz uzyskane rezultaty za pomocą opisywanego w niniejszym rozdziale algorytmu. Testy przeprowadzone zostały tylko za pomocą zbiorów Solomona. Wyniki uzyskane za pomocą programowania dynamicznego są dużo gorsze od obecnie znanych wyników światowych, dlatego testowanie algorytmu na trudniejszych testach Cordeau lub testach rzeczywistych zostało pominięte.

W pierwszej kolumnie tabel znajduje się nazwa każdego testu, w drugiej kolumnie przedstawione są najlepsze wyniki światowe, a w trzeciej kolumnie wyniki testów niniejszego algorytmu.

- w pierwszej z nich są inicjały autora lub autorów najlepszego światowego wyniku. Lista tych autorów oraz prace, gdzie te wyniki zostały opublikowane znajdują się w tabeli 3.21.

- W drugiej i trzeciej kolumnie znajdują się najlepsze rezultaty danych testów z podziałem na liczbę tras oraz całkowitą długość tras.

Kolumna „Uzyskane wyniki” z wynikami niniejszego algorytmu także została podzielona na dwie podkolumny, w których znajdują się rezultaty testów z podziałem na liczbę tras oraz całkowitą długość tras.

Nazwa testu	Wyniki światowe			Uzyskane wyniki	
	Autor	Liczba tras	Dług. Tras	Liczba tras	Dług. Tras
C101	RT	10	828,94	11	925,34
C102	RT	10	828,94	11	885,34
C103	RT	10	828,06	12	1023,45
C104	RT	10	824,78	10	824,78
C105	RT	10	828,94	10	838,56
C106	RT	10	828,94	11	967,80
C107	RT	10	828,94	10	828,94
C108	RT	10	828,94	11	1001,05

C109	RT	10	828,94	12	957,01

Tabela 3.9. Wyniki testów dla grupy C10x (źródło: opracowanie własne)

Nazwa testu	Wyniki światowe			Uzyskane wyniki	
	Autor	Liczba tras	Dług. Tras	Liczba tras	Dług. Tras
C201	RT	3	591,56	3	608,09
C202	RT	3	591,56	4	585,07
C203	RT	3	591,17	4	607,02
C204	RT	3	590,60	3	590,60
C205	RT	3	588,88	3	597,90
C206	RT	3	588,49	3	604,09
C207	RT	3	588,29	3	617,45
C208	RT	3	588,32	3	607,89

Tabela 3.10. Wyniki testów dla grupy C20x (źródło: opracowanie własne)

Nazwa testu	Wyniki światowe			Uzyskane wyniki	
	Autor	Liczba tras	Dług. Tras	Liczba tras	Dług. Tras
R101	H	19	1645,79	20	1815,56
R102	RT	17	1486,12	18	1513,09
R103	LLH	13	1292,68	14	1312,04
R104	M	9	1007,24	10	1204,15
R105	RT	14	1377,11	15	1381,13
R106	M	12	1251,98	13	1432,10
R107	S97	10	1104.66	11	1198,07
R108	BBB	9	960.88	10	1003,04
R109	HG	11	1194.73	12	1318,46
R110	M	10	1118,59	11	1203,09
R111	RGP	10	1096,72	11	1099,86
R112	GTA	9	982,14	11	1004,83

Tabela 3.11. Wyniki testów dla grupy R1x (źródło: opracowanie własne)

Nazwa testu	Wyniki światowe			Uzyskane wyniki	
	Autor	Liczba tras	Dług. Tras	Liczba tras	Dług. Tras
R201	HG	4	1252,37	5	1345,45
R202	RGP	3	1191,70	5	1215,41
R203	M	3	939,54	4	1103,04
R204	BVH	2	825,52	4	915,34
R205	RGP	3	994,42	4	998,02
R206	SSSD	3	906,14	4	956,12
R207	BVH	2	890,61	4	1003,23
R208	M	2	726,75	3	867,96
R209	H	3	909,16	4	945,17
R210	M	3	939,34	4	966,40
R211	BVH	2	892,71	4	1015,23

Tabela 3.12. Wyniki testów dla grupy R2x (źródło: opracowanie własne)

Nazwa testu	Wyniki światowe			Uzyskane wyniki	
	Autor	Liczba tras	Dług. tras	Liczba tras	Dług. tras
RC101	TBGGP	14	1696,94	14	1854,34
RC102	TBGGP	12	1554,75	13	1687,34
RC103	S98	11	1261,67	12	1324,65
RC104	CLM	10	1135,48	11	1275,90
RC105	BBB	13	1629,44	13	1814,23
RC106	BBB	11	1424,73	12	1654,23
RC107	S97	11	1230,48	12	1340,12
RC108	TBGGP	10	1139,82	11	1230,00

Tabela 3.13. Wyniki testów dla grupy RC10x (źródło: opracowanie własne)

Nazwa testu	Wyniki światowe			Uzyskane wyniki	
	Autor	Liczba tras	Dług. Tras	Liczba tras	Dług. tras
RC201	M	4	1406,91	4	1518,23
RC202	CC	3	1365,65	3	1371,02

RC203	CC	3	1049,62	4	1103,04
RC204	M	3	798,41	4	815,98
RC205	M	4	1297,19	4	1345,78
RC206	H	3	1146,32	5	1312,06
RC207	BVH	3	1061,14	4	1108,09
RC208	IKMUY	3	828,14	3	915,64

Tabela 3.14. Wyniki testów dla grupy RC20x (źródło: opracowanie własne)

Dla 56 przeprowadzonych testów Solomona w trzech przypadkach udało się wyrównać najlepsze wyniki światowe. W pozostałych przypadkach wyniki różnią się liczbą tras oraz ich długością.

Słabe wyniki osiągnięte za pomocą tej metody spowodowały, że autor niniejszej rozprawy zaprzestał badań w tym kierunku.

3.3. Zastosowane heurystyki do rozwiązywania problemów dostaw

Metaheurystyki takie jak symulowane wyżarzanie, algorytmy genetyczne, lub przeszukiwanie tabu zostały z powodzeniem użyte w celu głębokiego przeszukiwania przestrzeni rozwiązań i ucieczki z optimum lokalnego. Algorytmy genetyczne należą do grupy algorytmów przeszukiwania lokalnego, a polepszanie rozwiązania odbywa się w sąsiedztwie rozwiązania bieżącego. Przeszukiwanie tabu oraz symulowane wyżarzanie należą do innej grupy metaheurytyk, które pozwalają na wybór gorszego rozwiązania w momencie osiągnięcia optimum lokalnego. W wielu przypadkach pozwala to na opuszczenie tego optimum w celu lepszego przeszukiwania przestrzeni rozwiązań.

Symulowane wyżarzanie jest rozwinięciem metod przeszukiwania lokalnego, które polegają na ulepszaniu istniejącego rozwiązania do momentu, gdy nie udaje się go dalej poprawić (Cérny, 1985; Solomon, 1987; Woch, 2004). Podstawową wadą przeszukiwania lokalnego jest możliwość utknięcia w minimum lokalnym optymalizowanej funkcji. Symulowane wyżarzanie pomimo znalezienia minimum lokalnego, pozwala na dalsze przejście „w górę”, a tym samym na bardziej efektywne poszukiwanie optimum funkcji kosztu.

Podstawy symulowanego wyżarzania zostały po raz pierwszy opisane w roku 1953 przez Metropolis i in. (Metropolis, 1953) w algorytmie symulującym chłodzenie ciał stałych. W procesach ochładzania, jak i stygnięcia metali zaobserwowano, że jego właściwości zależą od szybkości chłodzenia. Przy powolnym stygnięciu cząsteczki ciała oddając energię rozkładają się w sposób bardziej systematyczny tworząc równomierne struktury. Natomiast jeśli chłodzenie będzie zbyt szybkie, to cząsteczki rozłożą się bardziej chaotycznie.

Algorytm Metropolis, wzorujący się na prawach termodynamiki, symuluje zmianę energii systemu podczas procesu wyżarzania. Jeśli energia spada, to system przenosi się do nowego stanu, jeżeli energia wzrasta, to nowy stan jest akceptowany zgodnie z prawdopodobieństwem danym wzorem 3.7:

$$p = e^{\frac{-\Delta E}{kT}} \tag{3.7}$$

gdzie: p to prawdopodobieństwo akceptacji, E - energia systemu, K - stała Boltzmana, a T to temperatura.

Proces wyżarzania jest powtarzany, aż do schłodzenia materiału.

Na początku lat osiemdziesiątych S. Kirkpatrick i in. (Kirkpatrick i in., 1983) oraz niezależnie od nich V. Cérny (Cérny, 1985) zaproponowali, aby algorytmu Metropolis użyć do znajdywania rozwiązań problemów optymalizacyjnych.

Większość praktycznych zastosowań symulowanego wyżarzania dotyczy przeszukiwania przestrzeni dyskretnych. Po raz pierwszy Vanderbilt i Louie (Vanderbilt, Louie, 1984) zaproponowali użycie tej metaheurystyki do przeszukiwania przestrzeni ciągłych. Przestrzeń rozwiązań problemu dostaw jest zbiorem dyskretnym, dlatego w niniejszej pracy nie ma omówionych badań będących kontynuacją prac Vanderbilt i Louie (Vanderbilt, Louie, 1984).

Chipperfield i in. (Chipperfield i in., 1997) wyróżnili pięć podstawowych cech, które powinien spełniać każdy algorytm symulowanego wyżarzania:

a. opis systemu,

b. mechanizm przejścia,

c. funkcja energii, funkcja kosztu,

d. kryteria akceptacji nowego rozwiązania,

e. harmonogram wyżarzania, chlodzenia.

Ogólny algorytm symulowanego wyżarzania dla minimalizacji funkcji *F*, gdzie *N* oznacza sąsiedztwo jest przedstawiony na rysunku 3.7, natomiast schemat blokowy na rysunku 3.8:

```
Krok 1. Stwórz rozwiązanie początkowe x0
Krok 2. Określ temperaturę początkową T0
Krok 3. Dopóki i < liczba iteracji
      Krok 3.1. licznik=0
      Krok 3.2. Dopóki licznik < nrep
           Krok 3.2.1. Wybierz rozwiązanie x z sąsiedztwa N
           Krok 3.2.2. d = F(x) - F(x0)
           Krok 3.2.3. Jeżeli d < 0
                       x0 = x
           Krok 3.2.4. W przeciwnym wypadku
                Krok 3.2.4.1. Wylosuj liczbę b z przedziału [0,1]
                Krok 3.2.4.2. Jeżeli b < exp(-d/T)
                              x0 = x; //przejście w górę
           Krok 3.2.5. licznik = licznik + 1;
      Krok 3.3. T = A(T) //zmiana temperatury
Krok 4. Zwróć rozwiązanie
```

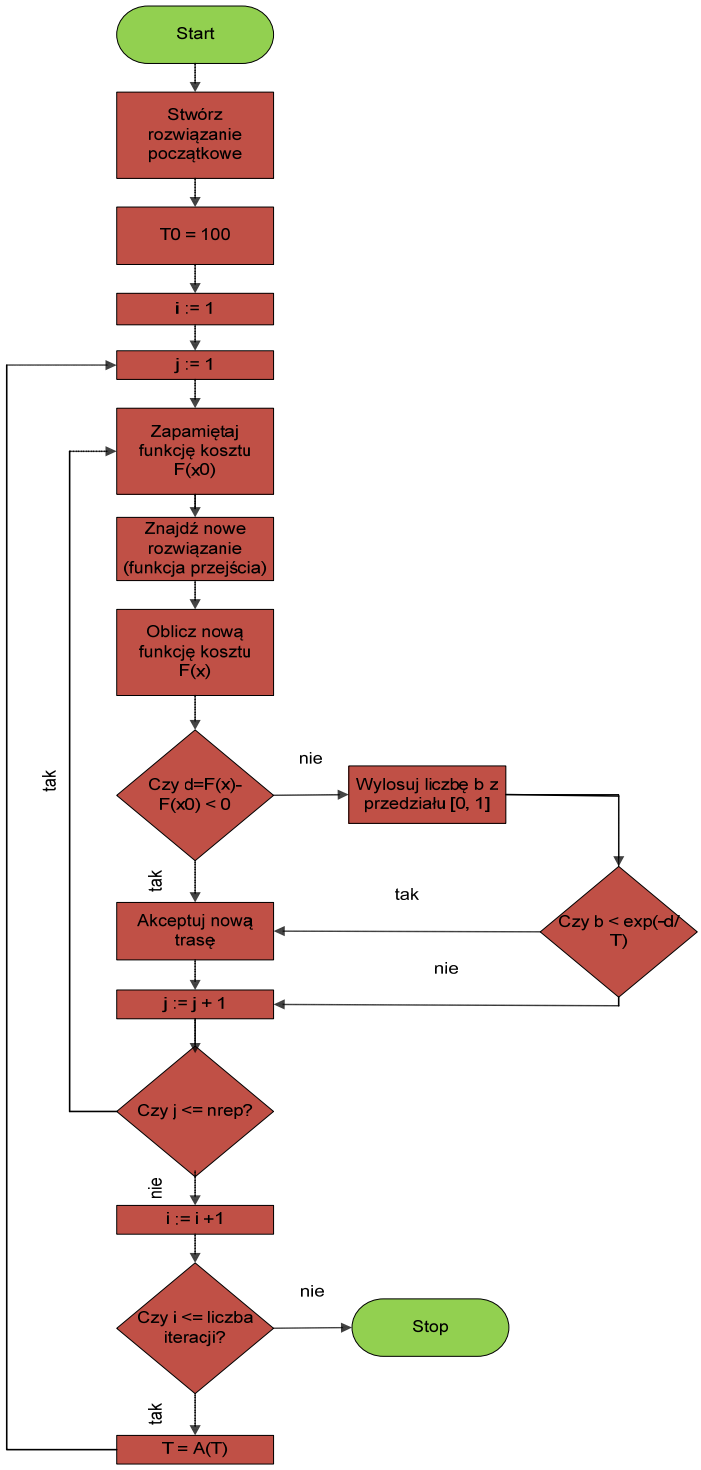
Rys. 3.7. Algorytm symulowanego wyżarzania (źródło: Chipperfield i in., 1997)

Ważną cechą symulowanego wyżarzania jest możliwość wyboru gorszego rozwiązania. Wybór taki jest dokonywany z prawdopodobieństwem danym wzorem 3.7. Rolę energii pełni tutaj różnica funkcji kosztu rozwiązań. Dzięki temu algorytm symulowanego wyżarzania może w określonych warunkach wyjść ze znalezionego minimum lokalnego i dalej podążać w kierunku rozwiązania optymalnego.

Zmiana stanu, czyli przejście z jednego potencjalnego rozwiązania do drugiego, jest realizowane za pomocą funkcji przejścia. Proces ten polega na znalezieniu rozwiązania sąsiedniego, co jest zależne od konkretnego problemu.

Wybór temperatury początkowej oraz funkcji zmiany temperatury bezpośrednio wpływa na zachowanie się algorytmu symulowanego wyżarzania. Prawdopodobieństwo wyboru gorszego rozwiązania jest zależne od temperatury. Jeśli końcowe rozwiązanie ma być jak najlepsze, temperatura początkowa musi być na tyle wysoka, aby umożliwić dokładne przeszukanie sąsiedztwa. Zaś zbyt wysoka temperatura początkowa może

spowodować spowolnienie pracy algorytmu poprzez przeszukiwanie zbyt dużej liczby potencjalnych rozwiązań.



Rys. 3.8. Schemat blokowy symulowanego wyżarzania (źródło: opracowanie własne na podstawie m.in. Cérny, 1985; Woch, 2004)

W początkowym etapie działania algorytmu, gdy temperatura jest wysoka, prawdopodobieństwo wyboru gorszego rozwiązania jest duże. Gdy temperatura maleje, prawdopodobieństwo również. Ten etap polega na ulepszaniu istniejącego rozwiązania. To właśnie znajduje odzwierciedlenie w drugim ważnym aspekcie algorytmu symulowanego wyżarzania, czyli w powolnym ochładzaniu.

Matematyczny model heurystyki symulowanego wyżarzania przedstawiają jednorodne łańcuchy Markowa.

Niech *X(k)* będzie zmienną oznaczającą wynik *k*-tej próby. W takim przypadku łańcuch Markova może być opisany za pomocą prawdopodobieństw warunkowych, które określają prawdopodobieństwo, że wynik k-tej próby jest przejściem ze stanu *i* do stanu *j* jest dany wzorem 3.8:

$$P_{ij}(k-1,k) = P[X(k) = j \mid X(k-1) = i] \tag{3.8}$$

W przypadku heurystyki symulowanego wyżarzania prawdopodobieństwo warunkowe P_{ij} nie zależy od *k*, czyli taki łańcuch Markova nazywa się łańcuchem jednorodnym. W procesie wyżarzania prawdopodobieństwa P_{ij} są prawdopodobieństwami przejść, a macierz prawdopodobieństw jednoskokowych przejść pomiędzy stanami $P=|R|*|R|$ jest macierzą przejść. Elementy macierzy *P* spełniają następujące warunki 3.9:

$$\begin{aligned} &\bigvee_{i,j} P_{ij} \geq 0 \\ &\bigvee_{i} \sum_{j \in R} P_{ij} = 1 \end{aligned} \tag{3.9}$$

Niech $a_i(k)$ oznacza prawdopodobieństwo wyniku *i* w *k-tej* próbie:

$$a_i(k) = P[X(k) = i] \tag{3.10}$$

czyli prawdopodobieństwo $a_i(k)$ można obliczyć w następujący sposób:

$$a_i(k) = \sum_{j \in R} a_j(k-1) P_{ji}(k) \tag{3.11}$$

W procesie symulowanego wyżarzania wartość temperatury T_n jest redukowana. Macierz P prawdopodobieństw P_{ij} jest zdefiniowana następująco:

$$\underset{i,j\in R}{\forall} P_{ij}(T_n) = \begin{cases} G_{ij}(T_n)A_{ij}(T_n) \Leftrightarrow j \neq i \\ 1 - \sum_{\substack{k=1 \\ k\neq 1}}^{|R|} G_{ik}(T_n)A_{ik}(T_n) \Leftrightarrow j = i \end{cases} \tag{3.12}$$

Z wzoru 3.12 wynika, że każde prawdopodobieństwo przejścia P_{ij} zależy od dwóch prawdopodobieństw warunkowych: prawdopodobieństwa G_{ij} wygenerowania stanu j ze stanu i oraz prawdopodobieństwa akceptacji A_{ij} stanu j wygenerowanego ze stanu i. Macierz G prawdopodobieństw G_{ij} jest nazywana macierzą generacyjną, a macierz A prawdopodobieństw A_{ij} macierzą akceptacji.

Prawdopodobieństwo G_{ij} nie zależy od temperatury:

$$\underset{i,j\in R}{\forall} G_{ij} = \left(\frac{1}{|R_i|}\right)\chi_{(R_i)}(j) \tag{3.13}$$

gdzie funkcja χ jest tzw. funkcją charakterystyczną zbioru stanów R, daną wzorem 3.14:

$$\chi_{(R)}(r) = \begin{cases} 1 \Leftrightarrow r \in R \\ 0 \Leftrightarrow r \notin R \end{cases} \tag{3.14}$$

Natomiast prawdopodobieństwo akceptacji dane jest wzorem 3.15:

$$A_{ij}(T_n) = \begin{cases} 1 & \Leftrightarrow \Delta E_{ij} \leq 0 \\ e^{-\frac{\Delta E_{ij}}{T}} & \Leftrightarrow \Delta E_{ij} > 0 \end{cases} \tag{3.15}$$

gdzie E oznacza energię systemu.

W niektórych implementacjach algorytmu symulowanego wyżarzania stosuje się wersję z zapamiętywaniem najlepszego rozwiązania. Algorytm ten wykorzystuje jeszcze jedno rozwiązanie oprócz bieżącego i sąsiedniego. Jest to zawsze rozwiązanie najlepsze,

które na początku działania algorytmu jest rozwiązaniem początkowym. W toku dalszego działania algorytmu rozwiązanie to zostaje zastąpione nowo znalezionym rozwiązaniem sąsiednim w przypadku, gdy jego koszt jest niższy niż koszt rozwiązania najlepszego. Na koniec działania algorytmu zwracane jest właśnie to rozwiązanie najlepsze. Dzięki tej modyfikacji końcowe rozwiązanie jest najlepszym możliwym znalezionym podczas działania algorytmu. Dlatego też ta wersja algorytmu została użyta w niniejszej pracy.

W niniejszej pracy w funkcji przejścia ze względu na dużo szerszy zakres omawianego problemu zdecydowano się użyć elementów heurystyki ALNS.

Adaptive Large Neighborhood Search to metoda oparta na przeszukiwaniu lokalnym, w której kilka prostych, niezależnych algorytmów konkuruje ze sobą w celu znalezienia najlepszego rozwiązania. W każdej iteracji wybierany jest jeden algorytm do zepsucia bieżącego rozwiązania i jeden do naprawienia tego rozwiązania. Nowe rozwiązanie jest zaakceptowane, jeśli spełnione są odpowiednie kryteria.

W niniejszej rozprawie zastosowano elementy ALNS w funkcji przejścia algorytmu symulowanego wyżarzania. Omówiona w dalszej części metoda zwiększających się promieni jest oparta o paradygmaty ALNS.

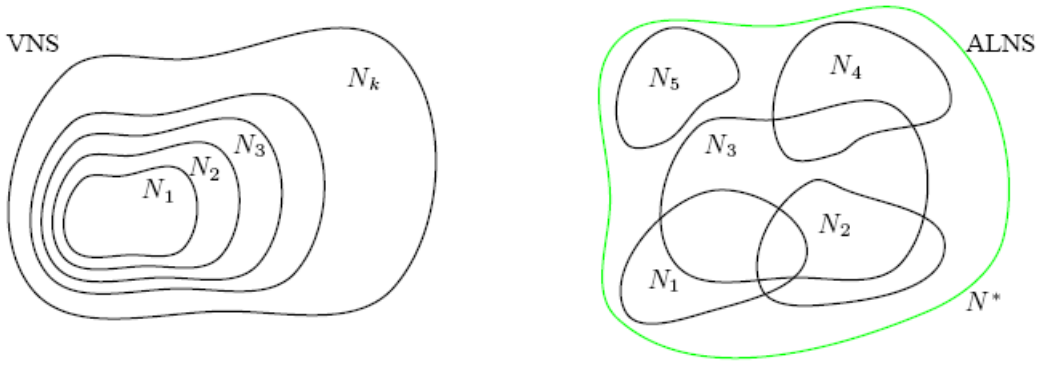
Ogólny algorytm ALNS jest przedstawiony na rys. 3.9:

```
Krok 1. Stwórz rozwiązanie początkowe x=x0
Krok 2. Powtarzaj
      Krok 2.1. Wybierz sąsiedztwo psujące oraz sąsiedztwo naprawiające
      Krok 2.2. Stwórz nowe rozwiązanie x' bazując na x oraz na
                wygenerowanych sąsiedztwach
      Krok 2.3. Jeżeli x' jest zaakceptowane to x=x'
      Krok 2.4. Zmodyfikuj zbiór punktów dla sąsiedztw psujących i
                naprawiających
      Krok 2.5. Jeśli x jest najlepszym rozwiązaniem to zapisz to jako
                najlepsze rozwiązanie
        Dopoki nie jest spełnione kryterium stopu
Krok 3. Zwróć rozwiązanie x
```

Rys. 3.9. Algorytm ALNS (źródło: Ropke, Pisinger, 2005)

Inaczej niż zmienne przeszukiwanie sąsiedztwa VNS, ALNS operuje na predefiniowanych zbiorach sąsiedztwa korespondujących z heurystykami niszczenia oraz

naprawiania rozwiązania. Sąsiedztwa te są zdefiniowane i wyszukiwane przez odpowiednie algorytmy. Różnice pomiędzy VNS, a ALNS przedstawia rysunek 2.4.4.



Rys. 3.10. Różnice pomiędzy VNS oraz ALNS (źródło: Ropke, Pisinger, 2005)

Na rysunku 3.10 widać, że w VNS przejścia pomiędzy kolejnymi sąsiedztwami oznaczonymi jako N_1, ..., N_k są od siebie zależne. W ALNS przeszukiwanie sąsiedztw jest bardziej szczegółowe, sąsiedztwa są niezależne od siebie i są one odnajdywane przez odpowiednie algorytmy.

W niniejszej pracy zastosowano elementy ALNS w funkcji przejścia algorytmu symulowanego wyżarzania.

3.4. Opis zastosowanej hybrydy symulowanego wyżarzania oraz adaptacyjnego przeszukiwania dużego sąsiedztwa

Autor niniejszej rozprawy rozpoczął swoje badania nad problemami dostaw w 2004 r. (Woch, 2004). Skupiono się wtedy na problemie dostaw z oknami czasowymi VRPTW. Prace nad problememami dostaw koncentrują się na dwóch głównych zagadnieniach, pierwsze to konstruowanie rozwiązania początkowego, a drugie jego ulepszanie. Użyta wtedy procedura generowania rozwiązania początkowego była zaczerpnięta z modelu zaproponowanego przez Solomona (Solomon, 1987). Procedura ta polegała na znajdywaniu najlepszego miejsca na trasach dla klientów posortowanych wg rosnącego okna czasowego. Algorytm ten był bardzo prosty w implementacji i generował stosunkowo dobre rozwiązania.

Funkcja przejścia zastosowana w wymienionym artykule (Woch, 2004) polegała na losowym usuwaniu grupy klientów z tras, a następnie wyszukiwaniu nowych najlepszych

miejsc na trasach. Dzięki tej metodzie udało się wyrównać kilka najlepszych światowych wyników testów Solomona (przede wszystkim z grup C1 oraz C2), niemniej jednak algorytm ten nie spełnił pokładanych w nim oczekiwań.

W 2008 r. autor rozbudował funkcję przejścia (Woch, Łebkowski, 2008) o głębsze przeszukiwanie sąsiedztwa. Zostało dodane ulepszanie tras wg własnego, oryginalnego algorytmu. Procedura generowania rozwiązania początkowego w opisywanym artykule (Woch, Łebkowski, 2008) nie została zmodyfikowana. Dzięki algorytmowi symulowanego wyżarzania, który wykorzystywał tą funkcję przejścia udało się wyrównać kilka kolejnych najlepszych wyników światowych testów Solomona, ale w dalszym ciągu osiągane wyniki różniły się od najlepszych rezultatów w niektórych przypadkach nawet liczbą tras.

Dopiero rozszerzenie badań na kolejne problemy dostaw (wielomagazynowe, różne pojemności pojazdów) zaskutkowało stworzeniem algorytmu w dzisiejszej, opisywanej w niniejszej pracy formie (Woch, Łebkowski, 2009). Zmodyfikowano funkcję generowania rozwiązania początkowego oraz funkcję przejścia. Nowa funkcja przejścia pozwala na przenoszenie całych grup klientów pomiędzy trasami. Zastosowana metoda grupuje klientów wg odległości od magazynu centralnego, dlatego ten algorytm został nazwany metodą zwiększających się promieni.

Zarówno nowa funkcja tworzenia rozwiązania początkowego, jak i metoda zwiększających się promieni zostały opisane w niniejszym rozdziale.

Procedura tworzenia rozwiązania początkowego jest wykorzystywana w pierwszym kroku algorytmu symulowanego wyżarzania. W niniejszej implementacji wykorzystano własną procedurę będącą modyfikacją procedury wykorzystanej w wersji algorytmu z roku 2004 (Woch, 2004), a jej ogólny algorytm jest przedstawiony na rysunkach 3.11 oraz 3.12:

```
Krok 1. Posortuj klientów rosnąco wg okna czasowego
Krok 2. Dla każdego klienta
      Krok 2.1. Dla i = 1 do MT_CLIENT
            Krok 2.1.1. Dla każdej trasy
                  Krok 2.1.1.1. Znajdź najlepsze miejsce na trasie dla
                  klienta
                  Krok 2.1.1.2. Jeżeli nowa długość trasy < poprzednia
                  długość
                        Zapamiętaj te trasę i miejsce
            Krok 2.1.2. Jeżeli znaleziono miejsce wstawienia
```

```
                    Wstaw klienta do tej trasy w określonym miejscu
          Krok 2.1.2. W przeciwnym wypadku
                    Wstaw klienta na początku nowo utworzonej trasy
          Krok 2.1.3. Usuń klienta z pierwotnej listy
     Krok 2.2. Dla i = Liczba_klientow_z_pierwotnej_listy w dół do
               LT_CLIENT
          Krok 2.2.1. Dla każdej trasy
               Krok 2.2.1.1. Znajdź najlepsze miejsce na trasie dla
               klienta
               Krok 2.2.1.2. Jeżeli nowa długość trasy < poprzednia
               długość
                    Zapamiętaj te trasę i miejsce
          Krok 2.2.2. Jeżeli znaleziono miejsce wstawienia
                    Wstaw klienta do tej trasy w określonym miejscu
          Krok 2.2.3. W przeciwnym wypadku
                    Wstaw klienta na początku nowo utworzonej trasy
          Krok 2.2.4. Usuń klienta z pierwotnej listy
Krok 3. Zapamiętaj znalezione rozwiązanie
```

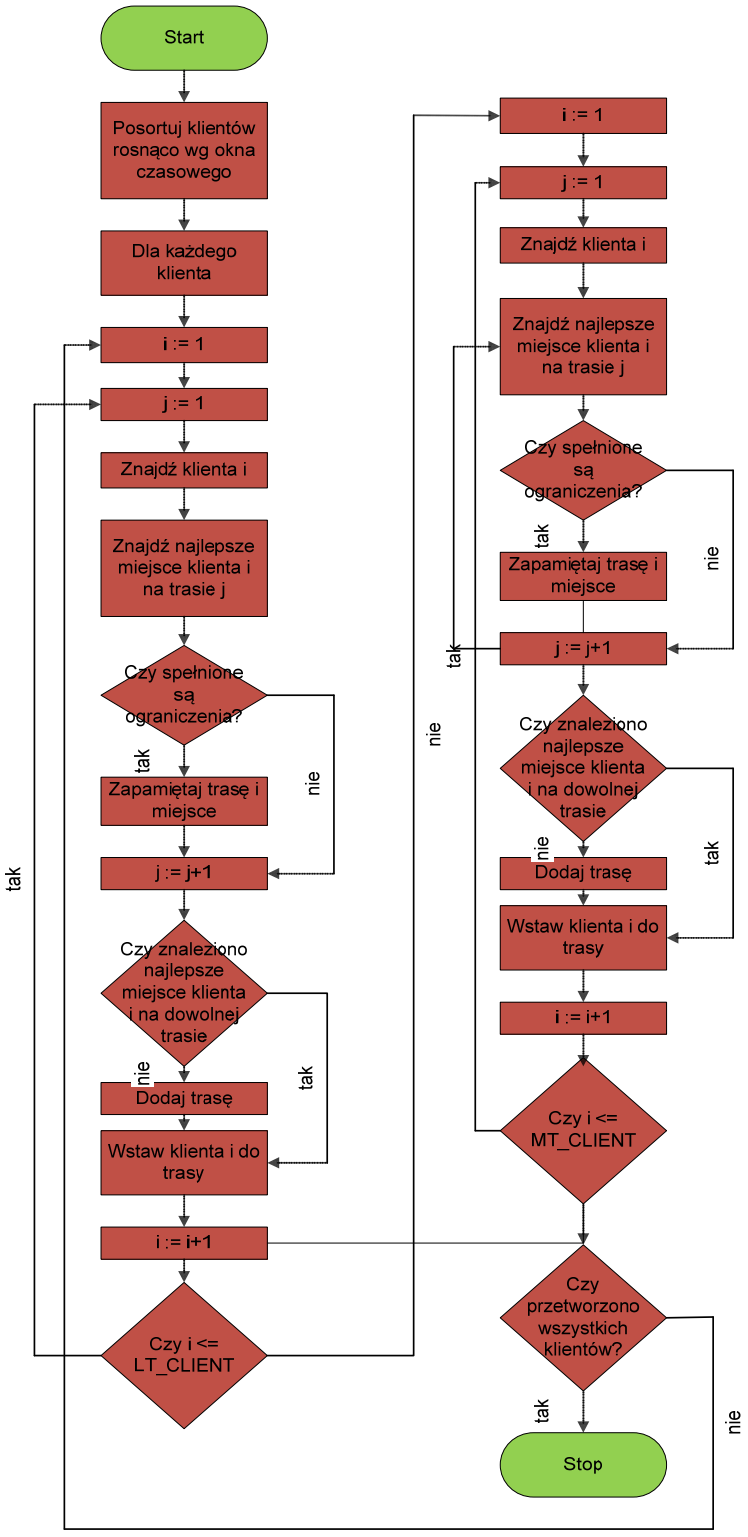
Rys. 3.11. Funkcja tworzenia rozwiązania początkowego (źródło: opracowanie własne)

Sortowanie wg rosnącego okna czasowego w kroku 1 pozwala na umieszczenie klientów z najmniejszym oknem czasowym na początku listy. Ci klienci są dzięki temu przetwarzani w pierwszej kolejności. W ramach przeprowadzanych eksperymentów próbowano różnych ustawień klientów, jednak przedstawione podejście dawało najlepsze rezultaty.

Okno czasowe klienta *i* jest obliczane wg wzoru 3.16:

$$T_i = L_i - E_i - S_i \tag{3.16}$$

gdzie: T_i – okno czasowe klienta *i,* L_i to najpóźniejszy możliwy czas zakończenia obsługi, E_i oznacza najwcześniejszy możliwy czas rozpoczęcia obługi, a S_i to czas obsługi.



Rys. 3.12. Funkcja tworzenia rozwiązania początkowego (źródło: opracowanie własne)

Algorytm generowania rozwiązania początkowego najpierw bierze pod uwagę kilku najbardziej kłopotliwych klientów – ich liczba zdefiniowana jest za pomocą parametru MT_CLIENT, a następnie przeszukuje kilku najmniej kłopotliwych klientów –

ich liczba definiowana jest za pomocą parametru LT_CLIENT. W toku doświadczeń i badań okazało się, że najbardziej efektywnymi wartościami parametrów MT_CLIENT oraz LT_CLIENT są odpowiednio 5% i 2% całkowitej liczby klientów. Dopasowywanie klientów z dużym oknem czasowym do klientów z małym oknem czasowym pozwala już na etapie tworzenia rozwiązania początkowego zredukować liczbę tras. Krok ten jest powtarzany aż do momentu, w którym wszyscy klienci są rozmieszczeni na trasach.

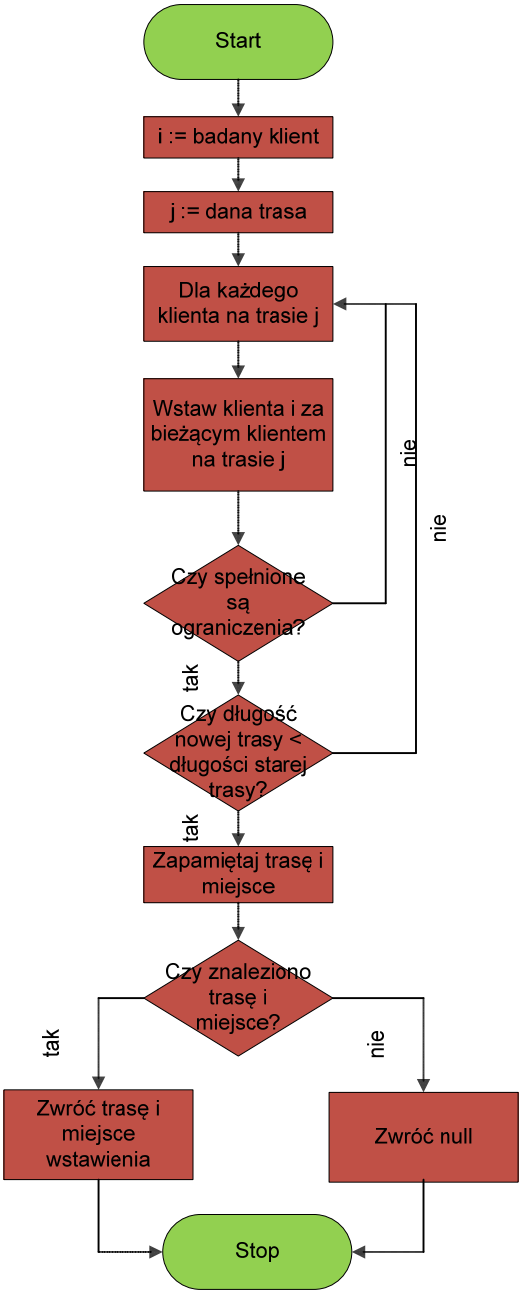
W procedurze wyznaczenia najlepszego miejsca na trasach do wstawienia posortowanych klientów wykorzystano własny algorytm przedstawiony na rysunkach 3.13 oraz 3.14:

```
Krok 1. Dla każdego klienta na trasie
      Krok 1.1. Wstaw badanego klienta za klientem bieżącym
      Krok 1.2. Jeżeli spełnione są warunki czasowe i nie jest
      przekroczona ładowność pojazdu oraz jeżeli nowa długość trasy <
      poprzednia długość
                        Zapamiętaj pozycję klienta na trasie
Krok 2. Zapamiętaj rozwiązanie
```

Rys. 3.13. Procedura wyznaczania najlepszego miejsca na trasie (źródło: opracowanie własne)

Funkcja wstawia badanego klienta za każdym odbiorcą na danej trasie sprawdzając, czy nie zostały przekroczone okna czasowe oraz dopuszczalna ładowność pojazdu. Jeżeli trasa jest poprawna to następuje porównanie długości trasy bieżącej z poprzednią.

Schemat blokowy funkcji wyznaczającej najlepsze miejsce na trasie został przedstawiony na rysunku 3.14.

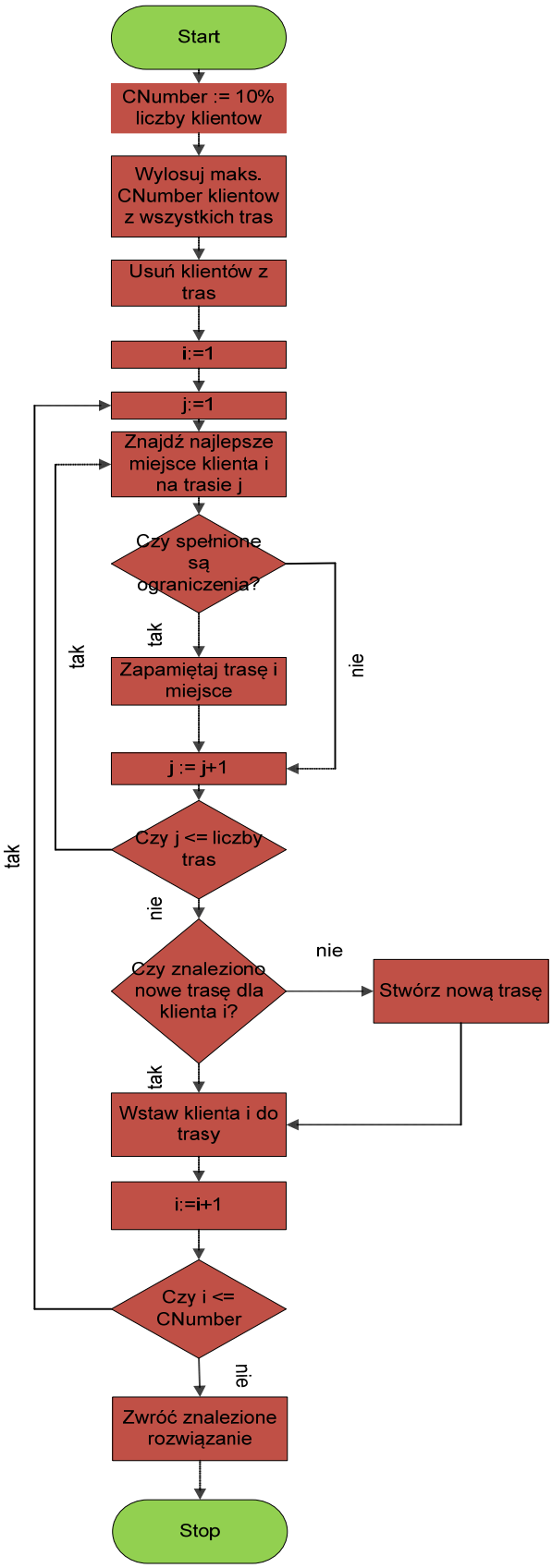


Rys. 3.14. Schemat blokowy funkcji znajdującej najlepsze miejsce na trasie (źródło: opracowanie własne)

Funkcja przejścia jest jednym z podstawowych elementów, który warunkuje sprawność algorytmu symulowanego wyżarzania. To od funkcji przejścia zależy zdolność algorytmu do wychodzenia z minimów lokalnych, co jest szczególnie istotne przy niskich wartościach temperatury. W niniejszej pracy postanowiono wykorzystać własną funkcję przejścia (rysunki 3.15, 3.16).

```
Krok 1. Wylosuj maks. liczbę klientów określoną parametrem CNUMBER w tym
celu
      Krok 1.1. Wylosuj trasę
      Krok 1.2. Wylosuj klienta Ci na trasie
      Krok 1.3. Zastosuj Metodę Zwiększających się Promieni do
      wylosowania sąsiedztwa danego klienta Ci
      Krok 1.4. Usuń wylosowanych klientów z tras
      Krok 1.6. Jeśli usunięto tylko jednego klienta wylosuj drugą trasę
      Krok 1.7. Wstaw wylosowanego klienta do drugiej trasy jeśli to jest
                możliwe
      Krok 1.8. Znajdź przeszkadzajacego klienta na drugiej trasie. Jeśli
                udalo się znaleźć zaznacz go do usunięcia.
      Krok 1.9. Dla zawadzającego klienta z drugiej trasy wykonaj kroki
      jak w kroku 1.3.
      Krok 1.10.Jeśli liczba klientów < CNUMBER wróć do kroku 1.1.
      Krok 1.11.Jeśli liczba klientów > CNUMBER zakończ działanie funkcji
      i wróc do rozwiązania wcześniejszego
Krok 2. Posortuj trasy wg dystansów rosnąco
Krok 3. Dla każdego wylosowanego klienta
      Krok 3.1. Dla każdej trasy
            Krok 3.1.1. Znajdź najlepsze miejsce dla bież. klienta na
            trasie
            Krok 3.1.2. Jeżeli nowa długość trasy < poprzednia długość
                        Zapamiętaj tę trasę i miejsce
      Krok 3.2. Jeżeli znaleziono miejsce wstawienia
                Wstaw klienta do tej trasy w określonym miejscu
      Krok 3.3. W przeciwnym wypadku
                Wstaw klienta na początku nowo utworzonej trasy
Krok 4. Dla każdej trasy
      Krok 4.1. Ulepsz pojedynczą trasę
Krok 5. Posortuj trasy wg liczby klientów rosnąco
Krok 6. Ulepsz wszystkie trasy
```

Rys. 3.15. Funkcja przejścia pomiędzy rozwiązaniami (źródło: opracowanie własne)



Rys. 3.16. Schemat blokowy funkcji przejścia pomiędzy rowiązaniami (źródło: opracowanie własne)

Zaletą zaprezentowanego algorytmu jest szybkość działania oraz dokładne przeszukiwanie przestrzeni rozwiązań. Elementem tej procedury jest zaproponowany przez autora sposób grupowania klientów w zależności od odległości od bieżącego klienta.

Metoda ta polega na zwiększaniu promienia odległości od bieżącego klienta i sprawdzaniu, czy wewnątrz okręgu stworzonego przez ten promień znajdują się jacyś klienci. Jeśli tak, to taki klient zostaje dodany do losowanych klientów wg określonego prawdopodobieństwa. Prawdopodobieństwo to zależy od promienia. Im mniejszy promień, tym prawdopodobieństwo wylosowania klienta jest większe. Sposób ten został nazwany metodą zwiększających się promieni.

Ten algorytm wspomaga przenoszenie całych grup klientów pomiędzy trasami, co ma szczególne znaczenie podczas niskich temperatur.

W toku badań stwierdzono, że najlepszą wartością dla parametru wyrażającego liczbę usuniętych klientów w pojedynczej iteracji CNUMBER jest 10-15 procent liczby wszystkich klientów. CNUMBER określa maksymalną liczbę klientów usuniętych z tras w pojedynczej iteracji. W procesie ich ponownego wstawiania do tras klienci umieszczani są w kolejności wylosowania. Autor we wcześniejszych wersjach algorytmu sortował wylosowanych klientów według rosnącego okna czasowego. Preferowanie klientów z węższymi oknami podczas ponownego wstawiania do tras miało za zadanie szybsze zmniejszanie liczby tras. Klienci z węższymi oknami czasowymi są trudniejsi do wstawienia niż klienci w szerszymi oknami, co powoduje, że liczba miejsc, gdzie można ich wstawić, jest mniejsza. Niemniej jednak podczas eksperymentów okazywało się, że tego typu sortowanie powodowało duża większą powtarzalność wygenerowanych rozwiązań. Także w wielu przypadkach wstawienie klienta z węższym oknem czasowym w miejsce, które w danym momencie okazywało się najlepsze docelowo, powodowało zwiększanie liczby tras, gdyż nie udawało się już wstawiać kolejnych klientów. Głównie ze względu na uniknięcie powtarzalności rozwiązań podczas kolejnych iteracji zdecydowano się na ponowne wstawianie klientów do tras według kolejności losowania. Spowodowało to polepszenie wygenerowanych rezultatów.

W kroku 4 wykonywana jest funkcja ulepszania pojedynczej trasy. Polega ona na znajdywaniu na trasie najlepszego położenia dla każdego klienta. W tym celu bieżący klient jest z tej trasy usuwany, a następnie znajdywane jest jego najlepsze położenie w tej marszrucie. Sposób znajdywania najlepszego miejsca na trasie przedstawiony jest na

rysunkach 3.13 oraz 3.14. Jeżeli najlepszym miejscem dla danego klienta jest jego położenie początkowe, to trasa nie zmienia się.

W kroku 5 funkcji przejścia trasy są sortowane wg liczby klientów na trasie, a w przypadku, gdy liczba klientów dla kilku tras jest taka sama, to porównywane są sumy żądań dostawy dla klientów z danej trasy. Sortowanie umożliwia ustawienie tras z najmniejszą liczbą klientów na początku, co jest bardzo przydatne w kroku 6, czyli ulepszaniu tras.

Metoda zwiększających się promieni analizuje, kto znajduje się w jakiej odległości od bieżącego klienta. Im mniejsza odległość tym prawdopodobieństwo wylosowania „sąsiada" bieżącego klienta jest większe.

Prawdopopodobieństwo to wyliczane jest ze wzoru 3.17:

$$P = 1/D_k \tag{3.17}$$

gdzie D_i to odległość od klienta dla wewnętrznej iteracji k wyliczana wg wzoru 3.18.

$$D_k = \begin{cases} D_0 \text{ dla } k=0 \\ Z+D_{k-1} \text{ dla } k>0 \end{cases} \tag{3.18}$$

D_0 to najmniejsza odległość pomiędzy dowolnymi dwoma klientami dla danej trasy. Z to współczynnik zwiększenia promienia. Eksperymenty i doświadczenia pokazały, że maksymalną liczbą iteracji $k_{max} = 10$, a współczynnik zwiększenia promienia Z to 10%.

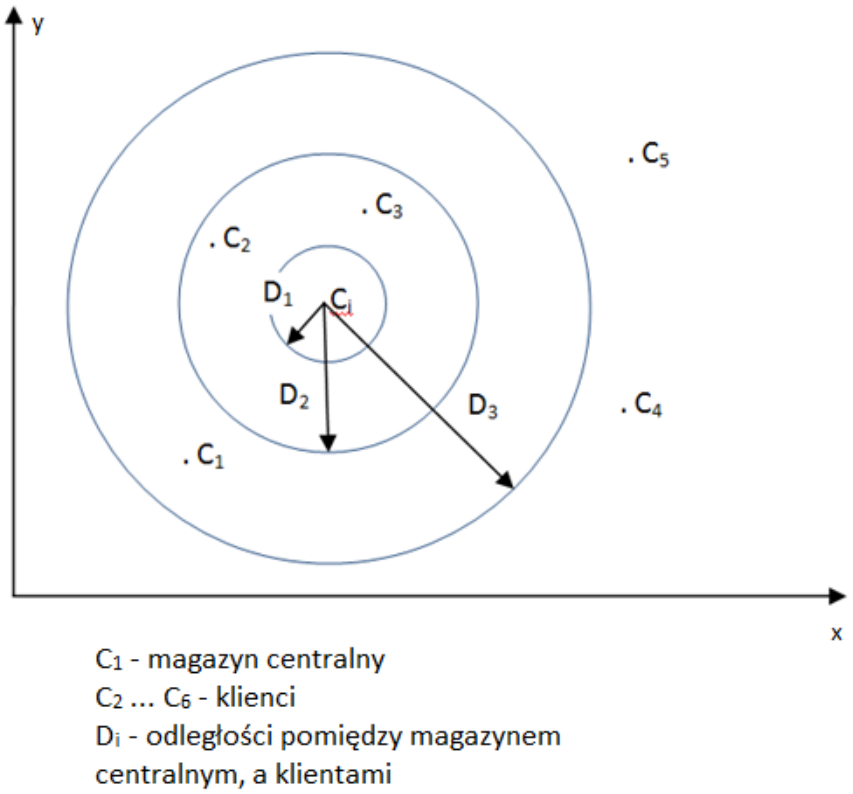
Schemat metody zwiększających się promieni jest przedstawiony na rysunku 3.17:

```
Krok 1. Dane są trasa i klient Ci
Krok 2. Znajdź D0 najmniejszą odległość pomiędzy 2 dowolnymi klientami na
trasie
Krok 3. Dla k = 0 do kmax
     Krok 3.1. Jeśli k <> 0 to
                Oblicz bieżącą odległość Dk
     Krok 3.2. Wylosuj klienta Ck z trasy takiego, że Ck jest różny od Ci
     Krok 3.3. Jeśli Ck znajduje się w odległości mniejszej lub równej Dk
                od klienta Ci to
          Krok 3.4.1. Z prawdopodobieństwem P dodaj klienta Ck do listy
                      klientów do usunięcia
     Krok 3.5. Jeśli Ck jest się w odległości większej Dk od klienta Ci to
```

```
        Krok 3.5.1. Wylosuj liczbę P z przedziału obustronnie
                    domkniętego [0, 1]
                    Dodaj klienta Ck do listy usuniętych klientów z
                    prawdopodobieństwem P
Krok 4. Zwróć listę usuniętych klientów
```

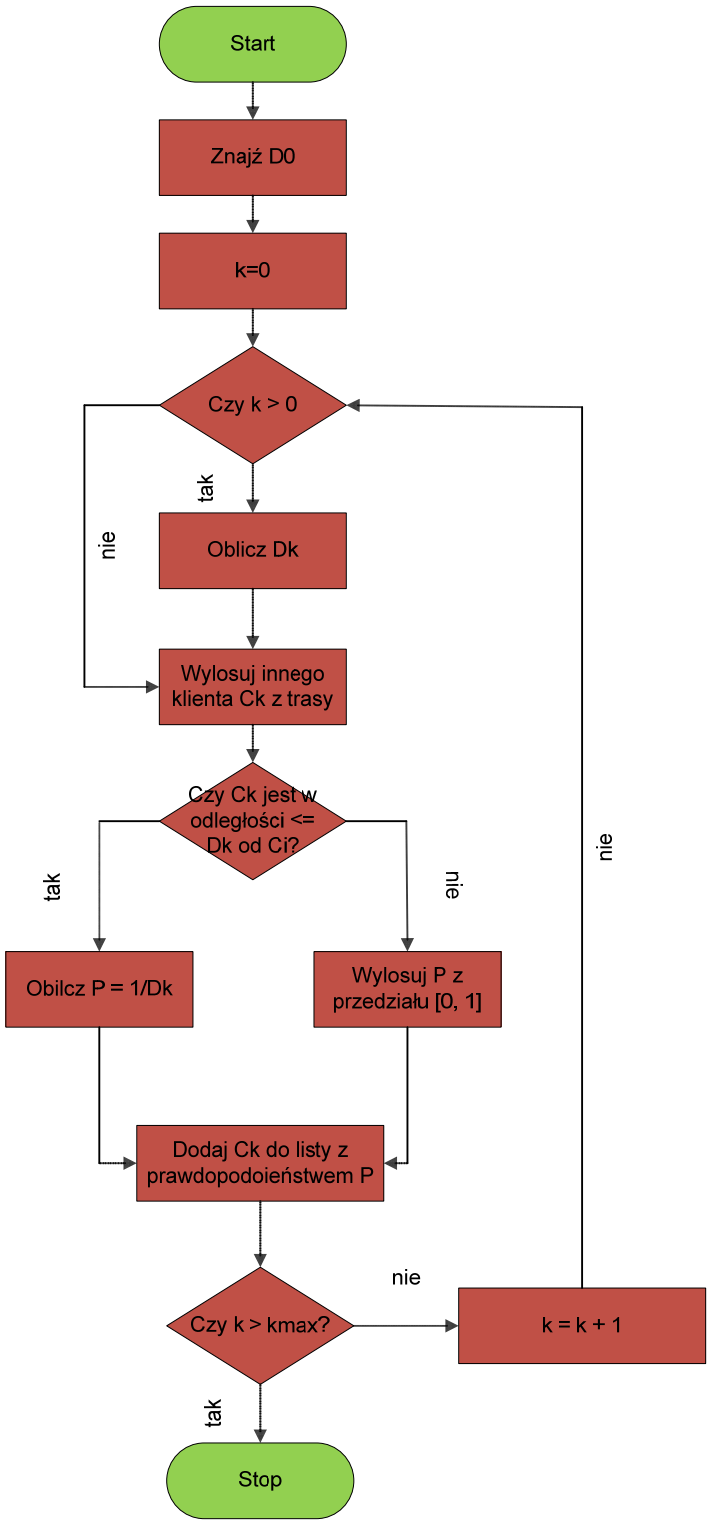
Rys. 3.17. Algorytm metody zwiększających się promieni (źródło: opracowanie własne)

Rysunek 3.18 ilustruje metodę zwiększających się promieni.



Rys. 3.18. Ilustracja metody zwiększających się promieni (źródło: opracowanie własne)

Rysunek 3.19 przedstawia schemat blokowy algorytmu zwiększających się promieni.



Rys. 3.19. Schemat blokowy metody zwiększających się promieni (źródło: opracowanie własne)

Opracowany przez autora niniejszej pracy algorytm ulepszania tras polega na usuwaniu po kolei każdego klienta z trasy i wstawianiu go w inne miejsce. Ogólny schemat algorytmu jest przedstawiony na rysunku 3.20.

```
Krok 1. Dla każdej trasy
     Krok 1.1. Dla każdego klienta na trasie
          Krok 1.1.1. Usuń klienta z trasy
               Krok 1.1.1.1. Dla każdej trasy za wyjątkiem trasy bieżącej
                         Krok 1.1.1.1.1. Znajdź najlepsze położenie
                         klienta
                         Krok 1.1.1.1.2. Jeżeli koszt nowego rozwiązania <
                         Koszt poprzedniego rozwiązania
                                   Zapamiętaj te trasę i miejsce
                         Krok 1.1.1.1.3. Jeżeli znaleziono miejsce
                         wstawienia
                                   Wstaw klienta do tej trasy w
                                   określonym miejscu
       Krok 1.2. Jeżeli podstawowa trasa jest pusta
                 Usuń trasę
       Krok 1.3. Jeżeli nie znaleziono miejsca wstawienia
                 Wstaw klienta do trasy, z której go usunięto
Krok 2. Zapamiętaj znalezione rozwiązanie
```

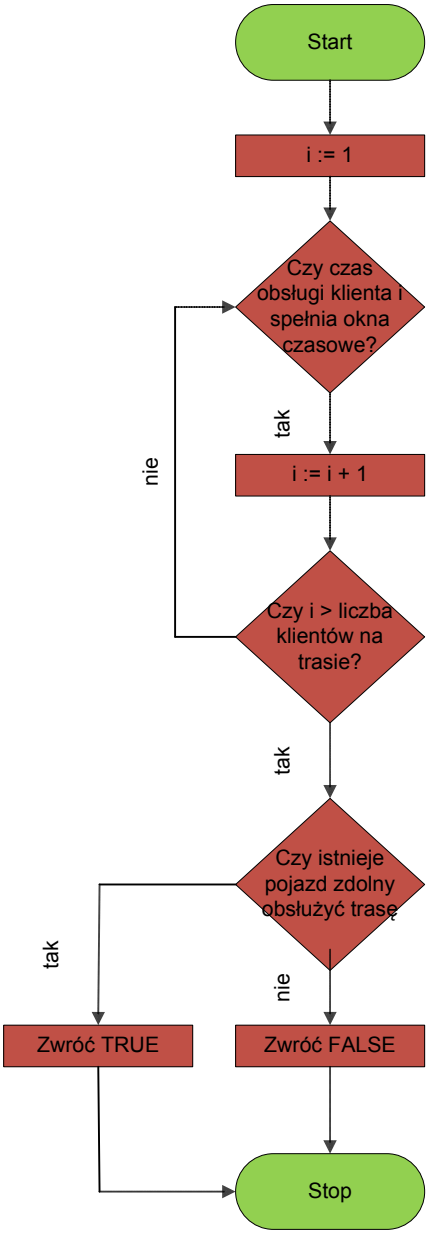
Rys. 3.20. Algorytm ulepszania tras (źródło: opracowanie własne)

Dzięki posortowaniu tras wg rosnącej liczby klientów na trasie jest możliwe usunięcie krótkich tras, które powstały we wcześniejszych krokach.

Przy każdorazowym dodaniu klienta do trasy sprawdzana jest poprawność trasy. Polega to na sprawdzeniu, czy trasa spełnia wszystkie zadane ograniczenia. Algorytm przedstawiony jest na rysunkach 3.21 oraz 3.22:

```
Krok 1. Dla każdego klienta na trasie
     Krok 1.1. Jeśli czas przybycia do klienta/magazynu nie spełnia
     warunku okna czasowego
                Przerwij działanie funkcji i zwróć FALSE
Krok 2. Sprawdź czy istnieje pojazd zdolny obsłużyc trasę, w tym celu
     Krok 2.1. Sprawdź czy istnieje pojazd o nośności większej bądź
     równej niż waga towarów i objętości większej bądź równej niż
     objętość towarów
Krok 3. Zwróć TRUE
```

Rys. 3.21. Algorytm sprawdzający trasy (źródło: opracowanie własne)



Rys. 3.22. Schemat blokowy funkcji sprawdzającej trasy (źródło: opracowanie własne)

W ostatnim kroku zastosowanego w niniejszej rozprawie algorytmu symulowanego wyżarzania następuje przypisanie pojazdów do obliczonych tras. Dla każdej znalezionej trasy system przypisuje pojazd na postawie następujących warunków:

- pojazd musi mieć większą bądź równą nośność niż całkowita waga wszystkich towarów dostarczanych do klientów na trasie,

- pojazd musi mieć większą bądź równą objętość od sumarycznej objętości wszystkich towarów dostarczanych do klientów na trasie,

- z listy pojazdów spełniających te dwa warunki wybierany jest pojazd o najmniejszym koszcie jednostkowym.

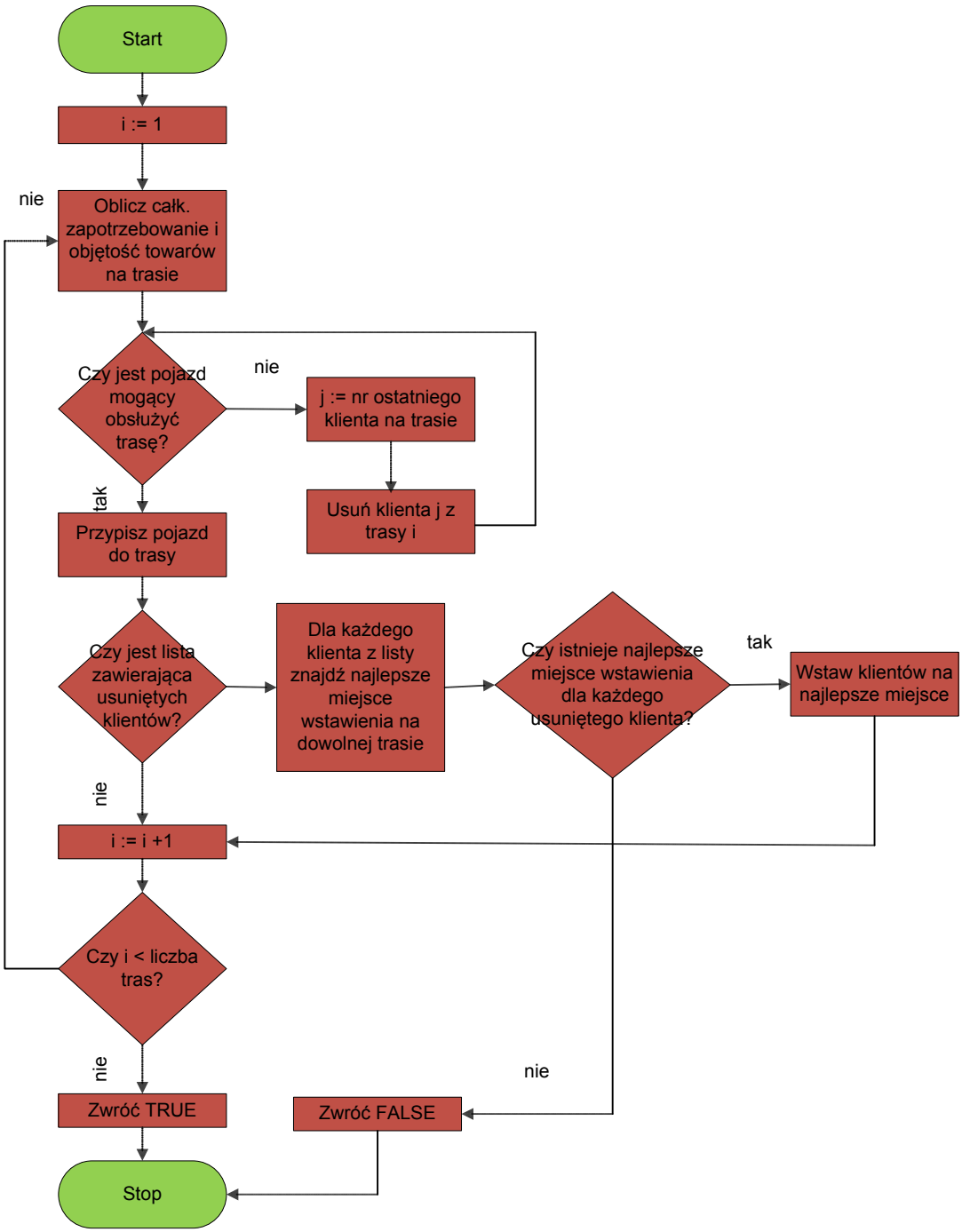
W sytuacji, gdy żaden pojazd nie spełnia warunków dotyczących minimalnej objętości, system próbuje znaleźć na trasie „blokującego" klienta i znaleźć dla niego inną trasę. Gdy algorytm nie jest w stanie przypisać usuniętych klientów do żadnych innych tras spełniających ograniczenia, wówczas zwracany jest komunikat o błędzie.

Algorytm funkcji przypisującej pojazdy jest przedstawiony na rysunku 3.23:

```
Krok 1. Dla każdej trasy
      Krok 1.1. Czy jest pojazd o nośności i objętości >= sumaryczne
                zapotrzebowanie (objętościowe i wagowe)
      Krok 1.1.1. Jeśli tak
                Przypisz pojazd do trasy
      Krok 1.1.2. Jeśli nie
           Krok 1.1.2.1. Dla każdego klienta na trasie poczynając od
                   ostatniego
                Krok 1.1.2.1.1. Usuwaj klientów z trasy, aż trasa
                     spełni warunki z punktu 1.1.1
                Krok 1.1.2.1.2. Jeśli trasa nie spełnia warunków
                     Zakończ działanie algorytmu z komunikatem o
                     błędzie.
                Krok 1.1.2.1.3. Dla każdego usuniętego klienta
                Krok 1.1.2.1.4. Znajdź najlepsze miejsce dla
                     klienta na trasach, za wyjątkiem bieżącej trasy.
           Krok 1.1.2.2. Czy dla każdego klienta znaleziono najlepsze
                        miejsce na trasach?
                Krok 1.1.2.2.1. Jeśli nie to zakończ działanie
                     algorytmu z komunikatem o błędzie.
Krok 2. Zapamiętaj rozwiązanie
```

Rys. 3.23. Algorytm funkcji przypisującej pojazdy (źródło: opracowanie własne)

Schemat blokowy funkcji przypisującej pojazdy jest przedstawiony na rysunku 3.24.



Rys. 3.24. Funkcja przypisująca pojazdy do tras (źródło: opracowanie własne)

3.5. Parametry ogólne algorytmu symulowanego wyżarzania

Symulowane wyżarzanie wykorzystuje szereg parametrów, które wpływają na jakość rozwiązania oraz szybkość pracy algorytmu. Wyróżnia się dwie grupy parametrów:

- parametry ogólne dla algorytmu symulowanego wyżarzania,

- parametry specyficzne dla rozwiązywanego problemu.

Parametrami specyficznymi dla problemu są:

- konfiguracja, czyli w przypadku problemów dostaw jest to zbiór klientów, pojazdów, magazynów centralnych,

- sąsiedztwo konfiguracji, czyli zbiór punktów osiągalny za pomocą jednego kroku funkcji przejścia,

- tzw. funkcja kosztu, czyli funkcja, dla której szukana jest minimalna wartość. Jest to funkcja, przy pomocy której rozwiązanie jest optymalizowane,

- funkcja generowania rozwiązania początkowego,

- tzw. funkcja przejścia, czyli funkcja pozwalająca na przejście z rozwiązania bieżącego do innego akceptowalnego rozwiązania.

Parametry ogólne algorytmu symulowanego wyżarzania to:

- temperatura początkowa T_0 lub funkcja wyliczania temperatury początkowej (krok 2 na rysunku 3.7),

- funkcja redukcji temperatury (krok 3.3 na rysunku 3.7),

- długość epoki, czyli liczba kroków dla jednej temperatury (krok 3.2 na rysunku 3.7),

- warunki zatrzymania algorytmu (krok 3 na rysunku 3.7).

W kolejnych podrozdziałach 3.5.1 do 3.5.4 omówione są poszczególne parametry.

3.5.1. Temperatura początkowa

Temperatura początkowa dobierana jest w taki sposób, aby w początkowych cyklach pracy algorytmu prawdopodobieństwo akceptacji gorszego rozwiązania było stosunkowo duże. W literaturze najczęściej temperatura początkowa jest stała lub jest ona wyliczana w taki sposób, aby gorsza konfiguracja została przyjęta ze stałym prawdopodobieństwem.

Na podstawie własnych doświadczeń autor niniejszej pracy zdecydował się na stałą temperaturę początkową, która wynosi 100 (Czech, Czarnas, 2002). Testy z różnymi sposobami obliczeń temperatury początkowej nie poprawiały generowanych rozwiązań, natomiast czas wykonywania algorytmu był dłuższy.

3.5.2. Funkcja redukcji temperatury

Najczęściej redukcja temperatury odbywa się wg następującego modelu (Solomon, 1987):

$$T_i = rT_{i-1}. \tag{3.19}$$

Temperatura bieżącego kroku zależy zatem od temperatury kroku wcześniejszego. Natomiast *r* jest stałą, która jest bliska, ale mniejsza od 1 (Laarhoven, Aarts, 1987).

Autorzy poszczególnych implementacji najczęściej zalecają wartości stałej *r* w przedziale od 0,85 do 0,95.

Osman (Osman, 1993) proponuje zwiększenie temperatury w sytuacji, gdy rozwiązanie nie zostało poprawione przez pewną liczbę kroków. Autor niniejszej pracy zdecydował się na podejście zaproponowane przez Osmana. Temperatura jest obniżana wg metody Laarhovena i Aartsa z parametrem r = 0.95, natomiast gdy przez liczbę kroków równą bądź większą niż 20% całkowitych kroków iteracji rozwiązanie nie jest poprawione, to temperatura jest zwiększana o wartość 10%.

Autor niniejszej rozprawy początkowo zastosował metodę Solomona opisaną wzorem 3.19 do redukcji temperatury. Podczas eksperymentów zdarzało się jednak, że algorytm utknął w minimum lokalnym. W tym celu zdecydowano na implementację metody Osmana.

3.5.3. Długość epoki

Długość epoki najczęściej dobierana jest w zależności od wielkości sąsiedztwa. Niemniej jednak w niniejszej pracy funkcja przejścia jest bardzo złożona i pozwala na głębokie zmiany w konfiguracji. Jest możliwe w jednym kroku przejście do innego akceptowanego rozwiązania, a decyzje podejmowane są częściowo w sposób losowy. Przy takiej budowie sąsiedztwo jest bardzo duże i wyliczenie liczby rozwiązań jest bardzo trudne.

Generalnie im dłuższa epoka, tym rozwiązanie końcowe jest lepsze, ale obserwując budowę algorytmu można zauważyć, że zamiast zwiększać długość epoki można wolniej zmniejszać temperaturę.

Na podstawie prób i doświadczeń autor niniejszej pracy zdecydował się na stałą długość epoki, która wynosi 100.

3.5.4. Warunki zatrzymania

Kryterium zatrzymania algorytmu w literaturze jest określane na wiele sposobów. Powszechnie stosowana metoda to zatrzymanie algorytmu, jeśli od pewnej liczby kroków nie ulepszono rozwiązania. Inny warunek mówi, że zatrzymanie algorytmu następuje, gdy liczba zaakceptowanych ruchów spadnie poniżej pewnego stałego procenta.

Autor w niniejszej pracy zastosował najprostszą metodę zatrzymania algorytmu, która polega na określeniu z góry liczby kroków. W niniejszej pracy wynosi ona 100. W rozprawie testowano także większe wartości, ale nie przynosiło to praktycznie żadnych efektów, natomiast całkowity czas pracy algorytmu był znacząco dłuższy.

3.6. Funkcja kosztu

Funkcja kosztu zastosowana w niniejszej pracy ma postać dany wzorem 3.20:

$$F = aR + bD + cK \tag{3.20}$$

gdzie a, b oraz c są stałymi, R oznacza liczbę tras, D to całkowita długość wszystkich tras, a K to współczynnik kosztu pojazdu. Postać ze wzoru 3.20 wynika wprost z modelu opisanego wzorem 2.8.

W niniejszej pracy parametr a = 10000, b = 100 oraz c = 1. Takie dobranie parametrów oznacza, że w pierwszej kolejności minimalizowana jest liczba tras, a dopiero później ich długość. W ostatniej kolejności minimalizowane są współczynniki kosztów użytych pojazdów. Parametry b i c w przedstawionej funkcji kosztu nie są pomijane, są natomiast brane pod uwagę dopiero przy porównywaniu dwóch rozwiązań z taką samą liczbą tras. Sposób dobrania parametrów wynika tylko i wyłącznie z kolejności optymalizowanych zmiennych. Jeśli priorytetem jest minimalizacja liczby tras, parametr a musi być dużo większy od parametru b oraz c. Wartości tych parametrów mają drugorzędne znaczenie w stosunku do różnicy pomiędzy nimi.

3.7. Weryfikacja algorytmu

Do celów testowych należało zdefiniować zestawy uproszczonych problemów z niewielką liczbą ograniczeń i kilkoma kryteriami optymalizacji. W oparciu o te problemy przeprowadzone zostały eksperymenty, na bazie których został opracowany algorytm.

Badanie efektywności zbudowanego w ten sposób algorytmu przeprowadzone zostało przy użyciu danych rzeczywistych, zestawów testowych Cordeau oraz wybranych testów Solomona. Dane rzeczywiste pochodzą z systemu ewidencji i śledzenia transportu w badanej firmie.

Pierwszy etap weryfikacji polegał na porównaniu rozwiązań wygenerowanych przez opracowany algorytm z istniejącymi najlepszymi wynikami światowymi testów opracowanych przez M. Solomona dotyczącymi problemu dostaw z oknami czasowymi, a następnie algorytm szukał rozwiązań optymalnych testów Cordeau, które dotyczą problemu dostaw z wieloma magazynami. W ostatnim etapie wygenerowane wysyłki zostały porównane z planami rzeczywistymi.

3.8. Problem testowy oraz otrzymane wyniki

Autor niniejszej rozprawy do testowania opisanego algorytmu użył trzech rodzajów testów: zestawów Solomona, Cordeau oraz zbiorów rzeczywistych.

Szczegółowy opis zbiorów testowych znajduje się w rozdziale 3.8.1. W kolejnych podrozdziałach 3.8.2 – 3.8.4 przedstawione są uzyskane wyniki każdego z opisywanych testów.

3.8.1. Zbiory testowe zastosowane do testowania problemów dostaw

Autor niniejszej rozprawy do testowania programu realizującego opisywany algorytm użył trzech rodzajów testów:

- 56 zestawów testowych opracowanych przez M. Solomona (Solomon, 1987) służących do testowania algorytmów rozwiązujących problem VRPTW,

- 20 zestawów testowych opracowanych przez Cordeau (Cordeau i in., 1997) służących do testowania algorytmów rozwiązujących problem MDVRPTW

(wielomagazynowy problem dostaw z oknami czasowymi, ang. Multi Depot Vehicle Routing Problem with Time Windows),

- 18 zestawów testów rzeczywistych pochodzących z badanego przedsiębiorstwa – jest to rozszerzony problem RDPTW o dodatkowe wymiary pojazdów i zamawianych towarów, a także uwzględnianie współczynnika kosztów przypisywanych do tras pojazdów.

Tego typu usystematyzowanie testów powoduje, że problem badany w niniejszej rozprawie jest najpierw uproszczony do VRPTW. Jeśli wyniki znalezione przez niniejszy algorytm są zadowalające, algorytm testowany jest na zestawach z problemu MDVRPTW, który uwzględnia wiele magazynów. Ostatnią grupę testów stanowią testy rzeczywiste analizujące wszystkie założenia oraz ograniczenia, czyli istnienie wielu magazynów oraz niejednorodnej floty pojazdów. Ponadto podczas testowania zestawów rzeczywistych brane jest pod uwagę trzecie kryterium minimalizacji funkcji kosztu oraz dodatkowe wymiary pojazdów i towarów.

Dla grupy testów Solomona należało przyjąć następujące założenia:

- wszystkie wysyłki są dokonywane z jednego centralnego magazynu,
- wszystkie samochody posiadają tą samą ładowność (flota jest homogeniczna),
- nie brane są pod uwagę wymiary pojazdów ani towarów.

Testy Solomona składają się z 56 instancji. Każda z nich zawiera jeden centralny magazyn oraz 100 klientów. Położenia magazynu oraz klientów zdefiniowane są w układzie kartezjańskim. Przyjmuje się także, że czasy podróży t_{ij} od klienta i do j są równe odległościom d_{ij} pomiędzy klientami.

Problemy testowe Solomona podzielone są na sześć grup. Testy C1 oraz C2 składają się z klientów, którzy rozmieszczeni są w sposób uporządkowany w sensie rozmieszczenia geograficznego. W zbiorach RC1 oraz RC2 część klientów pogrupowanych jest w klastry, a część jest losowa, natomiast w zbiorach R1 oraz R2 wszyscy klienci rozmieszczeni są losowo w sposób nieuporządkowany wg rozkładu jednostajnego.

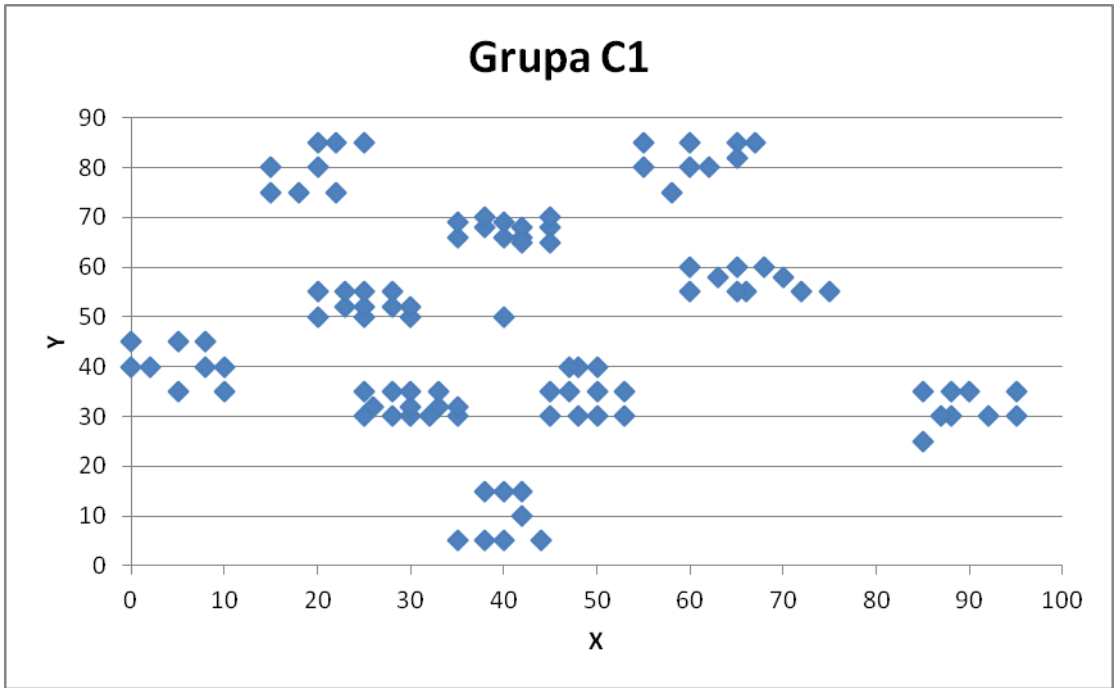
Klienci z grup C1, R1 oraz RC1 posiadają węższe okna czasowe, co powoduje, że liczba znalezionych tras jest większa. Im węższe okna czasowe klientów tym mniej ich się „zmieści” na trasie, więc tras w rozwiązaniu będzie więcej. Zbiory C2, R2 oraz RC2 posiadają klientów z szerszymi oknami czasowymi, co pozwala na umieszczenie większej liczby klientów na każdej trasie.

Rysunki 3.25 do 3.30 przedstawiają rozmieszczenie klientów z grup testowych Solomona. Osie X oraz Y przedstawiają współrzędne położenia geograficznego klientów. Klienci zaznaczeni są za pomocą niebieskich rombów.

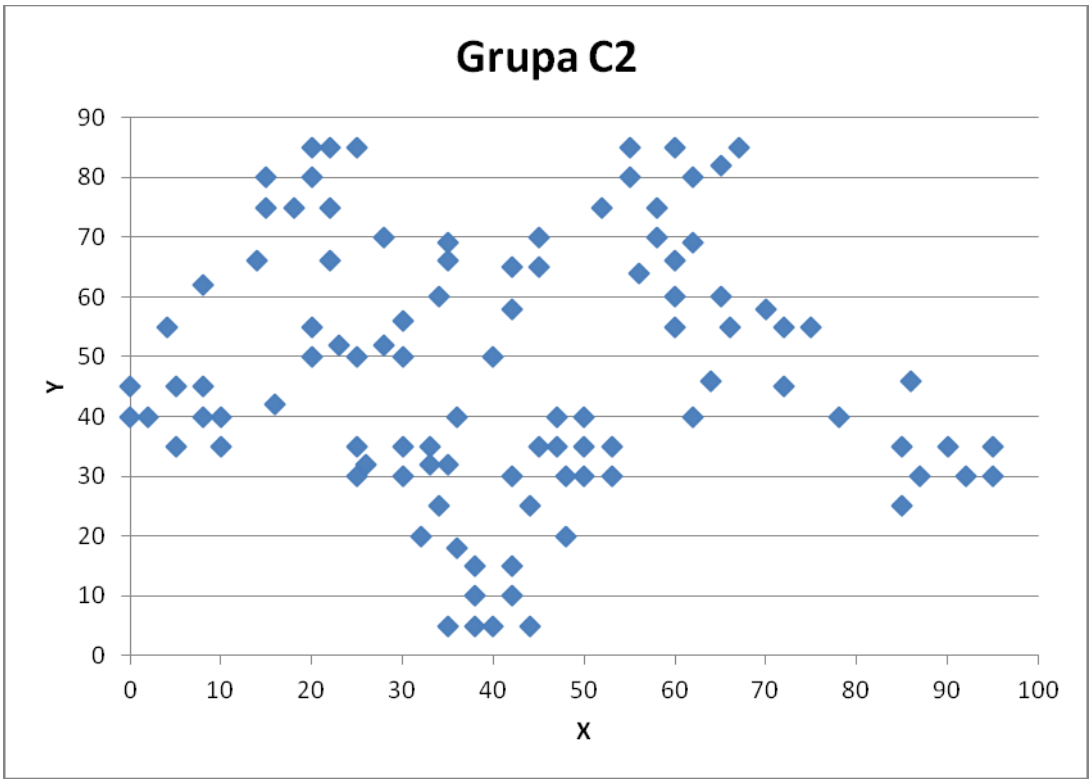
Jak wynika z rysunków 3.25 – 3.30, klientów z grup C1 oraz C2 łatwo pogrupować w tzw. klastry. W testach R1 oraz R2 klienci rozmieszczeni są w sposób całkowicie losowy, a z rysunków dla grup RC1 oraz RC2 wynika, że część klientów tworzy klaster, część jest położona losowo.

Pełne zestawy wszystkich 56 testów Solomona zostały umieszczone w załączniku 1. Zestawy testowe (załącznik 1) składają się z wierszy, w których znajdują sie klienci (numery wierszy od 1 do 100) oraz magazyn centralny (numer wiersza 0). Kolumny przedstawiają odpowiednio:

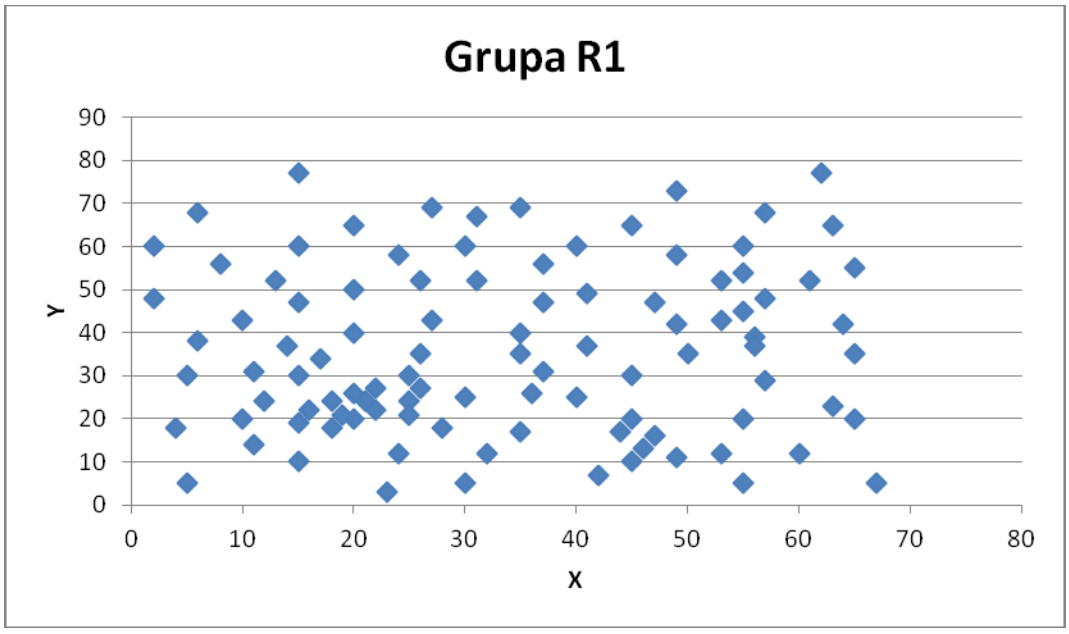
- nr porządkowy klienta lub magazynu,
- współrzędną X położenia geograficznego,
- współrzędną Y położenia geograficznego,
- zapotrzebowanie klienta na dany towar,
- początek okna czasowego,
- koniec okna czasowego,
- czas obsługi.



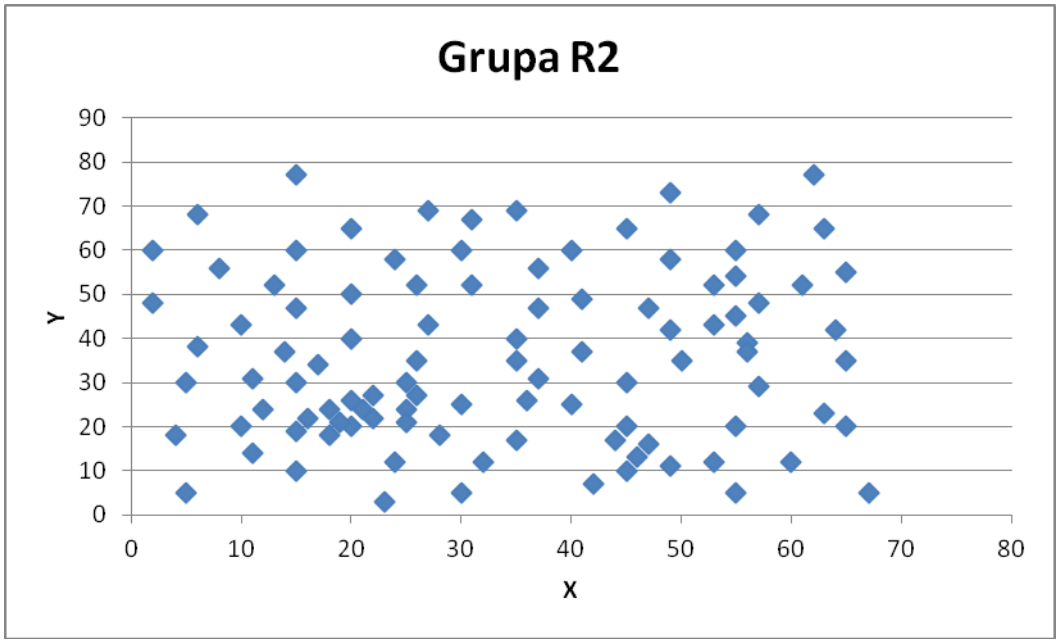
Rys. 3.25. Położenie klientów grupy C1(źródło: opracowanie własne)



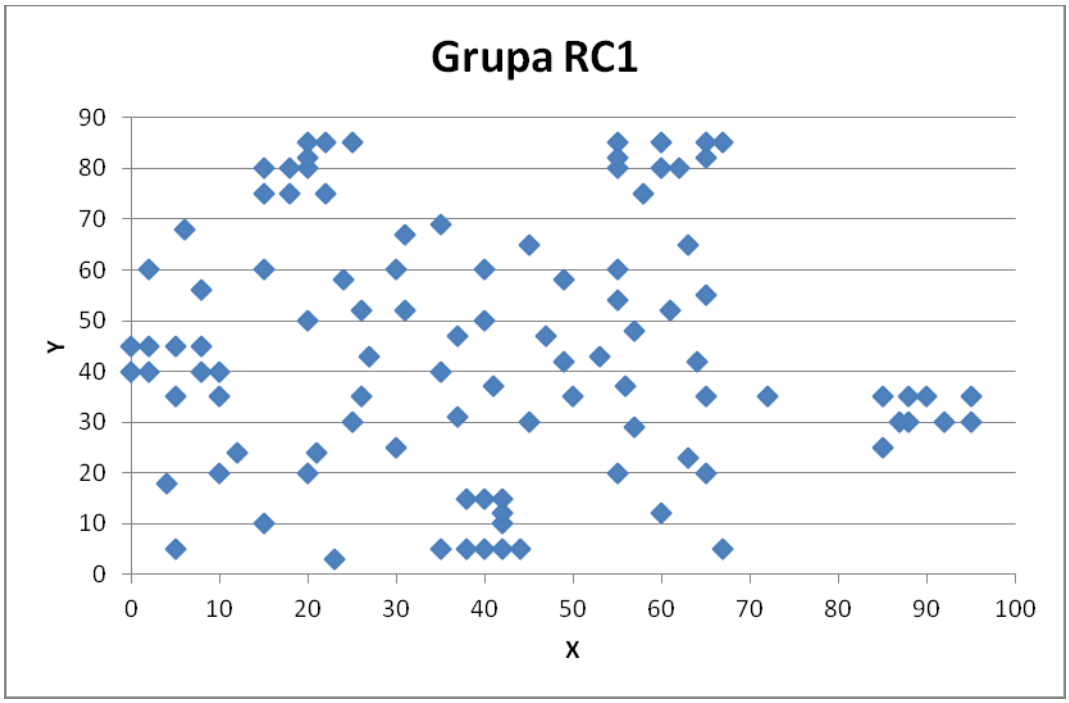
Rys. 3.26. Położenie klientów grupy C2(źródło: opracowanie własne)



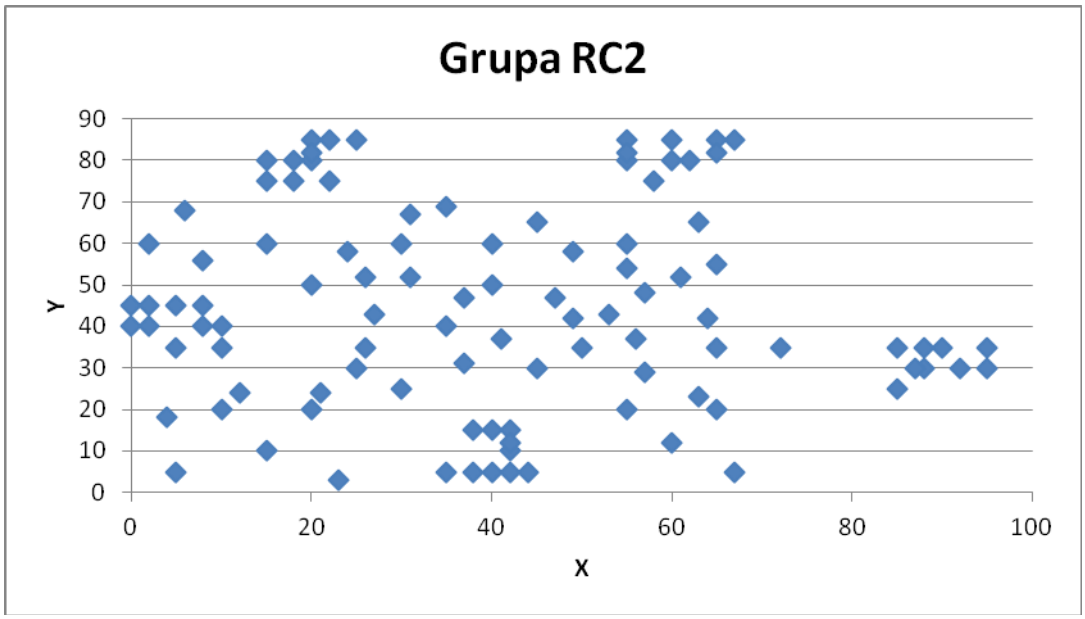
Rys. 3.27. Położenie klientów grupy R1(źródło: opracowanie własne)



Rys. 3.28. Położenie klientów grupy R2(źródło: opracowanie własne)



Rys. 3.29. Położenie klientów grupy RC1(źródło: opracowanie własne)



Rys. 3.30. Położenie klientów grupy RC2(źródło: opracowanie własne)

Uzyskane wyniki testów Solomona zostały porównane i zestawione z najlepszymi aktualnie znanymi wynikami światowymi. Lista tych wyników jest dostępna w Internecie na stronie „Transportation Optimization Portal" znajdującej się pod adresem http://www.top.sintef.no/VRP/bknown.html oraz na stronie Mariusa Salomona „VRPTW Benchmark Problems" (http://w.cba.neu.edu/~msolomon/problems.htm).

Wyniki uzyskane w niniejszej rozprawie przedstawione są w rozdziale 3.8.2, natomiast szczegółowe trasy zawiera załącznik 4.

W przypadku kolejnej grupy testów, jakimi są testy Cordeau badany problem został uproszczony do wielomagazynowego problemu dostaw. Problem ten ze względu na istnienie wielu magazynów centralnych jest dużo trudniejszy niż problem dostaw z oknami czasowymi. Testy te charakteryzują się losowym rozmieszczeniem klientów oraz magazynów.

Zestawy testów Cordeau zostały umieszczone w załączniku 2. Zestawy testowe składają się z wierszy, w których znajdują sie klienci oraz magazyny centralne. W przypadku tych testów magazyny centralne umieszczone są na końcu listy. Kolumny przedstawiają odpowiednio:

- nr porządkowy klienta lub magazynu,
- współrzędną X położenia geograficznego,
- współrzędną Y położenia geograficznego,

- zapotrzebowanie klienta na dany towar,
- początek okna czasowego,
- koniec okna czasowego,
- czas obsługi.

Uzyskane wyniki testów Cordeau zostały przedstawione w rozdziale 3.8.3. Tam też zostały one zestawione i porównane z najlepszymi obecnie znanymi wynikami światowymi. Natomiast załącznik 5 zawiera szczegółowe trasy testów.

3.8.2. Wyniki testów Solomona

W tabelach 3.15 – 3.20 przedstawione zostały aktualne najlepsze wyniki światowe oraz uzyskane rezultaty za pomocą algorytmu opisywanego w niniejszej rozprawie. W pierwszej kolumnie tabel znajduje się nazwa każdego testu, w drugiej kolumnie przedstawione są najlepsze wyniki światowe, a w trzeciej kolumnie wyniki testów niniejszego algorytmu.

Kolumna „Wyniki światowe" została podzielona na trzy dodatkowe kolumny:

- w pierwszej z nich są inicjały autora lub autorów najlepszego światowego wyniku. Lista tych autorów oraz prace, gdzie te wyniki zostały opublikowane, znajdują się w tabeli 3.21.

- W drugiej i trzeciej kolumnie znajdują się najlepsze rezultaty danych testów z podziałem na liczbę tras oraz całkowitą długość tras.

Kolumna „Uzyskane wyniki" z wynikami uzyskanymi zapomocą opisywanego algorytmu także została podzielona na dwie kolumny, w których znajdują się rezultaty testów z podziałem na liczbę tras oraz całkowitą długość tras.

Nazwa testu	Wyniki światowe			Uzyskane wyniki	
	Autor	Liczba tras	Dług. Tras	Liczba tras	Dług. Tras
C101	RT	10	828,94	10	828,94
C102	RT	10	828,94	10	828,94
C103	RT	10	828,06	10	828,06
C104	RT	10	824,78	10	824,78

C105	RT	10	828,94	10	828,94
C106	RT	10	828,94	10	828,94
C107	RT	10	828,94	10	828,94
C108	RT	10	828,94	10	828,94
C109	RT	10	828,94	10	828,94

Tabela 3.15. Wyniki testów dla grupy C10x (źródło: opracowanie własne)

Nazwa testu	Wyniki światowe			Uzyskane wyniki	
	Autor	Liczba tras	Dług. Tras	Liczba tras	Dług. Tras
C201	RT	3	591,56	3	591,56
C202	RT	3	591,56	3	591,56
C203	RT	3	591,17	3	591,17
C204	RT	3	590,60	3	590,60
C205	RT	3	588,88	3	588,88
C206	RT	3	588,49	3	588,49
C207	RT	3	588,29	3	588,29
C208	RT	3	588,32	3	588,32

Tabela 3.16. Wyniki testów dla grupy C20x (źródło: opracowanie własne)

Nazwa testu	Wyniki światowe			Uzyskane wyniki	
	Autor	Liczba tras	Dług. tras	Liczba tras	Dług. tras
R101	H	19	1645,79	19	1650,80
R102	RT	17	1486,12	17	1486,12
R103	LLH	13	1292,68	13	1292,68
R104	M	9	1007,24	9	1007,31
R105	RT	14	1377,11	14	1377,11
R106	M	12	1251,98	12	1257,96
R107	S97	10	1104.66	10	1104.66
R108	BBB	9	960.88	9	960,88
R109	HG	11	1194.73	11	1197,42
R110	M	10	1118,59	10	1118,84

R111	RGP	10	1096,72	10	1096,72
R112	GTA	9	982,14	9	982,14

Tabela 3.17. Wyniki testów dla grupy R1x (źródło: opracowanie własne)

Nazwa testu	Wyniki światowe			Uzyskane wyniki	
	Autor	Liczba tras	Dług. Tras	Liczba tras	Dług. Tras
R201	HG	4	1252,37	4	1252,37
R202	RGP	3	1191,70	3	1191,70
R203	M	3	939,54	3	939,50
R204	BVH	2	825,52	2	825,52
R205	RGP	3	994,42	3	994.42
R206	SSSD	3	906,14	3	906,14
R207	BVH	2	890,61	2	905,28
R208	M	2	726,75	2	726.82
R209	H	3	909,16	3	909.16
R210	M	3	939,34	3	939,37
R211	BVH	2	892,71	2	885,71

Tabela 3.18. Wyniki testów dla grupy R2x (źródło: opracowanie własne)

Nazwa testu	Wyniki światowe			Uzyskane wyniki	
	Autor	Liczba tras	Dług. tras	Liczba tras	Dług. tras
RC101	TBGGP	14	1696,94	14	1696,94
RC102	TBGGP	12	1554,75	12	1554,75
RC103	S98	11	1261,67	11	1261,67
RC104	CLM	10	1135,48	10	1135,48
RC105	BBB	13	1629,44	13	1629,44
RC106	BBB	11	1424,73	11	1424,73
RC107	S97	11	1230,48	11	1230,48
RC108	TBGGP	10	1139,82	10	1139,82

Tabela 3.19. Wyniki testów dla grupy RC10x (źródło: opracowanie własne)

Nazwa testu	Wyniki światowe			Uzyskane wyniki	
	Autor	Liczba tras	Dług. Tras	Liczba tras	Dług. tras
RC201	M	4	1406,91	4	1406,94
RC202	CC	3	1365,65	3	1367,00
RC203	CC	3	1049,62	3	1049,62
RC204	M	3	798,41	3	798.46
RC205	M	4	1297,19	4	1297,65
RC206	H	3	1146,32	3	1146,32
RC207	BVH	3	1061,14	3	1061,14
RC208	IKMUY	3	828,14	3	828,71

Tabela 3.20. Wyniki testów dla grupy RC20x (źródło: opracowanie własne)

Tabela 3.21 przedstawia prace, w których te wyniki zostały zaprezentowane oraz ich autorów. Symbole w pierwszej kolumnie pochodzą od pierwszych liter nazwisk autorów danej pracy.

BBB	Berger J., Barkaoui M., Bräysy O.: A Parallel Hybrid Genetic Algorithm for the Vehicle Routing Problem with Time Windows. Working paper, 2001, Defense Research Establishment Valcartier, Canada.
BVH	Bent R., Van Hentenryck P.: A Two-Stage Hybrid Local Search for the Vehicle Routing Problem with Time Windows. Technical Report CS-01-06, 2001, Department of Computer Science, Brown University.
CC	Czech Z. J., Czarnas P.A.: Parallel Simulated Annealing for the Vehicle Routing Problem with Time Windows. Proc. 10th Euromicro Workshop on Parallel, 2002, Distributed and Network-based Processing, Canary Islands, Spain, s. 376-383.
CLM	Cordeau J. F., Laporte G., Mercier A.: A Unified Tabu Search Heuristic for Vehicle Routing Problems with Time Windows. Working Paper CRT-00-03, 2000, Centre for Research on Transportation, Montreal, Canada.
GTA	Gambardella L. M., Taillard E., Agazzi G.: MACS-VRPTW: A Multiple Ant Colony System for Vehicle Routing Problems with Time Windows. New Ideas in Optimization, 1999, McGraw-Hill, London, s. 63-76.
H	J. Homberger: Verteilt-parallele Metaheuristiken zur Tourenplanung. Gaber, Wiesbaden, 2000.

HG	Homberger J., Gehring H.: Two Evolutionary Metaheuristics for the Vehicle Routing Problem with Time Windows. INFOR, 1999, VOL. 37, s297-318.
IKMUY	Ibaraki T., Kubo M., Masuda T., Uno T., Yagiura M.: Effective Local Search Algorithms for the Vehicle Routing Problem with General Time Windows. Working Paper, 2001, Department of Applied Mathematics and Physics, Kyoto University, Japan.
LLH	Li H., Lim A., Huang J.: Local Search with Annealing-like Restarts to Solve the VRPTW. Working Paper, 2001, Department of Computer Science, National University of Singapore.
M	Mester D.: An Evolutionary Strategies Algorithm for Large Scale Vehicle Routing Problem with Capacitate and Time Windows Restrictions. Working Paper, 2002, Institute of Evolution, University of Haifa, Israel.
RT	Rochat Y., Taillard E. D.: Probabilistic Diversification and Intensification in Local Search for Vehicle Routing. Journal of Heuristics, 1995 1, s. 147-167.
RGP	Rousseau L. M., Gendreau M., Pesant G.: Using Constraint-Based Operators to Solve the Vehicle Routing Problem with Time Windows. Journal of Heuristics, forthcoming.
SSSD	Schrimpf G., Schneider J., Stamm-Wilbrandt H., Dueck G.: Record Breaking Optimization Results Using the Ruin and Recreate Principle. Journal of Computational Physics 159, 2000, s. 139-171.
S97	Shaw P.: A New Local Search Algorithm Providing High Quality Solutions to Vehicle Routing Problems. Working Paper, 1997, University of Strathclyde, Glasgow, Scotland.
S98	Shaw P.: Using Constraint Programming and Local Search Methods to Solve Vehicle Routing Problems. Principles and Practice of Constraint Programming - CP98, 1998, Springer-Verlag, New York, s. 417-431.
TBGGP	Taillard E., Badeau P., Gendreau M., Geurtin F., Potvin J. Y.: A Tabu Search Heuristic for the Vehicle Routing Problem with Time Windows. Transportation Science, 1997, 31, s. 170-186.

Tabela 3.21. Lista prac z najlepszymi uzyskanymi wynikami problemu dostawy (źródło: opracowanie własne na podstawie Transportation Optimization Portal http://www.sintef.no/projectweb/top)

Dla 56 przeprowadzonych testów Solomona w dwóch przypadkach algorytm zaproponowany przez autora niniejszej pracy poprawił wyniki światowe. Są to testy R203 oraz R211. W 42 przypadkach uzyskane wyniki równe są najlepszym wynikom światowym. W żadnym przypadku uzyskane wyniki nie różniły się od najlepszych wyników światowych liczbą tras, a pozostałe 12 testów różni się tylko i wyłącznie

dystansem, przy tym największa różnica w stosunku do wyników światowych wynosi 0.48%.

Poniżej zestawione są trasy poprawiające obecne najlepsze wyniki światowe. Poszczególne liczby wchodzące w skład tras oznaczają numer klienta z danego zbioru testowego Solomona. Szczegółowe wyniki dla najlepszych uzyskanych testów M. Solomona zostały przedstawione w załączniku 4.

R203:

Trasa 1: 0-27-52-18-45-46-36-64-11-62-88-31-30-76-3-79-78-9-35-34-29-68-12-26-13-58-0; **Trasa 2:** 0-94-95-97-92-98-37-42-57-15-43-14-44-38-86-16-85-59-99-96-6-84-83-8-48-47-49-19-10-63-90-32-66-20-70-7-82-17-61-91-100-93-5-60-89-0; **Trasa3:** 0-50-33-81-65-71-51-1-69-39-67-23-72-73-21-40-53-87-2-41-22-75-56-74-4-55-25-54-24-80-77-28-0

R211:

Trasa 1: 0-95-59-92-98-42-15-2-21-73-72-39-67-23-75-22-41-57-87-94-6-53-40-12-76-29-79-33-81-9-71-65-66-20-51-35-34-78-3-77-68-80-24-54-55-25-4-56-74-58-0; **Trasa 2:** 0-28-27-52-69-31-30-63-64-11-19-62-88-7-82-18-83-84-5-99-85-61-16-44-14-38-86-17-45-8-46-48-47-36-49-90-32-10-70-1-50-26-13-97-96-37-43-100-91-93-60-89-0

3.8.3. Wyniki testów Cordeau

W tabeli 3.22 przedstawione zostały aktualne najlepsze wyniki światowe oraz uzyskane rezultaty za pomocą algorytmu opisywanego w niniejszej rozprawie. W pierwszej kolumnie tabel znajduje się nazwa każdego testu, druga kolumna przedstawia liczbę odbiorców, w kolumnie trzeciej znajduje się liczba centralnych magazynów, a w kolumnie 4 liczba pojazdów przypadających na jeden magazyn. W kolejnej kolumnie znajdują się odpowiednio najlepsze obecnie znane wyniki światowe oraz wyniki uzyskane za pomocą niniejszego algorytmu. W ostatniej kolumnie przedstawione jest odchylenie procentowe uzyskanego wyniku w stosunku do najlepszego wyniku światowego.

Nazwa testu	Liczba klientów	Liczba magazynów	Liczba pojazdów na magazyn	Wyniki światowe	Uzyskane wyniki	Odchylenie %
PR01	48	4	2	1074,12	1083,98	0,92
PR02	96	4	3	1762,36	1762,36	0,00
PR03	144	4	4	2385,94	2385,94	0,00
PR04	192	4	5	2840,59	2852,99	0,44
PR05	240	4	6	3018,38	3018,38	0,00
PR06	288	4	7	3670,13	3670,13	0,00
PR07	72	6	2	1418,22	1418,22	0,00
PR08	144	6	3	2099,49	2102,61	0,15
PR09	216	6	4	2737,82	2752,61	0,54
PR10	288	6	5	3507,26	3507,26	0,00
PR11	48	4	1	957,48	961,59	0,43
PR12	96	4	2	1478,51	1486,26	0,52
PR13	144	4	3	2011,24	2014,06	0,14
PR14	192	4	4	2202,08	2228,64	1,21
PR15	240	4	5	2494,57	2525,20	1,23
PR16	288	4	6	2901,02	2940,73	1,37
PR17	72	6	1	1201,73	1208,25	0,54
PR18	144	6	2	1792,61	1803,22	0,59
PR19	216	6	3	2285,10	2288,38	0,14
PR20	288	6	4	3079,16	3091,54	0,40

Tabela 3.22. Wyniki testów MDVRPTW (źródło: opracowanie własne)

W 6 przypadkach wyniki uzyskane przez niniejszy algorytm pokrywają się z najlepszymi wynikami światowymi. W pozostałych przypadkach wyniki uzyskane przez niniejszy algorytm różnią się długością tras, niemniej jednak różnica w żadnym przypadku nie przekracza 1,4%.

Szczegółowe rozwiązania najlepszych testów zostały przedstawione w załączniku 5.

3.8.4. Wyniki testów opartych na danych rzeczywistych

W tabeli 3.23 przedstawione zostały aktualne wyniki testów na danych rzeczywistych pochodzących z badanej firmy. Celowo do testów zostały wybrane tylko wysyłki do minimum 80 klientów.

W pierwszej kolumnie tabeli znajduje się całkowita liczba klientów do obsłużenia w ramach danej dostawy. W drugiej liczba magazynów centralnych, z których następują wysyłki towarów. W kolejnych kolumnach wyniki osiągnięte przez nadany algorytm zostały porównane z wynikami logistyków firmy. Jak wynika z tabeli 3.23, wyniki osiągnięte przez badany algorytm są zdecydowanie lepsze zarówno pod względem czasu trwania obliczeń, jak i pod względem znalezionych rozwiązań.

		Służby zakładowe			System NAV		
Liczba klientów	Liczba mag. Centralnych	Czas obliczeń [h]	Liczba tras	Dług. Tras	Czas obliczeń [h]	Liczba tras	Dług. tras
80	1	4,5	15	1980	0,2	10	1650
80	1	4	16	1870	0,2	11	1695
85	2	5	15	1993	0,2	11	1630
86	2	4,5	14	1987	0,2	10	1599
95	1	3,5	16	1887	0,2	12	1545
100	1	3,5	15	1901	0,2	11	1435
100	2	5	12	1805	0,5	8	1608
110	1	5,5	18	2030	0,3	12	1162
110	1	4,5	19	2045	0,5	14	1745
120	2	4,5	19	2087	0,3	13	1698
130	2	4,5	22	3487	0,5	16	2876
133	2	7	28	3700	0,5	17	2929
140	3	7,5	22	2876	1	15	2018
140	2	6,5	23	3245	1,2	16	2459
143	3	5,5	25	3467	1	18	2654
150	2	8.5	32	4150	1	19	3564

150	3	8.5	33	4080	1	22	3218
150	3	9	34	4093	1,2	23	3356

Tabela 3.23. Lista wyników testów rzeczywistych – porównanie z wynikami uzyskanymi przez służby zakładowe (źródło: opracowanie własne)

Wyniki testów rzeczywistych zostały przedstawione (tabela 3.23) w celu pokazania różnicy pomiędzy czasem obliczeń oraz rezultatami uzyskanymi przez opisywany w niniejszej pracy algorytm w porównaniu z pracą logistyków badanej firmy.

	Jopt		Quagga		Paragon		System NAV	
Liczba klientów	Liczba tras	Dług. Tras	Liczba tras	Dług. Tras	Liczba tras	Dług. Tras	Liczba tras	Dług. tras
80	10	1698	10	1694	10	1791	10	1650
80	11	1787	11	1813	11	1780	11	1695
85	11	1841	11	1801	11	1680	11	1630
86	10	1712	10	1745	10	1599	10	1599
95	12	1548	12	1598	12	1598	12	1545
100	11	1487	11	1491	12	1454	11	1435
100	8	1701	8	1701	8	1654	8	1608
110	12	1193	13	1165	12	1236	12	1162
110	16	1902	15	1801	14	1956	14	1745
120	14	1754	13	1698	14	1803	13	1698
130	16	3102	16	2901	17	3001	16	2876
133	18	3156	18	3044	17	2990	17	2929
140	15	2134	15	2018	15	2412	15	2018
140	19	2569	17	2459	17	2845	16	2459
143	18	2759	18	2715	19	2701	18	2654
150	20	3700	19	3564	19	3601	19	3564
150	25	3298	23	3291	23	3291	22	3218
150	24	3415	24	3431	23	3423	23	3356

Tabela 3.24. Lista wyników testów rzeczywistych – porównanie z wynikami uzyskanymi przez inne systemy komputerowe (źródło: opracowanie własne)

W tabeli 3.24 porównano wyniki testów z tabeli 3.23 uzyskane przez algorytm opisywany w niniejszej pracy (kolumna System NAV) z innymi aplikacjami komercyjnym rozwiązującymi problemy dostaw: JOpt, Paragon oraz Quagga. Czasy obliczeń były do siebie bardzo zbliżone, jednak można zauważyć, że wyniki uzyskane za pomocą niniejszego algorytmu są znacząco lepsze od uzyskanych za pomocą porównywanych aplikacji.

Otrzymane rezultaty są zadowalające i pozwalają sądzić, że użyty algorytm będący hybrydą symulowanego wyżarzania oraz ALNS jest bardzo dobrym narzędziem do planowania wysyłek.

4. Implementacja algorytmu w systemie Microsoft Dynamics NAV 2009

Aplikacja testująca niniejszy algorytm zwana „Planowaniem wysyłek" została zaimplementowana jako rozszerzenie systemu Microsoft Dynamics NAV 2009.

Microsoft Dynamics NAV 2009 jest systemem składającym z tzw. modułów tworzących jedną całość. Moduły te pozwalają na pełną obsługę firm i są to:

- księgowość i finanse,
- zarządzanie sprzedażą, rozliczenia z klientami,
- zarządzanie relacjami z klientami i kontaktami (moduł marketingowy),
- zakupy i rozliczenia z dostawcami,
- zarządzanie zasobami ludzkimi,
- zlecenia,
- zarządanie serwisem,
- produkcja.

„Planowanie wysyłek" stworzone przez autora niniejszej pracy jest modułem NAV w pełni zintegrowanym z pozostałymi elementami systemu.

Aplikacja NAV została napisana przez firmę Microsoft w specjalnie do tego celu opracowanym języku programowania C/AL (język aplikacji C/SIDE, ang. C/SIDE Application Language) w środowisku programistycznym C/SIDE (Zintegrowane środowisko programistyczne klient-serwer, ang. Client/Server Integrated Development Environment). Tworząc nowe moduły i funkcjonalności w systemie NAV używa się języka C/AL w celu zachowania pełnej integralności systemu oraz zgodności z jego standardami.

W wersji NAV 2009 wprowadzono nowy interfejs użytkownika, tzw. RTC (klient zorientowany zadaniowo, ang. Role Tailored Client). Obecnie użytkownik ma do wyboru dwa rodzaje interfejsów, tzw. klient klasyczny (ang. Classic Client) oraz klient RTC.

W rozdziale 4 opisany jest sposób implementacji modułu „Planowanie wysyłek". Opisywane rozszerzenie systemu NAV może być wykorzystywane przez użytkowników pracujących zarówno w kliencie klasycznym, jak i tym używającym interfejsu RTC.

4.1. Implementacja algorytmu

4.1.1. Ustawienia modułu „Planowanie wysyłek”

Wszystkie funkcje nowego modułu „Planowanie wysyłek” znajdują się w menu Sprzedaż&Należności, Przetwarzanie zamówień, Transport (ang. Sales&Receivables, Order Processing, Transport).

W menu Ustawienia znajdują się formatki z ustawieniami modułu.

W pierwszej z nich, „Ustawienia transportu”, należy zdefiniować parametry wykorzystywane przez algorytm opracowany dla potrzeb niniejszej rozprawy.

Formatka składa się z dwóch zakładek: „Symulowane Wyżarzanie” (ang. Simulated Annealing) oraz „Numeracja” (ang. Numbering). W zakładce „Symulowane Wyżarzanie” znajdują się następujące pola:

- Liczba iteracji (ang. No. of Iterations) – pole definujące liczbę iteracji algorytmu symulowanego wyżarzania.

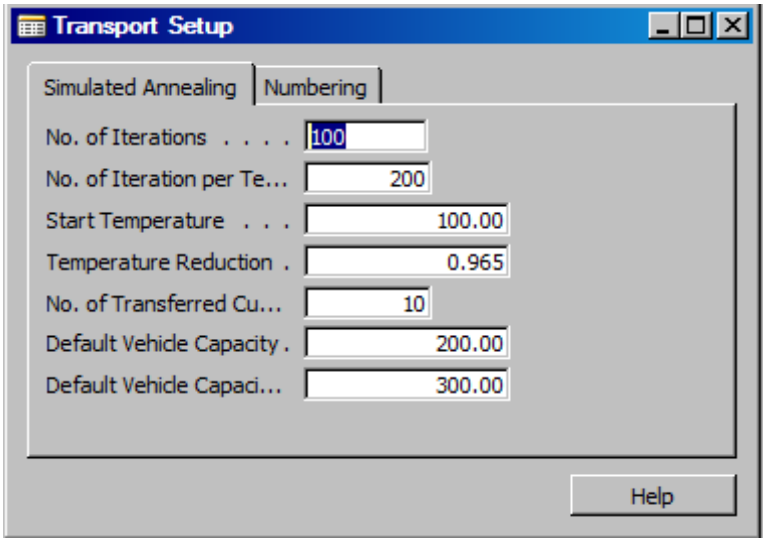
- Liczba iteracji dla temperatury (ang. No. of Iterations per Temperature) – pole określające długość epoki, czyli liczbę iteracji dla bieżącej temperatury.

- Temperatura początkowa (ang. Start Temperature) – temperatura początkowa dla algorytmu symulowanego wyżarzania.

- Liczba przenoszonych klientów (ang. No. of Transferred Customers) – parametr używany w funkcji przejścia służący do określenia maksymalną liczbę losowanych klientów w celu zmiany ich rozmieszczenia na trasach, czyli zmiany przyporządkowania ich do innych tras.

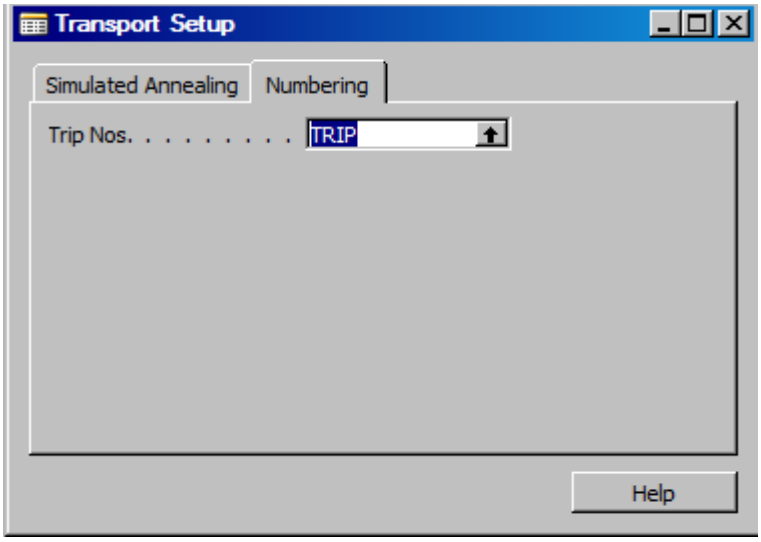
- Domyślna nośność pojazu [kg] (ang. Default Vehicle Capacity [kg]) – parametr, który jest używany w sytuacji, gdy nie ma zdefiniowanych pojazdów. Wg tego założenia wszystkie pojazdy posiadają tę samą nośność, a liczba pojazdów jest nieograniczona.

- Domyślna pojemność pojazdu [m3] (ang. Default Vehicle Capacity [m3]) – parametr, który jest używany w sytuacji, gdy nie ma zdefiniowanych pojazdów. Wg tego założenia wszystkie pojazdy posiadają tą samą pojemność, a liczba pojazdów jest nieograniczona.



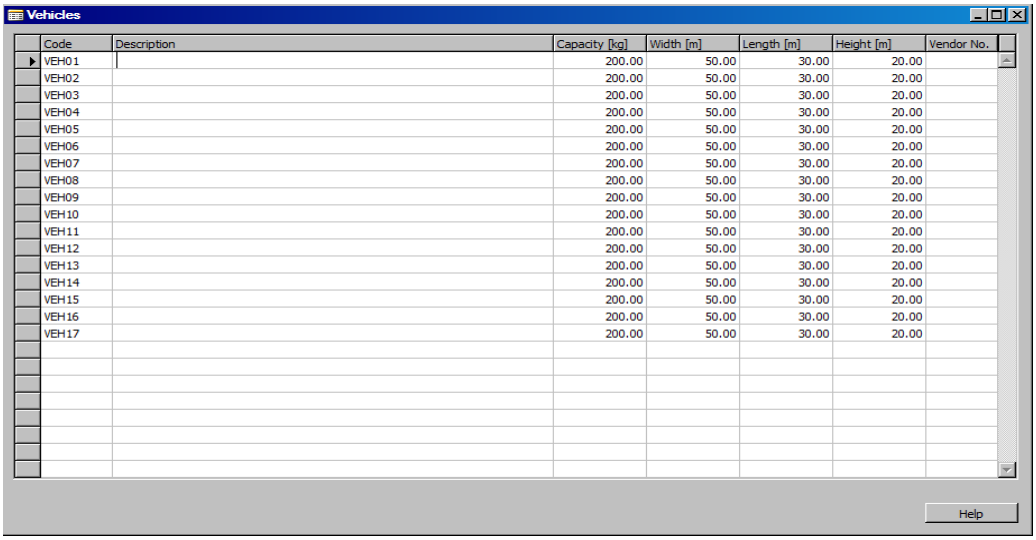
Rys. 4.1. Okno ustawień transportu – zakładka „symulowane wyżarzanie” (ang. Simulated Annealing) (źródło: opracowanie własne)

W zakładce Numeracja (ang. Numbering) znajduje się jedno pole, które określa sposób automatycznego przypisywania unikalnych numerów dla generowanych wysyłek.



Rys. 4.2. Okno ustawień transportu – zakładka „Numeracja” (ang. Numbering) (źródło: opracowanie własne)

Kolejne okno w menu Setup (ang. Setup) służy do stworzenia listy pojazdów, którymi firma dysponuje. Jest to lista zawierająca zarówno pojazdy własne, jak i pojazdy wynajmowane do obsłużenia bieżącej wysyłki.



Rys. 4.3. Okno definicji pojazdów (źródło: opracowanie własne)

Formatka zawiera następujące pola:

- Kod (ang. Code) – unikalna referencja pojazdu,

- Opis (ang. Desription) – dowolny, wprowadzony przez użytkownika opis pojazdu np. marka, typ, itp.,

- Nośność [kg] (ang. Capacity [kg]) – maksymalna i nieprzekraczalna nośność pojazdu używana w algorytmie do sprawdzania czy data trasa może być obsłużona przez bieżący pojazd,

- Szerokość [m] (ang. Width [m]), Długość [m] (ang. Length [m]), Wysokość [m] (ang. Height [m]) – maksymalne i nieprzekraczalne wymiary pojazdu używane w algorytmie do sprawdzania, czy data trasa może być obsłużona przez bieżący pojazd,

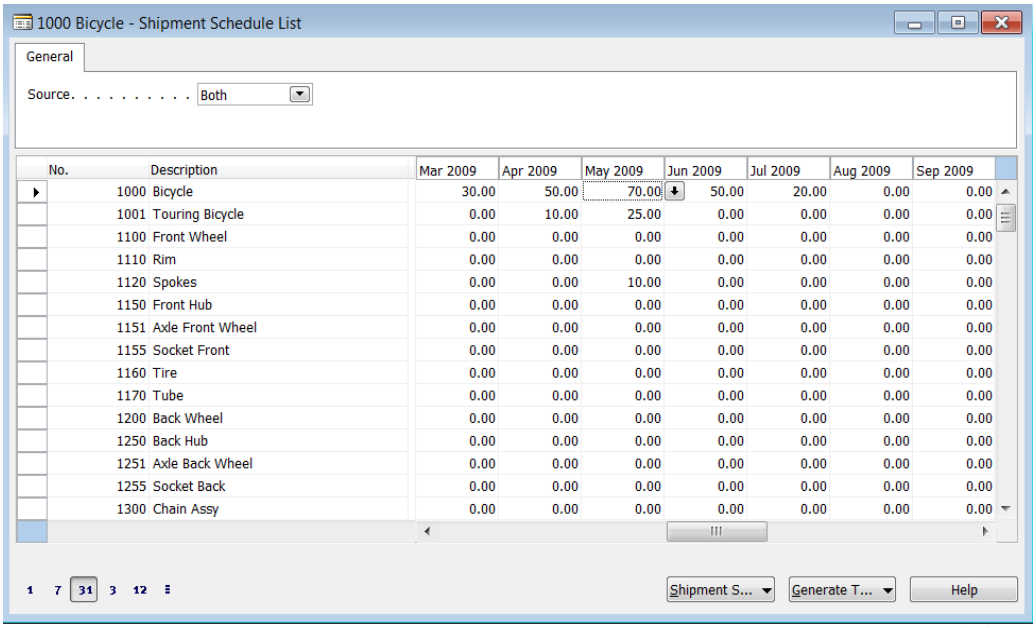
- Nr dostawcy (ang. Vendor No.) – pole informacyjne nieużywane przez algorytm optymalizacji tras zawierające unikalny numer wewnętrzny dostawcy danego pojazdu, np. firmy logistycznej,

- Współczynnik kosztu (ang. Cost Factor) – pole używane podczas minimalizacji funkcji kosztu. Jeśli np. możliwe jest przypisanie kilku samochodów do jednej trasy, priorytet uzyskuje samochód z mniejszym kosztej jednostkowym.

4.1.2. Tworzenie listy wysyłek

Tworzenie listy wysyłek odbywa się w momencie, gdy wszystkie zamówienia sprzedaży dla danego okresu zostały zaksięgowane i zostały stworzone dokumenty wysyłki. Księgowanie i tworzenie dokumentów wysyłek jest standardową funkcjonalnością systemu NAV i nie jest elementem niniejszej pracy.

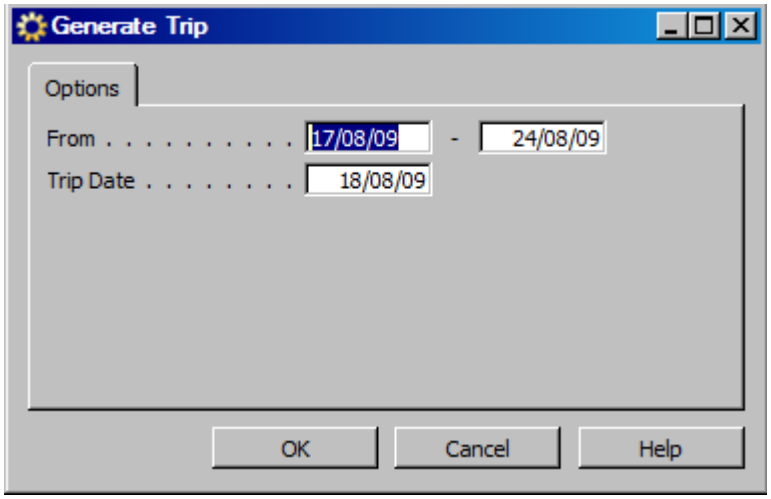
Dodane dla potrzeb niniejszej pracy okno „Lista harmonogramów wysyłek" (ang. Shipment Schedule List) zestawia wszystkie wygenerowane i zaksięgowane zamówienia sprzedaży w zadanych okresach. Dla przykładu rysunek 4.4 pokazuje miesięczne zestawienie towarów do wysyłek. Dla potrzeb analitycznych możliwe jest także przedstawienie wysyłek w ujęciu dziennym, tygodniowym, miesięcznym, kwartalnym oraz rocznym. Towary do wysyłki zawsze przedstawiane są w tzw. bazowych jednostkach miary. Bazowa jednostka miary towaru to jednostka miary, dla której tworzone są zapisy ilościowe oraz kosztowe danego zapasu. Podczas księgowania transakcji magazynowych system NAV przelicza ilość towaru z innych jednostek miary na jednostkę bazową.



Rys. 4.4. Przykładowy harmonogram wysyłek - widok w ujęciu miesięcznym (źródło: opracowanie własne)

Następnie na podstawie powyższej listy generowana jest wysyłka. Odbywa się to za pomocą funkcji znajdującej się w menu Generuj Wysyłkę, Generuj (ang. Generate Trip, Generate). Wysyłki tworzone są zawsze dla przedziału zadanych dat Od-Do (and. From,

To), a sama wysyłka planowana jest na dzień wpisany w polu Data wysyłki (ang. Trip Date).

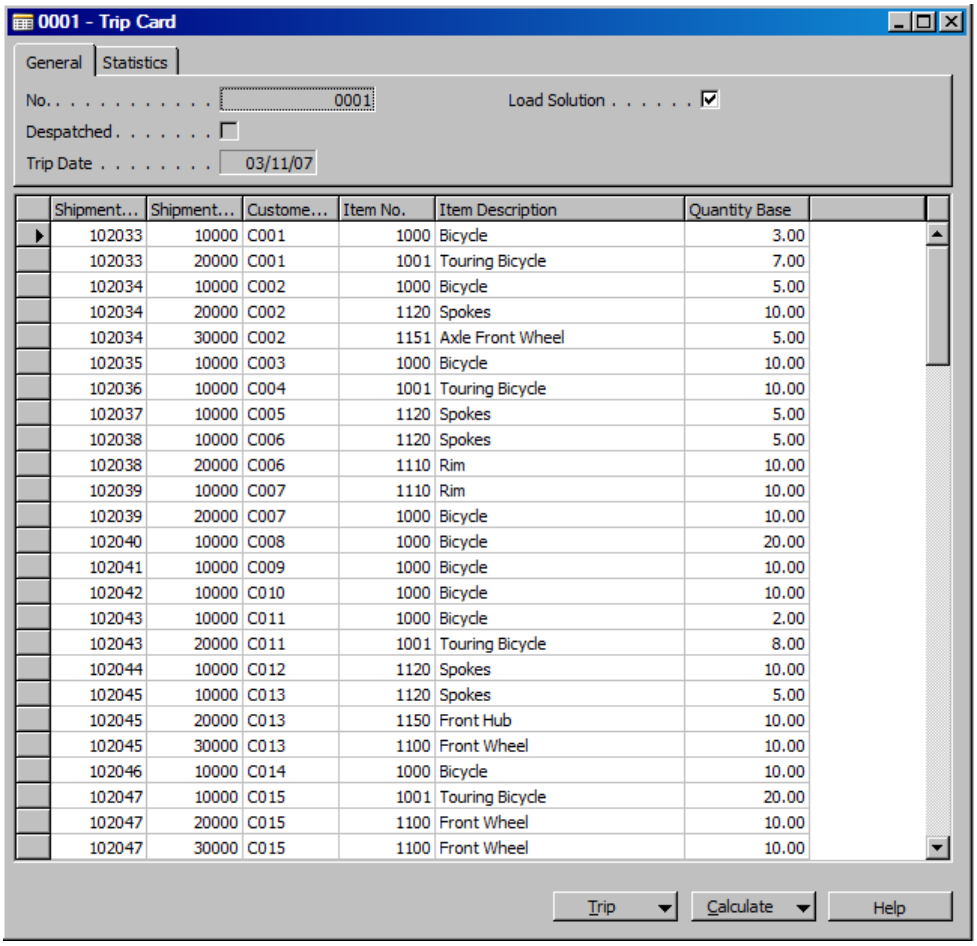


Rys. 4.5. Wyznaczenie wysyłki (źródło: opracowanie własne)

4.1.3. Wyznaczenie oraz obliczanie tras

Z warunków narzuconych przez firmę wynika, że transport obejmuje okres jednego dnia. Jest to efektem rzeczywistego zapotrzebowania – w badanej firmie wszystkie samochody zawsze powracają w ten sam dzień do magazynu centralnego. Dane z harmonogramu wysyłek zbierane są dla dowolnego okresu (rysunek 4.5), ale sama wysyłka tworzona jest na jeden dzień. Tworzenie kartoteki wysyłki zależy od uzgodnionych z klientami terminów dostaw.

Po wygenerowaniu wysyłki system stworzył kartotekę zawierającą listę klientów do obsłużenia danego dnia oraz listę towarów, które każdy klient zamowił. Pole Nr (ang. No.) to unikalny numer nadany automatyczne przez NAV na podstawie zdefiniowanej przez użytkownika serii numeracji, „Data wysyłki” (ang. Trip Date) określa, kiedy towary należy wysłać do klienta. Zaznaczone pole Wysłane (ang. Despatched) oznacza, że dla danej kartoteki już zostało dokonane obliczenie tras i przypisanie pojazdów. Pole „Wczytaj rozwiązanie” (ang. Load Solution) pozwala na wczytanie wcześniej przeliczonych tras jako rozwiązania początkowego dla algorytmu symulowanego wyżarzania. Pola te znajdują się na zakładce „Ogólne” (ang. General).



Shipment...	Shipment...	Custome...	Item No.	Item Description	Quantity Base
102033	10000	C001	1000	Bicycle	3.00
102033	20000	C001	1001	Touring Bicycle	7.00
102034	10000	C002	1000	Bicycle	5.00
102034	20000	C002	1120	Spokes	10.00
102034	30000	C002	1151	Axle Front Wheel	5.00
102035	10000	C003	1000	Bicycle	10.00
102036	10000	C004	1001	Touring Bicycle	10.00
102037	10000	C005	1120	Spokes	5.00
102038	10000	C006	1120	Spokes	5.00
102038	20000	C006	1110	Rim	10.00
102039	10000	C007	1110	Rim	10.00
102039	20000	C007	1000	Bicycle	10.00
102040	10000	C008	1000	Bicycle	20.00
102041	10000	C009	1000	Bicycle	10.00
102042	10000	C010	1000	Bicycle	10.00
102043	10000	C011	1000	Bicycle	2.00
102043	20000	C011	1001	Touring Bicycle	8.00
102044	10000	C012	1120	Spokes	10.00
102045	10000	C013	1120	Spokes	5.00
102045	20000	C013	1150	Front Hub	10.00
102045	30000	C013	1100	Front Wheel	10.00
102046	10000	C014	1000	Bicycle	10.00
102047	10000	C015	1001	Touring Bicycle	20.00
102047	20000	C015	1100	Front Wheel	10.00
102047	30000	C015	1100	Front Wheel	10.00

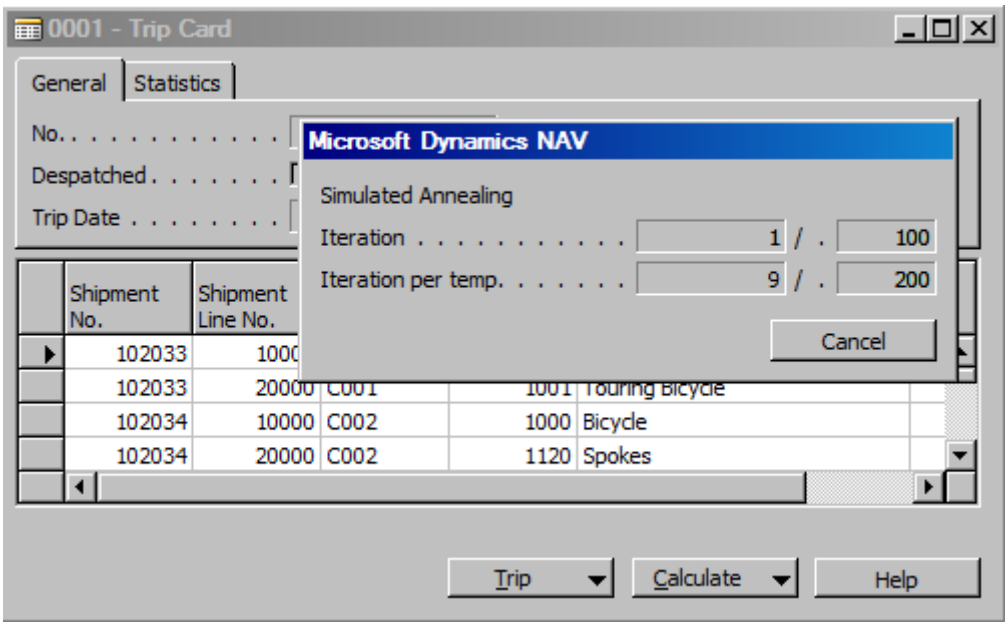
Rys. 4.6. Kartoteka wysyłki (źródło: opracowanie własne)

Kolejna sekcja formatki (rysunek 4.6) to wiersze. Wiersze grupują ilości każdego towaru, który ma być wysłany dla danego klienta w ramach bieżącej wysyłki.

Ilości podane są w bazowej jednostce miary definiowanej w kartotece zapasu, także wszystkie obliczenia wewnątrz prezentowanego w niniejszej pracy algorytmu symulowanego wyżarzania opierają sie o tzw. jednostkę bazową zapasu (ang. Base Unit of Measure). Wynika to ze specyfiki systemu Microsoft Dynamics NAV, który tworzy zapisy księgi zapasu (ang. Item Ledger Entries) oraz zapisy wyceny (ang. Value Entries) właśnie w jednostce bazowej. Różne zapasy mogą posiadać różne jednostki bazowe.

Pierwsze dwie kolumny kartoteki wysyłki, czyli „Nr wysyłki” (ang. Shipment No.) oraz „Nr wiersza wysyłki” (ang. „Shipment Line No.”), pozwalają na dokładne znalezienie źródła danego wiersza wysyłki, czyli dokumentu wysyłki i jego konkretnego wiersza.

Aby obliczyć trasy, należy w menu „Oblicz” (ang. Calculate) wybrać opcję „Oblicz trasy” (ang. Calculate Routes). Po potwierdzeniu system wykona obliczenia bazując na opisanym w niniejszej pracy algorytmie symulowanego wyżarzania.

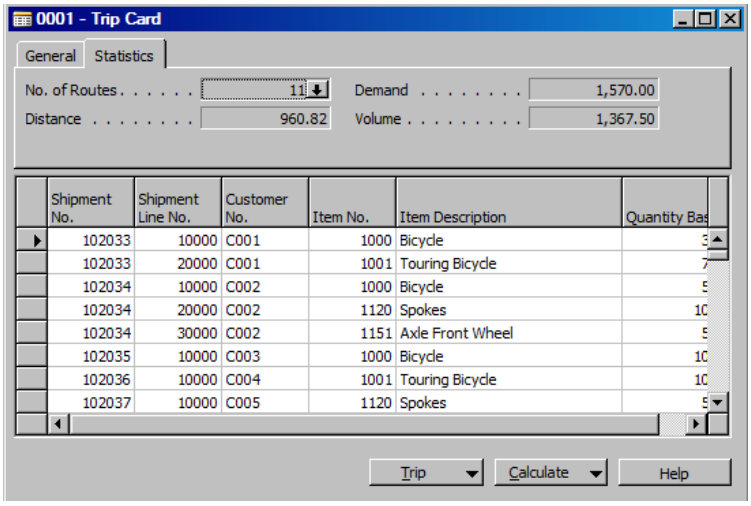


Rys. 4.7. Obliczanie tras oraz okno postępu (źródło: opracowanie własne)

W przypadku, gdy wysyłki jednego klienta proponowane są z różnych magazynów, algorytm sprawdzi optymalny magazyn wysyłki dla danego klienta i zaproponuje przesunięcie międzymagazynowe pozostałych towarów.

Rysunek 4.7. przedstawia postęp obliczeń. W wierszu „Iteracja” (ang. Iteration) widać bieżący numer iteracji w stosunku do całkowitej liczby, a wiersz „Iteracja dla temperatury” (ang. Iteration per Temp.”) to postęp algorytmu dla danej epoki.

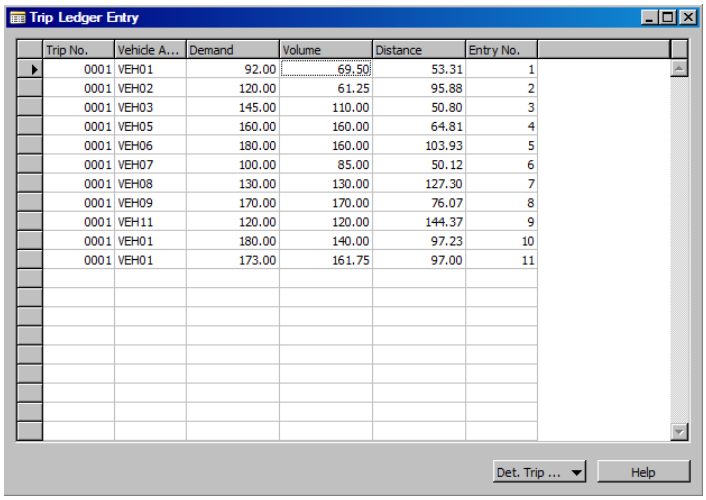
Po zakończeniu obliczeń w zakładce Statystyki (ang. Statistics) można sprawdzić wyniki (rysunek 4.8). W polu „Liczba tras” (ang. No. of Routes) widnieje informacja, ile tras zostało opracowanych przez system. Pole „Odległość” (ang. Distance) to całkowita liczba przejechanych kilometrów dla wszystkich tras danej wysyłki, „Zapotrzebowanie” (ang. Demand) to całkowita waga, a „Objętość” (ang. Volume) to całkowita objętość wszystkich transportowanych towarów.



Rys. 4.8. Wyniki obliczeń algorytmu symulowanego wyżarzania (źródło: opracowanie własne)

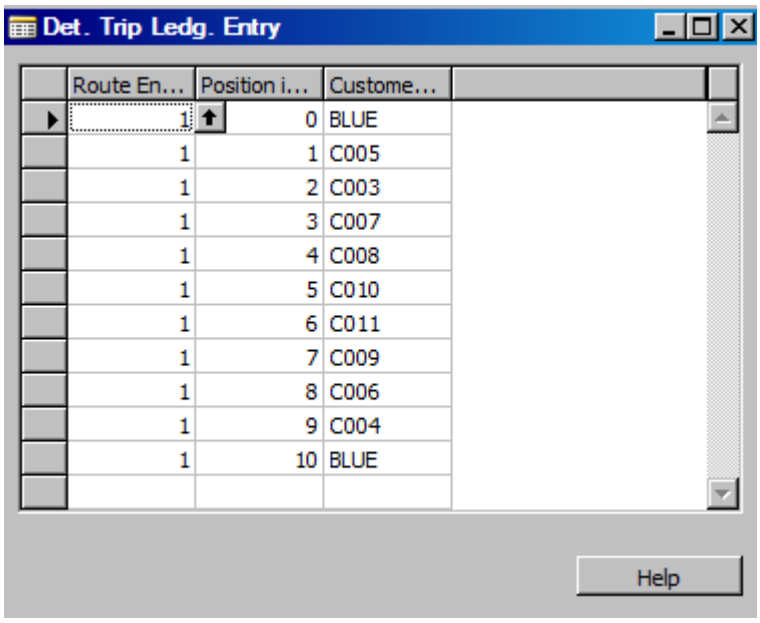
Wybierając opcję „Wysyłka” (ang. Trip), „Zapisy księgi” (ang. Ledger Entries) lub naciskając klawisz strzałki (ang. Drill Down Button) po prawej stronie dowolnego z pól w zakładce Statystyki wyświetlone zostanie okno „Zapisy księgi wysyłki” (ang. Trip Ledger Entries).

Zapisy księgi wysyłki przedstawiają wszystkie obliczone przez algorytm trasy. Na przykładowym rysunku 4.8 system wyliczył 11 tras, przypisał do nich pojazd – pole „Przypisany pojazd” (ang. Vehicle Assigned). W polach „Zapotrzebowanie” (ang. Demand), „Objętość” (ang. Volume), Odległość (ang. Distance) pokazane są odpowiednio zapotrzebowanie wagowe i objętościowe wszystkich klientów na danej trasie oraz całkowity dystans przejechany przez każdy pojazd.



Rys. 4.9. Zapisy księgi wysyłki (źródło: opracowanie własne)

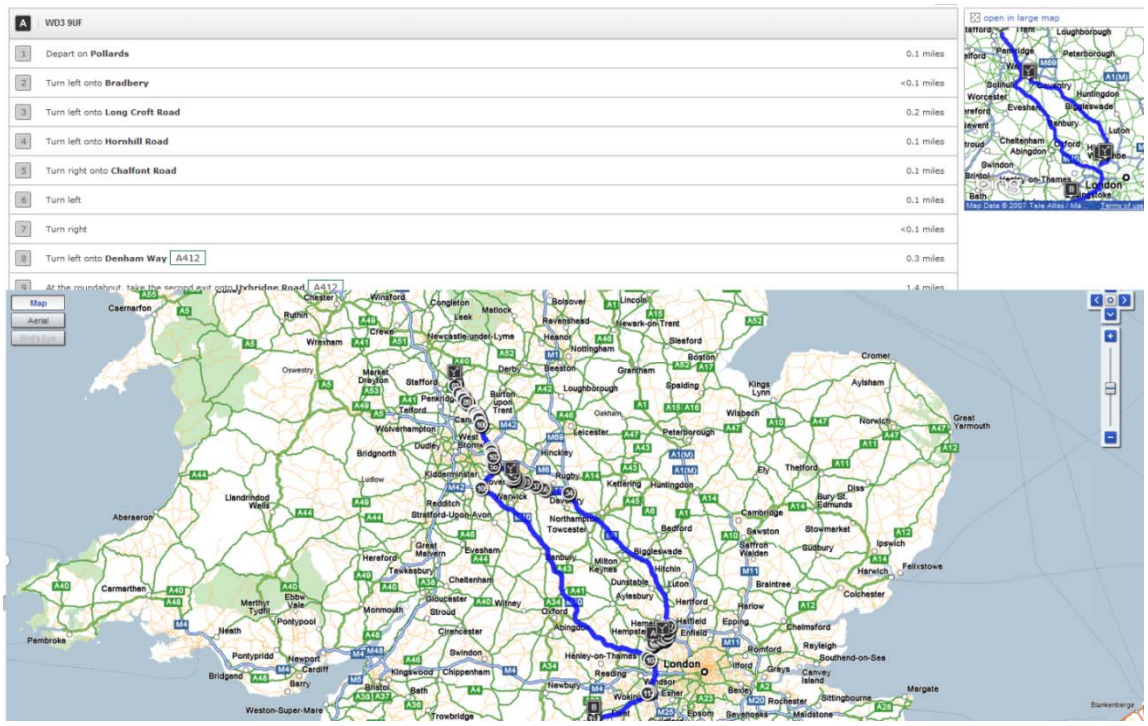
Aby prześledzić szczegóły każdej obliczonej trasy, należy wybrać opcję „Szczegółowe zapisy księgi wysyłki” (ang. Detailed Trip Ledger Entries) znajdującą się w menu „Szczeg. Zapisy księgi Wysyłki” (ang. Det. Trip Ledger Entries).



Rys. 4.10. Szczegóły wysyłki (źródło: opracowanie własne)

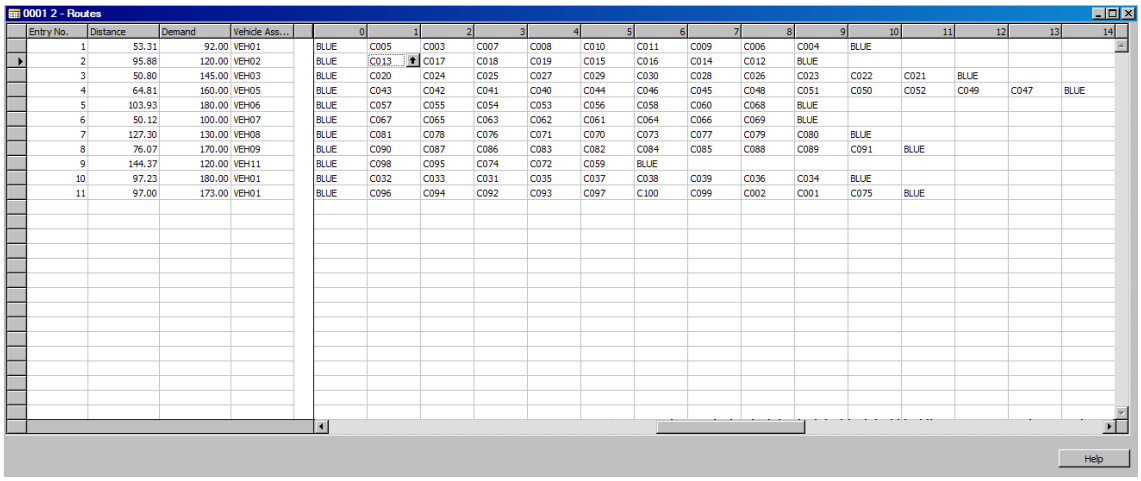
Rysunek 4.10 wyświetla z jakiego magazynu należy wyjechać i do jakiego wrócić oraz w jakiej kolejności jechać do poszczególnych klientów na bieżącej trasie.

Na formatce „Zapisy księgi wysyłki” istnieje możliwość obejrzenia danej trasy na mapie w systemie GIS (system informacji geograficznej, ang. Geographical Information System). W obecnej chwili autor niniejszej rozprawy dokonał integracji Microsoft Dynamics NAV z multimap.com.



Rys. 4.11. Przegląd obliczonych tras oraz szczegóły podróży przedstawione na mapie w systemie multimap.com (źródło: opracowanie własne)

Z kartoteki wysyłki po wykonaniu obliczeń jest możliwość przeglądu szczegółów wszystkich tras w jednym oknie. Rysunek 4.12 pokazuje wszystkie trasy w ramach danej wysyłki wraz ze szczegółami dotyczącymi odwiedzanych klientów w ujęciu tabelarycznym.

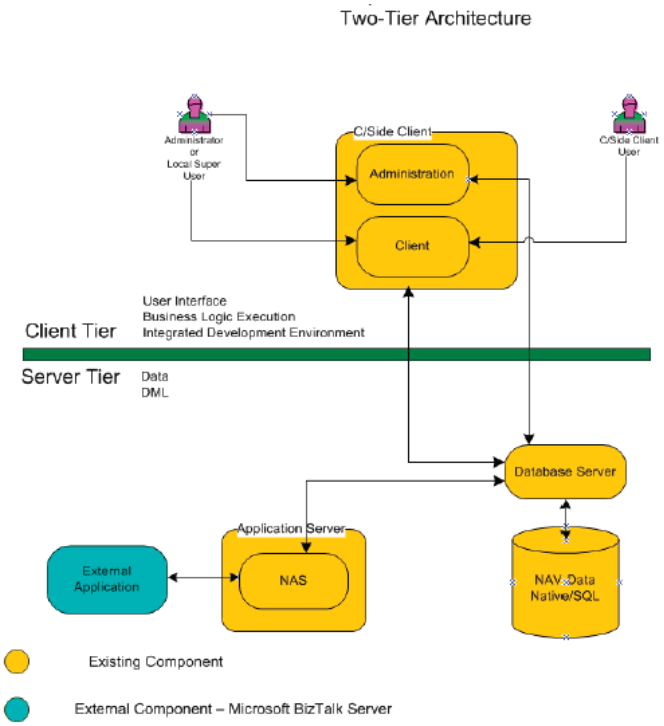


Entry No.	Distance	Demand	Vehicle Ass...	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	53.31	92.00	VEH01	BLUE	C005	C003	C007	C008	C010	C011	C009	C006	C004	BLUE				
2	95.88	120.00	VEH02	BLUE	C013	C017	C018	C019	C015	C016	C014	C012	BLUE					
3	50.80	145.00	VEH03	BLUE	C020	C024	C025	C027	C029	C030	C028	C026	C023	C022	C021	BLUE		
4	64.81	160.00	VEH05	BLUE	C043	C042	C041	C040	C044	C046	C045	C048	C051	C050	C052	C049	C047	BLUE
5	103.93	180.00	VEH06	BLUE	C057	C055	C054	C053	C056	C058	C060	C068	BLUE					
6	50.12	100.00	VEH07	BLUE	C067	C065	C063	C062	C061	C064	C066	C069	BLUE					
7	127.30	130.00	VEH08	BLUE	C081	C078	C076	C071	C070	C073	C077	C079	C080	BLUE				
8	76.07	170.00	VEH09	BLUE	C090	C087	C086	C083	C082	C084	C085	C088	C089	C091	BLUE			
9	144.37	120.00	VEH11	BLUE	C098	C095	C074	C072	C059	BLUE								
10	97.23	180.00	VEH01	BLUE	C032	C033	C031	C035	C037	C038	C039	C036	C034	BLUE				
11	97.00	173.00	VEH01	BLUE	C096	C094	C092	C093	C097	C100	C099	C002	C001	C075	BLUE			

Rys. 4.12. Zestawienie wszystkich tras dla danej wysyłki (źródło: opracowanie własne)

4.2. Budowa modułu „Planowanie wysyłek”

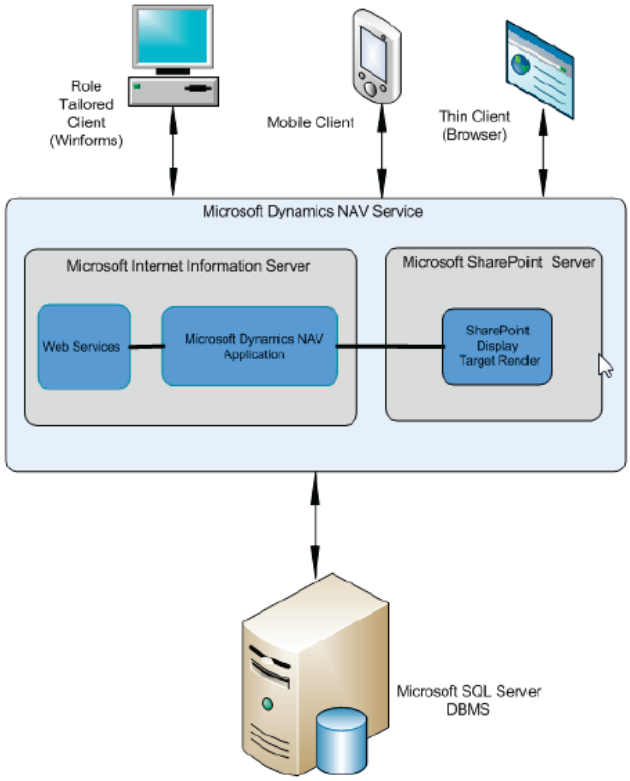
System NAV w wersji 2009 posiada 2 interfejsy użytkownika, tzw. klienta klasycznego oraz interfejsu RTC (klient zorientowany zadaniowo, ang. Role Tailored Client). Klient klasyczny bezpośrednio łączy się z serwerem bazy danych. Warstwa przetwarzania, manipulacji danymi jest po stronie serwera bazodanowego, a warstwa aplikacji oraz prezentacji po stronie klienta. Dwuwarstwową architekturę klient/serwer przedstawia rysunek 4.13.



Rys. 4.13. Dwuwarstwowa architektura systemu NAV przy użyciu interfejsu klasycznego (źródło: Microsoft, 2008)

Architektura trójwarstwowa składa się z interfejsu użytkownika zwanego klientem RTC, serwera bazy danych oraz pośredniej wartwy serwera NAV, który służy jako serwer aplikacji. W tym modelu klient jest używany tylko i wyłącznie do prezentacji danych. Logika biznesowa, sprawdzanie uprawnień użytkownika do danych jest przetwarzane w warstwie aplikacji. Model trójwarstwowy przedstawia rysunek 4.14.

Aplikacja „Planowanie wysyłek” została napisana dla obu interfesów użytkownika.



Rys. 4.14. Architektura wielowarstwowa systemu NAV przy interfejsie RTC (źródło: Microsoft, 2008)

Podrozdziały 4.2.1 oraz 4.2.2 opisują szczegółowo zmienione oraz dodane elementy systemu NAV w celu stworzenia modułu „Planowanie wysyłek".

4.2.1. Budowa aplikacji

Język programowania C/AL wbudowany w system Microsoft Dynamics NAV 2009 jest językiem strukturalnym z pewnymi cechami obiektowości. Nie ma możliwości definiowania własnych klas oraz zdarzeń, nie ma także dziedziczenia, ale programiści mogą korzystać z pewnej predefiniowanej liczby obiektów.

Do obiektów sytemu NAV należą:

- tabele (ang. Table),
- formatki (ang. Form),
- raporty (ang. Report),
- dataporty (ang. Dataport),
- XMLPorty (ang. XMLPort),

- jednostki kodu (ang. Codeunit),
- menu (ang. MenuSuite),
- strony (ang. Page).

Tabele służą do przechowywania danych, składają się one ze struktury (właściwości, pól, kluczy, triggerów pól oraz tabeli oraz procedur i funkcji C/AL) oraz danych. Formatki służą do przeglądu, modyfikacji, usuwania oraz dodawania danych. Formatki używane są w kliencie klasycznym. Raporty służą do przedstawiania danych w formie wydruków oraz do manipulowania danymi. Za pomocą raportu jest możliwe dodanie rekordów w tabelach, ich usuwanie czy modyfikacja. Dataporty i XMLPorty służą do eksportu oraz importu danych do i z plików systemu Windows. Przy czym dataporty pozwalają na operacje na plikach tekstowych, a XMLPorty na plikach w formacie xml. Jednostki kodu sa to zgrupowane procedury i funkcje użytkownika napisane w języku C/AL. Te funkcje oraz procedury mogą być używane przez pozostałe obiekty systemu NAV. Menu zawiera definicję menu systemu NAV. Strony są odpowiednikiem formatek dla interfejsu RTC i służą do przeglądu, modyfikacji, usuwania oraz dodawania danych.

Microsoft Dynamics NAV jest aplikacją będącą kontynuacją systemu Navision Financials, a następnie Navision Attain stworzonego przez duńską firmę Navision. System ten w 2002 został zakupiony przez Microsoft i obecnie jest przez nich rozwijany.

Aplikacja NAV składa się z pewnej liczby predefiniowanych obiektów. Firmy wdrażające NAV posiadają licencję, która pozwala im na zmiany standardowego kodu oraz dopisywanie własnych modułów. W celu usystematyzowania pracy w NAV każdy obiekt posiada unikalny numer w ramach danego typu obiektu. Czyli np. nie mogą istnieć dwie tabele o numerach 32, ale może istnieć tabela 32 oraz formatka 32. Przyjmuje się, że wszystkie obiekty w zakresie numeracji od 50000 do 99999 są obiektami dodawanymi programistów systemu NAV. Oznacza to, że obiekty o numerach niższych niż 50000 opisane w niniejszym rozdziale są obiektami standardowymi i zostały zmodyfikowane przez autora niniejszej pracy, natomiast obiekty na numeracji 60000 i powyżej zostały dodane przez tegoż autora.

Do budowy aplikacji omówionej w niniejszej pracy autor zmodyfikował oraz dodał następujące obiekty:

Typ obiektu	ID	Nazwa obiektu	Opis
Table	14	Location	Definicja magazynów centralnych
Table	18	Customer	Definicja klientów
Table	60000	Shipment Document	Lista dokumentów WZ
Table	60001	Trip Line	Wiesze kartoteki wysyłki
Table	60002	Transport Setup	Ustawienia modułu
Table	60003	Trip Header	Nagłówek kartoteki wysyłki
Table	60004	Vehicle	Kartoteka pojazdów
Table	60006	Combined Item	Tabela pomocnicza - lista zapasów do bieżącej wysyłki
Table	60007	Customer Affected	Tabela pomocnicza - lista klientów do bieżącej wysyłki
Table	60008	Route	Obliczone przez system trasy
Table	60009	Route Customer	Obliczone przez system szczegóły trasy - lista klientów przypadających dla danej trasy
Form	21	Customer Card	Definicja klientów
Form	5703	Location Card	Definicja magazynów centralnych
Form	60000	Shipment Schedule List	Lista dokumentów WZ przygotowanych do wysyłki
Form	60001	Shipment Document List	Lista dokumentów WZ
Form	60002	Transport Setup	Ustawienia modułu
Form	60003	Trip Card	Kartoteka wysyłki
Form	60004	Trip List	Lista wysyłek
Form	60005	Trip Subform	Wiesze kartoteki wysyłki
Form	60006	Vehicles	Lista pojazdów
Form	60007	Trip Ledger Entry	Zapisy księgi wysyłki
Form	60008	Det. Trip Ledg. Entry	Szczegółowe zapisy księgi wysyłki
Form	60009	Routes	Obliczone przez system trasy
Report	60000	Generate Trip	Tworzenie kartoteki wysyłki na podstawie zebranych dokumentów WZ

Report	60001	Calculate Trips	Skrypt uruchamiający obliczanie wysyłek, algorytm symulowanego wyżarzania
Codeunit	60000	Trip Management	Procedury i funkcje wspomagające tworzenie i zarządzanie kartoteką wysyłek
Codeunit	60001	Simulated Annealing	Procedury i funkcje algorytmu symulowanego wyżarzania

Tabela 4.1. Lista zmienionych i dodanych obiektów w systemie NAV (źródło: opracowanie własne)

4.2.2. Szczegółowy opis zmian

Table 14 Location – Tabela służy do przechowywania informacji o magazynach firmy. Zmiany w tabeli Location:

Dodane pola w tabeli Location przedstawia tabela 4.2.

Field No.	Field Name	Data Type	Length
60000	Position X	Integer	
60001	Position Y	Integer	
60002	Default Start Time	Time	
60003	Default End Time	Time	
60005	Default Int. Start Time	Integer	
60006	Default Int. End Time	Integer	

Tabela 4.2. Lista dodanych pól w tabeli Location (źródło: opracowanie własne)

W tabeli tej dokonywane są także przeliczenia wartości pól 60002 oraz 60003 z typu Time na Integer. W tym celu została dodana procedura wywoływana z trigerów OnValidate() wymienionych pól.

Tabela 18 Customer – Tabela służy do przechowywania informacji o klientach firmy, ich nazwy, adresy, ustawienia księgowe, itp. Zmiany w tabeli Customer:

Dodane pola w tabeli Customer przedstawia tabela 4.3.

Field No.	Field Name	Data Type	Length
60000	Position X	Integer	
60001	Position Y	Integer	
60002	Default Start Time	Time	
60003	Default End Time	Time	
60005	Default Int. Start Time	Integer	
60006	Default Int. End Time	Integer	

Tabela 4.3. Lista dodanych pól w tabeli Customer (źródło: opracowanie własne)

W tabeli tej dokonywane są także przeliczenia wartości pól 60002 oraz 60003 z typu Time na Integer. W tym celu została dodana procedura wywoływana z trigerów OnValidate() wymienionych pól

Tabela 60000 Shipment Document – Tabela zaproponowana przez autora niniejszej aplikacji oraz algorytmu. Jest to tabela pomocnicza, wykorzystywana wewnętrznie przez NAV. Gromadzi ona informacje o zaksięgowanych oraz niezaksięgowanych dokumentach wysyłki. Struktura tabeli:

Enabled	Field No.	Field Name	Data Type	Length	Key
Yes	1	Type	Option		PK1
Yes	2	Document No.	Code	20	PK2
Yes	3	Customer No.	Code	20	
Yes	4	Customer Name	Text	50	

Tabela 4.4. Lista pól tabeli Shipment Document (źródło: opracowanie własne)

Tabela 60001 Trip Line – Tabela zaproponowana przez autora niniejszej aplikacji oraz algorytmu. Tabela zbiera i przechowuje wiersze danej wysyłki, czyli zestawienia

towarów, które mają być wysłane do klientów w ramach dokumentów wysyłki. Struktura tabeli 60001 jest następująca:

Enabled	Field No.	Field Name	Data Type	Length	Key
Yes	1	Trip No.	Code	20	PK1
Yes	2	Line No.	Integer		PK2
Yes	3	Shipment No.	Code	20	
Yes	4	Shipment Line No.	Integer		
Yes	5	Item No.	Code	20	
Yes	6	Quantity Base	Decimal		
Yes	7	Item Description	Text	30	
Yes	8	Customer No.	Code	20	
Yes	9	Location Code	Code	10	

Tabela 4.5. Lista pól w tabeli Trip Line (źródło: opracowanie własne)

Tabela 60002 Transport Setup –Tabela posiada tylko jeden rekord z ustawieniami modułu. Są to seria numeracji dla kartotek wysyłek, a także parametry algorytmu symulowanego wyżarzania. Struktura tabeli jest następująca:

Enabled	Field No.	Field Name	Data Type	Length	Key
Yes	1	Primary Key	Code	10	PK
Yes	2	Trip Nos.	Code	10	
Yes	3	No. of Iterations	Integer		
Yes	4	No. of Iteration per Temp.	Integer		
Yes	5	Temperature Reduction	Decimal		
Yes	6	Start Temperature	Decimal		
Yes	7	No. of Transferred Customers	Integer		
Yes	8	Default Vehicle Capacity [kg]	Decimal		

Yes	9	Default Vehicle Capacity [m3]	Decimal		

Tabela 4.6. Lista pól w tabeli Transport Setup (źródło: opracowanie własne)

Tabela 60003 Trip Header – Tabela zaproponowana przez autora niniejszej aplikacji oraz algorytmu. Tabela zbiera i przechowuje ogólne dane wysyłki takie jak planowana data wysyłki oraz obliczona liczba tras, całkowity dystans, itp. Struktura tabeli 60003 jest następująca:

Enabled	Field No.	Field Name	Data Type	Length	Key
Yes	1	No.	Code	20	PK
Yes	2	Despatched	Boolean		
Yes	3	Trip Date	Date		
Yes	4	Distance	Decimal		
Yes	5	No. of Routes	Integer		
Yes	6	Load Solution	Boolean		
Yes	7	Demand	Decimal		
Yes	8	Volume	Decimal		

Tabela 4.7. Lista pól w tabeli Trip Header (źródło: opracowanie własne)

Tabela 60004 Vehicle – Tabela zaproponowana przez autora niniejszej aplikacji oraz algorytmu. Tabela zbiera i przechowuje dane dotyczące pojazdów, które mogą być wykorzystywane podczas planowania tras. Struktura tabeli 60004 jest następująca:

Enabled	Field No.	Field Name	Data Type	Length	Key
Yes	1	Code	Code	10	PK
Yes	2	Description	Text	30	
Yes	3	Capacity [kg]	Decimal		
Yes	4	Width [m]	Decimal		

Yes	5	Length [m]	Decimal		
Yes	6	Height [m]	Decimal		
Yes	7	Vendor No.	Code	20	
Yes	8	Capacity [m3]	Decimal		
Yes	9	Unit Cost	Decimal		

Tabela 4.8. Lista pól w tabeli Vehicle (źródło: opracowanie własne)

Tabela 60006 Combined Item – Tabela zaproponowana przez autora niniejszej aplikacji oraz algorytmu. Jest to tabela wykorzystywana wewnętrznie przez system do grupowania zapotrzebowania każdego klienta wg asortymentu w ramach jednej wysyłki. Struktura tabeli 60006 jest następująca:

Enabled	Field No.	Field Name	Data Type	Length	Key
Yes	1	Entry No.	Integer		PK
Yes	2	Customer No.	Code	20	
Yes	3	Item No.	Code	20	
Yes	4	Quantity	Decimal		
Yes	5	Volume	Decimal		
Yes	6	Location Code	Code	10	

Tabela 4.9. Lista pól w tabeli Combined Item (źródło: opracowanie własne)

Tabela 60007 Customer Affected – Tabela zaproponowana przez autora niniejszej aplikacji oraz algorytmu. Jest to tabela wykorzystywana wewnętrznie przez system do grupowania klientów, którzy mają być obsłużeni w ramach danej wysyłki. Struktura tabeli 60007 jest następująca:

Enabled	Field No.	Field Name	Data Type	Length	Key
Yes	1	Customer No.	Code	20	PK

Yes	2	Position X	Integer		
Yes	3	Position Y	Integer		
Yes	4	Default Start Time	Integer		
Yes	5	Default End Time	Integer		
Yes	6	Service Time	Integer		
Yes	7	Demand	Decimal		
Yes	9	Entry No.	Integer		
Yes	10	Volume	Decimal		
Yes	11	Central Depot	Boolean		
Yes	12	Location Code	Code	10	

Tabela 4.10. Lista pól w tabeli Customer Affected (źródło: opracowanie własne)

Tabela 60008 Route – Tabela zaproponowana przez autora niniejszej aplikacji oraz algorytmu. Przechowuje wyniki algorytmu symulowanego wyżarzania takie jak liczba tras, całkowity dystans, zapotrzebowanie oraz kod pojazdu przypisany do danej trasy. Struktura tabeli 60008 jest następująca:

Enabled	Field No.	Field Name	Data Type	Length	Key
Yes	1	Entry No.	Integer		PK
Yes	2	Demand	Decimal		
Yes	3	Distance	Decimal		
Yes	4	Vehicle Assigned	Code	10	
Yes	5	Trip No.	Code	20	
Yes	6	Volume	Decimal		

Tabela 4.11. Lista pól w tabeli Route (źródło: opracowanie własne)

Tabela 60009 Route Customer – Tabela zaproponowana przez autora niniejszej aplikacji oraz algorytmu. Przechowuje obliczone przez algorytm symulowanego wyżarzania szczegóły każdej trasy. Struktura tabeli 60009 jest następująca:

Enabled	Field No.	Field Name	Data Type	Length	Key
Yes	1	Route Entry No.	Integer		PK1
Yes	2	Position in Route	Integer		PK2
Yes	3	Customer No.	Code	20	
Yes	4	Trip No.	Code	20	
Yes	5	Central Depot	Boolean		

Tabela 4.12. Lista pól w tabeli Route Customer (źródło: opracowanie własne)

Formatki (ang. Forms) to okna zaprojektowane do wprowadzania, usuwania oraz modyfikowania danych. Do formatek 21 oraz 5703 zostały dodane dodatkowe pola z tablic 18 oraz 14, dodane również przez autora niniejszej pracy formatki 60000 do 60009 służą tylko i wyłącznie do manipulowania danymi i nie posiadają żadnej dodatkowej logiki biznesowej.

Raporty w środowisku C/SIDE oprócz prezentacji danych w formie posiadają bardzo szerokie możliwości manipulacji danymi, czyli usuwanie, modyfikacja oraz wstawianie rekordów. W aplikacji opisywanej w niniejszej pracy autor dodał dwa obiekty typu raport, które służą jako skrypty, czyli przetwarzają dane.

Raport 60000 Generate Trip – Skrypt tworzy kartotekę wysyłki na podstawie wybranych przez użytkownika dokumentów wysyłki. Dla każdego wiersza dokumentu tworzony jest odpowiadający mu wiersz wysyłki.

Raport 60001 Calculate Trip – Skrypt, który rozpoczyna obliczanie trasy za pomocą opisywanego w niniejszej pracy algorytmu symulowanego wyżarzania.

Codeunity w systemie Microsoft Dynamics NAV są obiektami zawierającymi procedury i funkcje napisane w języku C/AL. Moduł planowania wysyłek posiada dwa codeunity:

Codeunit 60000 Trip Management – codeunit zawiera następujące funkcje i procedury:

Trigger OnRun() – wbudowany trigger uruchamiany za pomocą polecenia C/AL RUN. W triggerze tym autor inicjuje zmienne, okno dialogowe pokazujące postęp pracy algorytmu oraz uruchamia codeunit obliczający trasy.

IntToTime – funkcja zamieniająca milisekundy – liczba typu Interger na czas w formacie hh:mm:ss

TimeToInt – funkcja zamieniająca zmienną typu Time (w formacie hh:mm:ss) na milisekundy – liczbę typu całkowitego.

CheckTrip – procedura sprawdzająca dla każdego wiersza wysyłki, czy wszyscy klienci posiadają zdefiniowane położenie oraz okno czasowe, a także czy każdy zapas wysyłany ma zdefiniowaną wagę, wymiary. Procedura także uruchamia procedurę CombineItems.

CreatePossibleVehicleList – procedura przygotowuje listę dostępnych pojazdów z tabeli Vehicle, które aktualnie nie są przypisane do żadnych innych wysyłek.

CombineItems – procedura grupuje wysyłane towary wg magazynu centralnego oraz klienta docelowego i zapisuje rekordy w tabeli tymczasowej „Combined Items". Tabela ta jest następnie używana jako jedna z danych wejściowych algorytmu symulowanego wyżarzania.

PrepareCustomerList – procedura przygotowuje listę klientów i zapisuje te dane w tymczasowej tabeli „Customer Affected". Tabela ta jest następnie używana jako jedna z danych wejściowych algorytmu symulowanego wyżarzania.

Codeunit 60001 Simulated Annealing – Jest to codeunit zawierający właściwy algorytm symulowanego wyżarzania. Danymi wejściowymi są tabele tymczasowe oraz parametry przygotowane przez codeunit „Trip Management" zawiera następujące funkcje i procedury:

Calculate – procedura startowa algorytmu sprawdzająca, czy rozwiązanie początkowe będzie załadowane z wcześniejszych obliczeń, zerująca zmienne pomocnicze, itp.

GenerateInitSolution – procedura tworząca rozwiązanie początkowe.

GetPositionX – funkcja zwracająca współrzędną X położenia klienta lub magazynu centralnego.

GetPositionY – funkcja zwracająca współrzędną X położenia klienta lub magazynu centralnego.

GetDistance – funkcja obliczająca odległość pomiędzy dwoma klientami, lub klientem i magazynem centralnym.

GetRoutesDistance – funkcja obliczająca długość zadanej trasy.

FindBestPlace – funkcja znajdująca najlepsze miejsce zadanego klienta na trasie. Jeśli nowe miejsce jest lepsze od starego, to funckja zwraca TRUE, w przeciwnym wypadku FALSE.

InsertCustomer – procedura wstawiająca zadanego klienta do zadanej trasy na pozycję daną parametrem Position.

RemoveCustomer – procedura usuwiającego klienta znajdującego się na zadanej pozycji z zadanej trasy.

CheckRoute – funkcja sprawdzająca czy zadana trasa spełnia wszystkie ograniczenia, czyli czy nie są przekroczone żadne okna czasowe, czy nie zostało przekroczone zapotrzebowanie klienta, wymiary pojazdów oraz czy jest dostępny pojazd do obsłużenia danej trasy. Gdy wszystkie ograniczenia są spełnione to funkcja zwraca wartość TRUE, w przeciwnym wypadku FALSE.

CalcSolutionCost – funkcja zwracająca koszt rozwiązania wg wzoru 20.

TransferRoute – procedura pomocnicza kopiująca trasy pomiędzy różnymi zmiennymi typu Record wskazującymi na tabele z trasami.

FindBestSolution – właściwy algorytm symulowanego wyżarzania.

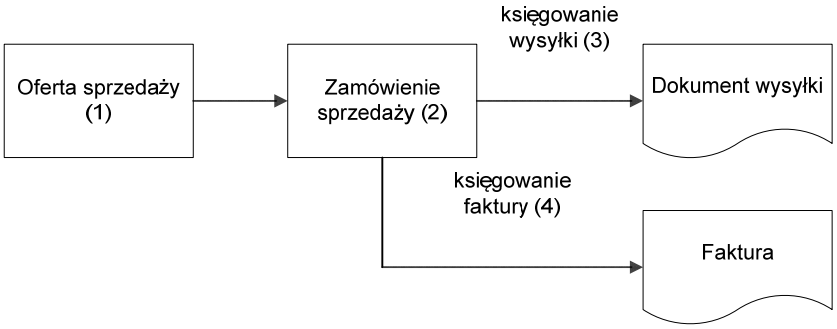
ReduceRoutesQty – Funkcja przejścia.

AssignVehicle – procedura przypisująca pojazdy do każdej obliczonej trasy. W przypadku, gdy nie ma możliwości przypisania pojazdu, procedura zwraca komunikat o błędzie.

Nadrzędnym celem podczas budowy modułu „Planowanie wysyłek” była pełna integracja z pozostałymi funkcjami systemu Microsoft Dynamics NAV 2009. Ponadto Moduł „Planowanie wysyłek” został napisany tak, że w minimalny sposób ingeruje w istniejące obiekty aplikacji, co pozwala po pierwsze uniknąć czasochłonnego upgradu

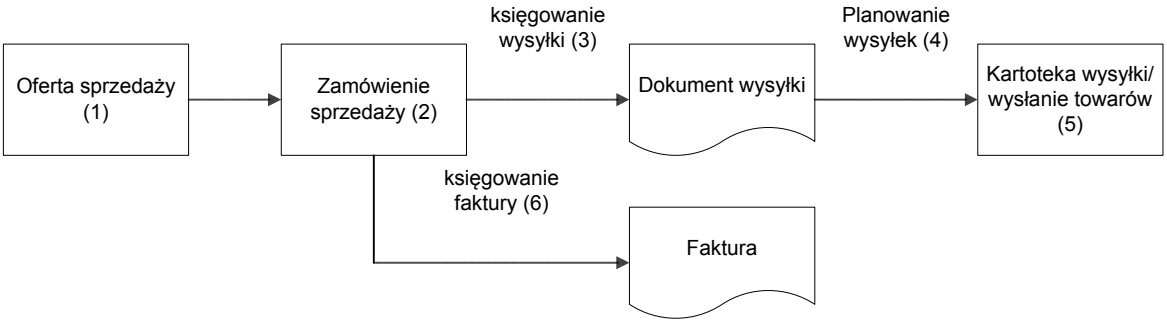
bazy danych, a po drugie potencjalne poprawki do bieżącego systemu wydawane przez Microsoft nie wymagają łączenia obiektów.

Z punktu widzenia użytkownika aplikacji NAV moduł „Planowanie wysyłek" jest rozszerzeniem bieżącej funkcjonalności sprzedaży. Proces sprzedaży zaimplementowany w systemie NAV przedstawiony jest na rysunku 4.15, natomiast proces z uwzględnieniem planowania wysyłek przedstawia rysunek 4.16.



Rys. 4.15. Proces sprzedaży w ogólnej wersji NAV (źródło: opracowanie własne)

Pierwszy krok w procesie sprzedaży to wysłanie do klienta oferty. Jest to krok opcjonalny, który może zostać pominięty. Po otrzymaniu potwierdzenia, wystawiane zostaje zamówienie sprzedaży. Następnie zamówienie jest potwierdzane (tzw. księgowanie wysyłki) i tworzona jest lista towarów do wysłania. Po otrzymaniu zamówionych towarów przez klienta wystawia się fakturę.



Rys. 4.16. Proces sprzedaży z uwzględnieniem planowania wysyłek (źródło: opracowanie własne)

W zmodyfikowanej wersji NAV po wygenerowaniu listy wysyłek, planowane są trasy oraz przypisywane są pojazdy (krok 4 oraz 5).

5. Wnioski końcowe i kierunki dalszych badań

5.1. Podsumowanie badań i wnioski

Przeprowadzone badania w ramach niniejszej rozprawy potwierdziły przyjętą koncepcję badawczą, że algorytm będący hybrydą symulowanego wyżarzania oraz adaptacyjnego przeszukiwania sąsiedztwa jest bardzo skuteczną metodą do rozwiązywania dwu i trójkryterialnych problemów dostaw.

Dla algorytmu symulowanego wyżarzania kluczowe znaczenie posiadają: funkcja generowania rozwiązania początkowego oraz tzw. funkcja przejścia.

Procedura tworzenia rozwiązania początkowego została stworzona w ramach niniejszej rozprawy i charakteryzuje się bardzo dużą wydajnością zarówno pod względem czasu obliczeń jak i jakości wygenerowanego rozwiązania.

Funkcja przejścia pozwala na przeszukiwanie przestrzeni rozwiązań. Głównym elementem funkcji przejścia jest opracowany przez autora niniejszej pracy sposób grupowania klientów wg odległości od magazynu centralnego, który obsługuje danego klienta w ramach bieżącej wysyłki. Algorytm ten został nazwany metodą zwiększających się promieni. Grupowanie klientów ma szczególne znaczenie przy niskich temperaturach, gdzie trudniej jest dokonać przejścia z bieżącego minimum lokalnego i zaakceptować gorsze rozwiązanie. Wspomniane grupowanie pozwala na przenoszenie całych grup pomiędzy trasami, a nie tylko pojedynczych klientów

Opracowany algorytm udowodnił swoją skuteczność wyrównując wiele znanych testów Solomona VRPTW oraz Cordeau MDVRPTW. W dwóch przypadkach poprawiono wyniki światowe testów Solomona, są to testy R203 oraz R211. Zanim niniejsza praca została opublikowana, ostatnie poprawy niektórych testów nastąpiły w roku 2005, dotyczy to R207 oraz R211. Test R203 nie został ulepszony od roku 2001. Coraz częściej pojawiały się opinie, że żadnego z testów Solomona nie da się ulepszyć. Fakt poprawienia aż dwóch testów za pomocą opisywanego algorytmu jest znacznym osiągnięciem.

W pracy zdefiniowany nowy problem logistyczny, który jest rozszerzeniem ogólnego problemu dostaw RDPTW o następujące parametry:

- wymiary pojazdów,
- wymiary dostarczanych towarów,
- wskaźnik kosztu użytkowanych pojazdów.

Wymiary pojazdów oraz dostarczanych towarów są jednym z elementów ograniczających tworzenie nowych tras, natomiast wskaźnik kosztu jest trzecim kryterium optymalizacji i wpływa na uzyskaną wartość funkcji kosztu.

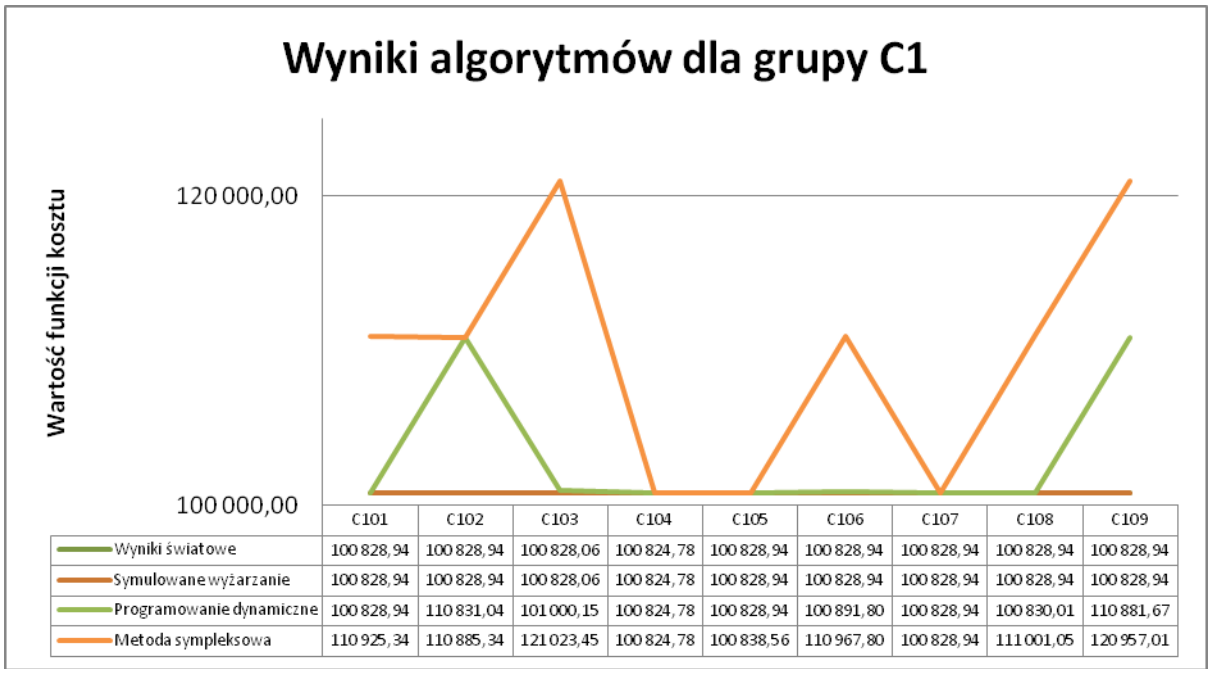
Na postawie wyników uzyskanych na danych testowych rzeczywistych można stwierdzić, że algorytm zbudowany w ramach niniejszej rozprawy charakteryzuje się bardzo dużą wydajnością i jego wdrożenie w krótkim czasie przyniesie wymierne oszczędności.

		Służby zakładowe		System NAV		
Liczba klientów	Liczba mag. Centralnych	Czas obliczeń [h]	Wartość funkcji kosztu	Czas obliczeń [h]	Wartość funkcji kosztu	Rożnica [%]
80	1	4,5	151980	0,2	101650	33.12%
85	2	5	151993	0,2	111630	26.56%
95	1	4,5	161887	0,21	121545	24.92%
100	1	5,5	151901	0,27	111435	26.64%
110	1	5,5	182030	0,3	121162	33.44%
133	2	7	283700	0,55	172929	39.05%
150	2	8.5	324150	1	193564	40.29%
150	3	8.5	334080	1	223218	33.18%

Tabela 5.1. Lista wyników testów rzeczywistych (źródło: opracowanie własne)

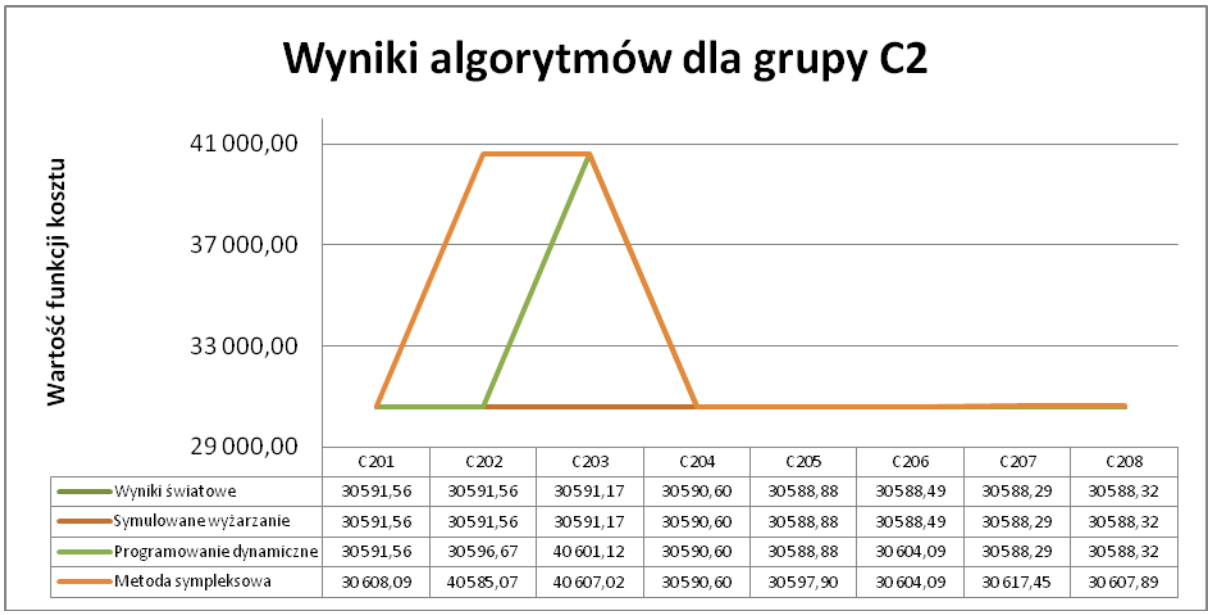
Z tabeli 5.1 wynika, że dzięki wdrożeniu modułu „Planowania wysyłek" znacząco spadły koszty planowania tras (kolumna „Różnica %") – od 25% do 40%. Spadła liczba wysyłanych pojazdów jak i całkowity przejechany przez nie dystans. Przekłada się to wprost na oszczędności poczynione przez badaną firmę oraz na polepszenie jej wizerunku na rynku.

W rozprawie przeprowadzono testy porównawcze algorytmu symulowanego wyżarzania z heurystykami wykorzystującymi programowanie dynamiczne oraz metodę sympleksową. Rysunki 5.1 do 5.6 przedstawiają wartości funkcji kosztów testów Solomona uzyskane za pomocą badanych algorytmów. Wyniki te zostały zestawione z najlepszymi obecnymi wynikami światowymi.



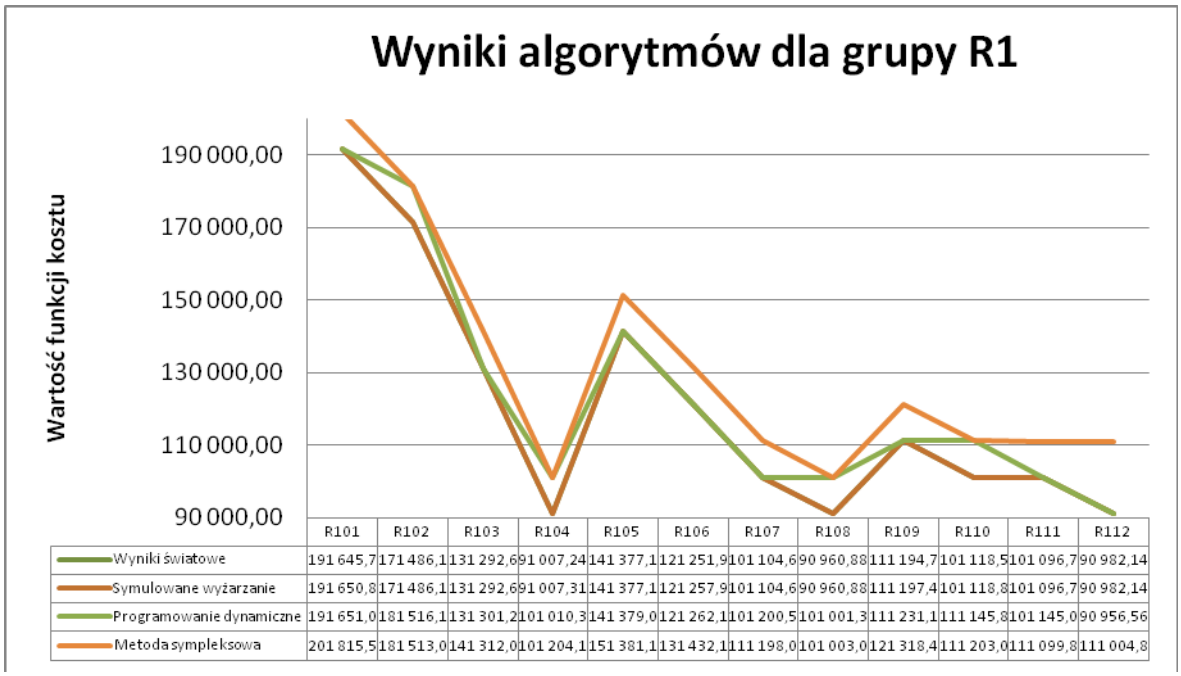
	C101	C102	C103	C104	C105	C106	C107	C108	C109
Wyniki światowe	100 828,94	100 828,94	100 828,06	100 824,78	100 828,94	100 828,94	100 828,94	100 828,94	100 828,94
Symulowane wyżarzanie	100 828,94	100 828,94	100 828,06	100 824,78	100 828,94	100 828,94	100 828,94	100 828,94	100 828,94
Programowanie dynamiczne	100 828,94	110 831,04	101 000,15	100 824,78	100 828,94	100 891,80	100 828,94	100 830,01	110 881,67
Metoda sympleksowa	110 925,34	110 885,34	121 023,45	100 824,78	100 838,56	110 967,80	100 828,94	111 001,05	120 957,01

Rys. 5.1. Zestawienie wartości funkcji kosztów dla grupy C1 (źródło: opracowanie własne)



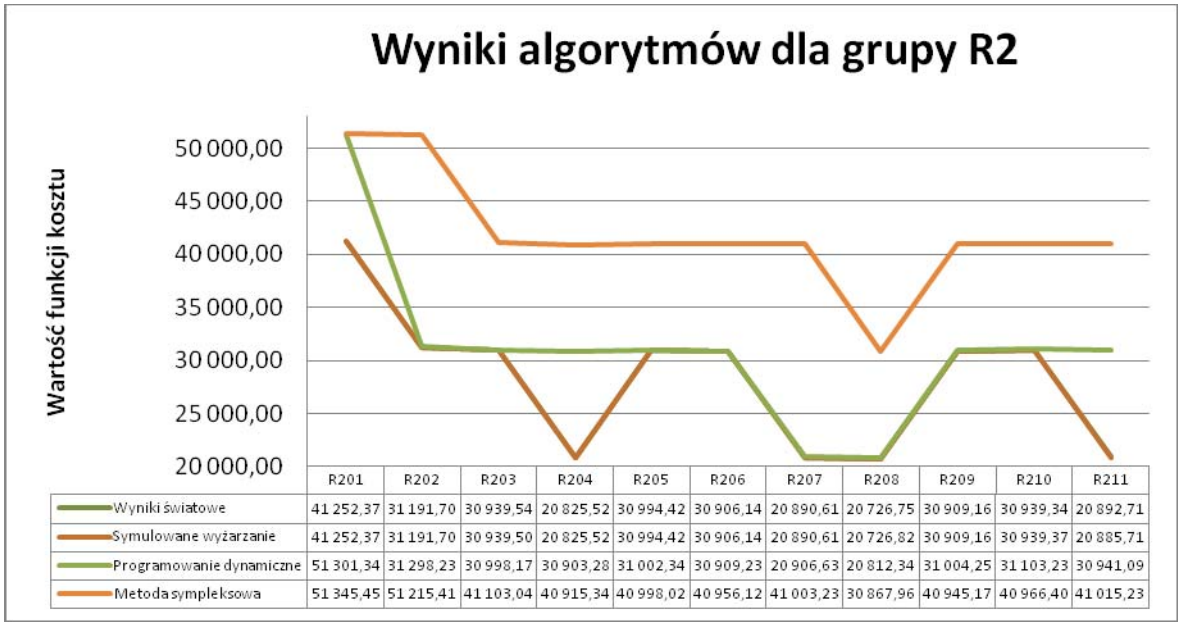
	C201	C202	C203	C204	C205	C206	C207	C208
Wyniki światowe	30591,56	30591,56	30591,17	30590,60	30588,88	30588,49	30588,29	30588,32
Symulowane wyżarzanie	30591,56	30591,56	30591,17	30590,60	30588,88	30588,49	30588,29	30588,32
Programowanie dynamiczne	30591,56	30596,67	40 601,12	30590,60	30588,88	30 604,09	30588,29	30588,32
Metoda sympleksowa	30 608,09	40585,07	40 607,02	30590,60	30597,90	30 604,09	30 617,45	30 607,89

Rys. 5.2. Zestawienie wartości funkcji kosztów dla grupy C2 (źródło: opracowanie własne)



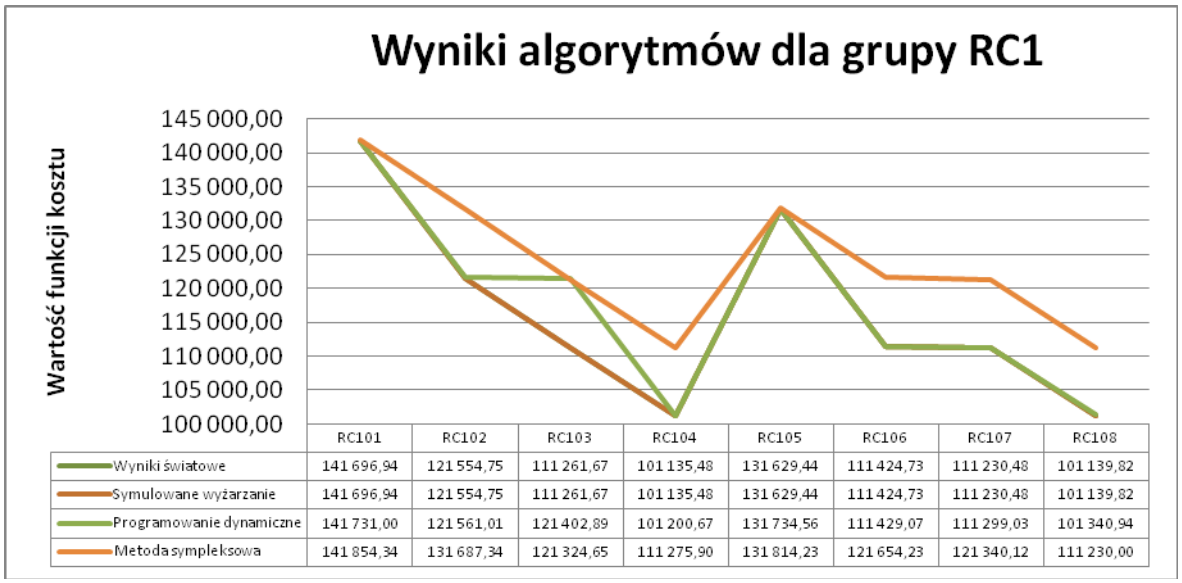
	R101	R102	R103	R104	R105	R106	R107	R108	R109	R110	R111	R112
Wyniki światowe	191 645,7	171 486,1	131 292,6	91 007,24	141 377,1	121 251,9	101 104,6	90 960,88	111 194,7	101 118,5	101 096,7	90 982,14
Symulowane wyżarzanie	191 650,8	171 486,1	131 292,6	91 007,31	141 377,1	121 257,9	101 104,6	90 960,88	111 197,4	101 118,8	101 096,7	90 982,14
Programowanie dynamiczne	191 651,0	181 516,1	131 301,2	101 010,3	141 379,0	121 262,1	101 200,5	101 001,3	111 231,1	111 145,8	101 145,0	90 956,56
Metoda sympleksowa	201 815,5	181 513,0	141 312,0	101 204,1	151 381,1	131 432,1	111 198,0	101 003,0	121 318,4	111 203,0	111 099,8	111 004,8

Rys. 5.3. Zestawienie wartości funkcji kosztów dla grupy R1 (źródło: opracowanie własne)



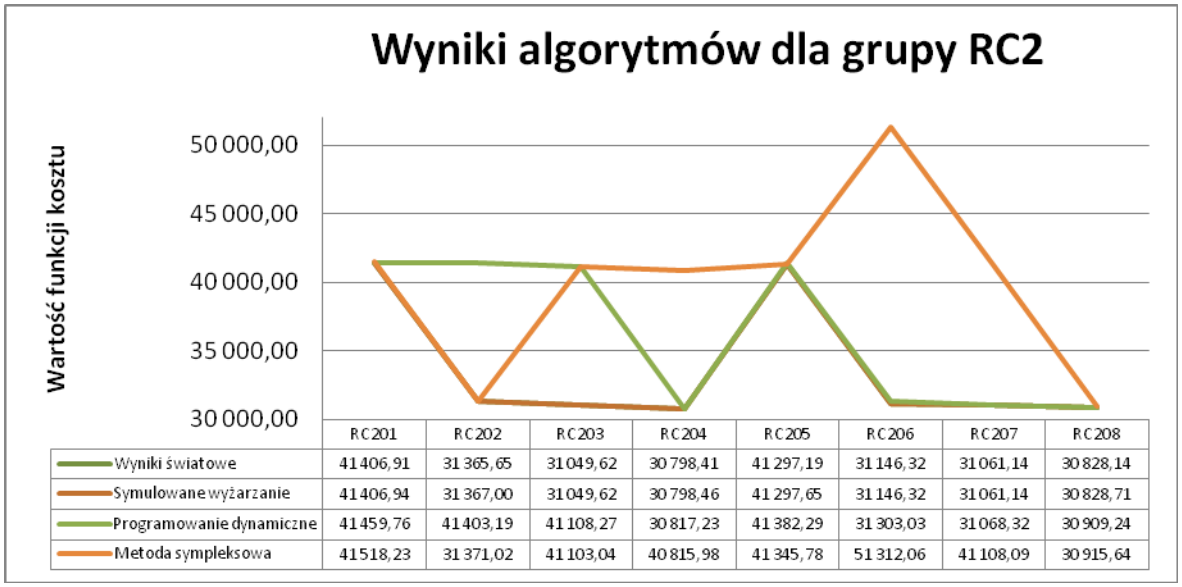
	R201	R202	R203	R204	R205	R206	R207	R208	R209	R210	R211
Wyniki światowe	41 252,37	31 191,70	30 939,54	20 825,52	30 994,42	30 906,14	20 890,61	20 726,75	30 909,16	30 939,34	20 892,71
Symulowane wyżarzanie	41 252,37	31 191,70	30 939,50	20 825,52	30 994,42	30 906,14	20 890,61	20 726,82	30 909,16	30 939,37	20 885,71
Programowanie dynamiczne	51 301,34	31 298,23	30 998,17	30 903,28	31 002,34	30 909,23	20 906,63	20 812,34	31 004,25	31 103,23	30 941,09
Metoda sympleksowa	51 345,45	51 215,41	41 103,04	40 915,34	40 998,02	40 956,12	41 003,23	30 867,96	40 945,17	40 966,40	41 015,23

Rys. 5.4. Zestawienie wartości funkcji kosztów dla grupy R2 (źródło: opracowanie własne)



	RC101	RC102	RC103	RC104	RC105	RC106	RC107	RC108
Wyniki światowe	141 696,94	121 554,75	111 261,67	101 135,48	131 629,44	111 424,73	111 230,48	101 139,82
Symulowane wyżarzanie	141 696,94	121 554,75	111 261,67	101 135,48	131 629,44	111 424,73	111 230,48	101 139,82
Programowanie dynamiczne	141 731,00	121 561,01	121 402,89	101 200,67	131 734,56	111 429,07	111 299,03	101 340,94
Metoda sympleksowa	141 854,34	131 687,34	121 324,65	111 275,90	131 814,23	121 654,23	121 340,12	111 230,00

Rys. 5.5. Zestawienie wartości funkcji kosztów dla grupy RC1 (źródło: opracowanie własne)



	RC201	RC202	RC203	RC204	RC205	RC206	RC207	RC208
Wyniki światowe	41406,91	31365,65	31049,62	30798,41	41297,19	31146,32	31061,14	30828,14
Symulowane wyżarzanie	41406,94	31367,00	31049,62	30798,46	41297,65	31146,32	31061,14	30828,71
Programowanie dynamiczne	41459,76	41403,19	41108,27	30817,23	41382,29	31303,03	31068,32	30909,24
Metoda sympleksowa	41518,23	31371,02	41103,04	40815,98	41345,78	51312,06	41108,09	30915,64

Rys. 5.6. Zestawienie wartości funkcji kosztów dla grupy RC1 (źródło: opracowanie własne)

Z rysunków 5.1 do 5.6 wynika, że wartości uzyskane za pomocą opisywanego algorytmu nie odbiegają od najlepszych wyników światowych – im niżej położony jest dany wykres, tym wyniki metody są lepsze. Ponadto można wysnuć wniosek, że użyta heurystyka z wykorzystaniem algorytmu sympleksowego nie jest tak efektywną metodą jak symulowane wyżarzanie, ale w większości testów charakteryzuje się lepszymi wynikami niż heurystyka z wykorzystaniem programowania dynamicznego.

Porównano także wyniki testów rzeczywistych z innymi aplikacjami komercyjnym rozwiązującymi problemy dostaw: JOpt, Paragon oraz Quagga. Wyniki uzyskiwane za pomocą algorytmu opisywanego w niniejszej pracy charakteryzują się dużo krótszymi trasami i w wielu przypadkach mniejszą liczbą tras. W ani jednym przypadku wynik uzyskany w systemie NAV nie był gorszy od innych porównywanych aplikacji komercyjnych.

W niniejszej pracy dokonano także integracji modułu „Planowania wysyłek" z systemem informacji geograficznej. Obecnie zaplanowane wysyłki w systemie NAV są zintegrowane z aplikacją multimap.com. Multimap pozwala na podgląd oraz wydruk map z zaznaczonymi trasami, przegląd i wydruk szczegółowych kroków każdej trasy.

Koszt pojedynczej iteracji zastosowanego algorytmu symulowanego wyżarzania jest proporcjonalny do n^2, gdzie n to całkowita liczba iteracji (Czech, 2001). Dlatego maksymalna złożoność obliczeniowa zastosowanego algorytmu wynosi:

$$O(n) = 100n^2 \tag{5.1}$$

Jak to zostało opisane w rozdziale 3.5.4, liczba kroków algorytmu jest stała i wynosi 100 dla jednej iteracji. Na podstawie uzyskanych wyników można stwierdzić, że algorytm będący hybrydą symulowanego wyżarzania oraz adaptacyjnego przeszukiwania sąsiedztwa jest bardzo wydajną metodą do rozwiązywania problemów optymalizacyjnych.

5.2. Dalsze kierunki badań

W niniejszej rozprawie podjęto interesujący, ale bardzo obszerny problem naukowy wymagający przeprowadzenia szczegółowych badań ukierunkowanych na poszukiwanie możliwości usprawnienia algorytmów rozwiązujących NP-zupełne problemy dostaw. Podjęto tylko wybrane wątki ogólnej koncepcji badawczej, wskazując celowość prowadzenia dalszych badań, zarówno w obszarze usprawnienia zbudowanego algorytmu, jak i stworzenia jego wersji przetwarzającej dane w sposób równoległy. Problemy te będą przedmiotem dalszych badań autora. Badany problem może podlegać dalszym uogólnienieniom poprzez dodanie niezerowego czasu załadunku w magazynach centralnych, pobieranie towarów od dostawców, itp.

W szczególności, badaniom zostanie poddana funkcja przejścia algorytmu symulowanego wyżarzania. Elementem wspomagającym bardziej efektywne przeszukiwanie przestrzeni rozwiązań może okazać się wprowadzenie kary za nieterminową dostawę. W obecnej wersji algorytmu trasy, dla których dostawa następuje poza oknem czasowym jednego lub więcej klientów nie są w ogóle brane pod uwagę. Akceptacja tego typu tras z uwzględnieniem kary za naruszenie okna czasowego może docelowo spowodować, że uzyskane rozwiązania będą ulepszone.

Kolejnym kierunkiem badań będzie próba wykonania wersji algorytmu, w którym wykorzystuje się przetwarzanie równoległe. Wykonanie funkcji przejścia z zaimplementową równoległą wersją metody zwiększających się promieni na kilku procesorach jednocześnie spowoduje wykonanie większej liczby iteracji algorytmu w tym samym czasie. Obecnie nie jest możliwa implementacja równoległa w języku C/AL dlatego należy stworzyć aplikację implementującą badany algorytm w innym języku programowania jak C++, C#, itp.

Dalsze badania będą również ukierunkowane na stworzenie algorytmu, który będzie w stanie dokonać samoadaptacji w czasie rzeczywistym. Obecnie jakakolwiek zmiana w zapotrzebowaniu przez klientów wymaga uruchomienia algorytmu od początku. Powoduje to wydłużenie całkowitego czasu pracy algorytmu.

Z punktu widzenia praktycznego planowane jest rozszerzenie aplikacji „Planowanie tras” o integrację z urządzeniami nawigacji satelitarnej.

Literatura

1. Ahuja R., Ergun Ö., Orlin J., Punnen A.: A Survey of Very Large Scale Neighborhood Search Techniques. Discrete Applied Mathematics, 123, s. 72–102, 2002.
2. Ai J., Kachitvichyanukul V.: A Particle Swarm Optimization for the Vehicle Routing Problem with Simultaneous Pickup and Delivery. Computer and Operational Research vol. 36(5), s. 1693–1702, Oxford, 2009.
3. Antes J., Derigs U.: A New Parallel Tour Construction Algorithm for the Vehicle Routing Problem with Time Windows. Working Paper, Department of Economics and Computer Science, University of Köln, Germany, 1995.
4. Bachem A., Hochstättler W., Malich M.: The Simulated Trading Heuristic for Solving Vehicle Routing Problems. Disc. App. Math. 65, 47–72, 1996.
5. Backer DE B., Furnon V., Kilby P., Prosser P., Shaw P.: Solving Vehicle Routing Problems Using Constraint Programming and Metaheuristics. J. of Heuristics 6, 501–523, 2000.
6. Baker E. K., Schaffer J. R.: Solution Improvement Heuristics for the Vehicle Routing and Scheduling Problem with Time Window Constraints. Am. J. Math. Mgmt. Sci. 6, 261–300, 1986.
7. Barnes J. W., Carlton W. B.: A Tabu Search Approach to the Vehicle Routing Problem with Time Windows. Presented at the Fall 1995 INFORMS Conference, New Orleans, LA, 1995.
8. Bellman R.: On the Theory of Dynamic Programming. Proceeding of the National Academy of Sciences, USA. 1952.
9. Bent R., Van Hentenryck P.: A Two-Stage Hybrid Local Search for the Vehicle Routing Problem with Time Windows. Technical Report CS-01-06, Department of Computer Science, Brown University, 2001.
10. Berger, Salois J. M., Begin R.: A Hybrid Genetic Algorithm for the Vehicle Routing Problem with Time Windows. Presented at the AI 98 12th Canadian Conference on Artificial Intelligence, Vancouver, Canada, June 1998.
11. Blanton J. L., Wainwright R. L.: Multiple Vehicle Routing with Time and Capacity Constraints using Genetic Algorithms. In Proceedings of the Fifth International Conference on Genetic Algorithms, S. Forrest (eds.). Morgan Kaufmann Publishing, San Francisco, 452–459, 1993.
12. Borgwart K.: The Simplex Algorithm: A Probabilistic Analysis. Algorithms and Combinatorics vol. 1, Springer-Verlag, 1987.

13. Bramel J., Simchi-Levi D.. Probabilistic Analyses and Practical Algorithms for the Vehicle Routing Problem with Time Windows. Opns. Res. 44, 501–509, 1996.
14. Brandão J.: Metaheuristic for the Vehicle Routing Problem with Time Windows. In Meta-heuristics – Advances and Trends in Local Search Paradigms for Optimization, S. Voss, S. Martello, I. H. Osman and C. Roucairol (eds.). Kluwer Academic Publishers, Boston, 19–36, 1999.
15. Brandão J.: A Tabu Search Algorithm for the Open Vehicle Routing Problem. European Journal of Operational Research, 157(3), s. 552–564, 2004.
16. Bräysy O.: A Hybrid Genetic Algorithm for the Vehicle Routing Problem with Time Windows. Licentiate thesis. Department of Mathematics and Statistics, University of Vaasa, Finland, 1999.
17. Bräysy O.: A New Algorithm for the Vehicle Routing Problem with Time Windows Based on the Hybridization of a Genetic Algorithm and Route Construction Heuristics. Proceedings of the University of Vaasa, Research papers 227, Vaasa, Finland, 1999.
18. Bräysy O., Berger J., Barkaoui M.: A New Hybrid Evolutionary Algorithm for the Vehicle Routing Problem with Time Windows, Presented at the Route 2000-Workshop, Skodsborg, Denmark, August 2000.
19. Bräysy O., Berger J., Barkaoui M.: A Parallel Hybrid Genetic Algorithm for the Vehicle Routing Problem with Time Windows. Working paper, Defense Research Establishment Valcartier, Canada, 2001.
20. Bräysy O., Gendreau M.: Metaheuristics for the Vehicle Routing Problemwith Time Windows, Oslo, 2001.
21. Bräysy O.: A Reactive Variable Neighborhood Search for the Vehicle Routing Problem with Time Windows, Oslo, 2001.
22. Bräysy O.: Five Local Search Algorithms for the Vehicle Routing Problem with Time Window, Vaasa, 2001.
23. Carlton W. B.: A Tabu Search Approach to the General Vehicle Routing Problem. Ph.D. Dissertation. Mechanical Engineering Department, University of Texas, Austin, U.S.A, 1995.
24. Caseau Y., Laburthe F.: Heuristics for Large Constrained Vehicle Routing Problems. J. of Heuristics 5, 281–303, 1999.
25. Cérny V.: A thermodynamical approach to traveling salesman problem: an efficient simulation algorithm. Journal of Optimization Theory and Applic., 45, s. 41–45, 1985.

26. Ceselli A., Righini G., Salani M.: A Column Generation Algorithm for a Vehicle Routing Problem with Economies of Scale and Addition Constraint. Italy, 2004.
27. Chabrier A.: Vehicle Routing Problem with Elementary Shortest Path Based Column Generation. Working Paper, ILOG, Madrid, 2003.
28. Chao I.-M., Golden B., Wasil E.: A New Heuristic for the Multi-Depot Vehicle Routing Problem that Improves Upon Best-Known Solutions. Am. J. Math. Mgmt. Sci., 13, s. 371–406, 1993.
29. Chao I.M., Golden B., Wasil E.: A Computational Study of a New Heuristic for the Site-Dependent Vehicle Routing Problem. INFOR, 37(3), s. 319–336, 1999.
30. Chiang W. C., Russell R. A.: Simulated Annealing Metaheuristics for the Vehicle Routing Problem with Time Windows. Annals Opns. Res. 63, 3–27, 1996.
31. Chiang W. C., Russell R. A.: A Reactive Tabu Search Metaheuristic for the Vehicle Routing Problem with Time Windows. INFORMS J. on Computing 9, 417–430, 1997.
32. Chipperfield A. J., Whidborne J. F., Fleming P.: Evolutionary Algorithms and Simulated Annealing for MCDM, 1997.
33. Clarke G., Wright J. W.: Scheduling of Vehicles from a Central Depot to a Number of Delivery Points. Opns. Res. 12, 568−581, 1964.
34. Cordeau J.-F., Gendreau M., Laporte G.: A Tabu Search Heuristic for Periodic and Multi-Depot Vehicle Routing Problems. Networks, 30, s. 105–119, 1997.
35. Cordeau J.-F., Desaulniers G., Desrosiers j., Solomon M., Soumis F.: The VRP with Time Windows, SIAM, Philadelphia, 1999.
36. Cordeau J.-F., Laporte G., Mercier A.: Unified Tabu Search Heuristic for Vehicle Routing Problems with Time Windows. Publication CRT-2000-03. University of Montreal, Canada, 2000.
37. Cordeau J.-F., Desaulniers G., Desrosiers J., Solomon M., Soumis F.: The VRP with Time Windows. Gerad, 2000.
38. Cordeau J.-F., Laporte G.: A Tabu Search Algorithm for the Site Dependent Vehicle Routing Problem with Time Windows. INFOR, 39, s. 292–298, Canada 2001.
39. Cordeau J-F., Desaulniers G., Desrosiers J., Solomon M., Soumis F.: VRP with Time Windows. The Vehicle Routing Problem, volume 9 of SIAM Monographs on DiscreteMathematics and Applications, chapter 7, s. 157–193. SIAM, Philidelphia, 2002.
40. Cordeau J.-F., Gendreau M., Hertz A., Laporte G., Sormany J-S.: New Heuristics for the Vehicle Routing Problem. Technical Report G-2004-33, GERAD, Montreal, Canada, 2004.

41. Cordone R., Wolfler-Calvo R.: A Heuristic for the Vehicle Routing Problem with Time Windows. Internal Report, Department of Electronics and Information, Polytechnic of Milan, Milan, Italy. To appear in J. of Heuristics, 1998.
42. Cormen T. H., Leiserson C. E., Rivest R. L., Stein C.: Introduction to Algorithms, 2nd ed. MIT Press & McGraw-Hill, USA, 2001.
43. Cortes J., Bullo F.: Nonsmooth Coordination and Geometric Optimization via Distributed Dynamical Systems. SIAM Review, 51(1), 2009.
44. Coyle J., Bardi E., Langrey J.: Zarządzanie logistyczne. PWE, Warszawa, 2002.
45. Czech, Z.J.: Parallel simulated annealing for the delivery problem. Proc. of the 9th EuromicroWorkshop on Parallel and Distributed Processing, Mantova, Italy, 219–226, February 7–9 2001.
46. Czech Z. J., Czarnas P.: Parallel simulated annealing for the vehicle routing problem with time windows. Proc. 10th Euromicro Workshop on Parallel, Distributed and Network-based Processing, Canary Islands, Spain, s. 376–383, January 9-11, 2002.
47. Dees A. W., Karger P.: Automated Rip-up and Reroute Techniques. Proceedings of the 19th conference on Design automation, s. 432–439. IEEE Press, 1982.
48. Desrochers M.: Shortest path problems with resource constraints. Technical Report GERAD G-88-27, École des Hautes Études Commerciales, Montréal, 1988.
49. Desrochers M., Soumis F., A column generation approach to the urban transit crew scheduling problem. Transport. Sci., 23, 1–13, 1989.
50. Desrochers J., Dumas Y., Solomon M.M., Soumis F.: Time Constrained Routing and Scheduling, Handbooks in Operations Research and Management Science 8: Network Routing, M.O. Ball, T.L. Magnanti, C.L. Monma, G.L. Nemhauser (eds), 35–139, Elsevier Science Publishers, Amsterdam, 1995.
51. Dorigo M., Gambardella L. M.: Ant colonies for the traveling salesman problem. TR/IRIDIA/1996-3, Belgium, 1996.
52. Dorigo M., Maniezzo V., Colorni A.: The Ant System: Optimization by a colony of cooperating agents. IEEE Transactions on Systems, Man, and Cybernetics–Part B, Vol.26, No.1, pp.1–13 1, 1996.
53. Duda J.: Algorytmy ewolucyjne w planowaniu operacyjnym na przykładzie odlewni żeliwa – praca doktorska. Kraków, 2003.
54. Dueck G.: New Optimization Heuristics: The Great Deluge Algorithm and the Record-to-Record Travel. Technical report, IBM Germany, Heidelberg Scientific Center, 1990.
55. Figliozzi M.A.: The Impact of congestion on commercial vehicle tours and costs. Transportation Research part E, 2009.

56. Figliozzi, M. A., Planning Approximations to the average length of vehicle routing problems with time window constraints. Transportation Research part B, 2009.
57. Frederickson G., Hecht M., Kim C.: Approximation Algorithm for Some Routing Problems. SIAM Journal of Computation, 7(2): 178-193, 1978.
58. Fu Z., Eglese R., Li L.: A Tabu Search Heuristic for the Open Vehicle Routing Problem. Technical Report 2003/042, Lancaster University Management School, Lancaster, United Kingdom, 2003.
59. Fukasawa R., Lysgaard J., Poggi de Aragão M., Reis M., Uchoa E., Werneck R.: Robust Branch-And-Cut-And-Price for the Capacitated Vehicle Routing Problem. Proceedings of IPCO X, Columbia University, 2004.
60. Gambardella L. M., Taillard E., Agazzi G., MACS-VRPTW: A Multiple Ant Colony System for Vehicle Routing Problems with Time Windows. New Ideas in Optimization, D. Corne, M. Dorigo and F. Glover (eds), 63-76, McGraw-Hill, London, 1999.
61. 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, Reports of the Department of Mathematical Information Technology, Series A. Collections, No. A 2/1999, K. Miettinen, M. Mäkelä and J. Toivanen (eds.). University of Jyväskylä, Jyväskylä, 57–64, 1999.
62. Gendreau M., Laporte G., Potvin J-Y.: Metaheuristics for the Capacitated VRP. The Vehicle Routing Problem, volume 9 of SIAM Monographs on Discrete Mathematics and Applications, Chapter 6, s. 129–154. SIAM, Philidelphia, 2002.
63. Glover F.: Multilevel Tabu Search and Embedded Search Neighborhoods for the Traveling Salesman Problem. Working Paper. College of Business & Administration, University of Colorado, Boulder, 1991.
64. Glover F.: New Ejection Chain and Alternating Path Methods for Traveling Salesman Problems. In Computer Science and Operations Research: New Developments in Their Interfaces, O. Balci, R. Sharda, and S. Zenios (eds.). Pergamon Press, Oxford, 449–509, 1992.
65. Golden B., Wasil E., KellyJ., Chao I.-M.: Metaheuristics in Vehicle Routing. Fleet Management and Logistics, s. 33–56, Kluwer, Boston, 1998.
66. Grabowski J., Nowicki E., Smutnicki C.: Metoda blokowa w zagadnieniach szeregowania zadań. Akademicka Oficyna Wydawnicza EXIT, Wydanie 1, Seria: Problemy współczesnej nauki. Teoria i zastosowania. ISBN: 83-87674-49-4. 2003.
67. Gribkovskaia I., Laporte G., Shyshou A.: The single vehicle routing problem with deliveries and selective pickups. Computer and Operational Research, vol. 35(9), Oxford 2008.

68. Homberger J., Gehring H.: Two Evolutionary Meta-heuristics for the Vehicle Routing Problem with Time Windows. INFORMS J. on Computing 37, s. 297–318, 1999.
69. Ibaraki T., Kubo M., Masuda T., Uno T., Yagiura M.: Effective Local Search Algorithms for the Vehicle Routing Problem with General Time Windows. Working Paper, Department of Applied Mathematics and Physics, Kyoto University, Japan, 2001.
70. Irnich S., Villeneuve D.: The Shortest Path Problem with Resource Constraints and k-cycle Elimination for k >= 3. Technical Report G-2003-55, GERAD, Montreal, Canada, September 2003.
71. Jozefowiez N., Laporte G., Semet F.: A multi-objective branch-and-cut algorithm and its application to the multi-modal traveling salesman problem. Lisbon, Portugal, 2010.
72. Jozefowiez N., Laporte G., Semet F.: Traveling salesman and multi-modality. Ajaccio, France, 2010.
73. Jozefowiez N., Laporte G., Semet F.: The multi-modal traveling salesman problem. Tristan VII, Tromso, Norway, 2010.
74. Jozefowiez N., Laporte G., Semet F: A branch-and-cut algorithm for the minimum labeling Hamiltonian cycle problem and two variants. Computers and Operations Research, 2011.
75. Kallehauge B., Larsen J., Madsen O. B. G.: Lagrangean Duality Applied on Vehicle Routing with Time Windows - Experimental Results. Technical Report IMM-REP-2000-8, Informatics and MathematicalModelling, Denmark, 2001.
76. Karaś R., Michlowicz E.: Logistyka w zarządzaniu zmianami technologicznymi na przykładzie okuć okiennych. Finanční a logistické řízení, s. 394–399, Ostrava 2007.
77. Karwat B., Kisiel P., Machnik R., Michlowicz E.: Modelowanie systemu transportu technologicznego przy wykorzystaniu pakietu komputerowego. Zarządzanie w przedsiębiorstwie: XIV międzynarodowa konferencja naukowo-techniczna: Szczyrk 2006.
78. Kilby P., Prosser P., Shaw P.: Guided Local Search for the Vehicle Routing Problem with Time Windows. In META-HEURISTICS Advances and Trends in Local Search Paradigms for Optimization, S. Voss, S. Martello, I. H. Osman and C. Roucairol (eds.). Kluwer Academic Publishers, Boston, 473–486, 1999.
79. Kirkpatrick S., Gellat C.D., Vecchi M.P.: Optimization by simulated annealing. 1983, Science, 220, s. 671-680.
80. Kontoravdis G. A., BARD J. F.: A GRASP for the Vehicle Routing Problem with Time Windows. J. on Computing 7, 10–23, 1995.
81. Krawczyk S.: Metody ilościowe w planowaniu (działalności przedsiębiorstwa) Tom I. Wydawnictwo C.H. Beck, Warszawa, 2001.

82. Krawczyk S.: Metody ilościowe w logistyce (przedsiębiorstwa) Tom II. Wydawnictwo C.H. Beck, Warszawa, 2001.
83. Laarhoven P. J. M., Aarts E.: Simulated Annealing - Theory and Applications Book Description. Springer, 1987.
84. Laporte G., Semet F.: Classical Heuristics for the Capacitated VRP. The Vehicle Routing Problem, volume 9 of SIAM Monographs on Discrete Mathematics and Applications, chapter 5, s. 109–128. SIAM, Philidelphia, 2002.
85. Lau H. C., Sim M., Teo K. M.: Vehicle Routing Problem with Time Windows and a limited Number of Vehicles. Elsevier, 2002.
86. Li H., Lim A., Huang J.: Local Search with Annealing-like Restarts to Solve the VRPTW. Working Paper, Department of Computer Science, National University of Singapore, 2001.
87. Li F., Golden B., Wasil E.: Very Large-Scale Vehicle Routing: New Test Problems, Algorithms, and Results. Computers and Operations Research, 2004.
88. Liu F-H., Shen S-Y.: A Route-neighborhood-based Metaheuristic for Vehicle Routing Problem with Time Windows. Eur. J. Opnl Res. 118, 485–504, 1999.
89. Lysgaard J., Letchford A., Eglese R.: A New Branch-And-Cut Algorithm for the Capacitated Vehicle Routing Problem. Mathematical Programming, Ser. A, s. 423–445, 2004.
90. Łebkowski P.: Algorytm ewolucyjny wyboru sekwencji operacji montażu mechanicznego. Zarządzanie i inżynieria produkcji pod red. M. Zaborowskiego, Warszawa, Wydawnictwa Naukowo-Techniczne, 247–254, 2004.
91. Łebkowski P.: Coloured petri nets for steel plant. CORS-INFORMS International : June 14–17, Toronto, Canada, 2009.
92. Mendoza J.E.: Solving real-world vehicle routing problems in uncertain environments. 4OR: A Quarterly Journal of Operations Research. 2010.
93. Mester D., Bräysy O.: Active Guided Evolution Strategies for Large-Scale Vehicle Routing Problems with Time Windows. Computer and Operation Research. 2004.
94. Mester D., Bräysy O., Dullaert W.: A Multi-parametric Evolution Strategies Algorithm for Vehicle Routing Problems. Working Paper, Institute of Evolution, University of Haifa, Israel, 2005.
95. Metropolis N., Rosenbluth A.W., Rosenbluth M.N., Teller A.H., Teller E.: Equation of state calculation by fast computing machines. Journal of Chem. Phys., 1953, 21, s. 1087–1091.
96. Michlowicz E.: Badania logistycznego systemu transportowego zintegrowanego z systemem produkcyjnym walcowni zimnej blach. Walcownictwo 1999.

97. Michlowicz E.: Badania symulacyjne zintegrowanego systemu transportowo-produkcyjnego walcowni zimnej blach. Hutnik Wiadomości Hutnicze, s. 413-416, 1999.
98. Michlowicz E.: Podstawy logistyki przemysłowej. Uczelniane Wydawnictwa Naukowo-Dydaktyczne, Kraków, 2002.
99. Michlowicz E.: Problem komiwojażera w optymalizacji zadań operatora logistycznego. Zastosowania teorii systemów, s. 203–212, Kraków 2005.
100. Michlowicz E., Zwolińska B.: Problemy tworzenia centrów logistycznych w Polsce. Wybrane Zagadnienia Logistyki Stosowanej, Kraków 2007.
101. Michlowicz E.: Komputerowe wspomaganie zadań operatora logistycznego – zaopatrzenie. Wybrane Zagadnienia Logistyki Stosowanej nr 5, s. 270-278, 2008.
102. Microsoft Corp.: Installation and Configuration in Microsofy Dynamics NAV 2009. 2008.
103. Mladenovic N., Hansen P.: Variable Neighborhood Search. Computers & Opns. Res. 24, 1097–1100, 1997.
104. Nachum O.E., Hadas Y.: Comparison of Two Algorithms for Stochastic Time-Dependent Vehicle-Routing Problem. Transportation Research Board Annual Meeting, s. 10-28, USA, 2010.
105. Nag B., Golden B., Assad A.: Vehicle Routing with Site Dependencies. Vehicle Routing: Methods and Studies, s. 149–159. Elsevier Science Publishers B.V, 1988.
106. Nowicki E.: Metoda tabu w problemach szeregowania zadań produkcyjnych. Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław, 1999.
107. Or I.: Traveling Salesman-Type Combinatorial Problems and their Relation to the Logistics of Regional Blood Banking. Ph.D. Thesis. Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, Illinois, 1976.
108. Osman I. H.: Metastrategy Simulated Annealing and Tabu Search Algorithms for the Vehicle Routing Problems. Annals Opns. Res. 41, 421–452, 1993.
109. Polacek M., Benkner S., Doerner K., Hartl R.: A Cooperative and Adaptive Variable Neighborhood Search for the Multi Depot Vehicle Routing Problem with Time Windows. Operations and Information Systems Vol.1(2), 2008.
110. Potvin J-Y., Rousseau J-M.: A Parallel Route Building Algorithm for the Vehicle Routing and Scheduling Problem with Time Windows. Eur. J. Opnl. Res. 66, 331–340, 1993.
111. Potvin J-Y, Robillard C.:. Clustering for Vehicle Routing with a Competitive Neural Network. Neurocomputing 8, 125–139, 1995.
112. Potvin J-Y., Rousseau J-M.: An Exchange Heuristic for Routing Problems with Time Windows. J. Opnl. Res. Society 46, 1433–1446, 1995.
113. Potvin J-Y, Dube D., Robillard C.: A Hybrid Approach to Vehicle Routing Using Neural Networks and Genetic Algorithms. Applied Intelligence 6, 241-252, 1996.
114. Potvin J-Y., Bengio S.: The Vehicle Routing Problem with Time Windows Part II: Genetic Search. J. on Computing 8, 165–172, 1996.
115. Novoa C., Storer R.: An Approximate Dynamic Programming Approach for the Vehicle Routing Problem with Stochastic Demands. Journal of Operational Research , USA, 2008.

116. Potvin J-Y., Kervahut T., Garcia B. L., Rousseau J-M.: The Vehicle Routing Problem with Time Windows Part I: Tabu Search. J. on Computing 8, 157–164, 1996.
117. Potvin J.-Y., Ichoua S., Gendreau M.: Vehicle dispatching with time-dependent travel times. European Journal of Operational Research 144, 379-396, 2003.
118. Potvin J.-Y., Ying X., Benyahia I.: Vehicle routing and scheduling with dynamic travel times. Computers and Operations Research 33, 1129-1137, 2006.
119. Renaud J., Laporte G., Boctor F.: A Tabu Search Heuristic for the Multi-Depot Vehicle Routing Problem. Computers and Operations Research, 23(3), s. 229–235, 1996.
120. Reeves C. R.: Modern heuristic techniques for combinatorial problems, McGraw-Hill, 1995.
121. Rochat Y., Taillard E.: Probabilistic Diversification and Intensification in Local Search for Vehicle Routing. J. of Heuristics 1, 147–167, 1995.
122. Ropke S., Pisinger D.: A general heuristic for vehicle routing problems. Technical report, Department of Computer Science, University of Copenhagen, 2005.
123. Rousseau L.-M., Gendreau M., Pesant G.: Using Constraint-Based Operators to Solve the Vehicle Routing Problem with Time Windows. Working Paper. Centre for Research on Transportation, University of Montreal, Canada. Journal of Heuristics, 2000.
124. Russell R.: Hybrid Heuristics for the Vehicle Routing Problem with Time Windows. Trans. Sci. 29, 156–166, 1995.
125. Russell R.: An Effective Heuristic for the M-tour Traveling Salesman Problem with Some Side Conditions. Opns. Res. 25, 517–524, 1997.
126. Russell S., Norvig P.: Artificial Intelligence, a Modern Approach. Rearson Education Inc., New Jersey, 2003.
127. Salani M.: Improved Dynamic Programming for the Vehicle Routing Problem with Time Windows. Italy, 2006.
128. Salani M.: Branch-and-Price Algorithms for Vehicle Routing Problem. Milano 2006.
129. Sariklis D., Powell S.: A Heuristic Method for the Open Vehicle Routing Problem. Journal of the Operational Research Society, 51, s. 564–573, 2000.
130. Savelsbergh M. W. P.: The Vehicle Routing Problem with Time Windows: Minimizing Route Duration.ORSA Journal on Computing 4, 1992, 146-154.
131. Schrimpf G., Schneider J., Stamm-Wilbrandt H., Dueck G.: Record Breaking Optimization Results Using the Ruin and Recreate Principle. Journal of Computational Physics 159, 139-171, 2000.
132. Schulze J., Fahle T.: A Parallel Algorithm for the Vehicle Routing Problem with Time Window Constraints. Annals Opns. Res. 86, 585–607, 1999.
133. Shaw P.: A New Local Search Algorithm Providing High Quality Solutions to Vehicle Routing Problems. Working Paper. Department of Computer Science, University of Strathclyde, Glasgow, Scotland, 1997.
134. Shaw P.: Using Constraint Programming and Local Search Methods to Solve Vehicle Routing Problems. In Principles and Practice of Constraint Programming – CP98, Lecture Notes in Computer Science, M. Maher and J.-F. Puget (eds.). Springer-Verlag, New York, 417–431, 1998.
135. Solomon M.: Algorithms for the Vehicle Routing and Scheduling Problem with Time Windows Constraints. Oper. Res., 1987, 35, s. 254-265.

136. Solomon M., Desrosiers J.: Time windows constrained routing and scheduling problems. Transp. Sci., 1988, 22, s. 1–13.
137. Stawowy A.: Wspomagana kmputerowo optymalizacja kontroli jakości. Materiały z V Krajowej Konferencji Informatyków. Rozwój metod i zastosowań informatyki. Poznań, 1988, 1989.
138. Stawowy A., Mazur Z., Obrzud J.: Zastosowanie algorytmu genetycznego w grupowaniu wyrobów, Mechanika, tom 15, zeszyt 2, Wydawnictwa AGH, Kraków 1996.
139. Stawowy A.: Algorytm ewolucyjny do problemu pakowania, XII Międzynarodowe Sympozjum "Zastosowania teorii systemów", Zakopane 1997.
140. Stawowy A., Duda J.: Evolutionary based system for production scheduling In foundry, Archives of Foundry Engineering, Vol. 8, Iss. 3, 2008, str. 99–104.
141. Taillard E.: Parallel Iterative Search Methods for Vehicle Routing Problems. Networks, 23, s. 661–673, 1993.
142. Taillard E., Badeau P., Gendreau M., Guertin F., Potvin J-Y.: A Tabu Search Heuristic for the Vehicle Routing Problem with Soft Time Windows. Trans. Sci. 31, 170–186, 1997.
143. Tang J., Pan Z., Fung R., Lau H.: Vehicle Routing Problem with Fuzzy Time Windows. Fuzzy Sets and Systems vol. 160(5), s. 683–695, Amsterdam 2009.
144. Thangiah Sam R., Kendall N., Juell P.: GIDEON: A Genetic Algorithm System for Vehicle Routing Problems with Time Windows. Proceedings of the Seventh IEEE Conference on Artificial Intelligence Applications, 1991, Miami, Florida, 322–328.
145. Thangiah S., Osman I., Sun T.: Hybrid Genetic Algorithm, Simulated Annealing and Tabu Search Methods for Vehicle Routing Problems with Time Windows. Working Paper UKC/IMS/OR94/4. Institute of Mathematics and Statistics, University of Kent, Canterbury, 1994.
146. Thangiah S.:Vehicle Routing with Time Windows Using Genetic Algorithms. In Application Handbook of Genetic Algorithms: New Frontiers, Volume II, L. Chambers (eds.). CRC Press, Boca Raton, 253–277, 1995.
147. Thangiah S. R., Osman I. H., Vinayagamoorthy R., Sun T.: Algorithms for the Vehicle Routing Problems with Time Deadlines. Am. J. Math. Mgmt. Sci. 13, 323–355, 1995.
148. Thangiah Sam R.: A Hybrid Genetic Algorithms, Simulated Annealing and Tabu Search Heuristic for Vehicle Routing Problems with Time Windows. Practical Handbook of Genetic Algorithms, Volume III: Complex Structures, L. Chambers (Ed.), CRC Press, 347–381, 1999.
149. Thompson P. M., Psaraftis H. N.:. Cyclic Transfer Algorithms for Multivehicle Routing and Scheduling Problems. Opns. Res. 41, 935–946, 1993.
150. Toth P., Vigo D.: The Vehicle Routing Problem. Philadelphia, 2001.
151. Vanderbilt D., Louie S. G.: A Monte Carlo Simulated Annealing Approach to Optimization over Continuous Variables. J. Comp. Physics 56(1984), 259–271, 1984.
152. Van Landeghem H. R. G.: A Bi-criteria Heuristic for the Vehicle Routing Problem with Time Windows. Eur. J. Opns. Res. 36, 217–226, 1988.
153. Voudouris C.: Guided Local Search for Combinatorial Problems. Ph.D. thesis. Department of Computer Science, University of Essex, Colchester, UK, 1997.
154. Voudouris C., Tsang E.: Guided Local Search. Eur. J. Opns. Res. 113, 80–119, 1998.

155. Wiens E. G.: Simplex Algoritm - Demonstartes Algorithm in Detail. UCLA, Departmentof Econimics, 1980.
156. Woch M.: Rozwiązanie problemu dostaw z oknami czasowymi za pomocą symulowanego wyżarzania. Studia Informatica 2(58), s. 67-81, Gliwice 2004.
157. Woch M., Łebkowski P.: Manufacturing Cost Management in Microsoft Dynamics NAV 5.0 – Case Study. XI Konferencja Logistyki Stosowanej "Total Logistic Management", Zakopane 6-8.12.2007.
158. Woch M.: Microsoft SQL Server Optimization for Microsoft Dynamics NAV. Studia Informatica Volume 28, Number 2 (71), s. 17-29, Gliwice 2007.
159. Woch M.: Administrator's Tools in Microsoft SQL Server Optimization for Microsoft Dynamics NAV. Studia Informatica Volume 28, Number 3A (72), s. 29–42, Gliwice 2007.
160. Woch M.: Przegląd badań wybranych problemów optymalizacyjnych. Studia Informatica Volume 28, Number 4 (74), s. 81-93, Gliwice 2007.
161. Woch M.: Zarządzanie kosztami produkcji w systemie Microsoft Dynamics NAV 4.0. Komputerowo Zintegrowane Zarządzanie pod red. Ryszarda Knosali, Oficyna Wydawnicza Polskiego Towarzystwa Zarządzania Produkcją, T. 2, s. 547–551, Opole 2008.
162. Woch M.: Zarządzanie kosztami zapasów w systemie zintegrowanym klasy ERP Microsoft Dynamics NAV 4.0. Komputerowo Zintegrowane Zarządzanie pod red. Ryszarda Knosali, Oficyna Wydawnicza Polskiego Towarzystwa Zarządzania Produkcją, T. 2, s. 552–557, Opole 2008.
163. Woch M., Łebkowski P.: Manufacturing Cost Management in Microsoft Dynamics NAV 5.0 – Case Study. 2008.
164. Woch M., Łebkowski P.: Rozwiązanie problemu dostaw z oknami czasowymi w systemie Microsoft Dynamics NAV 5.0. XI Międzynarodowa Konferencja Naukowa Zarządzanie Przedsiębiorstwem – Teoria i Praktyka, Kraków 27-28.11.2008.
165. Woch M., Łebkowski P.: Sequential Simulated Annealing for the Vehicle Routing Problem with Time Windows.Decision Making in Manufacturing and Services, vol. 3, no. 1-2., s. 87–100, 2009.
166. Woch M., Łebkowski P.: Symulowane wyżarzanie w rozwiązywaniu dwukryterialnych problemów optymalizacyjnych. Współczesne problemy zarządzania przedsiębiorstwem — Konferencja Międzynarodowa, Szczyrk, 10-12.06.2010.
167. Woch M., Łebkowski P.: Zastosowanie algorytmu symulowanego wyżarzania do rozwiązania problemu dostaw z oknami czasowymi. Logistyka, ISSN 1231-5478, nr 2 s. 1–9, Kraków, 2010.
168. Woch M., Łebkowski P.: Algorytm dwukryterialnej optymalizacji w rozwiązywaniu problemów dostaw. Zarządzanie produkcją – planowanie, wytwarzanie, optymalizacja i kontrola = Management of production – planning, production, optimization and control : monografia, red. nauk. Ryszard Barcik, Marek Dudek, Wiesław Waszkielewicz, Wydawnictwo Akademii Techniczno-Humanistycznej, ISBN 978-83-62292-76-9, s. 151–164, Bielsko-Biała, 2010.