Spis treści

[1. Wstęp 2](#_Toc90148786)

[2. Problemy obliczeniowe 3](#_Toc90148787)

[2.1. Problem spełnialności (SAT) 3](#_Toc90148788)

[2.2. Problem komiwojażera (TSP) 4](#_Toc90148789)

[2.3. Programowanie nieliniowe (NLP) 5](#_Toc90148790)

[2.4. Problem planowania projektu z wieloma wymaganymi umiejętnościami i ograniczonymi zasobami (MSRCPSP) 7](#_Toc90148791)

[3. Elementu problemu 8](#_Toc90148792)

[3.1. Model i wielkość przestrzeni poszukiwań 8](#_Toc90148793)

[3.2. Sąsiedztwo 9](#_Toc90148794)

[3.3. Funkcja oceny, cel i ograniczenia 11](#_Toc90148795)

[4. Metody rozwiązywania problemów 14](#_Toc90148796)

[4.1. Przeszukiwanie lokalne 14](#_Toc90148797)

[4.2. Algorytmy zachłanne 14](#_Toc90148798)

[4.3. Symulowane wyżarzanie 14](#_Toc90148799)

[4.4. Algorytmy genetyczne 14](#_Toc90148800)

[5. Badania 15](#_Toc90148801)

[5.1. Opis sposoby realizacji projektu i metodyki badań 15](#_Toc90148802)

[5.2. Otrzymane wyniki 15](#_Toc90148803)

[6. Podsumowanie 16](#_Toc90148804)

[7. Bibliografia 17](#_Toc90148805)

# Wstęp

# Problemy obliczeniowe

W życiu napotyka się na wiele problemów które są trudne do rozwiązania, zarówno dla człowieka, jak i dla komputerów. Powodów takiego stanu rzeczy może być kilka, między innymi:

Liczba potencjalnych rozwiązań dla danego problemu jest tak duża, że nie jest możliwe przeszukanie wszystkich rozwiązań w zadowalającym nas czasie. Czasem potrafiło by to zająć miliony lat dla niektórych problemów, ale nie raz nawet parę minut 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 ciężko 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 poniższym rozdziale zostaną przedstawione trzy pomocnicze problemy i jeden główny, który jest tematem tej pracy magisterskiej.

## Problem spełnialności (SAT)

Jednym z podstawowych zagadnień rachunku zdań w logice jest problem spełnialności. Polega on na znalezieniu takich wartości zmiennych, które mogą przyjmować tylko wartości prawda lub fałsz, dla których dana formuła logiczna będzie spełniona. Przykładowy fragment takiej formuły może wyglądać następująco:

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).

## Problem komiwojażera (TSP)

W problemie komiwojażera (skrót jego nazwy to TSP) mamy zdefiniowaną listę miast i odległości pomiędzy każda parą z nich. 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, który pozwala na odwiedzenie każdego miasta więcej niż raz. Przykładowe rozwiązanie może wyglądać następująco:

Obraz zawierający tekst, zewnętrzne, ruch uliczny, jasne

Opis wygenerowany automatycznie

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 nim 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).

## Programowanie nieliniowe (NLP)

Kolejna klasą problemów, są te związane z programowaniem nieliniowym. Polegają one na zalezieniu minimum, maksimum, lub punktów zerowych dla danej funkcji. 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:

Możemy zwizualizować tą funkcję przykładowo dla (Michalewicz i Fogel, 2004). Rozwiązaniom niespełniającym ograniczeń nadano zerową wartość:



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.

## Problem planowania projektu z wieloma wymaganymi umiejętnościami i ograniczonymi zasobami (MSRCPSP)

Zbiór danych dla problemu planowania projektów z wieloma wymaganymi umiejętnościami i ograniczonymi zasobami, z którego korzystam w swojej pracy, 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 (P.B., M. i K., 2015). Na nim skupiłem swoje badania, których założenia i wyniki zostaną przedstawione w nadchodzących rozdziałach.

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:

* 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. Dzięki temu jego twórcy dostarczają oparte na rzeczywistości, ale zanonimizowane, zestawy danych odpowiadające rzeczywistym projektom. Jest ich łącznie 42 - 6 łatwiejszych problemów i 36 pełnowymiarowych, które posiadają do:

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

# Elementu problemu

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

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

Zostaną one przestawione w poniższych podrozdziałach na przykładzie omówionych w poprzednim rozdziale przypadków.

## Model i wielkość przestrzeni poszukiwań

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 . Dla 100 zmiennych jest to wartość rzędu .

W przypadku TSP najczęściej spotykanym modelem jest permutacja liczb naturalnych od do . 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 . Dla porównania z poprzednim problemem, dla 100 miast wartość ta jest rzędu

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 , co dla równań z 100 zmiennymi dało by rząd wielkości .

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. Dla takiego sposobu potencjalnych rozwiązań jest , gdzie to liczba zasobów, a to liczba zadań. Dla 100 zasobów i 100 zadań daje to rząd wielkości potencjalnych rozwiązań równy

## 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: i maksymalnej odległości równej , każdy z poniższych punktów byłby sąsiadem:

* (zmiana wartości pierwszej zmiennej).
* (zmiana wartości drugiej zmiennej).
* (zmiana wartości trzeciej zmiennej).
* (zmiana wartości pierwszej zmiennej).
* (zmiana wartości drugiej zmiennej).

Dla MSRCPSP zdecydowałem się na definicje sąsiedztwa inspirowaną wyżej 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.

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.

## Funkcja oceny, cel i ograniczenia

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. 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 – 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:

* 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. 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.

# Metody 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 zgodnie z nazwa metodami heurystycznymi. W związku z tym, w przeciwieństwie do klasycznych algorytmów, nie gwarantują one znalezienia optymalnego rozwiązania.

## Algorytm zachłanny

## Przeszukiwanie lokalne

## Symulowane wyżarzanie

## Algorytm genetyczny

# Badania

## Opis sposoby realizacji projektu i metodyki badań

## Otrzymane wyniki

# Podsumowanie

# Bibliografia

Bryant, R. E., German, S., Velev, M. N. i 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

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

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

He, J., Chen, T. i 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

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. W *Modern Heuristic Search Methods.* John Wiley.

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

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

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

Rego, C., Gamboa, D., Glover, F. i 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