Uniwersytet Ekonomiczny
we Wrocławiu

KIERUNEK STUDIÓW

INFORMATYKA W BIZNESIE

Marcin Żerko

Nr albumu 175694

PRACA MAGISTERSKA

Zastosowanie metaheurystyk do rozwiązywania problemu planowania projektów z wieloma wymaganymi umiejętnościami i ograniczonymi zasobami

Promotor:

dr hab. prof. UEW Helena Dudycz
Katedra Technologii Informacyjnych

WROCŁAW 2022

Application of metaheuristics to solve multi-skill resource-constrained project scheduling problem

The aim of this study is the examination of the use of metaheuristics to solve the multi-skill resource-constrained project scheduling problem, which itself is an NP-hard problem. In the first chapter, a computational problem and its elements are defined. Then, in the second chapter, an overview is made of the computational methods that are compared in the study. The third chapter presents the research assumptions and describes the manner of their implementation. In the fourth chapter, the results of the study are presented, along with their analysis. Based on the results of this experiment, it is shown that the metaheuristic methods analyzed in this work can be used to generate satisfactory solutions to the problem.

Spis treści

Wstęp 5

1. Problemy obliczeniowe w realizacji projektów 6

1.1. Zdefiniowanie projektu 6

1.2. Problem spełnialności 8

1.3. Problem komiwojażera 9

1.4. Programowanie nieliniowe 11

1.5. Problem planowania projektu z wieloma wymaganymi umiejętnościami i ograniczonymi zasobami 12

1.6. Elementy problemu 14

1.6.1. Model i wielkość przestrzeni poszukiwań 14

1.6.2. Sąsiedztwo 15

1.6.3. Funkcja oceny rozwiązania, cel i ograniczenia rozwiązania problemu 17

2. Metody heurystyczne do rozwiązywania problemów 20

2.1. Przegląd metod heurystycznych do rozwiązywania problemów 20

2.2. Algorytm zachłanny 21

2.3. Przeszukiwanie lokalne 23

2.4. Symulowane wyżarzanie 24

2.5. Algorytm genetyczny 26

3. Założenia realizacji badania 28

3.1. Cel badania i zastosowana procedura badania 28

3.2. Przyjęte założenia dla eksperymentu 29

3.3. Opis sposobu realizacji eksperymentu 30

4. Wyniki i wnioski z przeprowadzonego eksperymentu 32

4.1. Wyniki z przeprowadzonego eksperymentu 32

4.2. Wizualizacja otrzymanych harmonogramów rozwiązań w badaniach 52

4.3. Wnioski z przeprowadzonego eksperymentu 54

Zakończenie 56
Literatura 57
Spis tabel 60
Spis rysunków 62
Spis równań 63
Oświadczenia 64

Wstęp

Problem planowania projektu z wieloma wymaganymi umiejętnościami i ograniczonymi zasobami składa się listy zadań, które muszą zostać wykonane w projekcie, wraz z listą zasobów, które zadania te mogą wykonywać. Każdy z zasobów ma swoją cenę, a także posiada określone umiejętności. Każde zaś zadanie, wymaga określonej umiejętności na określonym minimalnym poziomie, oraz może mieć zadania, które muszą zostać wykonane przed jego rozpoczęciem. Celem w tym problemie jest stworzenie takiego harmonogramu, który zająłby najmniej czasu, kosztował by najmniej lub spełniał naraz oba te warunki, jak najlepiej.

Niestety jest to problem należący do grupy problemów NP-trudnych. Oznacza to, że przegląd zupełny rozwiązań i wybranie tego, które najlepiej spełnia określone wymagania jest niemożliwy ze względu na ograniczenia czasowe. Dlatego celem tej pracy jest sprawdzenie możliwości zastosowania metaheurystyk do rozwiązywania problemu planowania projektów z wieloma wymaganymi umiejętnościami i ograniczonymi zasobami, czyli generacji satysfakcjonujących harmonogramów w satysfakcjonującym czasie.

Niniejsza praca składa się z czterech rozdziałów, wstępu, zakończenia, spisu literatury, tabel oraz rysunków.

W pierwszym rozdziale zostaje na początku przytoczona definicja projektów, następnie opisany zostaje problem obliczeniowy wraz z innymi przykładowymi, oraz zostają zdefiniowane ich elementy, takie jak model, definicja sąsiedztwa, czy funkcja oceny rozwiązania, jego cel i ograniczenia. W drugim rozdziale zostają zdefiniowane metody heurystyczne do rozwiązywania tego typu problemów, początkowo zostaje przytoczona ogólna ich definicja, a następnie przestawione są cztery konkretne metaheurystyki. W rozdziale trzecim omówiono założenia realizacji badania, jego cel, zastosowana procedura badawcza, oraz przyjęte założenia i opis sposobu realizacji eksperymentu. Zaś w rozdziale czwartym, przestawione zostają wyniki i wnioski z przeprowadzonego eksperymentu, wraz z wizualizacja przykładowych otrzymanych harmonogramów. Na koniec w zakończeniu zostaje podsumowanie badania, wraz z dalszymi możliwościami jego kontynuacji.

Praca powstała przy wykorzystaniu dostępnych tradycyjnych źródeł literatury w postaci opracowań zwartych, artykułów naukowych, fragmentów książek, standardów technicznych oraz źródeł internetowych. Realizując badanie empiryczne zastosowano metodę badawczą eksperymentu.

1. Problemy obliczeniowe w realizacji projektów

1.1. Zdefiniowanie projektu

Zarządzanie projektami jako obszar nauki jest stosunkowo młode. Jednakże, różnego typu projekty istniały już w czasach starożytności. Wszak prawie każdy zbiór czynności wykonywanych w określonym celu, można nazwać projektem. Przykładem mogą być przedsięwzięcia budowlane takie jak budowa piramid w Egipcie (Pietras i Szmit, 2003). Dzisiaj pojęcie projektu sięga znacznie szerzej, ponieważ każde przedsiębiorstwo jest niejako jednym dużym projektem, jednak jego istota pozostała niezmienna.

Pojęcie projektu i zarządzania nim w sferze biznesowej pojawiło się w skutek szeroko pojętych zmian w sposobie działania firm. Przedsiębiorstwa musiały dostosować się do szybko zmieniającego się globalnego rynku, liczniejszej konkurencji oraz szybkiego rozwoju technologii (Pietras i Szmit, 2003). To wszystko wpłynęło na konieczność dokładnego określenia czym jest projekt. Definiuje się go w następujący sposób (Trocki, 2012):

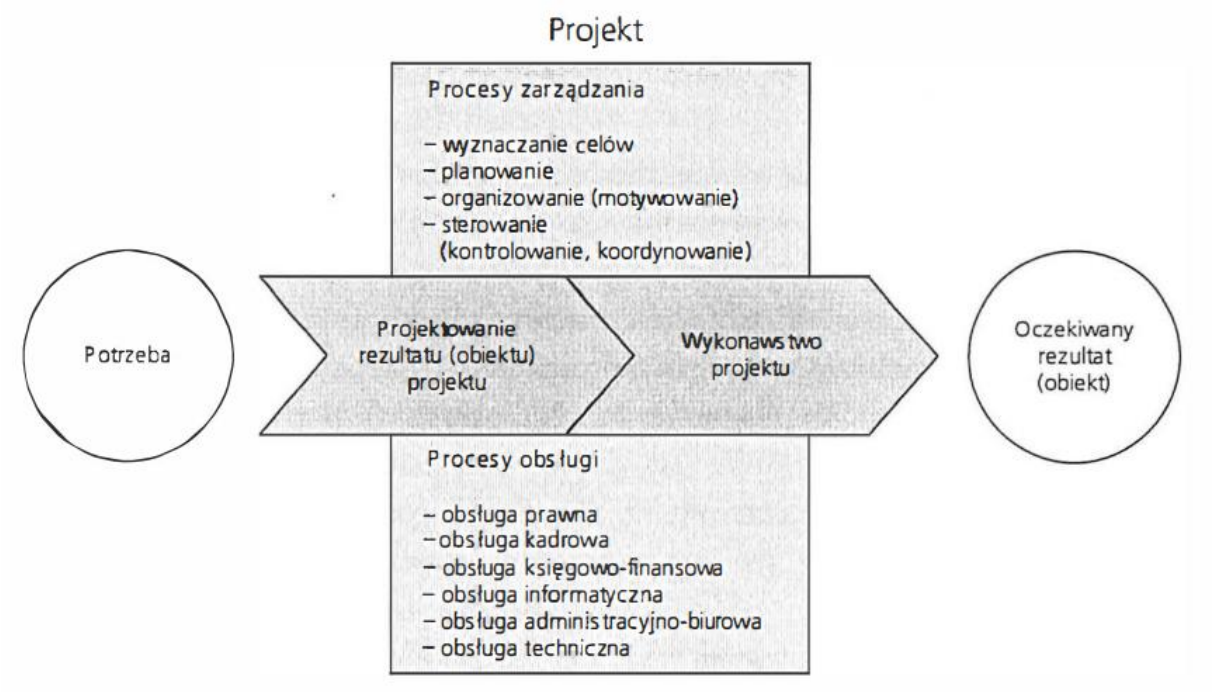
[Projekty] są to niepowtarzalne przedsięwzięcia o wysokiej złożoności, określone co do okresu ich wykonania – z wyróżnionym początkiem i końcem – wymagające zaangażowania znacznych, lecz limitowanych środków (rzeczowych, ludzkich, finansowych, informacyjnych), realizowane przez zespół […].

Ta definicja skupia się na paru warunkach, które trzeba spełnić, aby dane przedsięwzięcie móc nazwać projektem. Dotyczą one: wymiaru czasowego (określony początek oraz koniec), rzeczowego (skończona ilość różnego rodzaju środków) oraz celowości (punkt, w którym projekt jest uznany za zakończony). Inną definicją zbieżną z tymi warunkami jest (Pietras i Szmit, 2003):

Można go [projekt] zdefiniować jako zbiór działań podejmowanych dla zrealizowania określonego celu i uzyskania konkretnego, wymiernego rezultatu. Często rezultat projektu nazywany jest produktem projektu. […] produkt definiuje wymiar techniczny, czasowy i finansowy projektu.

Według tej definicji, to produkt projektu, który ma powstać w skutek zrealizowania postawionych celów, pozwala określić przedstawione wyżej warunki realizacji przedsięwzięcia, takie jak jego początek i koniec czy przeznaczone na niego środki finansowe.

Podkreślane jest znaczenie realizacji zdefiniowanego celu w projektach (Wyrozębski, Juchniewicz i Metelski, 2012), bowiem wypełnienie go świadczy o sukcesie bądź porażce podjętej inicjatywy. Realizację celu projektu można rozpatrywać w następujących parametrach: jakość produktu końcowego i poziom spełnienia przez niego wymagań, czas realizacji oraz jej koszt. Aby parametry te zostały spełnione na satysfakcjonującym poziomie, konieczne jest skuteczne zarządzanie procesami składającymi się ich realizację. Można zaliczyć do nich widoczne na poniższym rysunku procesy: operacyjne (projektowanie i wykonawstwo projektu), zarządzania oraz obsługi.



Rysunek 1: Rodzaje działań związanych z realizacją projektów

Źródło: (Grucza, Trocki, Bukłaha, Juchniewicz i Wyrozębski, 2009)

Procesy te mają miejsce na każdym etapie cyklu życia projektu. Ów cykl składa się zazwyczaj z następujących faz (Trocki, 2012):

1. Studia wstępne, które mają na celu zdefiniowanie wymagań, problemów, określenie kierunków rozwiązań, nakreślenie wstępnej struktury projektu oraz jego kosztu i czasu trwania.
2. Analiza systemowa, składająca się z przeglądu zadań, funkcji systemu i podsystemów oraz badania otoczenia systemu.
3. Planowanie systemu czyli określenie jego specyfikacji i koncepcji, ponadto sporządzenie dokumentów (planów, rysunków) i zdefiniowanie etapów wdrożenia.
4. Realizacja systemu, której celem jest stworzenie oraz przetestowanie prototypu.

5. Wdrożenie systemu. Podczas tej fazy system jest odbierany oraz uruchamiany w środowisku użytkownika.
6. Eksploatacja systemu.

Jednymi z najtrudniejszych etapów w tym cyklu są etapy początkowe, które opierają się na analizie obecnego stanu oraz możliwości jego poprawy i zaplanowania całego projektu. Wymaga to u odpowiedzialnych za to osób bardzo rozwiniętych umiejętności analitycznych oraz pomysłowości. Jednakże, czasem problemy napotkane na tym etapie mogą być trudne do rozwiązania, zarówno dla człowieka, jak i dla komputerów. Powodów takiego stanu rzeczy może być kilka (Garey i Johnson, 1979), między innymi:

- Liczba potencjalnych rozwiązań dla danego problemu jest tak duża, że nie jest możliwe przeszukanie wszystkich rozwiązań w zadowalającym nas czasie. Niekiedy mogłoby to zająć miliony lat dla niektórych problemów, ale nawet jak zajmuję tylko parę minut, to może być zbyt długim oczekiwaniem – przykładowo przy wyznaczaniu nowej trasy dla nawigacji samochodowej.
- Problem jest tak skomplikowany, że modele, które używamy, są zbyt uproszczone, aby dać sensowny rezultat – przykładowo trudno jest przewidzieć pogodę na rok do przodu, gdyż jest tak wiele zmiennych, że jest to praktycznie niemożliwe do zrobienia przy zachowaniu jakiejkolwiek dokładności i idącej za tym użyteczności dla takiej prognozy.
- Ograniczenia, które są nałożone na rozwiązania są tak skomplikowane, że problematyczne jest w ogóle stworzenie takiego, które było by prawidłowe i nie łamało żadnych zasad – przykładowo tylko jeden klucz może odszyfrować poprawnie zaszyfrowane dane.

W kolejnych punktach tego rozdziału zostaną przedstawione trzy problemy: problem spełnialności, problem komiwojażera i programowanie nieliniowe. Natomiast w ostatnim, problem planowania projektu z wieloma wymaganymi umiejętnościami i ograniczonymi zasobami.

1.2. Problem spełnialności

Jednym z podstawowych zagadnień rachunku zdań w logice jest problem spełnialności (Boolean satisfiability problem, skrót SAT). Polega on na znalezieniu takich wartości

zmiennych, które mogą przyjmować tylko wartości prawda lub fałsz, dla których dana formuła logiczna będzie spełniona (Vizel, Weissenbacher i Malik, 2015). Przykładowy fragment takiej formuły może wyglądać następująco:

$$F(x) = (x_{56} \vee \bar{x}_{18}) \wedge (x_1 \vee \bar{x}_{99} \vee \bar{x}_{72}) \wedge (\bar{x}_{42} \vee x_{13} \vee \bar{x}_{37} \vee \bar{x}_{56}) \wedge \dots$$

Problem ten, tak samo jak kolejne przedstawione w tej pracy, był pierwszym który udowodniono, że należy do grupy problemów NP-zupełnych (Cook, 1971). Oznacza to, że:

- Najlepsze rozwiązanie nie może zostać znalezione w czasie wielomianowym.
- Sprawdzenie, czy dane rozwiązanie jest poprawne jest możliwe w czasie wielomianowym.

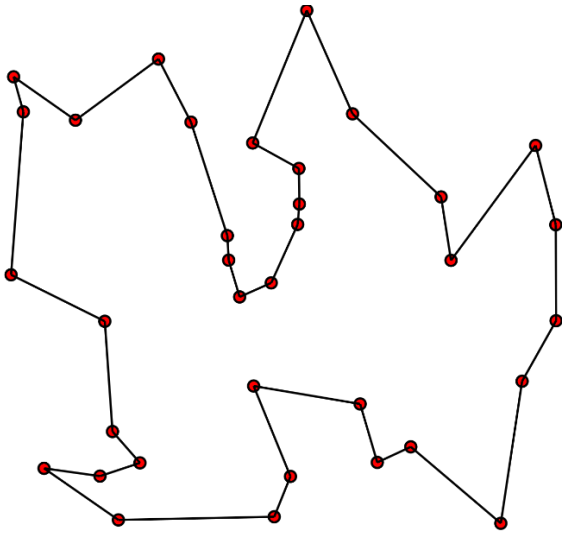
Tak długo, jak formuła nie jest skomplikowana, to możemy sprawdzić po prostu każdą możliwą opcję. Jednak, jak łatwo zauważyć, liczba kombinacji rośnie wykładniczo – dla każdej zmiennej są możliwe dwa stany, więc potencjalnych rozwiązań jest tyle ile dwa do potęgi. Rośnie to na tyle szybko, że już dla 50 takich zmiennych, liczba możliwych stanów jest większa niż biliard. Już taka ilość jest bardzo trudna do sprawdzenia nawet w przypadku pomocy komputerów, a dla ludzi bez nich dosłownie niemożliwa.

Jednym z najczęściej spotykanych obecnie zastosowań dla silników rozwiązujących ten problem są zagadnienia związane z projektowaniem układów cyfrowych. Dzięki temu można chociażby znaleźć bardziej optymalne rozłożenia komponentów w układach scalonych (Nam, Sakallah i Rutenbar, 2002), oraz przeprowadzić formalną weryfikację mikroprocesorów potokowych (Bryant, German, Velev i Murray, 1999). Jak widać jest to problem spotykany obecnie powszechnie w trakcie realizacji projektów. I właśnie dlatego konieczne było opracowanie takich metod heurystycznych, które pozwalają na znalezienie odpowiednio dobrego rozwiązania takiego problemu w akceptowalnym czasie.

1.3. Problem komiwojażera

W problemie komiwojażera (travelling salesman problem, skrót TSP) mamy zdefiniowaną listę miast i odległości pomiędzy każda parą z nich (Beardwood, Halton i Hammersley, 1959). Polega on na znalezieniu najkrótszej możliwej takiej trasy, która odwiedza każde miasto

dokładnie raz, a na sam koniec wraca do miejsca początkowego. Jest to szczególny przypadek problemu marszrutyzacji (Dantzig i Ramser, 1959), który pozwala na odwiedzenie każdego miasta więcej niż raz. Przykładowe rozwiązanie przedstawiono na poniższym rysunku.



Rysunek 2: Przykładowa ścieżka dla TSP

Źródło: (Michalewicz i Fogel, 2004)

Optymalnych rozwiązań dla problemu jest zawsze więcej niż jeden. Nawet jeżeli istnieje tylko jeden optymalny cykl dla takiego grafu, to zawsze możemy wygenerować nowe rozwiązanie, zaczynając podróż z innego miasta, a także można je odwrócić. Przykładowo, jeżeli przedstawimy rozwiązanie jako listę odwiedzanych miast, to jeżeli pierwsza była by optymalnym rozwiązaniem, to wszystkie następujące po niej także takie będą:

- 1-3-5-2-4.
- 3-5-2-4-1.
- 5-2-4-1-3, itd.

Występują w tym problemie pewnie uproszczenia w stosunku do prawdziwego życia. Jednym z nich jest fakt, że w rzeczywistości nie każde miasto musi mieć pomiędzy sobą bezpośrednią drogę. Dodatkowo nie zawsze koszt pokonania takiej ścieżki jest taki sam w obie strony. Może on także zależeć od godziny, w jakiej dana podróż się odbywa.

Problem ten został pierwszy raz zdefiniowany w latach 30 i jest jednym z najczęściej używanym punktów odniesienia dla nowo powstających metod optymalizacji. Ze względu na jego popularność, mimo tego, że jest on NP-kompletny, istnieje wiele algorytmów i heurystyk które pozwoliły wygenerować najlepsze możliwe rozwiązania nawet dla przypadków składających się z dziesiątek tysięcy miast. Często używanym zbiorem instancji problemu o

rożnym poziomie trudności jest TSPLIB, a największym całkowicie rozwiązanym przypadkiem jest problem składający się z 85 900 miast (Rego, Gamboa, Glover i Osterman, 2011). Dzięki powstaniu takiego zbioru danych, jest możliwość skutecznego porównania ze sobą dwóch, lub większej ilości metod optymalizacyjnych ze sobą, a następnie wybór takiej, która jest najbardziej optymalna dla danego przypadku.

1.4. Programowanie nieliniowe

Kolejna klasą problemów, są te związane z programowaniem nieliniowym (Nonlinear programming, skrót NLP). Polegają one na zalezieniu minimum, maksimum, lub punktów zerowych dla danej funkcji (Bazaraa i Shetty, 1979). Dodatkowo na przestrzeń rozwiązań mogą zostać nałożone ograniczenia, w postaci równości, lub nierówności, które muszą zostać spełnione, aby dane rozwiązanie było poprawne.

Zmienne mogą przyjmować dowolne wartości z zakresu liczb rzeczywistych. W związku z tym istnieje nieskończoność potencjalnych rozwiązań – ze względu na to, że zawsze możemy podać taką wartość, której jeszcze nie rozwiązywaliśmy. Nie każde z tych rozwiązań za to musi być optymalne lub w ogóle poprawne. W szczególności możemy dla danego problemu nie wiedzieć, czy w ogóle istnieje takie rozwiązanie które spełnia wszystkie podane ograniczenia i dana metoda może próbować znaleźć jakiekolwiek które będzie poprawne.

Przykładowym problemem może być znalezienie maksimum dla następującej funkcji (Keane, 1996), z następującymi ograniczeniami:

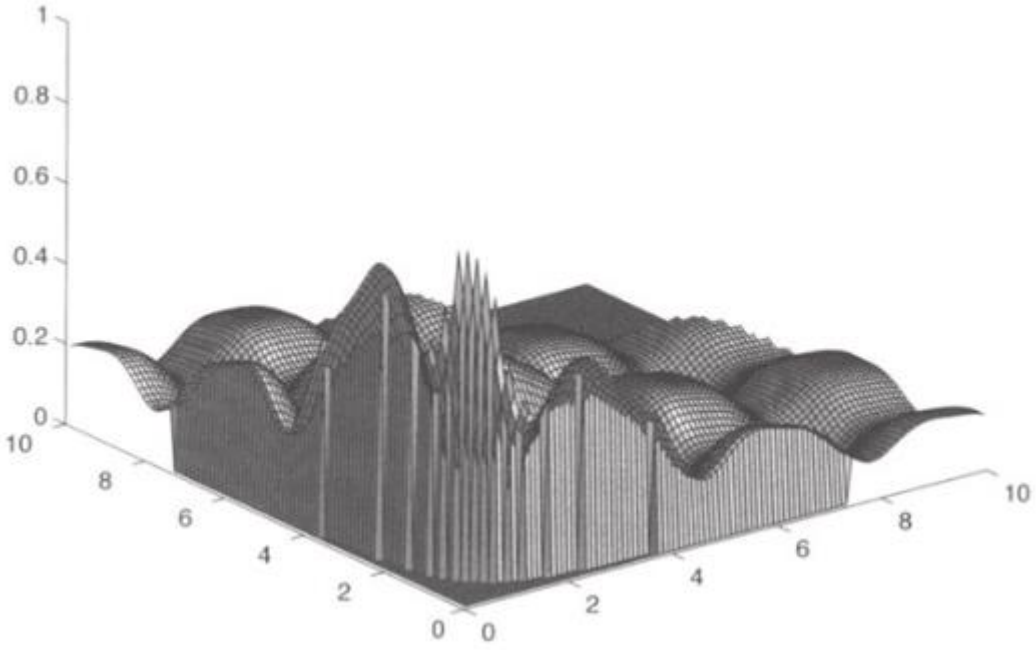
$$G2(X) = \left| \frac{\sum_{i=1}^{n} cos^4(x_i) - 2\prod_{i=1}^{n} cos^2(x_i)}{\sqrt{\sum_{i=1}^{n} ix_i^2}} \right|$$

$$\prod_{i=1}^{n} x_i \geq 0.75, \qquad \sum_{i=1}^{n} x_i \leq 7.5n, \qquad 0 \leq x_i \leq 10, \qquad 1 \leq i \leq n$$

Równanie 1: Przykładowa funkcja dla programowania nieliniowego

Źródło: (Keane, 1996)

Możemy zwizualizować tą funkcję przykładowo dla $n = 2$ (Michalewicz i Fogel, 2004). Rozwiązaniom niespełniającym ograniczeń nadano zerową wartość (Rysunek 3).



Rysunek 3: Wykres przykładowej funkcji dla programowania nieliniowego

Źródło: (Michalewicz i Fogel, 2004)

Jak widać powierzchnia tego wykresu jest bardzo nieregularna, zwłaszcza im bliżej początków osi. Dodatkowo możemy zaobserwować tutaj nagły spadek do zera, co oznacza niepoprawne rozwiązania. Jest on bardzo blisko fragmentu gdzie znajdują się najlepsze rozwiązania według tego wykresu, co pokazuje, że granica pomiędzy globalnym maksimum, a nieprawidłowym rozwiązaniem może być bardzo cienka. W związku z tym niektóre metody rozwiązań koncentrują się właśnie na tych krawędziach, uważając je za rejony z największym potencjałem. Oczywiście wszystko to zależy od konkretnego przypadku dla danego równania.

1.5. Problem planowania projektu z wieloma wymaganymi umiejętnościami i ograniczonymi zasobami

Zbiór danych dla problemu planowania projektów z wieloma wymaganymi umiejętnościami i ograniczonymi zasobami (Multi-Skill Resource-Constrained Project Scheduling Problem, skrót MSRCPSP) został zdefiniowany przez naukowców związanych z

Politechniką Wrocławską. Ich celem było stworzenie takiego problemu, która miał by jednoczesny balans prostoty implementacji i wierności rzeczywistej sytuacji (Skowroński, Myszkowski i Podlodowski, 2013).

Celem klasycznego problemu planowania projektów było przydzielenie zasobów do zadań w taki sposób, aby zminimalizować czas i/lub koszt wykonania projektu. Jednakże pojawiają się w nim także ograniczenia (Santos i Tereso, 2011):

- Niektóre zadania, mogą do rozpoczęcia pracy nad nimi wymagać zakończenia innych zadań.
- Dany zasób może zostać przydzielony tylko do jednego zadania naraz i musi je skończyć w całości, bez możliwości dzielenia pracy pomiędzy dwoma rozpoczętymi zadaniami.

Problem ten został rozszerzony przez dodanie umiejętności. Każdy zasób posiada pewien zbiór umiejętności na określonym poziomie, a każde zadanie wymaga jedną umiejętność na określonym poziomie. W związku z tym zadanie może mieć przydzielony dany zasób, tylko jeżeli posiada on daną umiejętność na równym lub wyższym poziomie niż to potrzebne.

Został on opracowany razem ze współpracy z inżynierami z firmy Volvo (Myszkowski, Skowroński i Sikora, 2015). Dzięki temu jego twórcy dostarczają oparte na rzeczywistości, ale zanonimizowane, zestawy danych odpowiadające rzeczywistym projektom. Jest ich łącznie 42: tzn. 6 łatwiejszych problemów i 36 pełnowymiarowych, które posiadają do:

- 200 zadań.
- 150 relacji pomiędzy zadaniami.
- 15 umiejętności na różnych poziomach.
- 40 zasobów.

Różne kombinacje tych ilości pozwalają na sprawdzenie metod obliczeniowych w różnych wypadkach. Przykładowo, jedna z metod może sobie radzić lepiej, gdy będzie więcej relacji pomiędzy zadaniami, a druga gdy będzie więcej zasobów

1.6. Elementy problemu

1.6.1. Model i wielkość przestrzeni poszukiwań

Aby użyć jakikolwiek algorytm lub heurystykę do rozwiązania danego problemu musimy najpierw zdefiniować pewne podstawowe koncepty dla danego problemu:

- Model dla problemu.
- Sąsiedztwo rozwiązań.
- Funkcję oceny rozwiązania.
- Cel, który chcemy osiągnąć.
- Potencjalne ograniczenia nałożone na rozwiązania.

Pierwsza rzeczą, którą trzeba zdefiniować w celu rozwiązania danego problemu obliczeniowego, jest zdefiniowanie jego modelu. Jest to określenie sposobu, w jaki możemy przedstawić alternatywne rozwiązania i umożliwić nam ich modyfikację. Nie istnieje jeden najlepszy model dla danego problemu. Jak udowodnili (Fogel i Ghozeil, 1997) wśród takich przedstawień problemu, które są dla siebie bijekcjami, żadna z nich nie daje przewagi w rezultatach nad innymi. W związku z tym najczęściej wybierane są takie, które są najbardziej intuicyjne dla danego problemu, co pozwala na ich łatwiejsze zrozumienie.

Przypadek SAT jest najprostszy do zamodelowania. Jego rozwiązaniem jest ciąg binarnych wartości, reprezentujących po kolei stany jakie przypisujemy poszczególnym zmiennym. W związku z tym możliwych rozwiązań jest dokładnie 2^n. Dla 100 zmiennych jest to wartość rzędu 10^{30}.

W przypadku TSP najczęściej spotykanym modelem jest permutacja liczb naturalnych od 1 do n. Każda z liczb jest przypisana do konkretnego miasta i ich kolejność w danym rozwiązaniu jest także kolejnością w której zostaną odwiedzone. W związku z tym, że w podstawowym problemie komiwojażera odległości pomiędzy miastami są symetryczne, to nie ma znaczenia dla danego rozwiązania czy lista miast zostanie przeprocesowana od lewej do prawej, czy od prawej do lewej. Dodatkowo także nie ma znaczenia od którego miasta zaczniemy taką podróż. W związku z tymi dwoma obserwacjami wielkość przestrzeni poszukiwań wynosi $\frac{(n-1)!}{2}$. Dla porównania z poprzednim problemem, dla 100 miast wartość ta jest rzędu 10^{155}.

Dla NLP teoretyczna przestrzeń przeszukiwań jest nieograniczona, ponieważ każda zmienna może przyjąć dowolną wartość ze zbioru liczb rzeczywistych. W związku z tym potrzeba każdą ze zmiennych odpowiednio poddać dyskretyzacji, aby było możliwe zastosowanie komputerów w celach obliczeniowych. W związku z tym możemy albo dokonać tego procesu samemu, na przykład dzieląc daną przestrzeń na określoną liczbę punktów w stałej odległości, lub skorzystać z precyzji jakie dają nam liczby zmiennoprzecinkowe na danych platformach obliczeniowych. W przypadku, gdy użylibyśmy standardowych liczb zmiennoprzecinkowych o podwójnej precyzji (IEEE Standard for Floating-Point Arithmetic, 2008) takich rozwiązań było by $n^{2^{64}}$, co dla równań z 100 zmiennymi dało by rząd wielkości $10^{10^{19}}$.

W przypadku MSRCPSP można zastosować różne potencjalne modele rozwiązania. Można przykładowo skupić się wyłącznie na priorytecie zadań, a do przydzielenia jakie zadanie ma być wykonane przez jaki zasób, zastosować podejście z użyciem prostego algorytmu zachłannego. Jest też możliwy sposób dokładnie odwrotny, czyli skupić się na przypisaniu zasobów do zadań i nie przejmować się tak dokładnie kolejnością zadań. Możliwe są także rozwiązania ustalające zarówno priorytet zadań jak i przypisania konkretnych zadań do nich. Dla takiego sposobu potencjalnych rozwiązań jest $t!\, r\hat{}t$, gdzie r to liczba zasobów, a t to liczba zadań. Dla 100 zasobów i 100 zadań daje to rząd wielkości potencjalnych rozwiązań równy 10^{357}.

1.6.2. Sąsiedztwo

Integralna częścią niektórych algorytmów oraz heurystyk jest pojęcie sąsiedztwa. Dwa rozwiązania są swoimi sąsiadami jeżeli są w pewien mierzalny sposób odpowiednio blisko siebie. Idąc dalej takie sąsiedztwem dla danego punktu w przestrzeni poszukiwań możemy nazwać wszystkie inne rozwiązania które spełniają taki warunek. Często takie punkty są uzyskiwane przez dokonanie jednej jak najmniejszej zmiany, ale konkretna definicja takiego warunku zależy od modelu problemu.

Dla SAT możemy zdefiniować je, poprzez odwrócenie wartości jednej ze zmiennych. Przykładowo jeżeli będziemy mieli rozwiązanie zapisane w postaci ciągu liczb binarnych: 01110, to sąsiedztwo dla niego będzie wyglądało następująco:

- 11110 (zamiana pierwszego bitu).
- 00110 (zamiana drugiego bitu).

- 01010 (zamiana trzeciego bitu).
- 01100 (zamiana czwartego bitu).
- 01111 (zamiana piątego bitu).

Dla TSP nie możemy zamienić tylko pojedynczego miasta na inne, ponieważ wtedy występowało by ono w rozwiązaniu więcej niż jeden raz, więc było by ono nieprawidłowe. W związku z tym najprościej jest zamienić dwa miasta miejscami. W zależności od tego ile sąsiadów chcemy wygenerować, możemy zamieniać miejscami tylko dwa miasta leżące obok siebie w danej ścieżce, lub dwa losowe miasta niezależnie od ich pozycji. W pierwszym wypadku, dla przykładowej ścieżki wyglądającej następująco: 1 – 3 – 4 – 5 – 2, zostały by wygenerowane następujące rozwiązania:

- 3 – 1 – 4 – 5 – 2 (zamiana pierwszego miasta z drugim).
- 1 – 4 – 3 – 5 – 2 (zamiana drugiego miasta z trzecim).
- 1 – 3 – 5 – 4 – 2 (zamiana trzeciego miasta z czwartym).
- 1 – 3 – 4 – 2 – 5 (zamiana czwartego miasta z piątym).
- 2 – 4 – 3 – 5 – 1 (zamiana piątego miasta z pierwszym).

Dla NLP jednym z podejść jest określenie maksymalnej odległości dla danych zmiennych dla której dwa punkty mogą dalej zostać zdefiniowane jako swoi sąsiedzi. Gdy mamy taką definicję, to możemy zmienić jedną z wartości punktów o losową wartość z przedziału nieprzekraczającej jej. Przykładowo dla rozwiązania składających się z trzech zmiennych o następujących wartościach: $(5.4, 2.34, 2.1)$ i maksymalnej odległości równej 0.12, każdy z poniższych punktów byłby sąsiadem:

- $(5.45, 2.34, 2.1)$ (zmiana wartości pierwszej zmiennej).
- $(5.4, 2.31, 2.1)$ (zmiana wartości drugiej zmiennej).
- $(5.4, 2.34, 2.12)$ (zmiana wartości trzeciej zmiennej).
- $(5.34, 2.34, 2.1)$ (zmiana wartości pierwszej zmiennej).
- $(5.4, 2.36, 2.1)$ (zmiana wartości drugiej zmiennej).

Dla MSRCPSP definicja sąsiedztwa zależy od modelu jaki zostanie wykorzystany. Dla listy priorytetów zadań, sąsiedztwo można zdefiniować przez zamianę priorytetów dwóch dowolnych zadań, podobnie jak w TSP. Za to dla listy przypisań zasobów można podejść do tego jako zmianę danego przypisanego zasobu, na dowolny inny zgodny z ograniczeniami. W przypadku gdy obecne są te dwie listy naraz, to sąsiedzi mogą być generowani na dwa wcześniej wymienione sposoby.

Mając zdefiniowane pojęcie sąsiedztwa możemy zdefiniować także pojęcie lokalnego optimum. Dane rozwiązanie jest w nim wtedy, gdy jest ono lepsze, lub co najmniej równe, niż wszystkie inne z jego sąsiedztwa. Najprostsze algorytmy poszukiwań rozwiązań, skupiają się tylko i wyłącznie na lokalnym optimum. Niestety w większości wypadków takie optimum nie jest jednocześnie globalnym. Wielkość zdefiniowanego sąsiedztwa pokazuje dla nich zależność pomiędzy skutecznością w poszukiwaniu rozwiązań, a czasem wykonania. Gdy takie sąsiedztwo jest niewielkie, to wtedy możemy szybko przeszukać wszystkie możliwości, jednakże taki algorytm może nie zauważyć jeszcze lepszego rozwiązania które jest tuż obok. W przeciwnym za to wypadku, taki algorytm może dojść do lepszych wyników w tej samej liczbie iteracji, jednakże czas jego wykonania może wzrosnąć do takiego stopnia, że będzie zupełnie bezużyteczny. W każdym wypadku taki rozmiar musi zostać dostosowany do konkretnego problemu dla którego ma zostać znalezione rozwiązanie.

1.6.3. Funkcja oceny rozwiązania, cel i ograniczenia rozwiązania problemu

Aby osiągnąć jak najlepsze rozwiązanie, należy zdefiniować funkcje oceny. W zależności od tego co chcemy osiągnąć, sposoby rozwiązania problemów dążą do jej minimalizacji, lub maksymalizacji. Odpowiednia funkcja powinna być jak najszybsza do obliczenia, nawet kosztem pewnych przybliżeń, ponieważ jest to element algorytmów i heurystyk który zwykle zajmuje najwięcej czasu w ich poszukiwaniach (He, Chen i Yao, 2015).

Można je podzielić na dwie typy (Michalewicz i Fogel, 2004). Pierwszym są porządkowe – pozwalają one na porównanie ze sobą dwóch rozwiązań. Drugim zaś typem są numeryczne – pozwalają one dodatkowo na określenie na ile jedno rozwiązanie jest lepsze od drugiego. Numeryczne pozwalają na większą elastyczność w projektowaniu odpowiedniego sposobu rozwiązania problemu, jednakże nie zawsze jest możliwe ich zastosowanie, a także mogą być droższe

Innym podziałem jest podział na funkcje statyczne i dynamiczne (Michalewicz i Fogel, 2004) – w tych pierwszych wartość oceny danego rozwiązania nie zmienia się, a w drugim przeciwnie. Przykładem jest próba opracowania najlepszego algorytmu do gry w szachy, gdzie jako funkcje oceny zwykle stosuje się wyniki w grze przeciwko innym, w związku z czym może ona się zmieniać w zależności od tego, z jakimi algorytmami zostanie on porównany.

Kolejną trudnością w projektowaniu takiej funkcji, jest fakt, że może nam zależeć na kilku rożnych kryteriach optymalizacji. Przykładowo przy wyborze samochodu może nam zależeć zarówno na jego cenie, jak i odpowiedniej mocy, wyposażeniu czy zużyciu paliwa. Aby

połączyć te kryteria razem, możemy zastosować sumę ważoną, przydzielając odpowiednią wagę dla każdego z celów, jaki chcemy osiągnąć i w ten sposób łącząc je w pojedynczą wartość.

W przypadku większości problemów trzeba także uwzględniać ograniczenia w możliwych rozwiązaniach. Aby sobie z nimi poradzić jest kilka możliwych rozwiązań, a w zależności od tego z jakim problemem mamy do czynienia, mogą zostać zastosowane inne sposoby (Michalewicz i Schoenauer, 1996):

- Zaprojektowanie takiego sposobu rozwiązania problemu, aby tworzyć tylko poprawne rozwiązania.
- Opracowanie sposobu naprawy nieprawidłowych rozwiązań tak, aby były z powrotem poprawne.
- Dodawanie kary do funkcji oceny gdy dane rozwiązanie nie spełnia ograniczeń.
- Stosowanie dwóch różnych funkcji oceny, w zależności od tego czy rozwiązanie jest poprawne czy nie.

Dla SAT liczba poprawnych rozwiązań jest bardzo mała, a w szczególności może nawet wynosić jeden. W związku z tym, sposoby rozwiązywania tego problemu muszą operować na nieprawidłowych rozwiązaniach. Najczęstszą spotykaną funkcja oceny jest ilość spełnionych części całego wyrażenia i w tym wypadku dążymy do jej maksymalizacji.

W przypadku TSP jako naturalna funkcję oceny możemy potraktować odległość danej drogi. W tym przypadku dążymy do jej minimalizacji. Ze względu na to, że wszystkie prawidłowe rozwiązania są swoimi permutacjami, to często spotyka się tak zaprojektowane algorytmy, aby uwzględniały to i produkowały tylko i wyłącznie poprawne rozwiązania.

Zaś w przypadku NLP funkcją oceny jest wartość samej funkcji która chcemy zoptymalizować. W zależności od konkretnego przypadku chcemy ją minimalizować lub maksymalizować. Także sposób obchodzenia się z ograniczeniami bardzo często zależy od charakteru funkcji nad którą pracujemy.

MSRCPSP jest problemem, który ma dwa różne kryteria optymalizacji. Można dążyć zarówno do minimalizacji czasu wykonania danego harmonogramu, jak i do minimalizacji jego kosztu. Minimalizacja kosztów jest łatwiejsza, od minimalizacji czasu, ponieważ można wybrać tylko najtańsze zasoby. Jednakże często spotykanym podejściem jest połączenie tych dwóch kryteriów naraz. Można w takim wypadku oba z nich znormalizować, przydzielić im odpowiednie wagi, a następnie policzyć ich sumę (Myszkowski, Skowroński i Sikora, 2015).

Jak widać w każdym z tych problemów dążymy do innego celu i napotykamy po drodze inne problemy. Część z nich nie ma ograniczeń wcale, część z nich polega na minimalizacji, a

część na maksymalizacji, a część z nich ma więcej niż jedno kryterium oceny. Oznacza to, że konieczny jest każdorazowy dobór odpowiedniej metody do próby rozwiązania danego problemu.

2. Metody heurystyczne do rozwiązywania problemów

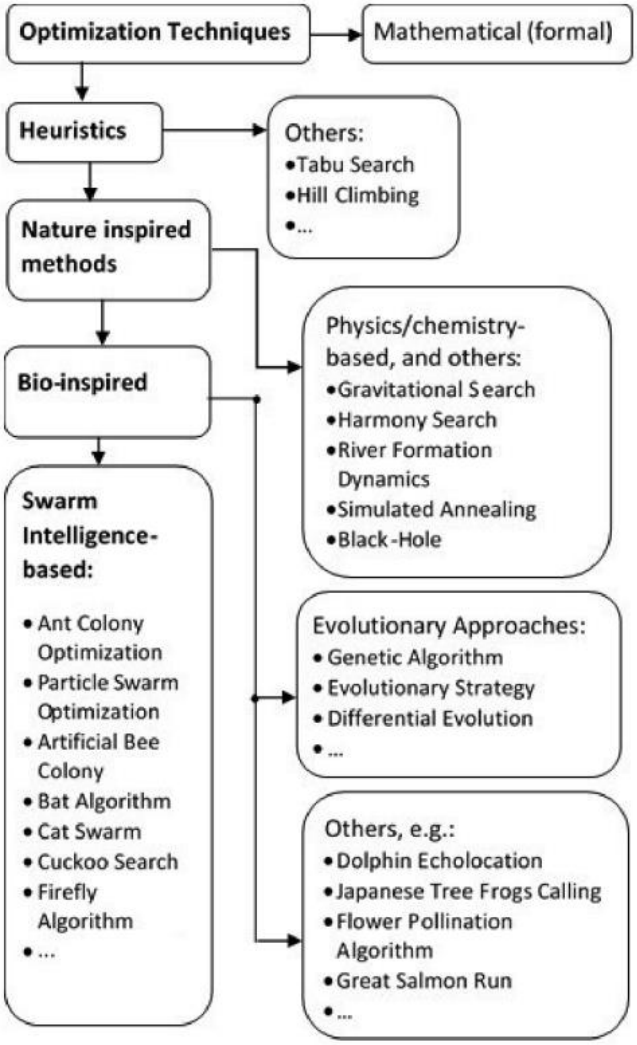
2.1. Przegląd metod heurystycznych do rozwiązywania problemów

Gdy zdefiniowaliśmy już elementy, z których składają się problemy obliczeniowe, to następnie możemy przejść do opisu potencjalnych metod, które pozwolą nam na uzyskaniu jak najlepszych rozwiązań. Ważny w tym wypadku jest odpowiedni balans pomiędzy skutecznością metody w poszukiwaniu co raz to lepszych rozwiązań, a także jej czasem działania. W skrajnym przypadku dla wielu problemów przegląd zupełny rozwiązań znalazł by dla nas to najbardziej optymalne rozwiązanie, jednak zajęło by to zbyt dużo czasu żeby było to w ogóle możliwe.

Dla przedstawionych wcześniej w tej pracy problemów nie są znane takie algorytmy, które pozwalały by na uzyskanie najlepszego rozwiązania w akceptowalnym czasie. W związku z tym w ich wypadku stosowane są metaheurystyki. Są to uniwersalne wysokopoziomowe podejścia, które dostarczają zbiór zasad, czy strategii w celu stworzenia heurystycznego algorytmu optymalizacyjnego (Sörensen i Glover, 2013). Otrzymane w ten sposób metody są nazywane metodami heurystycznymi i w przeciwieństwie do klasycznych algorytmów, nie gwarantują one znalezienia optymalnego rozwiązania.

Metody heurystyczne można podzielić na dwie kategorie: takie, które operują tylko na kompletnych rozwiązaniach i takie, które operują także na niekompletnych (Caserta i Voss, 2010). W tym pierwszym wypadku jeżeli zatrzymamy działanie metody przed czasem, to zawsze otrzymamy prawidłowe rozwiązanie, w drugim wypadku może to być niemożliwe.

Wiele z tych metod jest inspirowana procesami zachodzącymi w naturze. Na rysunku 4 przedstawiono ich taksonomię.



Rysunek 4: Taksonomia metod inspirowanych naturą

Źródło: (Slowik i Kwasnicka, 2018)

Nie każda metoda może zostać zastosowana do każdego problemu. Każda z nich ma swoje mocne i słabe strony, a także pewne ograniczenia do czego może zostać zastosowana. W związku z tym, do każdego problemu który chcemy rozwiązać, należy każdorazowo dobrać metody.

2.2. Algorytm zachłanny

Algorytmy zachłanne opierają się na tworzeniu pełnego rozwiązania krok po kroku. Powodem ich popularności jest ich prostota. Główna idea stojąca za nimi jest następująca: za

każdym razem gdy trzeba podjąć decyzję o dołożeniu nowej części do rozwiązania, wybierz taką, która dla danego częściowego rozwiązania da najlepszy rezultat w danym momencie. To podejście zakłada heurystycznie, że każde gdy będziemy podążać najlepszymi krokami w danym momencie, to osiągniemy finalnie najlepsze rozwiązanie - lecz oczywiście jest to dość krótkowzroczna metoda, ponieważ nie zawsze tak będzie. Stąd bierze się nazwa algorytmu zachłannego.

Dla SAT można opracować następujący algorytm zachłanny: dla każdej zmiennej, w dowolnej kolejności, dopasuj taką jej wartość prawda lub fałsz, która w danym momencie sprawi, że jak największa liczba podrównań będzie spełniona. W przypadku remisu, można wybrać tą wartość na przykład losowo, lub na przemian. Niestety takie podejście napotka na problem na takim prostym przykładowym równaniu (Równanie 2).

$$F(x) = \bar{x_1} \wedge (x_1 \vee x_2) \wedge (x_1 \vee x_3)$$

Równanie 2: Przykładowe równanie dla SAT

Źródło: opracowanie własne

Jak widać, w tym wypadku, takie podejście przydzieliło by najpierw wartość $prawda$ dla zmiennej x_1, ponieważ spełnione wtedy zostały by drugie i trzecie podrównania. Niestety, wtedy pierwsze z nich nie będzie spełnione i cały przypadek pozostanie bez rozwiązania, nie zależnie od tego jakie wartości zostaną przydzielone dla innych zmiennych. Można by było ulepszać ten algorytm, dodając co raz to kolejne reguły, jednakże nigdy nie będzie możliwe stworzenie takiego algorytmu zachłannego, który by pozwolił na znalezienie rozwiązania w każdy wypadku. Wynika to z przedstawionego wcześniej faktu, iż problem SAT jest NP-zupełny i jego rozwiązanie w czasie wielomianowym jest niemożliwe.

Podobnie jest w przypadku NLP. Można było by opracować algorytm, który starał by się wiele razy dla każdej ze zmiennych dobrać najlepsza wartość w danym momencie, tworząc pewnego rodzaju wielowymiarowe przeszukiwanie liniowe. Jednakże problem z takim podejściem jest taki, jak z innymi algorytmami genetycznymi – patrzą one krótkowzrocznie i pomijają interakcje jakie występują pomiędzy różnymi zmiennymi (Himmelblau, 1972).

Za to w przypadku TSP wygląda to trochę inaczej – o wiele łatwiejsze tutaj jest opracowanie takiego algorytmu zachłannego, który generuje zawsze poprawne rozwiązania. W związku z tym istnieją takie metody rozwiązywania problemów, które przykładowo zaczynają swoje działanie na zbiorze początkowym który został wygenerowany przez algorytm genetyczny, zamiast na losowych rozwiązaniach. Jedną z najprostszych procedur, jaka przychodzi tutaj do

głowy, jest zaczęcie w dowolnym losowym mieście i następnie odwiedzanie zawsze tego miasta, do którego jest najbliżej i w które jeszcze nie jest na wygenerowanej ścieżce. Oczywiście tak jak poprzednio, sposób ten jest chciwy i zawsze wybierze najkrótszą w danej chwili ścieżkę, nie ważne czy byłaby ona w końcowym rozrachunku optymalna.

Dla MSRCPSP w zależności od zastosowanej funkcji oceny, można przykładowo zastosować prostego algorytmu podobnego dla tego dla TSP. Polega on na budowie harmonogramu krok po kroku, przez dobieranie nowych elementów do niego, w ten sposób, aby jego ocena zwiększyła się o jak najmniejszą wartość. Jest to bardzo ograniczony sposób, który zupełnie pomija konsekwencje takiego działania, jednak jest od jednocześnie bardzo prosty do implementacji i szybki w działaniu.

2.3. Przeszukiwanie lokalne

Przeszukiwanie lokalne skupia się na wyjaśnionym już wcześniej koncepcie sąsiedztwa. Zamiast robić przegląd zupełny rozwiązań, polega ono na modyfikacji tych już istniejących, w celu ich poprawy. Początkowe rozwiązania mogą zostać wygenerowane losowo, lub przykładowo wybrane z określonych regularnie punktów w ich przestrzeni poszukiwań. Sposób działania tej metody jest następujący: wygeneruj początkowe rozwiązanie, a następnie wybierz jednego, lub więcej, z jego sąsiadów i jeżeli będzie któryś z nich lepszy, to skup swoje poszukiwania na nim.

Metoda ta wymaga dobrania odpowiedniej definicji sąsiedztwa. Gdy potencjalnych sąsiadów będzie mało, to ich przejrzenie będzie szybkie, jednak może on łatwiej utknąć w lokalnym minimum. A gdy będzie ich zbyt dużo, to jego wykonanie może trwać po prostu zbyt długo. W szczególnym wypadku, gdyby w takim sąsiedztwie znalazło by się każde inne rozwiązanie, metoda ta zamieniła by się w przegląd zupełny. Nie ma jednej uniwersalnej wartości jaka pasuje do każdego przypadku i tak jak w przypadku innych, musi ona być zawsze strojona pod konkretny określony problem, do którego chcemy znaleźć jak najlepsze rozwiązanie.

Problemem tego podejścia jest fakt, że bardzo łatwo może ono utknąć w lokalnym minimum i nie ma żadnego sposobu na wyjście z niego, gdy już tam wpadnie. Mimo tych wad jest to jedna z najszybszych i najłatwiejszych do implementacji metod. W związku z tym jest czasami

łączona z innymi, bardziej skomplikowanymi, aby lepiej przygotować początkową pulę rozwiązań, zamiast przykładowo je losować. Analogicznie także może zostać ona wykorzystana na sam koniec innych działania algorytmów czy metaheurystyk, aby zobaczyć czy nie przegapiły one prostych zmian, które mogły by jeszcze wprowadzić finalne poprawki.

Sposób działania tej metody dla każdego z omówionego w tej pracy problemu jest taki sam. To czym różnią się one między sobą, to różne definicje sąsiedztwa, ze względu na różne modele dla każdego z problemów. W związku z tym, jest ona bardzo prosta w implementacji i uniwersalna – można tą metodę wykorzystać w każdym przypadku.

2.4. Symulowane wyżarzanie

Aby poprawić działanie poprzedniej metody zostało opracowane symulowane wyżarzanie. Jak wiele innych technik rozwiązywania takich problemów zostało one zainspirowane prawdziwymi zjawiskami występującymi w przyrodzie – w tym wypadku procesami zachodzącym podczas obróbki cieplnej metali (Kirkpatrick, Gelatt i Vecchi, 1983).

Metoda ta bazuje na sposobie działania przeszukiwania lokalnego z jedną, ale istotną zmian. Jeżeli funkcja oceny $F(x)$ dla nowego wygenerowanego rozwiązania v_n zwraca wartość gorszą niż dla obecnego v_c, to może zostać ono i tak zaakceptowane z prawdopodobieństwo równym: $e^{\frac{F(v_c)-F(v_n)}{T}}$, gdzie T to malejąca z czasem temperatura (Kirkpatrick, Gelatt i Vecchi, 1983). Równanie to ma zastosowanie w przypadku problemów minimalizacji, w przypadku maksymalizacji różnica ta musi zostać odwrócona. Zmiana ta pozwala na wyjście z lokalnego minimum i potencjalne trafienie w przyszłości na rejony z lepszymi rozwiązaniami, niż te na które metoda trafiła na początku.

Im ocena nowego, ale gorszego rozwiązania, jest bliższa ocenie tego poprzedniego, tym większa jest szansa na jego zaakceptowanie. Przykładowe prawdopodobieństwa przedstawia pierwsza z poniższych tabel (Tabela 1).

Im większa tym różnica, tym szansa na akceptacje nowego rozwiązania staje się coraz mniejsza. W przypadku gdy jest tylko nieznacznie gorsze, jest ona dość duża, a w wypadku przeciwnym, jest ona minimalna i dąży do zera. Ma na to także wpływ wysokość temperatury w danym momencie, co jest przedstawione w drugiej z tabel (Tabela 2).

Tabela 1: Przykładowe prawdopodobieństwa dla akceptacji nowego rozwiązania w symulowanym wyżarzaniu przy zmiennej ocenie nowego rozwiązania

$F(v_c)$	T	$F(v_n)$	$e^{\frac{F(v_n)-F(v_c)}{T}}$
100	10	100	100%
100	10	101	90,48%
100	10	102	81,87%
100	10	105	60,65%
100	10	110	36,79%
100	10	120	13,53%
100	10	150	0,67%

Źródło: opracowanie własne

Tabela 2: Przykładowe prawdopodobieństwa dla akceptacji nowego rozwiązania w symulowanym wyżarzaniu przy zmiennej temperaturze

$F(v_c)$	T	$F(v_n)$	$e^{\frac{F(v_n)-F(v_c)}{T}}$
100	100	110	90,48%
100	50	110	81,87%
100	20	110	60,65%
100	10	110	36,79%
100	5	110	13,53%
100	2	110	0,67%

Źródło: opracowanie własne

Funkcja temperatury jest funkcja malejąca i jej wzór, tak samo jak wysokość temperatury początkowej, muszą zostać dopasowane indywidualne do danego problemu dla jakiego mają zostać znalezione rozwiązania. Wraz z postępem prac tej metody, temperatura ta maleje i związku z tym, prawdopodobieństwo także dąży do zera. W przypadku swojej implementacji tego algorytmu do problemu harmonogramowania zdecydowałem na określenie temperatury bieżącej jako iloraz temperatury maksymalnej, przez numer bieżącej iteracji metody.

2.5. Algorytm genetyczny

Ostatnia z przedstawionych w tej pracy metod jest najbardziej skomplikowana. Podobnie jak poprzednia, jest tak samo inspirowana procesami zachodzącymi w przyrodzie, a konkretnie działaniem naturalnej ewolucji wśród żywych organizmów. Schemat jej działania można opisać ją w następujący sposób (Mitchell, 1996):

1. Wygeneruj określoną liczbę losowych osobników.
2. Powtarzaj przez określoną liczbę iteracji:
    a. Przeprowadź proces selekcji.
    b. Przeprowadź proces krzyżowania.
    c. Przeprowadź proces mutacji.
3. Zwróć najlepsze rozwiązanie.

Początkowe osobniki zwykle generowane są w sposób losowy. Pierwszym krokiem działania tej metaheurystyki jest selekcja. Metod na przeprowadzenie jej jest wiele i zwykle mogą zostać zastosowane do wielu problemów. Przykładowe sposoby to (Goldberg i Deb, 1991):

- Selekcja turniejowa – polega ona na wybraniu określonej liczby losowych osobników z populacji, a następnie wybraniu najlepszego z nich. Proces ten powtarza się tak długo aż to potrzebne. W tym wypadku im większy rozmiar turnieju, tym mniejsza szansa na wybranie mniej optymalnych osobników
- Selekcja rankingowa – polega ona na uszeregowaniu wszystkich osobników z populacji i wybraniu tylko tych najlepszych
- Selekcja ruletkowa – polega ona na losowaniu osobników z populacji, gdzie im lepsza ocena danego osobnika, tym większa jego szansa na wylosowanie go
- Selekcja elitystyczna – polega ona na wybraniu najlepszego osobnika z populacji i zachowanie go w niezmienionej postaci do końca iteracji metody

Selekcja taka może wybrać liczbę osobników w nowej populacji równej poprzedniej i wtedy niektóre osobniki będą dla siebie identyczne, lub wybrać mniejszą i braki uzupełnić losowo wygenerowanymi nowymi rozwiązaniami.

Następnym krokiem jest proces krzyżowania, którego celem jest stworzenie nowych rozwiązań które będą łączyły cechy innych dwóch, lub większej ilości, rozwiązań i dzięki temu uzyskanie potencjalnie lepszych osobników (Akter, Nahar, Shahadat i Andersson, 2019). W przeciwieństwie do poprzedniego, jest on wykonywany z pewnym prawdopodobieństwem i nie

wszystkie osobniki będą brały udział w nim. W przypadku problemu SAT proces ten jest najprostszy – część wartości jest brana z pierwszego rozwiązania, a pozostałe z drugiego. W przypadku NLP oprócz takiego podejścia, można zamiast tego chociażby uśrednić wartości dla krzyżowanych osobników – efektem tego będzie powstanie dwóch identycznych osobników, o ile nie zostaną złamane żadne dodatkowe ograniczenia. W przypadku TSP krzyżowanie jest bardziej skomplikowane – w związku z tym, że dane miasto nie może zostać odwiedzone dwa razy, należy zastosować takie operatory których efektem działania będzie zachowanie prawidłowej kolejności. Zaś dla przypisania priorytetów zdecydowałem się na implementacje operatora krzyżowania pozycji (Syswerda, 1991): wybiera on część priorytetów z pierwszego osobnika, a brakujące uzupełnia zgodnie z kolejnością z drugiego.

Ostatnim krokiem jest proces mutacji, którego celem jest taka zmiana cech dotychczasowych osobników, aby być może udało się wyjść z lokalnego minimum lub maksimum (Otman, Abouchabaka i Tajani, 2012). Podobnie jak poprzedni, jest on wykonywany z prawdopodobieństwem, jednak zwykle mniejszym niż to, które występuje w przypadku krzyżowania. Schemat działania jest bardzo podobny do procesu generowania nowych sąsiadów, który został opisany w jednym z poprzednich rozdziałów. Gdy dany osobnik zostanie wybrany do poddania się mutacji, to jest generowany jego sąsiad który go zastąpi. Należy odpowiednio dobrać szansę na zadziałanie tego operatora. Zbyt niska wartość prawdopodobieństwa mutacji spowoduje, że populacje szybko utkną w lokalnych minimach, a zbyt wysoka sprawi, że wręcz przeciwnie, nie zdążą one osiągnąć żadnych dobrych rezultatów przed kolejną taką modyfikacją.

Jak widać metoda ta pozwala na wiele różnych modyfikacji, ale i wymaga odpowiedniego ustawienia tak samo wielu różnych parametrów. W związku z tym, zaletą jej jest fakt, że można ją zastosować do rozwiązywania różnych typów problemów. Jest to jednocześnie też jej wadą, ponieważ trudne jest określenie najbardziej optymalnego takiego zestawu i jest to różne w zależności od problemu który chcemy rozwiązać, a nawet od konkretnych danych dla danego problemu.

3. Założenia realizacji badania

3.1. Cel badania i zastosowana procedura badania

W poniższym rozdziale zostaną przedstawione założenia realizacji eksperymentu oraz sposób jego realizacji. Badanie to zostało przeprowadzone dla zdefiniowanego problemu MSRCPSP przy użyciu przedstawionych w poprzedniej części pracy metod obliczeniowych.

Pierwszym celem przeprowadzonego badania empirycznego jest znalezienie, zaimplementowanie i dostrojenie różnych metod heurystycznych, aby porównać wyniki ich skuteczności. Drugim zaś, jest sprawdzenie, czy jest możliwe przy użyciu tych metod uzyskanie bliskich optymalnym harmonogramów w akceptowalnym czasie, dzięki czemu można byłoby te metody zastosować, nie tylko dla sztucznego zbioru danych, ale także dla prawdziwych problemów tego typu. Badanie przeprowadzono według następującej procedury:

1. Identyfikacja problemu.
2. Zdefiniowanie elementów problemu.
3. Wybór metod heurystycznych.
4. Implementacja metod heurystycznych.
5. Dobranie parametrów dla metod heurystycznych.
6. Przeprowadzanie eksperymentu.
7. Analiza wyników eksperymentu.
8. Wizualizacja wyników.
9. Sformułowanie wniosków.

Symulowane wyżarzanie i algorytm genetyczny zostały wybrane jako metody heurystyczne, które miały potencjał osiągnąć dobre wyniki. Na pierwszą zdecydowałem się, ponieważ jest ona bardzo prosta w implementacji i dodatkowo ma małą liczbę dostosowywalnych parametrów, jednakże pozwalając na osiągnięcie bardzo dobrych wyników. W przypadku drugiej zdecydowałem się z powodu jej elastyczności. Daje ona bardzo wiele możliwych kombinacji używanych operatorów, co pozwala ją dostosować do różnych problemów, ale także jednocześnie bardzo wiele parametrów, które należy odpowiednio dostroić, aby uzyskać jak najlepsze wyniki.

Przeszukiwanie lokalne i algorytm zachłanny zostały wybrane jako prostsze metody, które zwykle osiągają gorsze wyniki, jednak mogą być dobrym porównaniem skuteczności dla innych metod. Pierwsza z nich jest podobna do symulowanego wyżarzania, jednak bez jego kluczowego elementu. Druga zaś jest metodą, która mogła by zostać wykorzystana gdyby człowiek bez żadnego wcześniejszego doświadczenia musiał złożyć samemu taki harmonogram.

3.2. Przyjęte założenia dla eksperymentu

W przypadku MSRCPSP zdecydowałem się na zastosowanie modelu składającego się z dwóch list. Pierwsza z nich to lista przypisań danych zasobów do danych zadań, a druga z nich to lista priorytetów, z jakimi mają zostać wykonane dane zadania. Sposób ten pozwala na pełną kontrolę nad rozwiązaniem - w przeciwieństwie do modeli opartych na np. samej liście priorytetów zadań. Nie wymaga on zastosowywania żadnych dodatkowych heurystyk przy obliczaniu momentu rozpoczęcia zadań. Wadą tego podejścia jest konieczność zaprojektowania dwóch różnych zachowań dla każdej z list, w przypadku każdej operacji jaką chcemy wykonać na takim rozwiązaniu.

Zdecydowałem się także na definicje sąsiedztwa inspirowaną wcześniej przedstawianymi przykładami. Dla listy priorytetów zadań, sąsiedztwo zdefiniowałem przez zamianę priorytetów dwóch dowolnych zadań, podobnie jak w TSP. Za to dla listy przypisań zasobów, zdecydowałem się na zmianę danego przypisanego zasobu, na dowolny inny zgodny z ograniczeniami. Każdy sąsiad może być oddalony od drugiego tylko o jedno naraz z tych dwóch transformacji. Generuje to dwa różniące się od siebie typy sąsiedztwa, jednak było to wymagane przez podwójny charakter przyjętego przeze mnie modelu.

Jako że minimalizacja kosztów jest łatwiejsza – można wybrać tylko najtańsze zasoby – postanowiłem się w mojej pracy skupić wyłącznie na minimalizacji czasu. Dodatkowo zaprojektowałem wszystkie zastosowane przeze mnie operator i reprezentację tak aby generowały wyłącznie poprawne rozwiązania, omijając w ten sposób problemy związane z naprawą lub eliminacja nieprawidłowych rozwiązań.

W przypadku algorytmu zachłannego, zdecydowałem się na implementacje podobnego do tego dla TSP. Sposób jego działania jest następujący: wybiera po kolei takie zadanie i

przypisuje taki do niego zasób, który w danym momencie powiększy czas wykonania całego harmonogramu o jak najmniejszą wartość. W wypadku kilku takich możliwości, jest ono losowane spośród z nich. Jest to jak widać bardzo prosty algorytm, jednak zdecydowałem się na jego implementacje, aby można było go porównać z innymi, także bardziej skomplikowanymi, metodami.

W swojej implementacji algorytmu genetycznego zdecydowałem się na wykorzystanie połączonej selekcji elitystycznej, z turniejową. Selekcja elitystyczna gwarantuje, że najlepsze rozwiązanie nigdy nie zginie, a selekcja turniejowa jest prosta w implementacji i dzięki możliwości sterowania wielkością tego turnieju, pozwala na dostosowanie jak często gorzej oceniane rozwiązania przejdą dalej. W przypadku krzyżowania zdecydowałem się na zastosowanie dwóch sposobów. Dla przypisania zasobów wybrałem następującą metodę: podziel rozwiązania na losowej wielkości dwie części, a następnie zamień je nawzajem częściami, tak aby powstały dwa osobniki z elementami z obu początkowych rozwiązań.

3.3. Opis sposobu realizacji eksperymentu

Każda z podanych wcześniej metod uruchomiłem dziesięć razy, na każdym z dostarczonych przez naukowców z Politechniki Wrocławskiej zbiorze danych. Następnie dla każdego uruchomienia została zapisana ocena najlepszego rozwiązania i dla uzyskanych w ten sposób rozwiązań dla danego zbioru została policzona wartość minimalna, średnia i odchylenie standardowe. Na większą ilość iteracji nie pozwoliła dostępna mi moc obliczeniowa mojego komputera.

Badania zajęły około 9 godzin, przy wykorzystaniu wielowątkowości, na komputerze wyposażonym w procesor Intel Core i9-9900K podkręconym w celu osiągniecia stałego taktowania 4,7 GHz, oraz posiadającym 32 GB pamięci operacyjnej DDR4 o taktowaniu 3600 MHz. Kod programu został napisany przy użyciu języka Java, a także został sprofilowany, aby usunąć potencjalne nieoptymalne implementacje algorytmów.

Parametry uruchomieniowe dla metod prezentują się następująco:

- Algorytm genetyczny:
    - Rozmiar populacji: 100.
    - Liczba iteracji: 10000.

    - Rozmiar selekcji turniejowej: 5.
    - Szansa na krzyżowanie: 100%.
    - Szansa na mutację: 50%.
- Przeszukiwane lokalne:
    - Ilość powtórzeń: 100000.
- Symulowane wyżarzanie:
    - Ilość iteracji: 1000000.
    - Temperatura maksymalna: 250000.
- Algorytm zachłanny:
    - Ilość powtórzeń: 100.

Parametry te zostały dobrane empirycznie, na bazie wcześniejszych doświadczeń na przestawionych wcześniej zbiorach danych. Zostały one dobrane tak, aby czas działania każdej z metod był zbliżony do siebie, a uzyskane wyniki jak najlepsze. Średni taki czas jednej iteracji metody wyniósł około 77 sekund.

Implementacje metod heurystycznych, jak i całe badanie, zostały wykonane bez użycia możliwości obliczeniowych, jakie potrafią dać nowoczesne karty graficzne. Ze względu na ich architekturę pozwalają one na o wiele szybsze równoległe obliczenia dla dużych zbiorów danych (Luo, Fujimura, El Baz i Plazolles, 2019). Jednakże implementacja wybranych przeze mnie metod w ten sposób nie jest trywialnym problemem i nie podjąłem się jej w tej pracy.

4. Wyniki i wnioski z przeprowadzonego eksperymentu

4.1. Wyniki z przeprowadzonego eksperymentu

Jak wspomniałem we wcześniejszym rozdziale, każda z metod heurystycznych została uruchomiona po 10 razy, na każdym z przypadków w zbiorze danych i został zapamiętany wynik, jaki uzyskał wygenerowany przez nią harmonogram. Następnie dla tych wszystkich wartości funkcji oceny została policzona wartość minimalna, średnia i odchylenie standardowe.

Dla zbioru danych 10_3_5_3 otrzymane wyniki przedstawiono w tabeli 3. Dla tego zbioru danych symulowane wyżarzanie osiągało stały i najlepszy wynik. Algorytm genetyczny miał drobne odchylenie standardowe, jednak wyniki minimalne były takie same jak dla symulowanego wyżarzania. Przeszukiwanie lokalne było w stanie osiągnąć tak dobre minimalne rozwiązanie jak poprzednie dwa algorytmy, jednak z o wiele większym odchyleniem standardowym. Algorytm zachłanny za to osiągał gorsze minimalne rozwiązanie, jednakże, miał on zerowe odchylenie standardowe.

Tabela 3: Wyniki dla zbioru danych 10_3_5_3

Metoda obliczeniowa	Min.	Śred.	Odch. Std.
Algorytm genetyczny	93	94,1	1,37
Algorytm zachłanny	99	99	0
Przeszukiwanie lokalne	93	105,6	7,96
Symulowane wyżarzanie	93	93	0

Źródło: opracowanie własne

Dla zbioru danych 10_5_8_5 otrzymane wyniki przedstawiono w tabeli 4. Wszystkie algorytmy osiągnęły takie same wyniki, poza przeszukiwaniem lokalnym, które miało większe odchylenie standardowe.

Tabela 4: Wyniki dla zbioru danych 10_5_8_5

Metoda obliczeniowa	Min.	Śred.	Odch. Std.
Algorytm genetyczny	80	80	0
Algorytm zachłanny	80	80	0
Przeszukiwanie lokalne	80	82,3	4,06
Symulowane wyżarzanie	80	80	0

Źródło: opracowanie własne

Dla zbioru danych 10_7_10_7 otrzymane wyniki przedstawiono w tabeli 5. Dla tego zbioru danych wszystkie algorytmy osiągnęły takie same wyniki.

Tabela 5: Wyniki dla zbioru danych 10_7_10_7

Metoda obliczeniowa	Min.	Śred.	Odch. Std.
Algorytm genetyczny	104	104	0
Algorytm zachłanny	104	104	0
Przeszukiwanie lokalne	104	104	0
Symulowane wyżarzanie	104	104	0

Źródło: opracowanie własne

Dla zbioru danych 15_3_5_3 otrzymane wyniki przedstawiono w tabeli 6. Dla tego zbioru danych wszystkie algorytmy osiągnęły takie same wyniki.

Tabela 6: Wyniki dla zbioru danych 15_3_5_3

Metoda obliczeniowa	Min.	Śred.	Odch. Std.
Algorytm genetyczny	230	230	0
Algorytm zachłanny	230	230	0
Przeszukiwanie lokalne	230	230	0
Symulowane wyżarzanie	230	230	0

Źródło: opracowanie własne

Dla zbioru danych 15_6_10_6 otrzymane wyniki przedstawiono w tabeli 7. Dla tego zbioru danych wszystkie algorytmy osiągnęły takie same wyniki.

Tabela 7: Wyniki dla zbioru danych 15_6_10_6

Metoda obliczeniowa	Min.	Śred.	Odch. Std.
Algorytm genetyczny	102	102	0
Algorytm zachłanny	102	102	0
Przeszukiwanie lokalne	102	102	0
Symulowane wyżarzanie	102	102	0

Źródło: opracowanie własne

Dla zbioru danych 15_9_12_9 otrzymane wyniki przedstawiono w tabeli 8. Dla tego zbioru danych wszystkie algorytmy osiągnęły takie same wyniki, poza przeszukiwaniem lokalnym, które miało większe odchylenie standardowe.

Tabela 8: Wyniki dla zbioru danych 15_9_12_9

Metoda obliczeniowa	Min.	Śred.	Odch. Std.
Algorytm genetyczny	90	90	0
Algorytm zachłanny	90	90	0
Przeszukiwanie lokalne	90	90,7	2,21
Symulowane wyżarzanie	90	90	0

Źródło: opracowanie własne

Dla zbioru danych 100_10_26_15 otrzymane wyniki przedstawiono w tabeli 9. Dla tego zbioru danych symulowane wyżarzanie osiągało najlepszy wyniki. Algorytm genetyczny miał minimalnie większe odchylenie standardowe, jednak wyniki minimalne były podobne jak dla symulowanego wyżarzania. Algorytm zachłanny osiągnął gorsze minimalne rozwiązania, wraz z większym odchyleniem standardowym. Przeszukiwanie lokalne osiągnęło podobne minimalne rozwiązania co poprzednia heurystyka, jednak z jeszcze większym odchyleniem.

Tabela 9: Wyniki dla zbioru danych 100_10_26_15

Metoda obliczeniowa	Min.	Śred.	Odch. Std.
Algorytm genetyczny	235	236,1	0,99
Algorytm zachłanny	270	280,3	7,56
Przeszukiwanie lokalne	272	308,8	37,37
Symulowane wyżarzanie	234	234,1	0,32

Źródło: opracowanie własne

Dla zbioru danych 100_10_27_9_D2 otrzymane wyniki przedstawiono w tabeli 10. Dla tego zbioru danych symulowane wyżarzanie osiągało najlepszy wyniki. Algorytm genetyczny miał minimalnie większe odchylenie standardowe, jednak wyniki minimalne były podobne jak dla symulowanego wyżarzania. Algorytm zachłanny osiągnął gorsze minimalne rozwiązania, wraz z większym odchyleniem standardowym. Przeszukiwanie lokalne osiągnęło podobne minimalne rozwiązania co poprzednia heurystyka, jednak z jeszcze większym odchyleniem.

Tabela 10: Wyniki dla zbioru danych 100_10_27_9_D2

Metoda obliczeniowa	Min.	Śred.	Odch. Std.
Algorytm genetyczny	208	210,5	2,17
Algorytm zachłanny	237	241,3	3,06
Przeszukiwanie lokalne	226	281,1	26,57
Symulowane wyżarzanie	207	208,4	0,84

Źródło: opracowanie własne

Dla zbioru danych 100_10_47_9 otrzymane wyniki przedstawiono w tabeli 11. Dla tego zbioru danych symulowane wyżarzanie osiągało najlepszy wyniki. Algorytm genetyczny miał minimalnie większe odchylenie standardowe, jednak wyniki minimalne były podobne jak dla symulowanego wyżarzania. Algorytm zachłanny osiągnął gorsze minimalne rozwiązania, wraz z większym odchyleniem standardowym. Przeszukiwanie lokalne osiągnęło podobne minimalne rozwiązania co poprzednia heurystyka, jednak z jeszcze większym odchyleniem.

Tabela 11: Wyniki dla zbioru danych 100_10_47_9

Metoda obliczeniowa	Min.	Śred.	Odch. Std.
Algorytm genetyczny	253	254,9	1,2
Algorytm zachłanny	274	277,7	2,31
Przeszukiwanie lokalne	272	299,3	29,39
Symulowane wyżarzanie	253	253,7	0,48

Źródło: opracowanie własne

Dla zbioru danych 100_10_48_15 otrzymane wyniki przedstawiono w tabeli 12. Dla tego zbioru danych symulowane wyżarzanie osiągało najlepszy wyniki. Algorytm genetyczny miał minimalnie większe odchylenie standardowe, jednak wyniki minimalne były podobne jak dla

symulowanego wyżarzania. Algorytm zachłanny osiągnął gorsze minimalne rozwiązania, wraz z większym odchyleniem standardowym. Przeszukiwanie lokalne osiągnęło podobne minimalne rozwiązania co poprzednia heurystyka, jednak z jeszcze większym odchyleniem.

Tabela 12: Wyniki dla zbioru danych 100_10_48_15

Metoda obliczeniowa	Min.	Śred.	Odch. Std.
Algorytm genetyczny	244	246,1	1,52
Algorytm zachłanny	280	283	4,32
Przeszukiwanie lokalne	262	296,6	35,46
Symulowane wyżarzanie	244	244,2	0,42

Źródło: opracowanie własne

Dla zbioru danych 100_10_64_9 otrzymane wyniki przedstawiono w tabeli 13. Dla tego zbioru danych symulowane wyżarzanie osiągało najlepszy wyniki. Algorytm genetyczny miał minimalnie większe odchylenie standardowe, jednak wyniki minimalne były podobne jak dla symulowanego wyżarzania. Algorytm zachłanny osiągnął gorsze minimalne rozwiązania, wraz z większym odchyleniem standardowym. Przeszukiwanie lokalne osiągnęło lepsze minimalne rozwiązania niż poprzednia heurystyka, jednak z jeszcze większym odchyleniem.

Tabela 13: Wyniki dla zbioru danych 100_10_64_9

Metoda obliczeniowa	Min.	Śred.	Odch. Std.
Algorytm genetyczny	242	243,9	1,1
Algorytm zachłanny	292	298,2	3,33
Przeszukiwanie lokalne	258	292,3	17,47
Symulowane wyżarzanie	242	242,8	0,63

Źródło: opracowanie własne

Dla zbioru danych 100_10_65_15 otrzymane wyniki przedstawiono w tabeli 14. Dla tego zbioru danych symulowane wyżarzanie osiągało najlepszy wyniki. Algorytm genetyczny miał minimalnie większe odchylenie standardowe, jednak wyniki minimalne były podobne jak dla symulowanego wyżarzania. Algorytm zachłanny osiągnął gorsze minimalne rozwiązania, wraz z większym odchyleniem standardowym. Przeszukiwanie lokalne osiągnęło lepsze minimalne rozwiązania co poprzednia heurystyka, jednak z jeszcze większym odchyleniem.

Tabela 14: Wyniki dla zbioru danych 100_10_65_15

Metoda obliczeniowa	Min.	Śred.	Odch. Std.
Algorytm genetyczny	245	246	0,82
Algorytm zachłanny	295	299,7	3,47
Przeszukiwanie lokalne	254	310,4	36,85
Symulowane wyżarzanie	243	244,2	0,63

Źródło: opracowanie własne

Dla zbioru danych 100_20_22_15 otrzymane wyniki przedstawiono w tabeli 15. Dla tego zbioru danych symulowane wyżarzanie osiągało najlepszy wyniki. Algorytm genetyczny miał minimalnie większe odchylenie standardowe, jednak wyniki minimalne były podobne jak dla symulowanego wyżarzania. Algorytm zachłanny osiągnął gorsze minimalne rozwiązania, wraz z większym odchyleniem standardowym. Przeszukiwanie lokalne osiągnęło podobne minimalne rozwiązania co poprzednia heurystyka, jednak z jeszcze większym odchyleniem.

Tabela 15: Wyniki dla zbioru danych 100_20_22_15

Metoda obliczeniowa	Min.	Śred.	Odch. Std.
Algorytm genetyczny	127	128,9	1,45
Algorytm zachłanny	157	159,4	1,43
Przeszukiwanie lokalne	162	185,2	22,75
Symulowane wyżarzanie	126	127,3	0,95

Źródło: opracowanie własne

Dla zbioru danych 100_20_23_9_D1 otrzymane wyniki przedstawiono w tabeli 16. Dla tego zbioru danych wszystkie algorytmy osiągnęły takie same wyniki, poza przeszukiwaniem lokalnym, które miało większe odchylenie standardowe.

Tabela 16: Wyniki dla zbioru danych 100_20_23_9_D1

Metoda obliczeniowa	Min.	Śred.	Odch. Std.
Algorytm genetyczny	172	172	0
Algorytm zachłanny	172	172	0
Przeszukiwanie lokalne	172	224,4	45,34
Symulowane wyżarzanie	172	172	0

Źródło: opracowanie własne

Dla zbioru danych 100_20_46_15 otrzymane wyniki przedstawiono w tabeli 17. Dla tego zbioru danych symulowane wyżarzanie osiągało najlepszy wyniki. Algorytm genetyczny miał minimalnie większe odchylenie standardowe, jednak wyniki minimalne były podobne jak dla symulowanego wyżarzania. Algorytm zachłanny osiągnął gorsze minimalne rozwiązania, wraz z większym odchyleniem standardowym. Przeszukiwanie lokalne osiągnęło podobne minimalne rozwiązania co poprzednia heurystyka, jednak z jeszcze większym odchyleniem.

Tabela 17: Wyniki dla zbioru danych 100_20_46_15

Metoda obliczeniowa	Min.	Śred.	Odch. Std.
Algorytm genetyczny	165	167	1,76
Algorytm zachłanny	191	196,9	3,14
Przeszukiwanie lokalne	193	214,9	20,05
Symulowane wyżarzanie	161	163,7	2,36

Źródło: opracowanie własne

Dla zbioru danych 100_20_47_9 otrzymane wyniki przedstawiono w tabeli 18. Dla tego zbioru danych symulowane wyżarzanie osiągało najlepszy wyniki. Algorytm genetyczny miał minimalnie większe odchylenie standardowe, jednak wyniki minimalne były podobne jak dla symulowanego wyżarzania. Algorytm zachłanny osiągnął gorsze minimalne rozwiązania, wraz z większym odchyleniem standardowym. Przeszukiwanie lokalne osiągnęło podobne minimalne rozwiązania co poprzednia heurystyka, jednak z jeszcze większym odchyleniem.

Tabela 18: Wyniki dla zbioru danych 100_20_47_9

Metoda obliczeniowa	Min.	Śred.	Odch. Std.
Algorytm genetyczny	125	125,6	0,84
Algorytm zachłanny	164	168,4	2,07
Przeszukiwanie lokalne	164	200,1	29,76
Symulowane wyżarzanie	124	125,1	0,88

Źródło: opracowanie własne

Dla zbioru danych 100_20_65_15 otrzymane wyniki przedstawiono w tabeli 19. Dla tego zbioru danych wszystkie algorytmy osiągnęły takie same wyniki, poza przeszukiwaniem lokalnym, które miało większe odchylenie standardowe.

Tabela 19: Wyniki dla zbioru danych 100_20_65_15

Metoda obliczeniowa	Min.	Śred.	Odch. Std.
Algorytm genetyczny	205	205	0
Algorytm zachłanny	205	205	0
Przeszukiwanie lokalne	205	229,3	23,51
Symulowane wyżarzanie	205	205	0

Źródło: opracowanie własne

Dla zbioru danych 100_20_65_9 otrzymane wyniki przedstawiono w tabeli 20. Dla tego zbioru danych symulowane wyżarzanie osiągało najlepszy wyniki. Algorytm genetyczny miał minimalnie większe odchylenie standardowe, jednak wyniki minimalne były podobne jak dla symulowanego wyżarzania. Algorytm zachłanny osiągnął gorsze minimalne rozwiązania, wraz z większym odchyleniem standardowym. Przeszukiwanie lokalne osiągnęło podobne minimalne rozwiązania co poprzednia heurystyka, jednak z jeszcze większym odchyleniem.

Tabela 20: Wyniki dla zbioru danych 100_20_65_9

Metoda obliczeniowa	Min.	Śred.	Odch. Std.
Algorytm genetyczny	126	128,3	1,77
Algorytm zachłanny	149	153,2	1,93
Przeszukiwanie lokalne	171	191	17,69
Symulowane wyżarzanie	125	125,5	0,53

Źródło: opracowanie własne

Dla zbioru danych 100_5_48_9 otrzymane wyniki przedstawiono w tabeli 21. Dla tego zbioru danych symulowane wyżarzanie osiągało najlepszy wyniki. Algorytm genetyczny miał minimalnie większe odchylenie standardowe, jednak wyniki minimalne były podobne jak dla symulowanego wyżarzania. Algorytm zachłanny osiągnął gorsze minimalne rozwiązania, wraz z większym odchyleniem standardowym. Przeszukiwanie lokalne osiągnęło podobne minimalne rozwiązania co poprzednia heurystyka, jednak z jeszcze większym odchyleniem.

Tabela 21: Wyniki dla zbioru danych 100_5_48_9

Metoda obliczeniowa	Min.	Śred.	Odch. Std.
Algorytm genetyczny	492	493,5	0,85
Algorytm zachłanny	513	518,5	3,57
Przeszukiwanie lokalne	509	526,8	15,86
Symulowane wyżarzanie	491	491,6	0,52

Źródło: opracowanie własne

Dla zbioru danych 100_5_64_15 otrzymane wyniki przedstawiono w tabeli 22. Dla tego zbioru danych symulowane wyżarzanie osiągało najlepszy wyniki. Algorytm genetyczny miał minimalnie większe odchylenie standardowe, jednak wyniki minimalne były podobne jak dla symulowanego wyżarzania. Algorytm zachłanny osiągnął gorsze minimalne rozwiązania, wraz z większym odchyleniem standardowym. Przeszukiwanie lokalne osiągnęło podobne minimalne rozwiązania co poprzednia heurystyka, jednak z jeszcze większym odchyleniem.

Tabela 22: Wyniki dla zbioru danych 100_5_64_15

Metoda obliczeniowa	Min.	Śred.	Odch. Std.
Algorytm genetyczny	482	484,4	1,07
Algorytm zachłanny	508	514,4	4,33
Przeszukiwanie lokalne	521	554,2	37,88
Symulowane wyżarzanie	482	482,7	0,48

Źródło: opracowanie własne

Dla zbioru danych 100_5_64_9 otrzymane wyniki przedstawiono w tabeli 23. Dla tego zbioru danych symulowane wyżarzanie osiągało najlepszy wyniki. Algorytm genetyczny miał minimalnie większe odchylenie standardowe, jednak wyniki minimalne były podobne jak dla symulowanego wyżarzania. Algorytm zachłanny osiągnął gorsze minimalne rozwiązania, wraz z większym odchyleniem standardowym. Przeszukiwanie lokalne osiągnęło lepsze minimalne rozwiązania niż poprzednia heurystyka, jednak z jeszcze większym odchyleniem.

Tabela 23: Wyniki dla zbioru danych 100_5_64_9

Metoda obliczeniowa	Min.	Śred.	Odch. Std.
Algorytm genetyczny	475	476,8	1,03
Algorytm zachłanny	504	508,3	4,19
Przeszukiwanie lokalne	495	527	32,15
Symulowane wyżarzanie	476	476	0

Źródło: opracowanie własne

Dla zbioru danych 100_5_20_9_D3 otrzymane wyniki przedstawiono w tabeli 24. Dla tego zbioru danych symulowane wyżarzanie osiągało najlepszy wyniki. Algorytm genetyczny miał minimalnie większe odchylenie standardowe, jednak wyniki minimalne były podobne jak dla symulowanego wyżarzania. Algorytm zachłanny osiągnął gorsze minimalne rozwiązania, wraz z większym odchyleniem standardowym. Przeszukiwanie lokalne osiągnęło lepsze minimalne rozwiązania niż poprzednia heurystyka, jednak z jeszcze większym odchyleniem.

Tabela 24: Wyniki dla zbioru danych 100_5_20_9_D3

Metoda obliczeniowa	Min.	Śred.	Odch. Std.
Algorytm genetyczny	387	388,3	1,06
Algorytm zachłanny	428	433,1	3,41
Przeszukiwanie lokalne	402	438,2	27,57
Symulowane wyżarzanie	387	387,9	0,32

Źródło: opracowanie własne

Dla zbioru danych 100_5_22_15 otrzymane wyniki przedstawiono w tabeli 25. Dla tego zbioru danych symulowane wyżarzanie osiągało najlepszy wyniki. Algorytm genetyczny miał minimalnie większe odchylenie standardowe, jednak wyniki minimalne były podobne jak dla symulowanego wyżarzania. Algorytm zachłanny osiągnął gorsze minimalne rozwiązania, wraz z większym odchyleniem standardowym. Przeszukiwanie lokalne osiągnęło lepsze minimalne rozwiązania niż poprzednia heurystyka, jednak z jeszcze większym odchyleniem.

Tabela 25: Wyniki dla zbioru danych 100_5_22_15

Metoda obliczeniowa	Min.	Śred.	Odch. Std.
Algorytm genetyczny	485	487,3	1,16
Algorytm zachłanny	526	534,5	5,84
Przeszukiwanie lokalne	487	530,4	48,26
Symulowane wyżarzanie	485	485,7	0,48

Źródło: opracowanie własne

Dla zbioru danych 100_5_46_15 otrzymane wyniki przedstawiono w tabeli 26. Dla tego zbioru danych symulowane wyżarzanie osiągało najlepszy wyniki. Algorytm genetyczny miał minimalnie mniejsze odchylenie standardowe, a wyniki minimalne były podobne jak dla symulowanego wyżarzania. Algorytm zachłanny osiągnął gorsze minimalne rozwiązania, wraz z większym odchyleniem standardowym. Przeszukiwanie lokalne osiągnęło lepsze minimalne rozwiązania niż poprzednia heurystyka, jednak z jeszcze większym odchyleniem.

Tabela 26: Wyniki dla zbioru danych 100_5_46_15

Metoda obliczeniowa	Min.	Śred.	Odch. Std.
Algorytm genetyczny	531	534,3	3,65
Algorytm zachłanny	620	630,4	5,64
Przeszukiwanie lokalne	557	601,2	41,96
Symulowane wyżarzanie	529	532,1	4,33

Źródło: opracowanie własne

Dla zbioru danych 200_10_128_15 otrzymane wyniki przedstawiono w tabeli 27. Dla tego zbioru danych symulowane wyżarzanie osiągało najlepszy wyniki. Algorytm genetyczny miał minimalnie większe odchylenie standardowe, jednak wyniki minimalne były podobne jak dla symulowanego wyżarzania. Algorytm zachłanny osiągnął gorsze minimalne rozwiązania, wraz z większym odchyleniem standardowym. Przeszukiwanie lokalne osiągnęło lepsze minimalne rozwiązania niż poprzednia heurystyka, jednak z jeszcze większym odchyleniem.

Tabela 27: Wyniki dla zbioru danych 200_10_128_15

Metoda obliczeniowa	Min.	Śred.	Odch. Std.
Algorytm genetyczny	461	463,4	2,37
Algorytm zachłanny	539	549,1	7,94
Przeszukiwanie lokalne	514	586,8	47,64
Symulowane wyżarzanie	460	460,9	0,32

Źródło: opracowanie własne

Dla zbioru danych 200_10_135_9_D6 otrzymane wyniki przedstawiono w tabeli 28. Dla tego zbioru danych symulowane wyżarzanie osiągało najlepszy wyniki. Algorytm genetyczny miał minimalnie większe odchylenie standardowe, jednak wyniki minimalne były podobne jak dla symulowanego wyżarzania. Algorytm zachłanny osiągnął gorsze minimalne rozwiązania, wraz z większym odchyleniem standardowym. Przeszukiwanie lokalne osiągnęło gorsze minimalne rozwiązania co poprzednia heurystyka, dodatkowo z jeszcze większym odchyleniem.

Tabela 28: Wyniki dla zbioru danych 200_10_135_9_D6

Metoda obliczeniowa	Min.	Śred.	Odch. Std.
Algorytm genetyczny	534	534	0
Algorytm zachłanny	589	596,2	4,61
Przeszukiwanie lokalne	630	705,3	59,08
Symulowane wyżarzanie	534	534	0

Źródło: opracowanie własne

Dla zbioru danych 200_10_50_15 otrzymane wyniki przedstawiono w tabeli 29. Dla tego zbioru danych symulowane wyżarzanie osiągało najlepszy wyniki. Algorytm genetyczny miał minimalnie większe odchylenie standardowe, jednak wyniki minimalne były podobne jak dla symulowanego wyżarzania. Algorytm zachłanny osiągnął gorsze minimalne rozwiązania, wraz z większym odchyleniem standardowym. Przeszukiwanie lokalne osiągnęło podobne minimalne rozwiązania co poprzednia heurystyka, jednak z jeszcze większym odchyleniem.

Tabela 29: Wyniki dla zbioru danych 200_10_50_15

Metoda obliczeniowa	Min.	Śred.	Odch. Std.
Algorytm genetyczny	486	486,8	0,63
Algorytm zachłanny	516	520	2,21
Przeszukiwanie lokalne	510	613,1	63,08
Symulowane wyżarzanie	485	485,5	0,53

Źródło: opracowanie własne

Dla zbioru danych 200_10_50_9 otrzymane wyniki przedstawiono w tabeli 30. Dla tego zbioru danych symulowane wyżarzanie osiągało najlepszy wyniki. Algorytm genetyczny miał minimalnie większe odchylenie standardowe, jednak wyniki minimalne były podobne jak dla symulowanego wyżarzania. Algorytm zachłanny osiągnął gorsze minimalne rozwiązania, wraz z większym odchyleniem standardowym. Przeszukiwanie lokalne osiągnęło podobne minimalne rozwiązania co poprzednia heurystyka, jednak z jeszcze większym odchyleniem.

Tabela 30: Wyniki dla zbioru danych 200_10_50_9

Metoda obliczeniowa	Min.	Śred.	Odch. Std.
Algorytm genetyczny	486	487,1	0,74
Algorytm zachłanny	501	503,3	1,34
Przeszukiwanie lokalne	507	616,5	84,02
Symulowane wyżarzanie	486	486	0

Źródło: opracowanie własne

Dla zbioru danych 200_10_84_9 otrzymane wyniki przedstawiono w tabeli 31. Dla tego zbioru danych symulowane wyżarzanie osiągało najlepszy wyniki. Algorytm genetyczny miał minimalnie większe odchylenie standardowe, jednak wyniki minimalne były podobne jak dla symulowanego wyżarzania. Algorytm zachłanny osiągnął gorsze minimalne rozwiązania, wraz z większym odchyleniem standardowym. Przeszukiwanie lokalne osiągnęło podobne minimalne rozwiązania co poprzednia heurystyka, jednak z jeszcze większym odchyleniem.

Tabela 31: Wyniki dla zbioru danych 200_10_84_9

Metoda obliczeniowa	Min.	Śred.	Odch. Std.
Algorytm genetyczny	507	507,5	0,53
Algorytm zachłanny	538	543,2	2,86
Przeszukiwanie lokalne	541	621,3	67,69
Symulowane wyżarzanie	506	506,8	0,63

Źródło: opracowanie własne

Dla zbioru danych 200_10_85_15 otrzymane wyniki przedstawiono w tabeli 32. Dla tego zbioru danych symulowane wyżarzanie osiągało najlepszy wyniki. Algorytm genetyczny miał minimalnie większe odchylenie standardowe, jednak wyniki minimalne były podobne jak dla symulowanego wyżarzania. Algorytm zachłanny osiągnął gorsze minimalne rozwiązania, wraz z większym odchyleniem standardowym. Przeszukiwanie lokalne osiągnęło gorsze minimalne rozwiązania co poprzednia heurystyka, dodatkowo z jeszcze większym odchyleniem.

Tabela 32: Wyniki dla zbioru danych 200_10_85_15

Metoda obliczeniowa	Min.	Śred.	Odch. Std.
Algorytm genetyczny	474	475,4	0,84
Algorytm zachłanny	497	501,3	2,06
Przeszukiwanie lokalne	554	624,4	60,16
Symulowane wyżarzanie	474	474	0

Źródło: opracowanie własne

Dla zbioru danych 200_20_145_15 otrzymane wyniki przedstawiono w tabeli 33. Dla tego zbioru danych symulowane wyżarzanie osiągało najlepszy wyniki. Algorytm genetyczny miał minimalnie większe odchylenie standardowe, jednak wyniki minimalne były podobne jak dla symulowanego wyżarzania. Algorytm zachłanny osiągnął gorsze minimalne rozwiązania, wraz z większym odchyleniem standardowym. Przeszukiwanie lokalne osiągnęło podobne minimalne rozwiązania co poprzednia heurystyka, jednak z jeszcze większym odchyleniem.

Tabela 33: Wyniki dla zbioru danych 200_20_145_15

Metoda obliczeniowa	Min.	Śred.	Odch. Std.
Algorytm genetyczny	236	237,4	1,51
Algorytm zachłanny	280	286	2,91
Przeszukiwanie lokalne	275	323,6	27,46
Symulowane wyżarzanie	236	236,9	0,74

Źródło: opracowanie własne

Dla zbioru danych 200_20_150_9_D5 otrzymane wyniki przedstawiono w tabeli 34. Dla tego zbioru danych wszystkie algorytmy osiągnęły takie same wyniki, poza przeszukiwaniem lokalnym, które miało większe odchylenie standardowe.

Tabela 34: Wyniki dla zbioru danych 200_20_150_9_D5

Metoda obliczeniowa	Min.	Śred.	Odch. Std.
Algorytm genetyczny	900	900	0
Algorytm zachłanny	900	900	0
Przeszukiwanie lokalne	900	912	21,71
Symulowane wyżarzanie	900	900	0

Źródło: opracowanie własne

Dla zbioru danych 200_20_54_15 otrzymane wyniki przedstawiono w tabeli 35. Dla tego zbioru danych symulowane wyżarzanie osiągało najlepszy wyniki. Algorytm genetyczny miał minimalnie większe odchylenie standardowe, jednak wyniki minimalne były podobne jak dla symulowanego wyżarzania. Algorytm zachłanny osiągnął gorsze minimalne rozwiązania, wraz z większym odchyleniem standardowym. Przeszukiwanie lokalne osiągnęło gorsze minimalne rozwiązania co poprzednia heurystyka, dodatkowo z jeszcze większym odchyleniem.

Tabela 35: Wyniki dla zbioru danych 200_20_54_15

Metoda obliczeniowa	Min.	Śred.	Odch. Std.
Algorytm genetyczny	258	260,3	1,57
Algorytm zachłanny	296	300,7	3,13
Przeszukiwanie lokalne	367	406,6	20,75
Symulowane wyżarzanie	257	258,6	0,84

Źródło: opracowanie własne

Dla zbioru danych 200_20_55_9 otrzymane wyniki przedstawiono w tabeli 36. Dla tego zbioru danych symulowane wyżarzanie osiągało najlepszy wyniki. Algorytm genetyczny miał minimalnie większe odchylenie standardowe, jednak wyniki minimalne były podobne jak dla symulowanego wyżarzania. Algorytm zachłanny osiągnął gorsze minimalne rozwiązania, wraz z większym odchyleniem standardowym. Przeszukiwanie lokalne osiągnęło gorsze minimalne rozwiązania co poprzednia heurystyka, dodatkowo z jeszcze większym odchyleniem.

Tabela 36: Wyniki dla zbioru danych 200_20_55_9

Metoda obliczeniowa	Min.	Śred.	Odch. Std.
Algorytm genetyczny	248	248,5	0,71
Algorytm zachłanny	277	280,5	2,46
Przeszukiwanie lokalne	296	337,9	23,85
Symulowane wyżarzanie	247	248	0,67

Źródło: opracowanie własne

Dla zbioru danych 200_20_97_15 otrzymane wyniki przedstawiono w tabeli 37. Dla tego zbioru danych symulowane wyżarzanie osiągało najlepszy wyniki. Algorytm genetyczny miał minimalnie większe odchylenie standardowe, jednak wyniki minimalne były podobne jak dla symulowanego wyżarzania. Algorytm zachłanny osiągnął gorsze minimalne rozwiązania, wraz z większym odchyleniem standardowym. Przeszukiwanie lokalne osiągnęło lepsze minimalne rozwiązania niż poprzednia heurystyka, jednak z jeszcze większym odchyleniem.

Tabela 37: Wyniki dla zbioru danych 200_20_97_15

Metoda obliczeniowa	Min.	Śred.	Odch. Std.
Algorytm genetyczny	336	336	0
Algorytm zachłanny	372	374,1	1,2
Przeszukiwanie lokalne	336	400,7	42,22
Symulowane wyżarzanie	336	336	0

Źródło: opracowanie własne

Dla zbioru danych 200_20_97_9 otrzymane wyniki przedstawiono w tabeli 38. Dla tego zbioru danych symulowane wyżarzanie osiągało najlepszy wyniki. Algorytm genetyczny miał minimalnie większe odchylenie standardowe, jednak wyniki minimalne były podobne jak dla

symulowanego wyżarzania. Algorytm zachłanny osiągnął gorsze minimalne rozwiązania, wraz z większym odchyleniem standardowym. Przeszukiwanie lokalne osiągnęło gorsze minimalne rozwiązania co poprzednia heurystyka, dodatkowo z jeszcze większym odchyleniem.

Tabela 38: Wyniki dla zbioru danych 200_20_97_9

Metoda obliczeniowa	Min.	Śred.	Odch. Std.
Algorytm genetyczny	241	243,2	1,55
Algorytm zachłanny	273	276,1	2,18
Przeszukiwanie lokalne	298	353,2	38,24
Symulowane wyżarzanie	241	242,3	0,95

Źródło: opracowanie własne

Dla zbioru danych 200_40_130_9_D4 otrzymane wyniki przedstawiono w tabeli 39. Dla tego zbioru danych wszystkie algorytmy osiągnęły takie same wyniki, poza przeszukiwaniem lokalnym, które miało większe odchylenie standardowe.

Tabela 39: Wyniki dla zbioru danych 200_40_130_9_D4

Metoda obliczeniowa	Min.	Śred.	Odch. Std.
Algorytm genetyczny	513	513	0
Algorytm zachłanny	513	513	0
Przeszukiwanie lokalne	513	539,7	48,74
Symulowane wyżarzanie	513	513	0

Źródło: opracowanie własne

Dla zbioru danych 200_40_133_15 otrzymane wyniki przedstawiono w tabeli 40. Dla tego zbioru danych symulowane wyżarzanie osiągało najlepszy wyniki. Algorytm genetyczny miał minimalnie większe odchylenie standardowe, jednak wyniki minimalne były podobne jak dla symulowanego wyżarzania. Algorytm zachłanny osiągnął gorsze minimalne rozwiązania, wraz z większym odchyleniem standardowym. Przeszukiwanie lokalne osiągnęło gorsze minimalne rozwiązania co poprzednia heurystyka, dodatkowo z jeszcze większym odchyleniem.

Tabela 40: Wyniki dla zbioru danych 200_40_133_15

Metoda obliczeniowa	Min.	Śred.	Odch. Std.
Algorytm genetyczny	136	137,6	1,51
Algorytm zachłanny	179	182	1,56
Przeszukiwanie lokalne	202	222,8	18,21
Symulowane wyżarzanie	139	140,3	0,95

Źródło: opracowanie własne

Dla zbioru danych 200_40_45_15 otrzymane wyniki przedstawiono w tabeli 41. Dla tego zbioru danych symulowane wyżarzanie osiągało najlepszy wyniki. Algorytm genetyczny miał minimalnie większe odchylenie standardowe, jednak wyniki minimalne były podobne jak dla symulowanego wyżarzania. Algorytm zachłanny osiągnął gorsze minimalne rozwiązania, wraz z większym odchyleniem standardowym. Przeszukiwanie lokalne osiągnęło gorsze minimalne rozwiązania co poprzednia heurystyka, dodatkowo z jeszcze większym odchyleniem.

Tabela 41: Wyniki dla zbioru danych 200_40_45_15

Metoda obliczeniowa	Min.	Śred.	Odch. Std.
Algorytm genetyczny	159	159	0
Algorytm zachłanny	170	170,9	0,57
Przeszukiwanie lokalne	188	221,6	20,53
Symulowane wyżarzanie	159	159	0

Źródło: opracowanie własne

Dla zbioru danych 200_40_45_9 otrzymane wyniki przedstawiono w tabeli 42. Dla tego zbioru danych symulowane wyżarzanie osiągało najlepszy wyniki. Algorytm genetyczny miał minimalnie większe odchylenie standardowe, jednak wyniki minimalne były podobne jak dla symulowanego wyżarzania. Algorytm zachłanny osiągnął gorsze minimalne rozwiązania, wraz z większym odchyleniem standardowym. Przeszukiwanie lokalne osiągnęło gorsze minimalne rozwiązania co poprzednia heurystyka, dodatkowo z jeszcze większym odchyleniem.

Tabela 42: Wyniki dla zbioru danych 200_40_45_9

Metoda obliczeniowa	Min.	Śred.	Odch. Std.
Algorytm genetyczny	138	141,1	1,97
Algorytm zachłanny	155	158,3	1,77
Przeszukiwanie lokalne	206	230,4	17,61
Symulowane wyżarzanie	141	141,6	0,7

Źródło: opracowanie własne

Dla zbioru danych 200_40_90_9 otrzymane wyniki przedstawiono w tabeli 43. Dla tego zbioru danych symulowane wyżarzanie osiągało najlepszy wyniki. Algorytm genetyczny miał minimalnie większe odchylenie standardowe, jednak wyniki minimalne były podobne jak dla symulowanego wyżarzania. Algorytm zachłanny osiągnął gorsze minimalne rozwiązania, wraz z większym odchyleniem standardowym. Przeszukiwanie lokalne osiągnęło gorsze minimalne rozwiązania co poprzednia heurystyka, dodatkowo z jeszcze większym odchyleniem.

Tabela 43: Wyniki dla zbioru danych 200_40_90_9

Metoda obliczeniowa	Min.	Śred.	Odch. Std.
Algorytm genetyczny	135	138,3	1,7
Algorytm zachłanny	165	169,8	2,35
Przeszukiwanie lokalne	203	235,3	25,11
Symulowane wyżarzanie	137	139,3	1,49

Źródło: opracowanie własne

Dla zbioru danych 200_40_91_15 otrzymane wyniki przedstawiono w tabeli 44. Dla tego zbioru danych symulowane wyżarzanie osiągało najlepszy wyniki. Algorytm genetyczny miał minimalnie większe odchylenie standardowe, jednak wyniki minimalne były podobne jak dla symulowanego wyżarzania. Algorytm zachłanny osiągnął gorsze minimalne rozwiązania, wraz z większym odchyleniem standardowym. Przeszukiwanie lokalne osiągnęło gorsze minimalne rozwiązania co poprzednia heurystyka, dodatkowo z jeszcze większym odchyleniem.

Tabela 44: Wyniki dla zbioru danych 200_40_91_15

Metoda obliczeniowa	Min.	Śred.	Odch. Std.
Algorytm genetyczny	131	133,5	1,43
Algorytm zachłanny	181	181,9	0,74
Przeszukiwanie lokalne	213	242	26,5
Symulowane wyżarzanie	132	135	1,63

Źródło: opracowanie własne

W tabeli 45 przedstawiono uśrednione wartości dla wszystkich zbiorów danych. Widać na nich, że najlepsze wyniki osiągnęło symulowane wyżarzanie, za równo jak chodzi o średnią osiągniętych minimalnych wyników, jak i o ich odchylenie standardowe. Algorytm genetyczny osiągnął odrobine gorsze wyniki w obu kategoriach. Algorytm zachłanny jak i przeszukiwanie lokalne osiągnęły za to o wiele gorsze rezultaty, jednakże ten pierwszy z nich miał o wiele niższe odchylenie standardowe, niż ten drugi z nich.

Tabela 45: Wyniki uśrednione dla wszystkich zbiorów danych

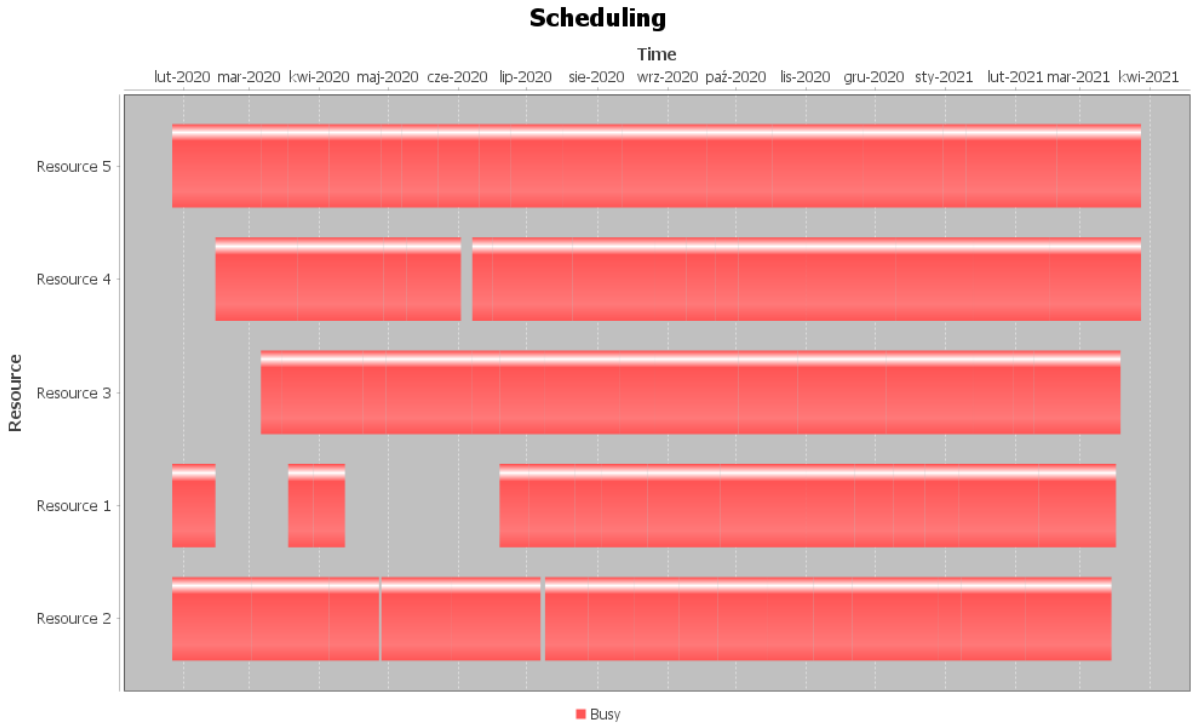
Metoda obliczeniowa	Min.	Śred.	Odch. Std.
Algorytm genetyczny	296,56	297,86	0,98
Algorytm zachłanny	323,98	327,78	2,42
Przeszukiwanie lokalne	325,91	363,72	30,38
Symulowane wyżarzanie	296,35	297,07	0,56

Źródło: opracowanie własne

Jak widać prezentowane wyniki, a w związku z tym skuteczność metod, różni się w zależności od danego zbioru testowego. Mimo tego, można zauważyć pewne układające się wzorce. Niektóre z metod bywają regularnie skuteczniejsze niż inne z nich.

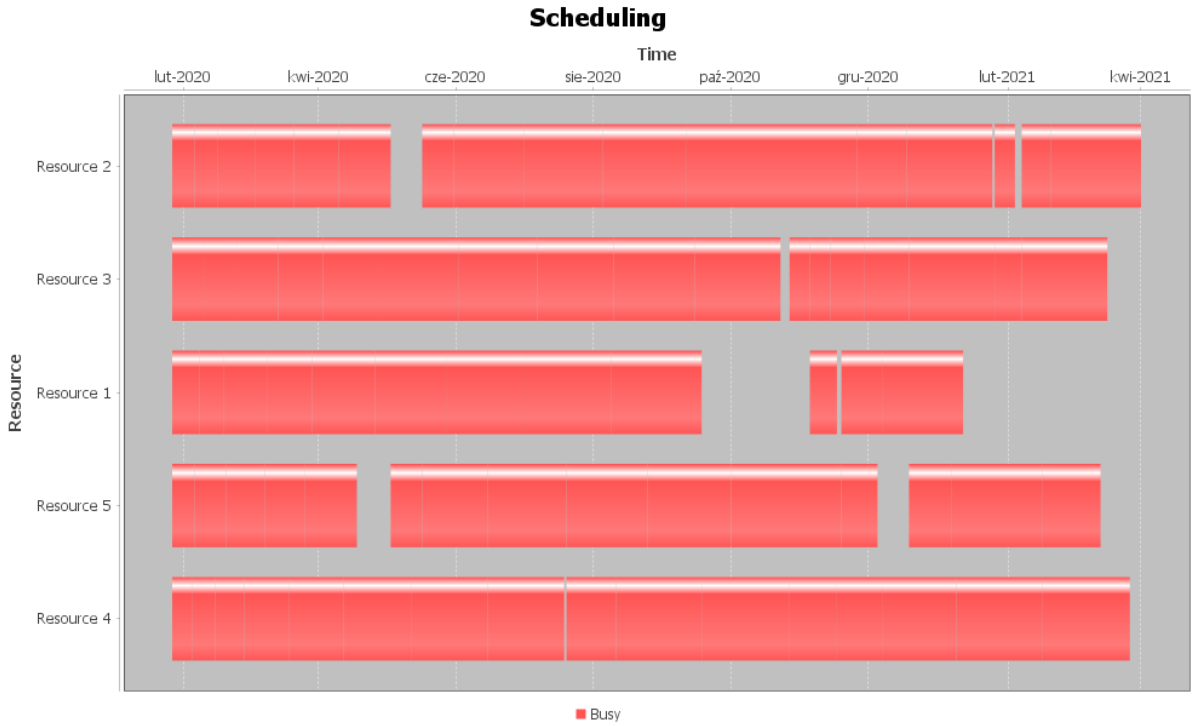
4.2. Wizualizacja otrzymanych harmonogramów rozwiązań w badaniach

Dodatkowo, aby zwizualizować różnice w generowanych rozwiązaniach, przy zastosowaniu biblioteki JFreeChart, postanowiłem wygenerować przykładowe harmonogramy dla metod w następującej kolejności: dla przeszukiwania lokalnego (Rysunek 5), dla algorytmu zachłannego (Rysunek 6), dla algorytmu genetycznego (Rysunek 7), oraz dla symulowanego wyżarzania (Rysunek 8). Czerwone paski symbolizują, że dany zasób pracuje nad jakimś zadaniem w danym przedziale czasu. Szare puste obszary pomiędzy oznaczają, że nie robi on nic i bezczynnie czeka. Ze względu na to, że biblioteka do wykresów, jaką zastosowałem wymagała podania dat, postanowiłem generować je od dnia 1 lutego 2020 i przyjąć jako jednostkę dzień pracy.



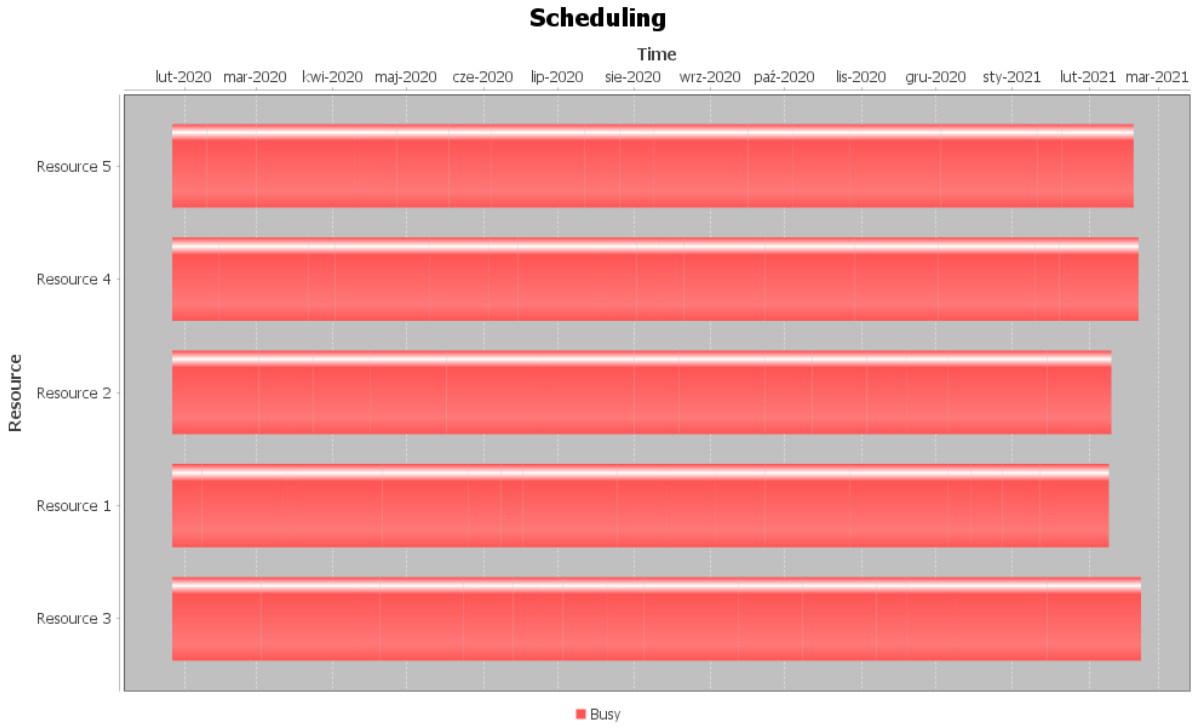
Rysunek 5: Przykładowy harmonogram dla przeszukiwania lokalnego

Źródło: opracowanie własne



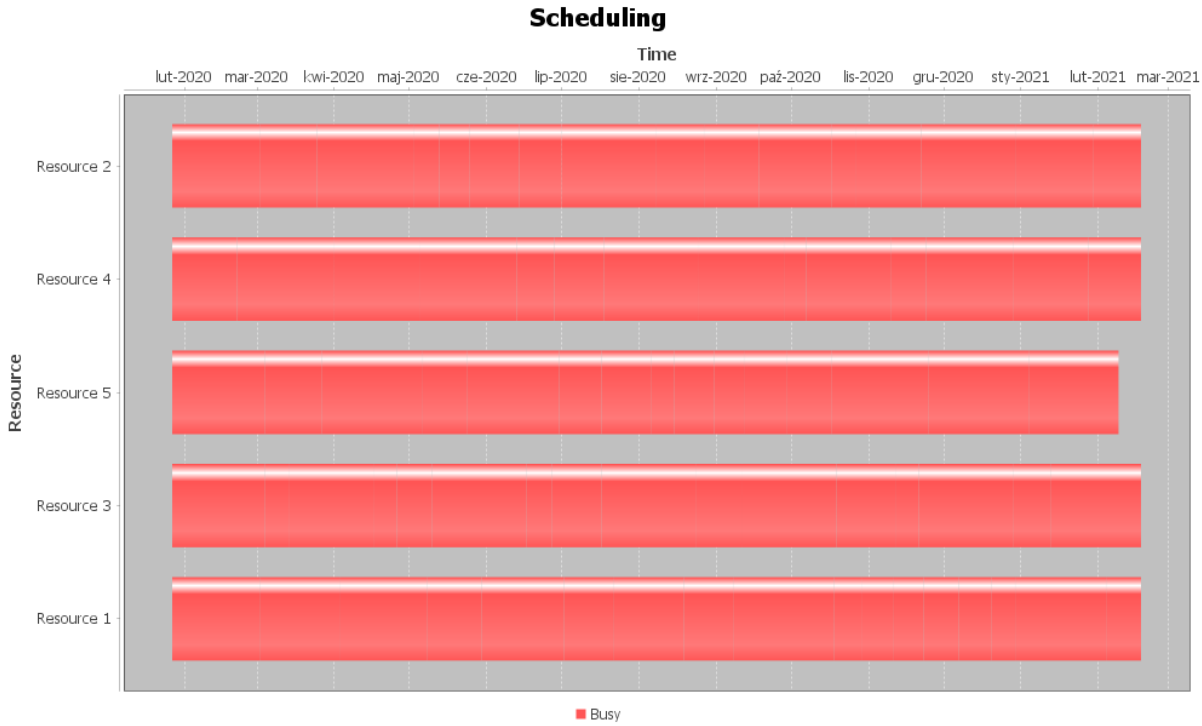
Rysunek 6: Przykładowy harmonogram dla algorytmu zachłannego

Źródło: opracowanie własne



Rysunek 7: Przykładowy harmonogram dla algorytmu genetycznego

Źródło: opracowanie własne



Rysunek 8: Przykładowy harmonogram dla symulowanego wyżarzania

Źródło: opracowanie własne

Na tych wykresach wyraźnie widać, dlaczego rozwiązania z pierwszej połowy są takie nieoptymalne – występują na nich przerwy, w których dane zasoby nie pracują nad żadnymi zasobami. Widać także, że te z drugiej połowy są bliskie optymalnym, ponieważ nie ma już tutaj praktycznie żadnych przerw, a wszystkie zasoby pracują przez podobny czas. Nawet te graficzne przedstawienia, pokazują, że te bardziej skomplikowane metody, jakimi są symulowane wyżarzanie i algorytm genetyczny osiągają lepsze wyniki, niż prostszy algorytm zachłanny i przeszukiwanie lokalne.

4.3. Wnioski z przeprowadzonego eksperymentu

Na podstawie przeprowadzonej analizy uzyskanych wyników z przeprowadzonego badania polegającego na dziesięciokrotnym uruchomieniu wybranych i dostrojonych przeze mnie metod heurystycznych, czyli: przeszukiwania lokalnego, symulowanego wyżarzania, algorytmu genetycznego i algorytmu zachłannego. Metody te zastosowano dla wybranego

zaprezentowanego wcześniej rozwiązywania problemu planowania projektów z wieloma wymaganymi umiejętnościami i ograniczonymi zasobami i sformułowano następujące wnioski.

Przeszukiwanie lokalnie osiągnęło zdecydowanie najgorsze rezultaty, z o wiele większym odchyleniem standardowym od reszty – pokazuje to, że jest to niestabilna metoda. Algorytm zachłanny osiągnął podobne rezultaty minimalne, jednakże osiągając o wiele lepsze rezultaty średnie, czyniąc go bardziej niezawodnym. Poprzednie dwie metody (tzn. przeszukiwanie lokalne i algorytm zachłanny) zostały zaimplementowane głownie w celu porównania ich z dwoma pozostałymi, bardziej skomplikowanymi metodami. Widać wyraźnie, że zarówno algorytm genetyczny, jak i symulowane wyżarzanie, osiągnęły znacznie lepsze rozwiązania, na lekką korzyść drugiego z nich.

W przypadku tych bardziej skomplikowanych metod osiągnięte wyniki pozwalają na twierdzenie, że mogą one być bardzo blisko do rozwiązań optymalnych, a czasem także je osiągając. Jest to oczywiście nie możliwe do weryfikacji w przypadku tak wielkich zbiorów danych, ze względu na to, że jest to problem NP-trudny, jednakże podobne minimalne rezultaty, wraz z bardzo małym odchyleniem standardowym wskazują na taki rozwój sytuacji.

Biorąc pod uwagę, że średni czas iteracji takiej metody wyniósł lekko ponad minutę, jest to zupełnie akceptowalny czas który użytkownicy mogą poczekać na otrzymanie gotowego rezultatu, który będzie lepszy niż te uzyskane o wiele prostszymi metodami. Oznacza to, że te dwie metody które osiągnęły lepsze rezultaty, czyli symulowane wyżarzanie i algorytm genetyczny, mogły by być z powodzeniem wykorzystywane w prawdziwym życiu. Dodatkowym atutem dla tych osiągniętych przez nich wyników, jest fakt, że zbiór tych danych został faktycznie opracowany bazie przeprowadzonych już projektów w firmie Volvo.

Zakończenie

Planowanie projektów jest dziedziną istotną w zarządzaniu i osiąganiu celów w każdym przedsiębiorstwie, a wraz z rozwojem technologii możliwe jest rozwiązywanie trudności z nim związanym na wiele różnych sposobów. Jednym z nich jest problem planowania projektu wymagającego użycia wielu umiejętności oraz ograniczonych zasobów. W powyższej pracy został przedstawiony proces wykorzystania w tym celu metaheurystyk.

Po przeanalizowaniu problemów obliczeniowych pojawiających się w procesie realizacji projektu (problem spełnialności, problem komiwojażera, programowanie nieliniowe, problem planowania projektu z wieloma wymaganymi umiejętnościami i ograniczonymi zasobami) udało się spełnić cel pracy, czyli zdefiniowanie ich elementów i zaimplementowanie zaprezentowanych metod generowania rozwiązań: algorytmu zachłannego, przeszukiwania lokalnego, symulowanego wyżarzania oraz algorytmu genetycznego.

Dzięki przeprowadzonemu w ramach procedury badawczej eksperymentowi, otrzymano rezultaty wskazujące na pozytywny wpływ wykorzystania metaheurystyk w opisanym problemie. Wygenerowane w ten sposób harmonogramy są bliskie optymalnym, a czas ich generacji jest akceptowalny do użytku w przypadkach występujących w rzeczywistości.

W dalszym kroku można by było bardziej rozbudować te metody, na przykład zaimplementować więcej operatorów dla algorytmów genetycznych, które być może pozwoliłyby na uzyskanie lepszych rezultatów. Można także spróbować polepszyć prędkość działania aplikacji, a optymalizacja ta mogłaby być osiągnięta przez ulepszanie jej kodu, zmianę języka programowania, czy też próbę wykorzystania możliwości obliczeniowych, na jakie pozwalają karty graficzne.

Można także zaimplementować i zbadać zupełnie nowe algorytmy czy metaheurystyki. Jednym z takich rozwiązań może być zastosowanie algorytmów rojowych – na przykład algorytm roju pszczół czy algorytm roju cząstek. Niestety jednak wymagają one zdefiniowania funkcji odległości pomiędzy dwoma rozwiązaniami, co dla wykorzystanej w pracy reprezentacji problemu nie jest możliwe, ponieważ nie da się zdefiniować miary odległości w przypadku kwestii przydziałów różnych zasobów do zadań.

Literatura

Akter, S., Nahar, N., Shahadat, M., & Andersson, K. (2019). A New Crossover Technique to Improve Genetic Algorithm and Its Application to TSP. *2019 International Conference on Electrical, Computer and Communication Engineering (ECCE)*, (pp. 1-6). doi:10.1109/ECACE.2019.8679367

Bazaraa, M. S., & Shetty, C. M. (1979). *Nonlinear programming. Theory and algorithms.* John Wiley & Sons.

Beardwood, J., Halton, J. H., & Hammersley, J. M. (1959). The shortest path through many points. *Mathematical Proceedings of the Cambridge Philosophical Society, 55*(4), pp. 299–327. doi:10.1017/S0305004100034095

Bryant, R. E., German, S., Velev, M. N., & Murray, N. V. (1999). *Microprocessor Verification Using Efficient Decision Procedures for a Logic of Equality with Uninterpreted Functions.* Springer Berlin Heidelberg. doi:10.1007/3-540-48754-9_1

Caserta, M., & Voss, S. (2010). Metaheuristics: Intelligent Problem Solving. In *Matheuristics: Hybridizing Metaheuristics and Mathematical Programming* (pp. 1-38). Springer US. doi:10.1007/978-1-4419-1306-7_1

Cook, S. (1971). *The complexity of theorem proving procedures.* Proceedings of the Third Annual ACM Symposium on Theory of Computing. doi:10.1145/800157.805047

Dantzig, G. B., & Ramser, J. H. (1959). The Truck Dispatching Problem. *Management Science, 1*(6), pp. 80-91.

Fogel, D., & Ghozeil, A. (1997). A note on representations and variation operators. *IEEE Transactions on Evolutionary Computation, 1*(2), pp. 159-161. doi:10.1109/4235.687882

Garey, M., & Johnson, D. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness.* New York: W.H. Freeman.

Goldberg, D. E., & Deb, K. (1991). A Comparative Analysis of Selection Schemes Used in Genetic Algorithms. *Foundations of Genetic Algorithms, 1*, 69-93. doi:10.1016/B978-0-08-050684-5.50008-2

Grucza, B., Trocki, M., Bukłaha, E., Juchniewicz, P., & Wyrozębski, P. (2009). *Zarządzanie w warunkach zmiany.* Demos Polska.

He, J., Chen, T., & Yao, X. (2015). On the Easiest and Hardest Fitness Functions. *IEEE Transactions on Evolutionary Computation, 19*(2), 295–305. doi:10.1109/tevc.2014.2318025

Himmelblau, D. M. (1972). *Applied Nonlinear Programming.* McGraw-Hill.

IEEE Standard for Floating-Point Arithmetic. (2008). 1-70. doi:10.1109/IEEESTD.2008.4610935

Keane, A. (1996). A brief comparison of some evolutionary optimization methods. In *Modern Heuristic Search Methods.* John Wiley.

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983, may). Optimization by Simulated Annealing. *Science, 220*(4598), pp. 671-680. doi:10.1126/science.220.4598.671

Luo, J., Fujimura, S., El Baz, D., & Plazolles, B. (2019). GPU based parallel genetic algorithm for solving an energy efficient dynamic flexible flow shop scheduling problem. *Journal of Parallel and Distributed Computing, 133*, 244-257. doi:https://doi.org/10.1016/j.jpdc.2018.07.022

Michalewicz, Z., & Fogel, D. B. (2004). *How to Solve It: Modern Heuristics.* Springer. doi:10.1007/978-3-662-07807-5

Michalewicz, Z., & Schoenauer, M. (1996). Evolutionary algorithms for constrained parameter optimization problems. *Evolutionary Computation, 4*(1), pp. 1-32.

Mitchell, M. (1996). *An Introduction to Genetic Algorithms.* MIT Press.

Myszkowski, P. B., Skowroński, M., & Sikora, K. (2015). A new benchmark dataset for Multi-Skill Resource-Constrained Project Scheduling Problem. In M. Ganzha, L. Maciaszek, & M. Paprzycki (Ed.), *Proceedings of the 2015 Federated Conference on Computer Science and Information Systems. 5*, pp. 129–138. ACSIS. doi:10.15439/2015F273

Nam, G.-J., Sakallah, K., & Rutenbar, R. (2002). A new FPGA detailed routing approach via search-based Boolean satisfiability. IEEE. doi:10.1109/TCAD.2002.1004311

Otman, A., Abouchabaka, J., & Tajani, C. (2012, 3). Analyzing the Performance of Mutation Operators to Solve the Travelling. *Int. J. Emerg. Sci., 2*.

Pietras, P., & Szmit, M. (2003). *Zarządzanie projektami. Wybrane metody i techniki* . Oficyna Księgarsko-Wydawnicza „Horyzont".

Rego, C., Gamboa, D., Glover, F., & Osterman, C. (2011). Traveling salesman problem heuristics: Leading methods, implementations and latest advances. *European Journal of Operational Research, 211*(3), 427-441. doi:10.1016/j.ejor.2010.09.010

Santos, M. A., & Tereso, A. P. (2011). On the Multi-mode, Multi-skill Resource Constrained Project Scheduling Problem – A Software Application. *Soft Computing in Industrial*

Applications (pp. 239-248). Springer Berlin Heidelberg. doi:10.1007/978-3-642-20505-7_21

Skowroński, M., Myszkowski, P., & Podlodowski, Ł. (2013). Novel heuristic solutions for Multi–Skill Resource–Constrained Project Scheduling Problem. *Annals of Computer Science and Information Systems, 1*, pp. 159-166.

Slowik, A., & Kwasnicka, H. (2018). Nature Inspired Methods and Their Industry Applications—Swarm Intelligence Algorithms. *IEEE Transactions on Industrial Informatics, 3*(14), 1004–1015. doi:10.1109/TII.2017.2786782

Sörensen, K., & Glover, F. (2013). Metaheuristics. In *Encyclopedia of Operations Research and Management Science* (pp. 960-970). Springer US.

Syswerda, G. (1991). Schedule optimization using genetic algorithms. In *Handbook of Genetic Algorithms* (pp. 332–349). Van Nostrand Reinhold.

Trocki, M. (2012). *Nowoczesne zarządzanie projektami.* Polskie Wydawnictwo Ekonomiczne.

Vizel, Y., Weissenbacher, G., & Malik, S. (2015). Boolean Satisfiability Solvers and Their Applications in Model Checking. *Proceedings of the IEEE, 103*(11), pp. 2021-2035. doi:10.1109/JPROC.2015.2455034

Wyrozębski, P., Juchniewicz, M., & Metelski, W. (2012). *Wiedza, dojrzałość, ryzyko w zarządzaniu projektami.* Oficyna Wydawnicza SGH.

Spis tabel

Tabela 1: Przykładowe prawdopodobieństwa dla akceptacji nowego rozwiązania w symulowanym wyżarzaniu przy zmiennej ocenie nowego rozwiązania 25
Tabela 2: Przykładowe prawdopodobieństwa dla akceptacji nowego rozwiązania w symulowanym wyżarzaniu przy zmiennej temperaturze 25
Tabela 3: Wyniki dla zbioru danych 10_3_5_3 32
Tabela 4: Wyniki dla zbioru danych 10_5_8_5 33
Tabela 5: Wyniki dla zbioru danych 10_7_10_7 33
Tabela 6: Wyniki dla zbioru danych 15_3_5_3 33
Tabela 7: Wyniki dla zbioru danych 15_6_10_6 34
Tabela 8: Wyniki dla zbioru danych 15_9_12_9 34
Tabela 9: Wyniki dla zbioru danych 100_10_26_15 34
Tabela 10: Wyniki dla zbioru danych 100_10_27_9_D2 35
Tabela 11: Wyniki dla zbioru danych 100_10_47_9 35
Tabela 12: Wyniki dla zbioru danych 100_10_48_15 36
Tabela 13: Wyniki dla zbioru danych 100_10_64_9 36
Tabela 14: Wyniki dla zbioru danych 100_10_65_15 37
Tabela 15: Wyniki dla zbioru danych 100_20_22_15 37
Tabela 16: Wyniki dla zbioru danych 100_20_23_9_D1 37
Tabela 17: Wyniki dla zbioru danych 100_20_46_15 38
Tabela 18: Wyniki dla zbioru danych 100_20_47_9 38
Tabela 19: Wyniki dla zbioru danych 100_20_65_15 39
Tabela 20: Wyniki dla zbioru danych 100_20_65_9 39
Tabela 21: Wyniki dla zbioru danych 100_5_48_9 40
Tabela 22: Wyniki dla zbioru danych 100_5_64_15 40
Tabela 23: Wyniki dla zbioru danych 100_5_64_9 41
Tabela 24: Wyniki dla zbioru danych 100_5_20_9_D3 41
Tabela 25: Wyniki dla zbioru danych 100_5_22_15 42
Tabela 26: Wyniki dla zbioru danych 100_5_46_15 42
Tabela 27: Wyniki dla zbioru danych 200_10_128_15 43
Tabela 28: Wyniki dla zbioru danych 200_10_135_9_D6 43
Tabela 29: Wyniki dla zbioru danych 200_10_50_15 44

Tabela 30: Wyniki dla zbioru danych 200_10_50_9 44
Tabela 31: Wyniki dla zbioru danych 200_10_84_9 45
Tabela 32: Wyniki dla zbioru danych 200_10_85_15 45
Tabela 33: Wyniki dla zbioru danych 200_20_145_15 46
Tabela 34: Wyniki dla zbioru danych 200_20_150_9_D5 46
Tabela 35: Wyniki dla zbioru danych 200_20_54_15 46
Tabela 36: Wyniki dla zbioru danych 200_20_55_9 47
Tabela 37: Wyniki dla zbioru danych 200_20_97_15 47
Tabela 38: Wyniki dla zbioru danych 200_20_97_9 48
Tabela 39: Wyniki dla zbioru danych 200_40_130_9_D4 48
Tabela 40: Wyniki dla zbioru danych 200_40_133_15 49
Tabela 41: Wyniki dla zbioru danych 200_40_45_15 49
Tabela 42: Wyniki dla zbioru danych 200_40_45_9 50
Tabela 43: Wyniki dla zbioru danych 200_40_90_9 50
Tabela 44: Wyniki dla zbioru danych 200_40_91_15 51
Tabela 45: Wyniki uśrednione dla wszystkich zbiorów danych 51

Spis rysunków

Rysunek 1: Rodzaje działań związanych z realizacją projektów 7
Rysunek 2: Przykładowa ścieżka dla TSP 10
Rysunek 3: Wykres przykładowej funkcji dla programowania nieliniowego 12
Rysunek 4: Taksonomia metod inspirowanych naturą 21
Rysunek 5: Przykładowy harmonogram dla przeszukiwania lokalnego 52
Rysunek 6: Przykładowy harmonogram dla algorytmu zachłannego 53
Rysunek 7: Przykładowy harmonogram dla algorytmu genetycznego 53
Rysunek 8: Przykładowy harmonogram dla symulowanego wyżarzania 54

Spis równań

Równanie 1: Przykładowa funkcja dla programowania nieliniowego 11
Równanie 2: Przykładowe równanie dla SAT 22

Oświadczenia

OŚWIADCZENIE AUTORA (AUTORÓW) PRACY

Świadom odpowiedzialności prawnej oświadczam, że niniejsza praca dyplomowa została napisana przeze mnie samodzielnie. Wszystkie dane, istotne myśli i sformułowania pochodzące z literatury (przytoczone dosłownie lub niedosłownie) są opatrzone odpowiednimi odsyłaczami.

Praca ta w całości ani w części, która zawierałaby znaczne fragmenty przedstawione w pracy jako oryginalne, nie była wcześniej przedmiotem procedur związanych z uzyskaniem tytułu zawodowego w wyższej uczelni.

Oświadczam, że tekst pracy dyplomowej wgrany do systemu APD jest identyczny z tekstem wydrukowanym złożonym w dziekanacie, o ile złożenie pracy w dziekanacie jest wymagane aktualnymi regulacjami Uczelni.

UWAGA: Oświadczenie składane w wersji elektronicznej w systemie APD

OŚWIADCZENIE PROMOTORA

Oświadczam, że niniejsza praca dyplomowa została przygotowana pod moim kierunkiem i spełnia warunki do przedstawienia jej w postępowaniu o nadanie tytułu zawodowego.

Jednocześnie oświadczam, że tematyka pracy jest zgodna z efektami uczenia się określonymi dla kierunku Autora pracy.

UWAGA: Oświadczenie składane w wersji elektronicznej w systemie APD