AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE
WYDZIAŁ ELEKTROTECHNIKI, AUTOMATYKI, INFORMATYKI I INŻYNIERII BIOMEDYCZNEJ

Projekt dyplomowy

Zarządzanie zasobami w planowaniu projektu
Resource Management in Project Planning

Autor: Filip Gąciarz
Kierunek studiów: Automatyka i Robotyka
Opiekun pracy: dr inż. Piotr Kadłuczka

Kraków, 2023

Spis treści

Wstęp

Niniejsza praca skupia się na problematyce zarządzania zasobami w planowaniu projektów. Zarządzanie jest nieodłącznym elementem każdego projektu, pozwalającym na przewidywanie, kierowanie oraz tworzenie harmonogramu. Dzięki niemu jesteśmy wstanie określić przed rozpoczęciem prac, jak powinniśmy rozdysponować pracę, jakie ryzyka mogą się wiązać z niedotrzymaniem terminów, a także określić budżet jakim dysponujemy, aby projekt przyniósł zysk. Inteligentne zarządzanie zasobami pozwala też na wyrównanie zapotrzebowania konkretnego zasobu np. ludzkiego w czasie trwania całego projektu. Pozwala to na maksymalizację dochodu przez redukcje kosztów spowodowaną m. in. zatrudnieniem pracownika na cały czas trwania procesu, mimo zapotrzebowania na jego pracę, jedynie na jednym jego etapie. Stworzenie takiego harmonogramu daje nam informację, ile konkretnego zasobu będziemy potrzebować w danym dniu i jaki zakres pracy powinien zostać wykonany w ciągu danego okresu, aby projekt zakończył się sukcesem.

Dlatego problem jest bardzo istotny i rozwiązanie go nie należy do najprostszych zadań. Bez sprawnego zarządzania nie jesteśmy wstanie przeprowadzić projektu, który to będzie w najwyższym stopniu zharmonizowany. Literatura wskazuje, że jedne z lepszych efektów w zarządzaniu zasobami i wyznaczaniu harmonogramu dla projektu można zaobserwować przy wykorzystaniu algorytmów metaheurystycznych. Przeszukują one przestrzeń rozwiązań w celu znalezienia rozwiązania quasi-optymalnego, a zarazem nie potrzebują tak dużych nakładów obliczeniowych jak algorytmy dokładne. Nie wypracowano jednak do tej pory uniwersalnego rozwiązania, które byłoby idealne dla każdego rodzaju projektu.

Celem pracy było stworzenie aplikacji wyrównującej zasoby w projekcie oraz wyznaczającej harmonogram, który wspomagałby kierownika w planowaniu projektu. Dodatkowo opracowanie jej w taki sposób, aby była ona elastyczna i możliwa do wykorzystania w wielu typach projektów, a osoba korzystająca sama mogła wybrać metody wyznaczania rozwiązania, priorytety czy ograniczenia.

Algorytm w oparciu o metodę ścieżki krytycznej oraz metaheurystyczny algorytm genetyczny dąży do poprawy wyniku oraz wyrównania, poprzez metody takie jak zamiana terminów, podział ciągłości zadań, rozdzielenie zadań na dłuży czas oraz wydłużenie czasu trwania przedsięwzięcia. Pozwala także na nadanie priorytetów dla konkretnych zasobów, które są interesujące dla użytkownika programu, a także konfiguracji algorytmu poprzez operowanie metodami selekcji czy dobranie odpowiedniej funkcji celu. Użytkownik sam definiuje wymagania przedsięwzięcia swojego zadania wraz z relacjami, czasami trwania i zapotrzebowaniem na zasoby, na których podstawie algorytm wyznacza przykładowe rozwiązanie końcowe, które może zostać wykorzystane przez kierownika projektu. Możliwe jest także dobranie przez użytkownika liczby iteracji algorytmu, jak i kryterium stopu, po którego osiągnięciu algorytm kończy pracę i zwraca znaleziony wynik.

Praca składa się z czterech rozdziałów, w których zostały opisane kolejno:

- Teoretyczny aspekt problemu jakim jest zarządzanie zasobami wraz z definicjami, zarysem historycznym podejmowania problemu wyrównywania, problematyką komputerowego kierowania projektem oraz modelem matematycznym opisywanego problemu.
- Schemat opracowanego algorytmu genetycznego z opisem każdego elementu odpowiedzialnego za szukanie poprawnego wyniku w toku działania programu.
- Szczegółową prezentację implementacji algorytmu wraz z pseudokodem i opisem funkcji w nim występujących.
- Eksperymenty obliczeniowe pokazujące działania programu dla różnych przypadków oraz scenariuszy przygotowanych na podstawie przykładowych dziedzin, gdzie algorytm może zostać wykorzystany.

Rozdział I: Zarządzanie zasobami w przedsiębiorstwie

W rozdziale zaprezentowano teoretyczne ujęcie problemu zarządzania zasobami w planowaniu projektu, a także tworzenia harmonogramu na jego podstawie. Przedstawiono najważniejsze definicje i elementy każdego przedsięwzięcia z wykorzystaniem zasobów. Wskazano również, w jaki sposób historycznie podejmowano próby rozwiązania problemy ich wyrównania oraz alokacji na przestrzeni lat. Pokazano istotne elementy dla komputerowego zarządzania procesem oraz dziedziny w jakich można zastosować ten typ zarządzania. Ponadto scharakteryzowano problem za pomocą modelu matematycznego.

1.1 Zarys problematyki zarządzania zasobami

Zarządzanie zasobami to proces dystrybucji i wykorzystania zasobów organizacji w najbardziej opłacalny sposób. Zasoby przedsiębiorstwa obejmują fundusze, produkty, sprzęt i siłę roboczą. Kluczem do zarządzania zasobami jest posiadanie przez organizację wystarczających zasobów do wspierania płynnego prowadzenia jej podstawowych działań, przy jednoczesnym zapewnieniu, że nie ma nadmiaru zasobów, aby nie zostały zmarnowane. Proces ten pozwala na przewidywanie, kierowanie cyklem życia projektu, a także jest wykorzystywany do kontrolowania, jak efektywnie godziny pracy są wykorzystywane przez firmę. [1] Każda działalność o charakterze gospodarczym dąży do maksymalizacji zysku poprzez redukcję kosztów i zwiększenie przychodów, aby przygotowany plan projektu był w jak najwyższym stopniu harmonizacji. Aby dobrze zrozumieć działanie mechanizmów systemu zarządzania zasobami warto jest znać podstawowe pojęcia.

Projekt, przedsięwzięcie (ang. Project) to unikalny zbiór współzależnych czynności (zadań, operacji), wraz z określonym momentem rozpoczęcia i zakończenia realizowany przez osobę prywatną lub organizację dla osiągnięcia przyjętych celów określonych wymaganiami, w ramach określonego czasu, kosztów i zasobów. [2]

Czynność to zadanie, operacja lub proces wymagające określonego czasu i (lub) zasobów dla zrealizowania. [2] Określenie jej wymaga przygotowania planu realizacji założonych celów z uwzględnieniem ograniczeń posiadanych zasobów.

Ma następujące cechy:

- określony czas trwania,
- terminy rozpoczęcia i zakończenia są umiejscowione i nie mogą wykraczać poza ramy ustalone przez czas określony dla całego przedsięwzięcia,
- zależności pomiędzy czynnościami - zdefiniowane relacjami następstwa,
- wymaganiami odnośnie tych zaangażowanych zasobów (ludzi, materiałów, maszyn),
- realizacją związane są określone koszty,
- powinny mieć przypisaną osobę lub organizację odpowiedzialną za ich wykonanie.

Zasoby to każda zdefiniowana zmienna (ludzie, finanse, materiały, maszyny), która jest wymagana do zakończenia działania i może ograniczać projekt. [2]

Zasoby można podzielić na: [3]

- Odnawialne – zależy od liczby jednostek, w każdej chwili jest ograniczona (robotnicy, zespoły robocze, maszyny),
- Nieodnawialne – zużycie w każdym przedziale czasu jest ograniczony (surowce, materiały budowlane),
- Podwójnie ograniczone – ograniczona jest zarówno liczba jednostek, jak i zużycie (np. środki finansowe),
- Przywłaszczalne – kiedy jednostka zasobu może zostać zabrana aktualnie wykonywanemu procesowi i przydzielona innemu.
- Nieprzywłaszczane – kiedy jednostka nie może zostać zabrana z aktualnie wykonywanego procesu,

Relacja odwzorowuje logiczny związek pomiędzy dwoma lub więcej czynnościami wykonywanymi ramach projektu. [2] Aby relacja była jednoznaczna konieczne jest podanie **poprzednika** (czyli czynności, która poprzez relację warunkuje rozpoczęcie lub zakończenie innej czynności), **następnika** (czynności, której możliwości realizacji są uwarunkowane poprzez relację) oraz rodzaju relacji.

Istnieją następujące rodzaje relacji: [4]

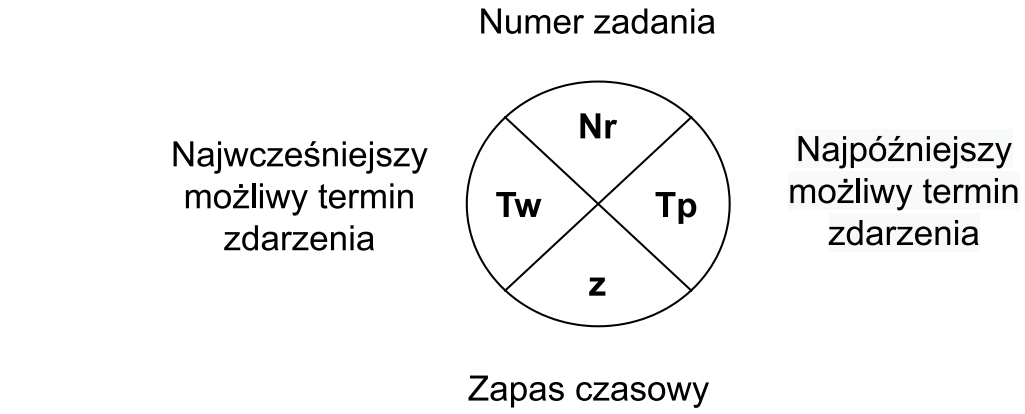
- koniec-początek (KP) – oznacza, że następnik może rozpocząć się nie wcześniej niż po zakończeniu poprzednika – jest to najpowszechniej stosowany w badaniach typ relacji,
- koniec-koniec (KK) – oznacza, że następnik może zakończyć się nie wcześniej niż po zakończeniu poprzednika,
- początek-początek (PP) – oznacza, że następnik może rozpocząć się nie wcześniej niż po rozpoczęciu poprzednika,
- początek-koniec (PK) – oznacza, że następnik może zakończyć się nie wcześniej niż po rozpoczęciu poprzednika,

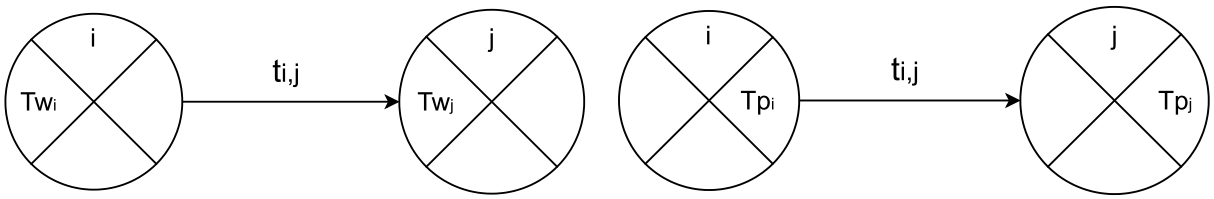
W ramach relacji zależności dopuszczalne jest stosowanie opóźnienia czasowego (ang. time lag). Może być ono wyrażane w jednostkach czasu bądź procentach. Wartość dodatnia wpływa na opóźnienie w czasie dostępności następnika, natomiast wartość ujemną pozwalając na wcześniejszą dostępność czynności. Wielkość opóźnienia określa wartość graniczą (maksymalną lub minimalną). Są to elementy niezbędne w przebiegu każdego przedsięwzięcia.

Znając relacje oraz zasoby, w szczególności czas trwania czynności w projekcie jesteśmy wstanie wykazać miejsca, których zamiany przyczynią się do optymalizacji. Jedną z metod wyznaczania terminów między kolejnymi procesami jest algorytm CPM.

Metoda ścieżki krytycznej (ang. Critical Path Method) jest to deterministyczna technika opracowana w latach 50. XX wieku w Wielkiej Brytanii i w Stanach Zjednoczonych. Opiera się na wykresie sieciowym, na którym dokonywane są obliczenia czasu trwania projektu i rezerw. Aby ją zastosować niezbędna jest skończona lista wszystkich czynności w projekcie, informacja o zapotrzebowaniu każdej z nich na zasoby, a także informacja o zależnościach między czynnościami. W tej technice korzystamy z wykresu typu AoA oraz wykonujemy kolejne etapy: [5]

1. Sporządzenie wykresu sieciowego
2. Wyliczenie terminów wystąpienia zdarzeń sieci
3. Wyznaczenie rezerw czasu i ścieżki krytycznej





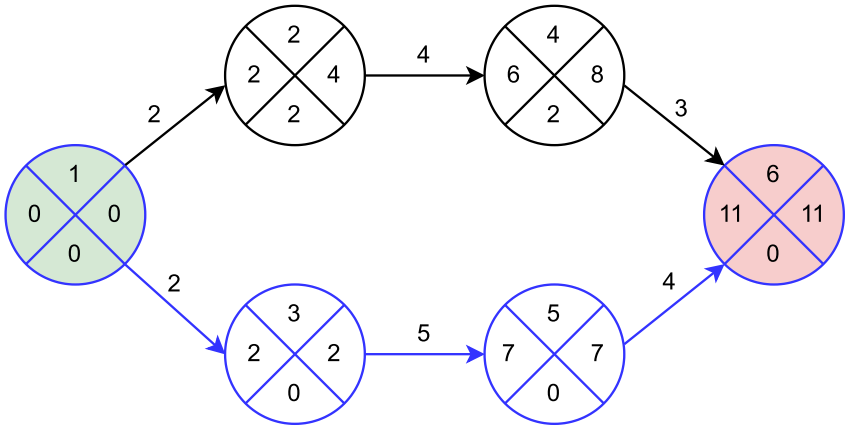
Rysunek 1 Przeliczanie terminów dla metody CPM

$$Tw_j = \max\{Tw_i + t_{i,j}\} \quad (1) \qquad Tp_i = \min\{Tp_j - t_{i,j}\} \quad (2)$$

Wyznaczenie najwcześniejszego terminu polega na przeliczeniu sieci po przenumerowanych węzłach i nadaniu terminu poprzez dodanie terminu poprzednika z czasem trwania, a w przypadku kilku czynności poprzedzających, poprzez dodanie czasu maksymalnego. Najpóźniejszy termin zdarzenia przeliczany jest od końca, poprzez odjęcie od terminu czasu trwania do tej czynności. Przy kilku czynnościach wybierana jest wartość minimalna.

$$z_i = Tp_i - Tw_i \quad (3)$$

Zapas jest różnicą pomiędzy terminem najpóźniejszy a terminem najwcześniejszym.



Rysunek 2 Wykres sieciowy dla metody CPM

Zielonym kolorem została oznaczona czynność początkowe, czerwonym czynność końcowa, natomiast na niebiesko została pokazana ścieżka krytyczna. Dla ścieżki krytycznej zapas zawsze jest równy 0.

Ścieżka krytyczna to sekwencja czynności w ramach projektu, która przebiega od zadania początkowego do końcowego oraz suma czasów jej trwania jest równa czasowi trwania projektu. [2]

Czynności wchodzące w skład tej ścieżki nie mogą ulec zmianie. Są one podstawą projektu, dlatego ich modyfikacja spowoduje zmianę czasu realizacji (np. opóźnienie przedsięwzięcia). Pozostałe czynności mogą być dostosowywane, muszą się one jednak zajmować w wyznaczonym zapasie. Niedotrzymanie wyznaczonych w ten sposób terminów może spowodować opóźnienie w założonym czasie realizacji projektu.

Istnieje wiele technik zarządzania zasobami, które w różny sposób próbują doprowadzić do sukcesu przedsięwzięcia. W projekcie chcemy skupić się na jak największej harmonizacji pracy poprzez wyrównanie zasobów. Obecnie najbardziej popularnymi technikami skupionymi na wyrównaniu są poziomowanie i alokacja:

Poziomowanie zasobów (ang. Resource leveling) to proces zmian harmonogramu w celu redukcji różnicy między maksimum a minimum wartości wymagań dotyczących zasobów. [2]

Alokacja zasobów (ang. Resource allocation) to planowanie alokacji czynności oraz wymaganych przez nie zasobów tak, aby ograniczenia dostępności zasobów i (lub) czas realizacji nie został przekroczone. [2]

Problem poziomowania zasobów określany w skrócie jako **RLP** dzieli się na dwa podstawowe kryteria, na podstawie których jego implementacja i zastosowania mogą mieć różne przeznaczenia:

- z ograniczeniami zasobowymi i bez,
- z relacjami pierwszeństwa i bez.

W implementacji będziemy skupiać się na RLP z relacjami pierwszeństwa, ponieważ zależy nam na tym, aby zadania były wykonywane z zachowaniem relacji i kolejności. Natomiast bez ograniczeń związanych z brakiem zasobów, które powodują komplikację w harmonizacji oraz przesunięcia w czasie realizacji. Uwzględnienie tego warunku, utrudniłoby osiągnięcie największego wygładzenia wykresu zapotrzebowania na zasoby.

Powyższe techniki pozwalają wyrównywać poprzez osiągnięcie stałego progu zapotrzebowania, dzięki czemu projekt w mniejszym stopniu jest narażony na niekontrolowane wydatki z powodu skoków zapotrzebowania. Wykorzystanie tych technik wiąże się z zastosowaniem poniższych strategii:

- Przerywanie ciągłości zadania w trakcie jego realizacji. Jest to możliwe do wprowadzenia w zadaniach, które nie muszą być wykonywane nieprzerwanie.
- Delegacja części zasobów do innego zadania. Pozwoli to na przyspieszenie realizacji czynności kosztem spowolnienia drugiej, ale zyskiem równoległości.
- Wydłużenie czasu przedsięwzięcia. Pozwala nam na zmniejszenie kosztów, ceną niezrealizowania projektu na uwcześnię ustalony deadline projektu. Jest to decyzja, która musi być uzgodniona obopólnie, ale może dać wiele zysku.
- Zamiana terminu rozpoczęcia i zakończenia czynności z rozpatrzeniem opóźnienia czasowego.
- Wydłużenie czasu realizacji czynności, kosztem zaoszczędzenia/wyrównania zasobów. Przydzielając miej zasobów (np. ludzkich) kosztem czasu.

Każda taka decyzja może mieć wiele czynników i wybór konkretnego sposobu powinien być adekwatny do rzeczywistego problemu i wymagań lub uzgodnień. Dlatego właśnie są one podejmowane na stopniu kierowniczym.

Przerzucenie zasobów do innego zadania powoduje przyspieszenie innej czynności i zrównoleglenie pracy na cały okres projektu. Jednakowoż nie każdy (np. pracownik) jest wstanie wykonywać każde zadanie z tą samą szybkością. Tym samym taka strategia poziomowanie ni może zostać wykorzystana we wszystkich przedsięwzięciach.

1.2 Stosowane algorytmy

Na przestrzeni lat wiele osób próbowało podjąć temat zarządzania zasobami na różne sposoby. Ich umiejętne dysponowanie zapewnia, że projekt zostanie ukończony na czas i po kosztach oraz, że jakość będzie taka jak wcześniej zdefiniowano. [6] W literaturze możemy natrafić na różne metody rozwiązywania problemu wyrównywania zasobów. Mogą one być podzielone na trzy grupy:

- Algorytmy dokładne
- Algorytmy heurystyczne
- Algorytmy metaheurystyczne

Pierwszą grupą metod do rozwiązywania RLP są **algorytmy dokładne**. Opierają się one na niejawnym wyliczeniu, programowaniu całkowitoliczbowym oraz technikach programowania dynamicznego. Takiego rodzaju wyliczenia podjęli się m in. Easa [7], która to zastosowała technikę programowania mieszalnej liczby binarnej i całkowitoliczbowej, Bandelloni i in. [8] korzystając z programowania dynamicznego, Rieck i in. [9] stosując formułę liniowego modelu mieszanych liczb całkowitych i techniki przetwarzania wstępnego z redukcją domeny opartą o formuły inteligentne czasu dyskretnego czy Ahuja [10]. Najpopularniejszą jednak techniką dokładną jest branch-and-bound (podziału i ograniczeń), ponieważ jest jedyną która pozwala uzyskać wynik optymalny przy akceptowalnym nakładzie obliczeniowym. Neumann i Zimmermann [11] opisali tę procedurę, dzięki czemu zredukowali zbiór wszystkich możliwych rozwiązań poprzez kolejne planowanie działań dla przybliżonego rozwiązania.

Ponieważ algorytmy dokładne potrzebowały dużych nakładów obliczeniowych zaczęto stosować **algorytmy heurystyczne**. Jest to grupa, która pozwala na uzyskanie dobrego wyniku (optimum lokalne), nie gwarantując przy tym, że znalezione rozwiązanie jest optymalne (optimum globalne). Algorytmy z tej grupy mają dwojaki rodzaj, ponieważ część służy do wykonalnego rozwiązania problemu, a część do ich ulepszania. Pierwszą procedurę zaproponowali Burgess i Killebrew [12] ustawiając metodę najmniejszych kwadratów jako miarę wydajności algorytmu. Ta procedura ponownie dostosowuje czas rozpoczęcia każdej czynności i redukuje zmienność. Na początku ustala najwcześniejszy czas rozpoczęcia czynności, a następnie zgodnie z priorytetem (najwcześniejszy czas zakończenia) wybiera najlepszy czas rozpoczęcia, czyli taki dla którego ma minimalną sumę kwadratów zużycia zasobów. Jest to jednak algorytm jednoprzebiegowy (raz przechodzący przez cały projekt), co nie daje nam pewności co do optymalnego wyniku. Problem ten został rozwiązany w późniejszych latach przez Burmana [13]. Inny algorytm zwany MOM, tym razem wieloprzebiegowy, zaproponował Harris [14], który ustalił minimalny moment jako miarę wydajności minimalizująca dzienne wahania wykorzystania zasobów przy niezmienionym całkowitym czasie trwania przedsięwzięcia. Ten algorytm był także udoskonalany w kolejnych latach metodą pakowania (PACK) [15], która rozpoznawała interakcje sieciowe w bardziej dogłębnej analizie. Podejście heurystyczne stosowali też Hiyassat [16], Jettendra i in. [17], czy Meridith i Mandel [18] których praca została wykorzystana w programach takich jak Primavera czy SuperProject.

Biorąc pod uwagę różnorodność problemów i zasobów, żadna pojedyncza metoda heurystyczna nie jest wstanie sobie poradzić z uzyskaniem optymalnego rozwiązania. Dlatego jak możemy przeczytać w *Scheduing construction projects using evlutionary algorithm* [19] konieczne jest stosowanie algorytmów **metaheurystycznych** do złożonych realistycznych projektów. Jest to grupa algorytmów wysokiego poziomu oparta na zachowaniach fizycznych, biologicznych czy zwierzęcych, które można zastosować dla dowolnego problemu optymalizacji. Najpopularniejszymi metodami metaheurystycznymi są: poszukiwanie tabu, symulowane wyżarzanie, algorytm ewolucyjny/genetyczny, wyszukiwanie rozporoszone, optymalizacja kolonii mrówek czy optymalizacja roju cząstek. Jednakże najpopularniejszymi stosowanymi algorytmami do RLP jest algorytm ewolucyjny (lub konkretniej genetyczny) oraz

symulowane wyżarzanie. Algorytm ewolucyjny wykorzystali np. Kounousis i Georgiou [20] do optymalizacji struktur czy Haidar i in. [21] do wyboru sprzętu do robót ziemnych. Jak wykazały badania algorytmy ewolucyjne mają potencjał na skuteczne rozwiązanie problemu związanego z harmonogramowaniem. Między innymi Li i Love [22] użyli go do optymalizacji czasu i kosztów, natomiast Hegazy [23] do optymalizacji i alokacji zasobów. Na przestrzeni lat zaprezentowano też metody hybrydowe, których promotorami byli Son i Skibniewski [24] czy Alsayegh i Hariga [25] którzy połączyli symulowane wyżarzanie i optymalizację roju cząstek. Próbowano również metod opartych na sztucznych sieciach neuronowych (Adeli i Karim) [26] czy systemach eksperckich (Kanet i Adelsberger) [27].

Jak widać RLP jest tematem, który próbowano i próbuje się rozwiązać wieloma sposobami. Jest on również problemem NP-trudnym nawet jak weźmiemy pod uwagę jeden zasób, dlatego znalezienie optymalnego rozwiązania nie jest proste. W pracy postarano się rozwiązać ten problem korzystając z algorytmu genetycznego, który to najbardziej pasuje to poziomowania zasobów w przedsiębiorstwie.

1.3 Komputerowe wspomaganie zarządzania zasobami

Celem projektu jest stworzenie aplikacji pozwalającej na komputerowe zarządzać zasobami oraz harmonizować pracę dla całego przebiegu projektu. Wyznaczenie takiego harmonogramu pracy dla danego przedsięwzięcia może w dużym stopniu ułatwić pracę kierownikowi, który jest odpowiedzialny za prowadzenie projektu. Dzisiejsze projekty stają wobec problemów takich jak zwiększona konkurencja, jakość czy braki zasobów. Co przyczynia się do zwiększenia ich stopnia trudności i złożoności oraz zapotrzebowania na specjalistyczną wiedzę. Powoduje to, że określenie deterministycznych zasobów dla konkretnego zadania staje się zadaniem niełatwym. Określenie czasu trwania dla konkretnego zadania również nie jest proste, ze względu na pojawiające się coraz częściej problemy. Powodują one brak możliwości znalezienia drogi alternatywnej, co może skutkować zatrzymaniem przedsięwzięcia na dłuższy czas. Jak możemy przeczytać w raporcie z przeprowadzonych badań The Standish Group [28] dla projektów informatycznych z 2020 roku, tylko jedna trzecia (31%) kończy się sukcesem, połowa (50%) została oceniona jako zagrożona (opóźniona, przekroczony budżetu, niewystarczająca jakość), a co piąty (19%) kończy się niepowodzeniem. Pokazuje to, jak trudnym zadaniem jest stworzenie harmonogramu projektu, który zakończyłby się pełnym sukcesem. Podczas realizacji mogą pojawić się także nieprzewidziane wydarzenia takie jak np. choroba pracownika, powodująca brak tego zasobu w planowanym okresie, przy równoczesnym braku możliwości natychmiastowego jego zastępstwa.

Przez te wszystkie problemy nie jesteśmy wstanie wyznaczyć harmonogramu, który by w pełni kontrolował projektem bez pomocy kierownika, który jest wstanie zareagować na kryzysy zaistniałe w projekcie. Komputerowe wersje programów mogą jedynie wspomagać projekt zarządzania zasobami, a także zaproponować scenariusze harmonogramu pracy, których wybór powinien dokonać kierownik na podstawie rzeczywistego problemu.

Klasyczne podejście do poziomowania zasobów niestety nie można wykorzystać w każdej branży. Przemysł IT wymyślił o wiele lepsze metody zarządzania projektami i zespołami takie jak Scrum czy Agile. Natomiast jest wiele dziedzin, gdzie wyrównanie jest ciągle potrzebne i korzystanie z jego technik daje wiele korzyści. Takimi dziedzinami są m.in.:

- Budownictwo
- Produkcja
- Sprzedaż
- Transport

1.4 Model matematyczny zagadnienia

Wyrównywanie oraz alokacja zasobów może pomóc nam osiągnąć satysfakcjonujące nas rezultaty, natomiast aby zamodelować, a następnie zaimplementować ten problem przyjęto następujące założenia:

- Terminy wykonania oraz czasy zadań są całkowitoliczbowe,
- Dostępność zasobów jest nieograniczona,
- Zapotrzebowanie na zasoby dla każdego zadania jest znane i stałe,
- Relacje kolejnościowe między zadaniami muszą być zachowane,
- Stałe wydajność zasobów (bilansowe z zapotrzebowaniem np. każdy pracownik wykona zadanie w tym samym czasie - nie jesteśmy wstanie w algorytmie określić zależności, że jedna osoba wykona coś szybciej, a druga wolniej)

W przedsięwzięciu będziemy posługiwać się czasem konkretnej czynności, który zostanie odgórnie określony przez użytkownika, a także konkretnymi rodzajami zasobów (jakie to zostały opisane w 1.1), które również będą mieć odpowiednie właściwości.

Przedsięwzięcie jest modelowane za pomocą skierowanego, niecyklicznego i spójnego unigrafu $G = \langle V, E \rangle$, gdzie V= {1,2,3,…n} jest zbiorem wierzchołków z jednym początkowym i końcowym, a $E \subset VxV$ to relacja dwuczłonowa określająca kolejność czynności. Aby wierzchołki były ułożone w dobrej kolejności została wykonana renumeracja, która to nadała im nowe wartości w zależności od kolejności. Zbiór rodzajów zasobów został określony jako R.

Oznaczenia do modelu:

i	Indeks zadania, $i \in V$
r	Indeks / typ zasobu, $r \in R$
z_{ir}	Zasób dla i-tego zadania i r-tego rodzaju
t_i	Czas wykonania i-tego zadania
t_{ij}	Czas trwania pomiędzy zadaniami i oraz j
Tw_i	Najwcześniejszy termin wystąpienia i-tego zadania
Tp_i	Najpóźniejszy termin wystąpienia i-tego zadania

T	Termin zakończenie projektu
s_i	Termin rozpoczęcia i-tego zadania
f_i	Termin zakończenia i-tego zadania
Z_{rt}	Dzienne zapotrzebowanie na r-tego zasobu w dniu t ($t = 1,2, \dots, T$)
b_{it}	Binarna zmienna decyzyjna przyjmująca 1, gdy i-te zadanie jest realizowane w dniu t ($t \in [Tw_i + 1, Tp_i]$) natomiast 0 w przeciwnym przypadku
w_z	Waga dla z-tego zasobu
M_z	Maksymalne zapotrzebowanie dla z-tego zasobu
$\overline{Z_r}$	Średni poziom zapotrzebowania r -tego zasób

Terminy najwcześniejszy (Tw_i) i najpóźniejszy (Tp_i) odpowiednich zadań zostaną wyznaczone za pomocą CMP. Dodatkowo zakładamy, że t_{ij} jest równe czasowi wykonania poprzedniej czynności t_i dla najwcześniejszego terminu, a $t_j$dla najpóźniejszego:

1. Termin najwcześniejszy pierwszego zadania jest równy 0:

$$Tw_1 = 0 \tag{4}$$

Dla pozostałych:

$$Tw_j = \max\{Tw_i + t_{i,j}\};\ \forall(i,j)\ \in\ E, t_{i,j} = t_i \tag{5}$$

2. Termin najpóźniejszego rozpoczęcia ostatniego zadania to, gdzie l to ostatnie zadnie:

$$Tp_l = Tw_l \tag{6}$$

Dla pozostałych:

$$Tp_i = \min\{Tp_j - t_{i,j}\};\ \forall(i,j)\ \in\ E, t_{i,j} = t_j \tag{7}$$

Wartość dziennego zapotrzebowania na zasoby możemy określić następująco:

$$Z_{rt} = \sum_{i\,\epsilon\,V} z_{ir} \cdot b_{it} \tag{8}$$

Dopuszczalne terminy rozpoczęcia (s_i), zakończenia(f_i) zadania oraz zmiennych binarnych (b_{it}) muszą spełniać ograniczenia:

1. Termin rozpoczęcia pierwszego zadania jest równy 0:

$$s_1 = 0 \tag{9}$$

2. Termin zakończenia przy założeniu, że zadnia są wykonywane bez przerwy jest sumą terminu rozpoczęcia i czasu wykonania:

$$f_i = s_i + t_i \tag{10}$$

3. Kolejne zadania mogą się rozpocząć po zakończeniu poprzedników:

$$s_j \geq f_j, \qquad \forall (i,j) \in E \tag{11}$$

4. Termin rozpoczęcia każdego zadania nie może być mniejszy od wyznaczonego najwcześniejszego terminu:

$$s_i \geq Tw_i \tag{12}$$

5. Termin zakończenia każdego zadania nie może być większy od wyznaczonego najpóźniejszego terminu:

$$f_i \leq Tp_i \tag{13}$$

6. Liczba dni, w których wykonywanie jest zadanie, jest równa czasowi jego wykonywania:

$$\sum_{t \in [Tw_i+1, Tp_i]} b_{it} = t_i \tag{14}$$

7. Zmienne b_{it} muszą przyjmować wartości binarne:

$$b_{it} \in \{0,1\} \tag{15}$$

Funkcja celu:

- Minimalizacja maksymalnego zapotrzebowania na zasoby w projekcie.

Ta funkcja celu pozwala nam na likwidacje nadwyżek zapotrzebowania na konkretny zasób (np. osób w projekcie), gdzie staramy się, aby ta liczba była stała przez cały okres projektu.

Dla jednego zasobu funkcje możemy zapisać:

$$\min(x_r) : x_r = Z_{rt}, \forall r \in R, t \in [1, T] \tag{16}$$

Natomiast w projekcie mamy wiele zasobów. Rozważamy więc problem optymalizacji wielokryterialnej. Pojawia się tutaj problem, kiedy jeden zasób jest w setkach, a drugi w tysiącach. Aby każdy zasób traktować z taką samą wagą zapotrzebowania zostaną podzielone przez maksymalne zapotrzebowanie jakie może wystąpić.

$$\min(y): y = \sum_{r \in R} \frac{x_r}{M_z} \tag{17}$$

Funkcja może też uwzględniać wagi dla poszczególnych zasobów tedy dodatkowo mamy

$$\min(y): y = \sum_{r \in R} w_z \cdot \frac{x_r}{M_z} \tag{18}$$

- Minimalizacja sumy odchyleń na dzienne zapotrzebowanie zasobu w projekcie.

Wahania poziomu zasobów utrudnia równomierną pracę w projekcie więc aby temu zapobiec można określić następujące równanie:

$$\min y: y = \sum_{t=1}^{T} \sum_{r \in R} |\overline{Z_r} - Z_{rt}| \tag{19}$$

Dodatkowo, aby problem był adekwatny do rzeczywistych zapotrzebowań, a także użytkownik aplikacji mógł sam decydować jakiego rozwiązania oczekuje zostaną dołożone funkcjonalności:

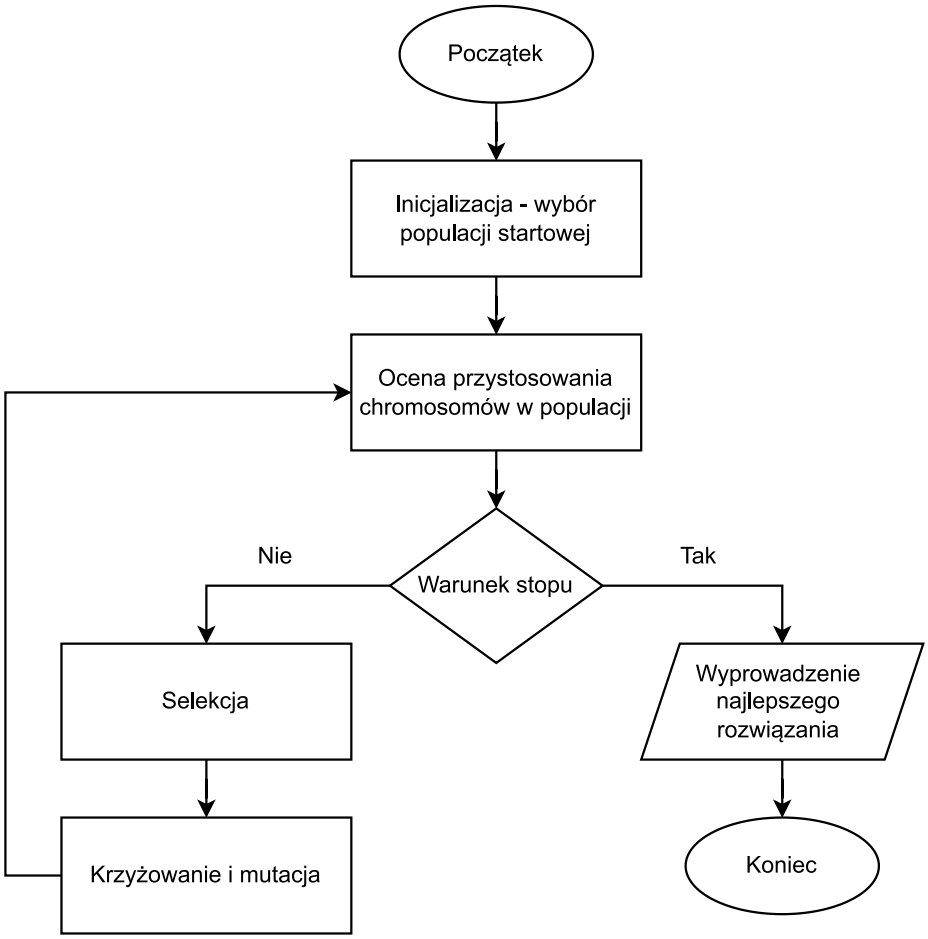
- Możliwość priorytetyzacji danego zasobu, aby w najmniejszym stopniu był on eksploatowany.
- Zabronienia możliwości rozdzielania zadania (rozerwania ciągłości) .
- Ograniczenie terminowości projektu – możliwość wyboru czy projekt może potrwać więcej niż ilość zakładanych dni.

Rozdział II: Opis opracowanego algorytmu

W rozdziale zaprezentowano z jakich elementów składa się opracowany algorytm genetyczny, który posłuży do rozwiązania problemu zarządzania zasobami. Przedstawiono kolejne etapy poszukiwania rozwiązania oraz omówiono każdy z nich. Informacje przedstawione w rozdziale pochodzą z [29].

2.1 Algorytm genetyczny

Algorytm genetyczny jest jedną z metod ewolucyjnych algorytmów, które to z założenia mają za zadanie odwzorowywać naturalny proces ewolucji świata przyrody. Jego twórcą jest Jhon Holland, który to starał się naśladować proces mutacji genetycznej zwierząt, gdzie przeżywały najsilniejsze osobniki, a u kolejnych pokoleń można było zauważyć mocne cechy poprzedników. Jest to więc algorytm oparty na dziedziczeniu i dobrze naturalnym, który łączy zasadę przeżycia najlepiej przystosowanych jednostek i zasadę losowej wymiary informacji między osobnikami. Składa się on z czterech działań, które wykonuje się cyklicznie jak pokazano to na poniższym rysunku:



Rysunek 3 Diagram czynność algorytmu genetycznego

Aby zrozumieć działanie algorytmu ważna jest znajomość podstawowych pojęć, które określają osobnika (postać rozwiązania problemu):

A	1	1	0	0	0	0	0	0	0	0	0	0	0	0
B	0	0	1	1	1	1	0	0	0	0	0	0	0	0
C	0	0	0	0	0	0	0	0	0	1	1	1	0	0
D	1	1	1	1	1	0	0	0	0	0	0	0	0	0
E	0	0	0	0	0	1	1	1	1	0	0	0	0	0
F	0	0	0	0	0	0	0	0	0	1	1	1	1	1
	1	2	3	4	5	6	7	8	9	10	11	12	13	14

Rysunek 4 Przykładowy osobnik problemu

Gen (oznaczony na żółto) – najmniejsza części chromosomu, decyduje o dziedziczności jednej lub kilku cech.

Chromosom (oznaczony na zielono) – uporządkowany ciąg genów, miejsce przechowywania genów osobnika.

Genotyp – zespół chromosomów, czyli kompletny i jednoznaczny opis zawarty w genach.

Osobnik – podstawowa jednostka, która jest zakodowanym zbiorem parametrów zadania. określająca potencjalne rozwiązanie.

Populacja – zbiór osobników zamieszkujących wspólne środowisko.

Fenotyp – zestaw parametrów cech rozwiązania zadania, które podlegają ocenie.

A	[1,2]	[1,2]	0	0	0	0	0	0	0	0	0	0	0	0
B	0	0	[4,3]	[4,3]	[4,3]	[4,3]	0	0	0	0	0	0	0	0
C	0	0	0	0	0	0	0	0	0	[3,2]	[3,2]	[3,2]	0	0
D	[3,2]	[3,2]	[3,2]	[3,2]	[3,2]	0	0	0	0	0	0	0	0	0
E	0	0	0	0	0	[4,1]	[4,1]	[4,1]	[4,1]	0	0	0	0	0
F	0	0	0	0	0	0	0	0	0	[2,2]	[2,2]	[2,2]	[2,2]	[2,2]
	1	2	3	4	5	6	7	8	9	10	11	12	13	14

Rysunek 5 Przykładowy fenotyp problemu

2.2 Konstrukcja rozwiązania początkowego / inicjalizacja populacji

Aby rozpocząć poszukiwanie najlepszego rozwiązania trzeba stworzyć populacje osobników, na podstawie której algorytm będzie się rozwijał oraz poszukiwał rozwiązania optymalnego. Najważniejsze na tym etapie jest zaplanowanie odpowiedniego kodowania dla problemu, którego wybór ma istotny wpływ na szybkość i jakość znajdowanych wyników. Wyróżniamy następujące warianty ułożenia genów takie jak:

- **Klasyczny –** geny na różnych pozycjach przechowują różne informacje.
- **Permutacyjny –** geny przechowują podobne informacje natomiast w wyniku działania programu zmieniają się one miejscami.
- **Drzewiasty –** chromosomy tworzą złożoną strukturę drzewiastą, gdzie w wyniku działania ulegają zmianom całe gałęzie genów.

W przypadku jak pokazano na *rysunku 1* osobnika mamy styczność z klasycznym kodowaniem, gdzie zmianom ulegają odpowiednie geny.

2.3 Selekcja

Podczas tego etapu wybierane są osobniki najlepiej przystosowane, które zostaną włączone do grupy rozrodczej, na której będą przeprowadzane kolejne operacje. Jest ona dokonywana na podstawie **funkcji przystosowania**, która to zwraca liczbę będącą oceną jakości przystosowania danego osobnika do danego problemu. Jest ona generowana na podstawie fenotypu, a nie rzeczywistych wartości. Wyróżniamy metody selekcji takie jak:

- **Metoda koła ruletki** – każdy osobnik otrzymuje wycinek koła ruletki, którego rozmiar jest wprost proporcjonalny do jego jakości, a następnie losowo wybierany jest element do grupy rozrodczej, gdzie im większy wycinek rozwiązanie posiadało tym większe prawdopodobieństwo na wylosowanie go.
- **Selekcja rankingowa –** osobniki sortowane są według jakości, a następnie do grupy rozrodczej przechodzi n najlepiej przystosowanych osobników.
- **Selekcja turniejowa –** populacje dzieli się na szereg dowolnie licznych grup, a następnie wybiera się najlepiej przystosowane osobniki.

Ważnym elementem jest też dobór wielkości tworzonej grupy rozrodczej do rozmiaru populacji, gdzie mały stosunek może doprowadzić do zaniku różnorodności, natomiast zbyt duży do wprowadzaniu w kolejnych rozwiązaniach zbyt dużej ilości słabych genów.

W programie używamy selekcji rankingowej i koła ruletki, aby sprawdzić, która metoda poradzi sobie lepiej ze znalezieniem rozwiązania, a także sortujemy dwie funkcje przystosowania opisane powyżej jako funkcje celu.

2.4 Operator krzyżowania

Jest jeden z procesów generowania nowej populacji z osobnikami, które zostały wybrane metodą selekcji do grupy rozrodczej. Odbywa się ono poprzez rozcięcie chromosomów rodziców w dowolnym punkcie, a następnie zmianie ich części między sobą.

Na przypadku rozwiązania krzyżowanie następuje dla konkretnych czynności, gdzie dla wybranych osobników są one różne i mogą zostać poddane wymianie nie zaburzając ograniczeń związanych z kolejnością.

2.5 Operator mutacji

Mutacja jest samoistną losową zmianą genotypu, w przyrodzie powstająca w wyniku błędu reprodukcji lub uszkodzeń fizycznych, przez co zwiększa różnorodność materiału genetycznego. W algorytmie symuluje się zachowanie takiej mutacji i dobranie prawdopodobieństwa jej zajścia jest sprawą indywidualną każdego programu, natomiast nie powinno być zbyt duże, gdyż w wynikach pojawi się dużo losowości i korzyści nie będą miały szans na utrwalenie. Występuje wiele metod mutacji natomiast dobranie właściwych zależy od specyfikacji programu.

Algorytm wybiera Pm – prawdopodobieństwo mutacji populacji, które zostaje poddane mutacji metodami alokacji, rozdzielania, delegacji czynności a także poprzez dodanie dodatkowego dnia. Bazowa jego wartość wynosi: $Pm = 10\%$

2.6 Kryterium zatrzymania algorytmu

Aby algorytm nie działał w nieskończoność potrzebne jest zdefiniowanie kryterium zatrzymania. Wyróżnia się dwa kryteria:

- Zatrzymanie poprzez ustaloną odgórnie liczbę iteracji algorytmu.
- Zatrzymanie, jeśli najlepszy osobnik spełnia kryterium zatrzymania (np. wartość zapotrzebowania spadnie poniżej ustalonej granicy).

W algorytmie mamy możliwość ustawienia obu parametrów zatrzymania w zależności od potrzeb użytkownika programu.

Rozdział III: Implementacja programu

W rozdziale zaprezentowano szczegółową implementację algorytmu opisanego w rozdziale drugim zrealizowaną w języku Python z wykorzystaniem bibliotek takich jak pandas, numpy oraz matplotlib. Przedstawiono wszystkie możliwości jakie ma użytkownik oraz jak algorytm zbliżą się do rozwiązania najbardziej optymalnego w każdym z elementów.

3.1 Pseudokod algorytmu

Struktury danych algorytmu:

- Z – Dane dla których chcemy rozwiązać problem (patrz *Tab 1*)
- Iter_k – Liczba iteracji algorytmu
- Size – rozmiar populacji początkowej
- Ws – Warunek stopu
- O – Ograniczenia algorytmu (wybrane przez użytkownika)

Tabela 1 Przykładowa baza danych dla problemu

Task	Description	Previous_task	Time	R1	R2	R3	R4	R5
A	Zadanie 1		3	3	400	35	2	1
B	Zadanie 2	A	2	2	500	40	4	1
C	Zadanie 3	A	3	3	500	55	3	3
D	Zadanie 4	B	5	4	600	63	5	2
E	Zadanie 5	B	5	5	400	45	2	3
F	Zadanie 6	D,E	5	6	200	34	2	1
G	Zadanie 7	C,F	4	2	500	23	2	1

Pseudokod algorytmu

```
c = CPM(Z)

Iter := 0

Roz = first_result()

population  = create_population (Roz,Size)

firness, resources = fitness_function(population,O)

while (Iter < Iter_k or Ws == BestSoluction):

    for  random  f, g in population:

            x, y = Cross (f,g)

            population += x

            population += y
```

```
    for  random f in population:
        f = mutation(f,O)
        population += f
    firness, resources = fitness_function(population,O)
    population, best_soluction = selection(population, fitness, O)
    Iter ++
return best_solution
```

3.2 Opis funkcji pseudokodu

CPM() – Klasa, której głównym zadaniem jest wyznaczenie ścieżki krytycznej dla algorytmu, oraz czynności przekazanych za pomocą tabeli danych. Tabela została dodana do programu za pomocą biblioteki pandas, która to pozwala na operacje związane z bazami danych. Aby wykonać wyznaczenie ścieżki, najpierw została stworzona macierz sąsiedztwa i każdej czynności został nadany nowy numer. Zostały wyznaczone najwcześniejsze i najpóźniejsze terminy dla każdej czynności według *wzorów nr 1 i 2*, a także zapas czasowy opisany *wzorem nr 3.* Została także zabezpieczona przed błędami takimi jak stworzenie wielu wierzchołków początkowych czy końcowych, a także odwróconą kolejnością relacji.

first_result() – Funkcja, która na podstawie wyznaczonych w klasie CPM() najwcześniejszych i najpóźniejszych terminów. Tworzy pierwszego osobnika populacji. Realizuje to poprzez przechodzenie po kolejnych czynnościach i dodawaniu bitu „1” do odpowiednich miejsc w macierzy rozwiązania.

create_population() – Funkcja, która za pomocą zapasów wyznaczonych w klasie CPM() tworzy populację startową, której wielkość zależy od parametru Size. Nowy osobnik tworzony jest na podstawie pierwszego osobnika wyznaczonego w funkcji first_result(). Losuje czynność, która nie znajduje się w ścieżce krytycznej, a następnie na podstawie zapasu do niej przypisanego, losuje o ile konkretna czynności zostanie przesunięta do przodu. Aby nie powstały tutaj problemy nakładania się zadań, które powinny zostać wykonane sekwencyjnie oraz zachowane zostały zasady kolejnościowe, algorytm sprawdza wszystkich kolejnych sąsiadów i przesuwa ich na możliwe pierwsze miejsce gdzie nie spowodowałyby konfliktu.

fitness_fuction() – Funkcja, której celem jest wyznaczenie oceny przystosowania dla otrzymanej populacji osobników. W pierwszym etapie dla każdego osobnika na podstawie fenotypu wyznacza macierz sum zapotrzebowania danego zasobu na konkretny dzień.

```
array([[4., 4., 6., 6., 3., 3., 3., 6., 6., 6., 6., 4., 4., 4.],
       [2., 2., 3., 3., 2., 2., 2., 3., 3., 3., 3., 2., 2., 2.]])
```

Rysunek 6 Przykładowa macierz sum zapotrzebowania dla dwóch zasobów

Następnie mamy do wyboru dwie funkcje celu, na podstawie których jest wyznaczona ocena konkretnego osobnika. Pierwszą funkcją jest opisana *wzorem nr 18*, która minimalizuje maksymalne zapotrzebowania na zasoby w projekcie. Drugą funkcja jest opisana *wzorem nr 19*, minimalizująca sumę odchyleń na dzienne zapotrzebowanie zasobu w projekcie. Zostały tutaj uwzględnione także wagi dla odpowiedniego zasobu, które możemy zdefiniować poprzez wybranie liczby, przez którą wartość jest mnożona.

selection() – Funkcja, która ma na celu selekcje wybraną liczbę osobników do nowej populacja wybraną metodą. W algorytmie mamy do wyboru dwie metody:

Metoda koła ruletki – Jak opisano we wcześniejszym rozdziale najpierw zostały wyznaczone wagi dla każdego z n osobników. Ponieważ funkcja celu jest u nas minimalizowana skorzystano ze wzoru:

$$val = \frac{\sum_1^n f}{f} \tag{20}$$

$$w = \frac{val}{\sum val}$$

Natępnie korzystając z funkcji z biblioteki random: **random.choice**, która pozwala na dodanie wag przy losowaniu, zostały wybrane elementy nowej populacji.

Selekcja rankingowa – lista przystosowania została posortowana za pomocą funkcji **sorted,** a następnie zostało wybranych n najlepszych osobników do populacji rozrodnej.

cross() – Funkcja, której celem jest krzyżowanie dwóch losowo wybranych osobników. Przechodzi ona po kolejnych czynności, nie będących w ścieżce krytycznej, jeśli czynności różnią się od siebie sprawdza czy jest wstanie wykonać krzyżowanie dla tych czynności. Wyznacza punkty krańcowe dla czynności i buduje macierz sprawdzając sąsiadów czynności jak przedstawiono na poniższym rysunku:

A	1	1	0	0	0	0	0	0	0	0	0	0	0	0
B	0	0	0	1	1	0	0	0	0	0	0	0	0	0
C	0	0	1	1	1	1	1	0	0	0	0	0	0	0
D	0	0	0	0	0	1	1	0	0	0	0	0	0	0
E	0	0	0	0	0	0	0	1	1	1	1	0	0	0
F	0	0	0	0	0	0	0	1	1	0	0	0	0	0
F	0	0	0	0	0	0	0	0	0	0	0	1	1	1
	1	2	3	4	5	6	7	8	9	10	11	12	13	14

A	1	1	0	0	0	0	0	0	0	0	0	0	0	0
B	0	0	1	1	0	0	0	0	0	0	0	0	0	0
C	0	0	1	1	1	1	1	0	0	0	0	0	0	0
D	0	0	0	0	0	0	0	1	1	0	0	0	0	0
E	0	0	0	0	0	0	0	1	1	1	1	0	0	0
F	0	0	0	0	0	0	0	0	0	1	1	0	0	0
F	0	0	0	0	0	0	0	0	0	0	0	1	1	1
	1	2	3	4	5	6	7	8	9	10	11	12	13	14

Rysunek 7 Przykład krzyżowania

Na zielono przedstawiono sąsiadów i stworzoną macierz, na ich podstawie dla czynności B (oznaczona kolorem pomarańczowym). Jeśli suma dla którejkolwiek z macierzy jest równa 0, to rozwiązanie można wstawić do drugiej funkcji:

A	1	1	0	0	0	0	0	0	0	0	0	0	0	0
B	0	0	0	1	1	0	0	0	0	0	0	0	0	0
C	0	0	1	1	1	1	1	0	0	0	0	0	0	0
D	0	0	0	0	0	0	0	1	1	0	0	0	0	0
E	0	0	0	0	0	0	0	1	1	1	1	0	0	0
F	0	0	0	0	0	0	0	0	0	1	1	0	0	0
F	0	0	0	0	0	0	0	0	0	0	0	1	1	1
	1	2	3	4	5	6	7	8	9	10	11	12	13	14

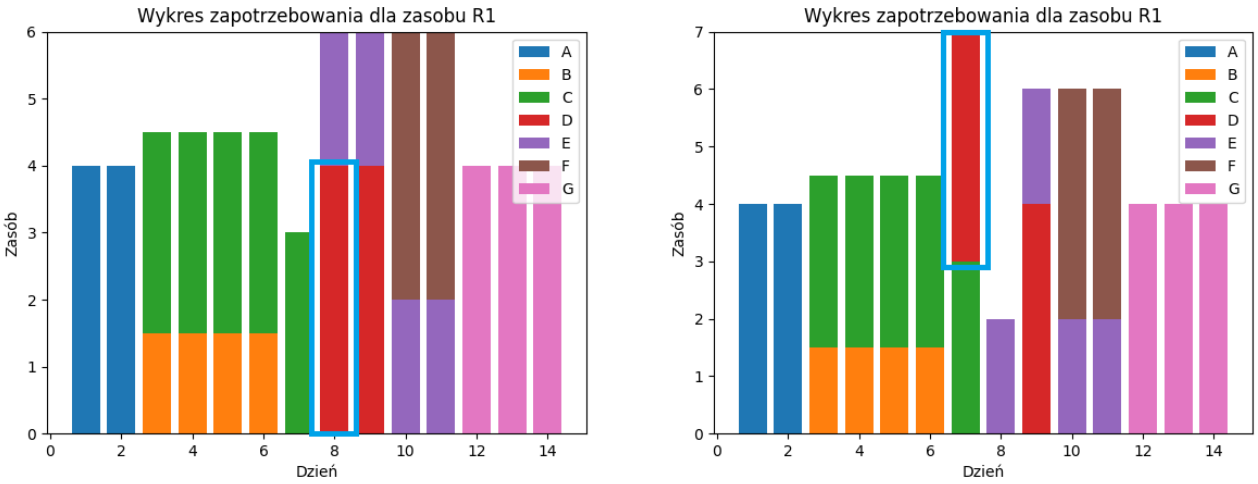
Rysunek 8 Rozwiązanie po krzyżowaniu

Na czerwono oznaczono czynność B z pierwszego rozwiązania wstawioną do drugiego rozwiąznia.

mutation() – Funkcja, która ma na celu zmienić tak losowo rozwiąznie, aby zbliżyło się do rozwiązania optymalnego. Posiada ona cztery rodzaje mutacji, które można uaktywnić lub wyłaczyć w zależności od ograniczeń podanych przez użytkownika. Pierwszym etapem przed wybraniem metody algorytm sprawdza dla konkretnego zasobu, które z dwóch maksimum czy minimum jest dalej od średniej dla algorytmu.

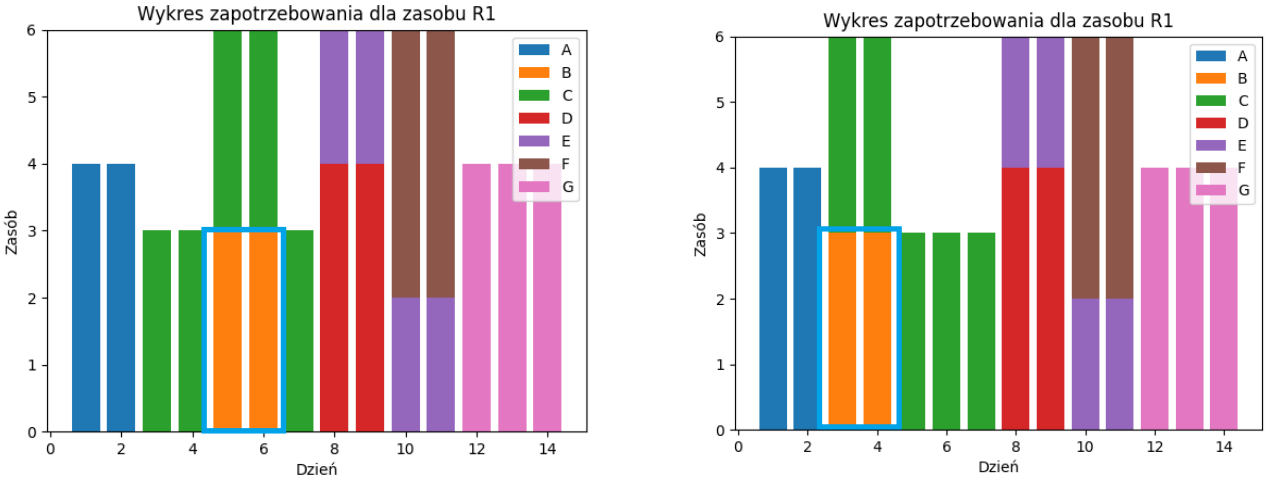
Jeśli minimum algorytm ma dwie opcje:

Rozdzielenie zasobów – algorytm szuka pierwszych możliwych sąsiadów zarówno z lewej jak i prawej strony, a następnie losowo wybiera tego który zostanie przeniesiony na jego miejsce, nie zaburzając przy tym zasad kolejnościowych.



Rysunek 9 Rozdzielanie zasobów - przykład

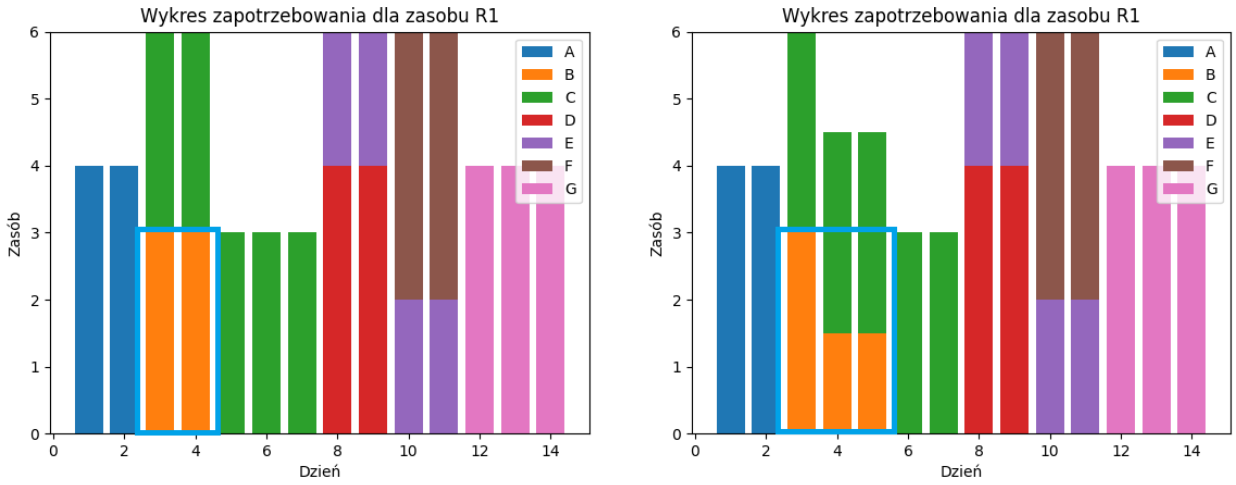
Alokacja zaosbów – w momencie kiedy zablokujemy opcje rozdzielania zadań, zostaje uaktwniona opcja przenoszenia całych czynności. Od miejsce gdzie zostało wykryte minimum, funkcja poszukuje najbliżego tym razem zadania, a następnie tak jak poprzednio zostaje ono przeniesione na nowe miejsce.



Rysunek 10 Alokacja zasobów - przykład

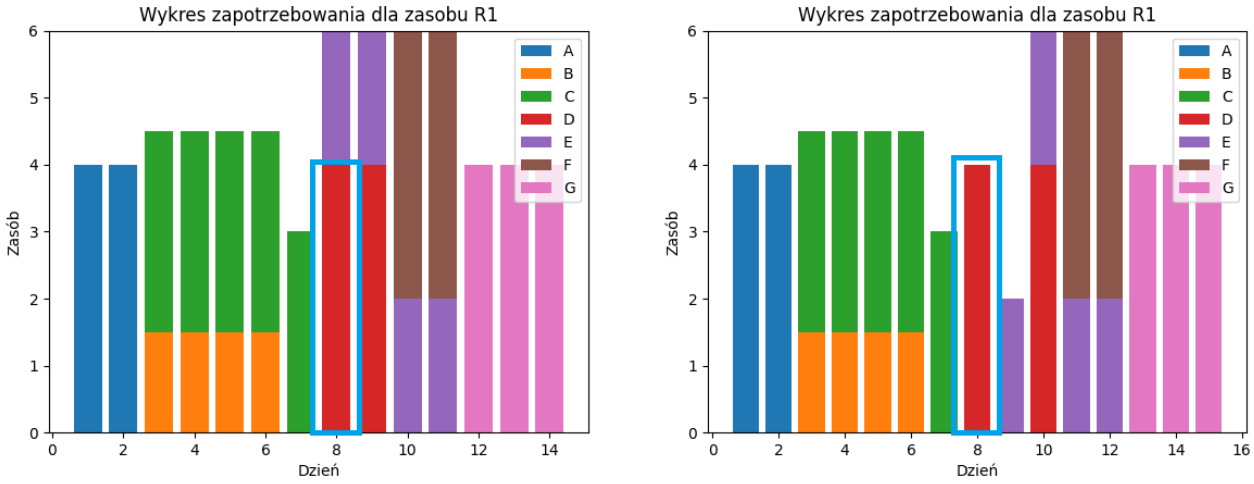
Jeśli maksimum algorytm ma także dwie opcje:

Podział zasobów – algorytm sprawdza czy obok maksimum jest miejsce, gdzie mógłby podzielić na dwa swoją ilość zasobów nie zaburzając kolejności. Sprawdza także miejsca obok krańców zadania. Jeśli mu się uda dzieli zadanie na dwa i w rozwiązaniu pojawiają się wartości „2”. Jeśli nie przechodzi do drugiej metody.



Rysunek 11 Podział zasobów - przykład

Dodanie dodatkowego dnia – Algorytm dodaje nowy dzień i przerzuca tam zasoby z maksimum, natomiast jest to równoznaczne z dodaniem ujemnych wartości do funkcji fitness().



Rysunek 12 Dodawanie nowego dnia - przykład

W momencie kiedy algorytm nie może wykonać funkcji maksimum, przeskakuje do funkcji minimum i stara się jeszcze przesunięciem znaleźć inne rozwiązanie problemu.

Rozdział IV: Eksperymenty obliczeniowe

W rozdziale zaprezentowano sposoby testowania oraz sprawdzania poprawności algorytmu, a także przeanalizowano program dla różnych wartości wejściowych. Następnie przeprowadzono testy porównawcze metod oraz wykazano, jakie parametry są wymagane dla najefektywniejszego działania stworzonego algorytmu.

4.1 Metodyka testowania

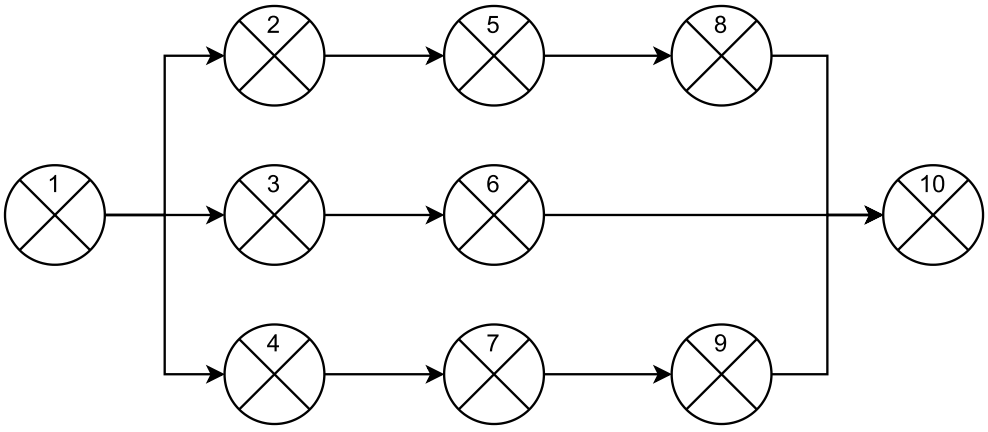
Aby można było stwierdzić, że stworzone oprogramowanie działa poprawne i można go użyć w przestrzeni komercyjnej niezbędne jest przeprowadzenie testów oraz analiz. Projekty informatyczne dzielą realizację testów na:

- Ze względu na cel (walidacja i weryfikacja)
- Ze względu na zakres (jednostkowe i systemu)
- Ze względu na sposób realizacji (losowe i specjalistyczne)

W planowaniu projektu kierownik planuje odpowiednie czynności oraz określa czas wraz z potrzebnymi zasobami, dlatego generacja losowych instrukcji testowych może spowodować, że znalezienie wyniku będzie nierealne czy bezsensowne. Dlatego dla programu przygotowano odpowiednie scenariusze testowe, które zasymulują realne projekty. Przetestowano je także dla różnych założeń projektowych, czego przekrój jest wstanie pokazać nam poprawność oraz problemy jakie program ma przy wyznaczaniu zapotrzebowania na zasoby.

Scenariusz nr 1

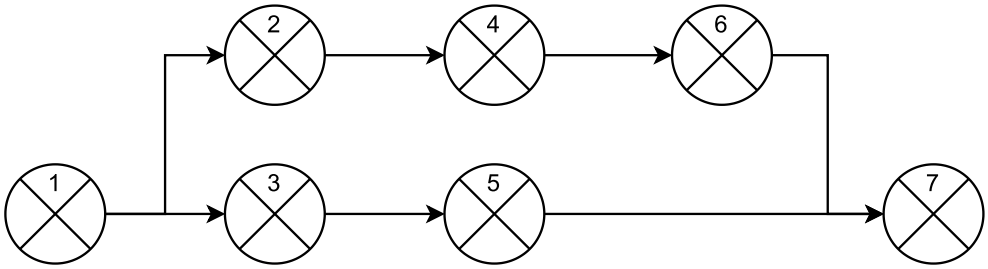
W tej wersji mamy do czynienia z projektem, w którym grupy zadań są wykonywane równolegle, oraz projekt zostanie zakończony sukcesem jeśli zakończą się one w ustalonym czasie. Taki typ projektu możemy zobaczyć np. w fabrykach, gdzie wszystkie składniki musza być gotowe w tym samym czasie przed ostatecznym złożeniem.



Rysunek 13 Scenariusz nr 1

Scenariusz nr 2

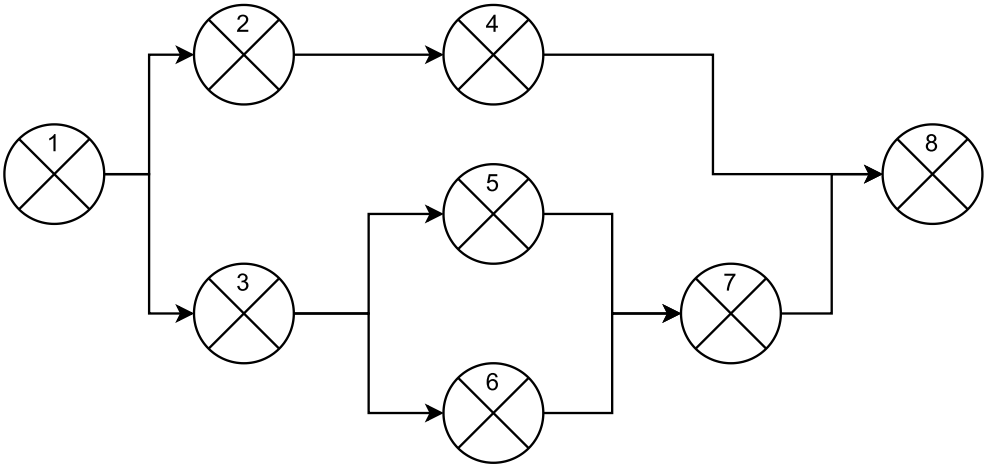
Kolejny scenariusz ma na celu sprawdzenie jak algorytm poradzi sobie w wyznaczeniu planu dla małych projektów, gdzie zadania są wykonywane także równolegle i muszą zakończyć się do danego momentu czasu.



Rysunek 14 Scenariusz nr 2

Scenariusz nr 3

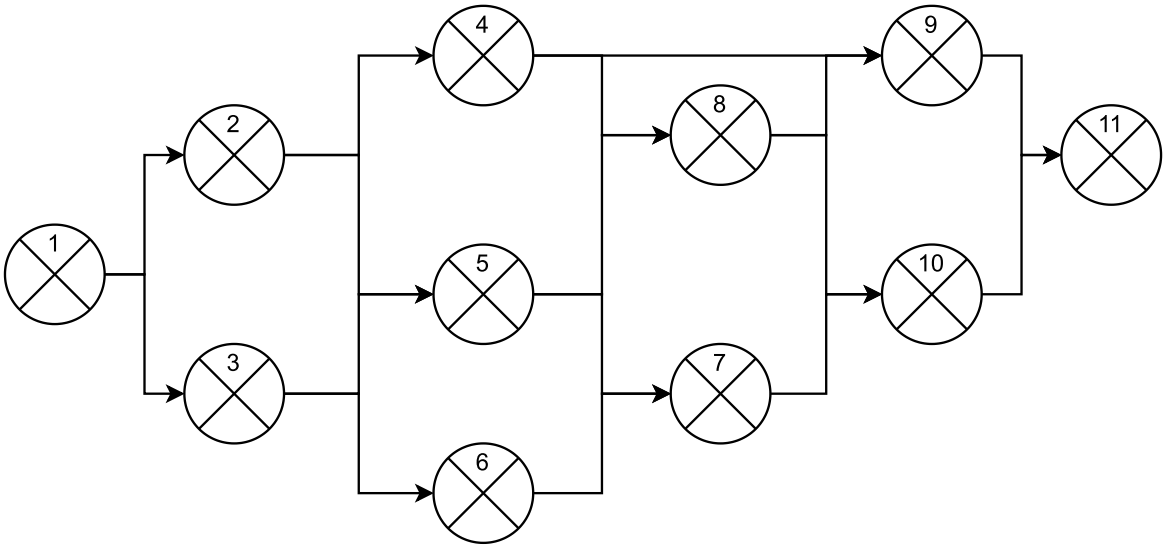
W tym scenariuszu chcemy sprawdzić, jak algorytm poradzi sobie z sytuacją, gdzie muszą być jeszcze bardziej restrykcyjnie zachowane relacje kolejnościowe, które ze sobą kolidują. Taki typ projektu wykorzystywany jest np. w budownictwie, gdzie bez wykonania kilku poprzednich zadań równocześnie nie jesteśmy wstanie pójść do przodu z projektem.



Rysunek 15 Scenariusz nr 3

Scenariusz nr 4

Ostatni już scenariusz ma na celu sprawdzenie jak poradzi sobie algorytm w projekcie, który ma wiele czynności synchronicznych. Ma to sprawdzić czy wyrównanie zasobów dla skomplikowanych projektów jest możliwe i co może zrobić algorytm, aby nie zaburzyć kolejności.



Rysunek 16 Scenariusz nr 4

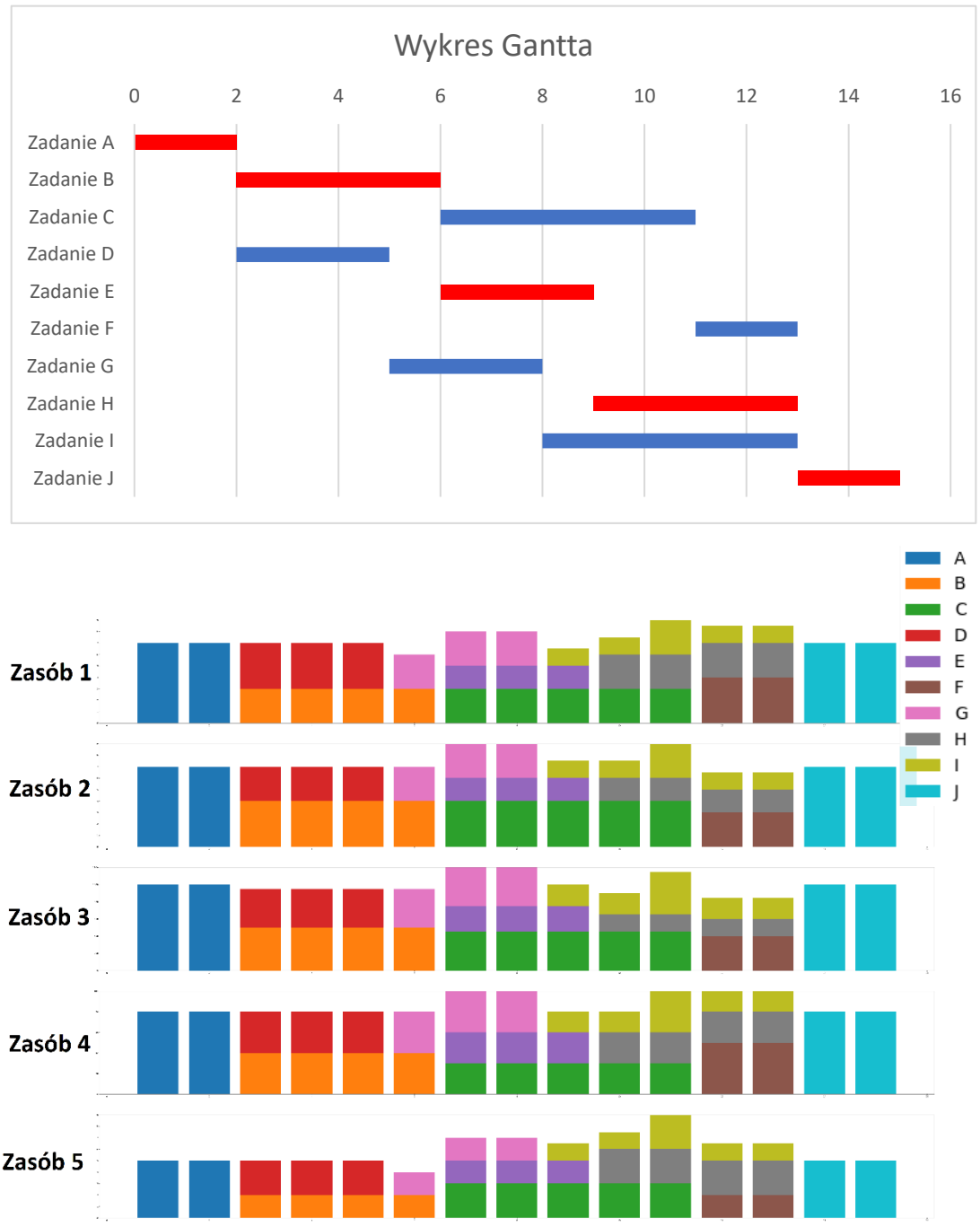
Zasoby oraz czasy dla każdego z scenariuszy zostały dobrane także z myślą o realnym projekcie, oraz zostały zmiennie dla potrzeb przetestowania sytuacji skrajnych.

4.2 Działanie algorytmu dla scenariuszy testowych

Pierwsze testy zostały wykonane dla wyżej zaprezentowanych scenariuszy, które to w sposób kompleksowy miały sprawdzić działanie program dla różnych rodzajów relacji oraz ułożeń czynności w projekcie. Dla scenariusza pierwszego zostały przygotowane dane:

Tabela 2 Dane dla scenariusza nr 1

Task	Description	Previous_task	Time	R1	R2	R3	R4	R5
A	Zadanie 1		2	7	7	1000	80	5
B	Zadanie 2	A	4	3	4	500	40	2
C	Zadanie 3	A	5	3	4	450	30	3
D	Zadanie 4	A	3	4	3	450	40	3
E	Zadanie 5	B	3	2	2	300	30	2
F	Zadanie 6	C	2	4	3	400	50	2
G	Zadanie 7	D	3	3	3	450	40	2
H	Zadanie 8	E	4	3	2	200	30	3
I	Zadanie 9	G	3	3	3	500	40	3
J	Zadanie 10	F,H,I	2	7	7	1000	80	5



Rysunek 17 Wynik dla scenariusza nr 1

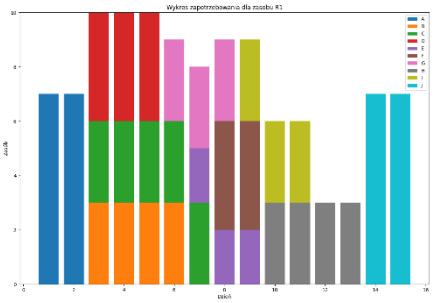
Dla zaprezentowanego scenariusza na podstawie macierzy został wyświetlony wykres Gantta oraz zapotrzebowania na zasób w danym dniu. Sam wykres Gantta, który można uznać za harmonogram pracy projektu jest generowany za pośrednictwem Excela. W kolejnych testach będziemy natomiast skupiać się na porównaniu początkowych wyników zapotrzebowania oraz końcowych dla konkretnych scenariuszy. Takie porównanie pozwoli zaobserwować w najlepszym stopniu jak wynik ewoluował oraz jak działa program dla każdego z przygotowanych scenariuszy.

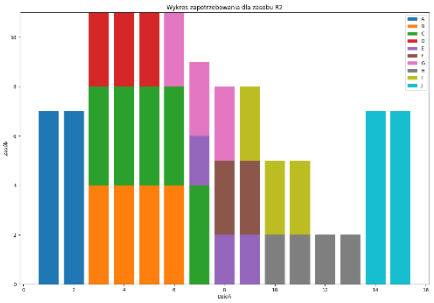
Każdy scenariusz był testowany dla :

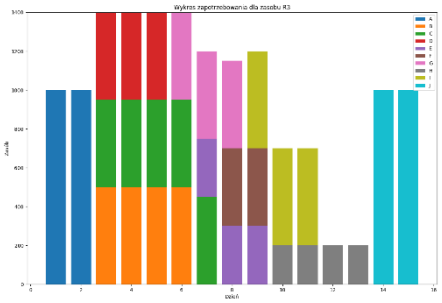
- funkcji celu minimalizującej odchylenie standardowe,
- selekcji metodą koła ruletki,
- 500 iteracji programu,
- przy możliwości stosowania wszystkich metod mutacji,
- bez nałożenia priorytetów dla konkretnego zasobu,

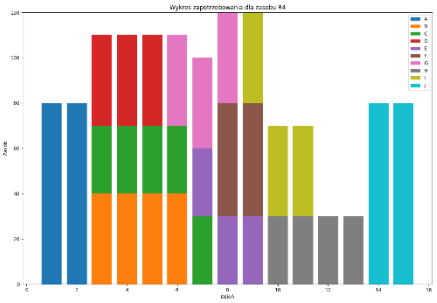
Scenariusz nr 1

Zapotrzebowanie początkowe

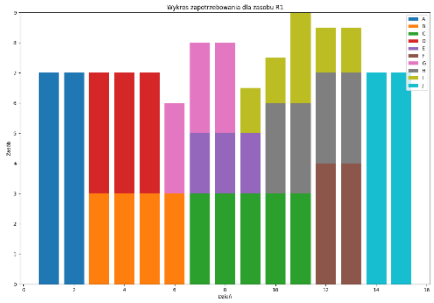


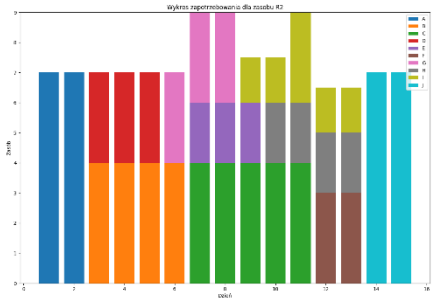


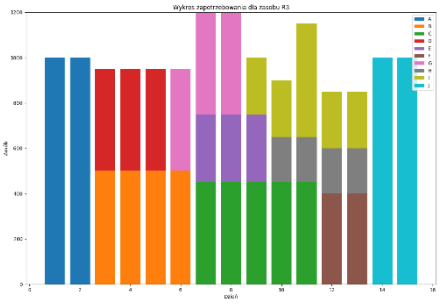


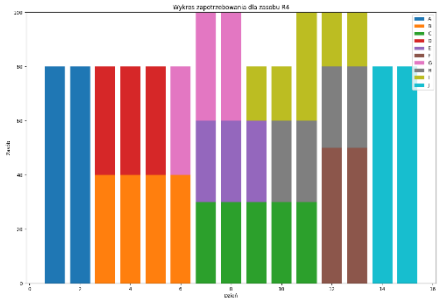


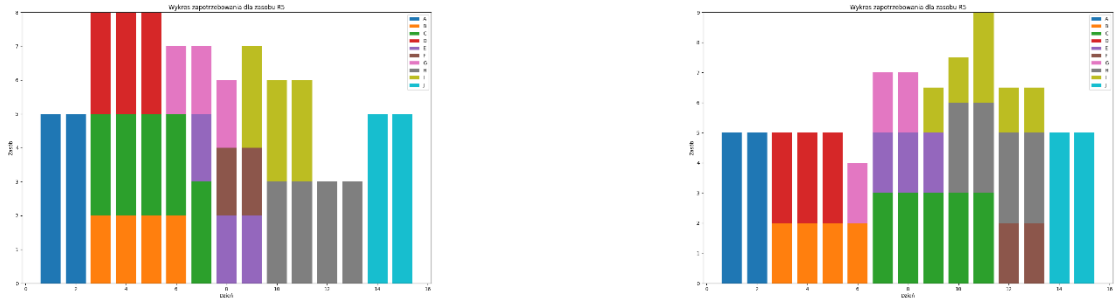
Zapotrzebowanie końcowe











Rysunek 18 Wyrównanie zasobów scenariusza pierwszego

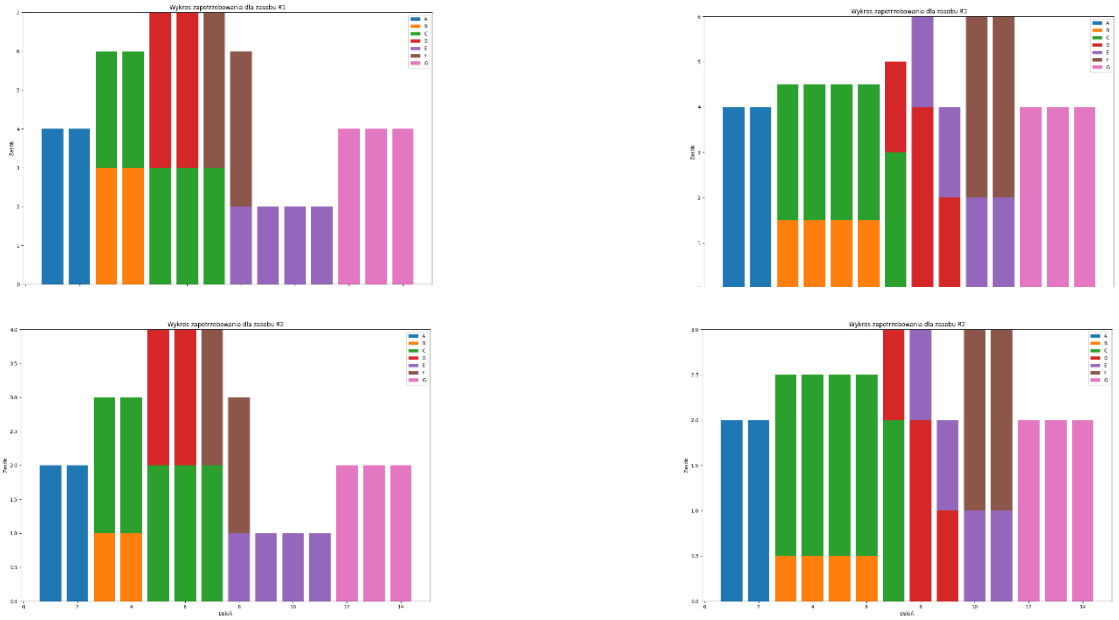
Jak można zauważyć program poradził sobie z wyrównaniem dla każdego zasobu, dla którego to było możliwe, możemy zaobserwować spadek zapotrzebowania zasobów na konkretny dzień. Ponieważ skala na wykresach jest różna efekt może być mylący, należy zwrócić uwagę na pierwszy zasób (kolor ciemnoniebieski), który nie może zostać poddany zmianie.

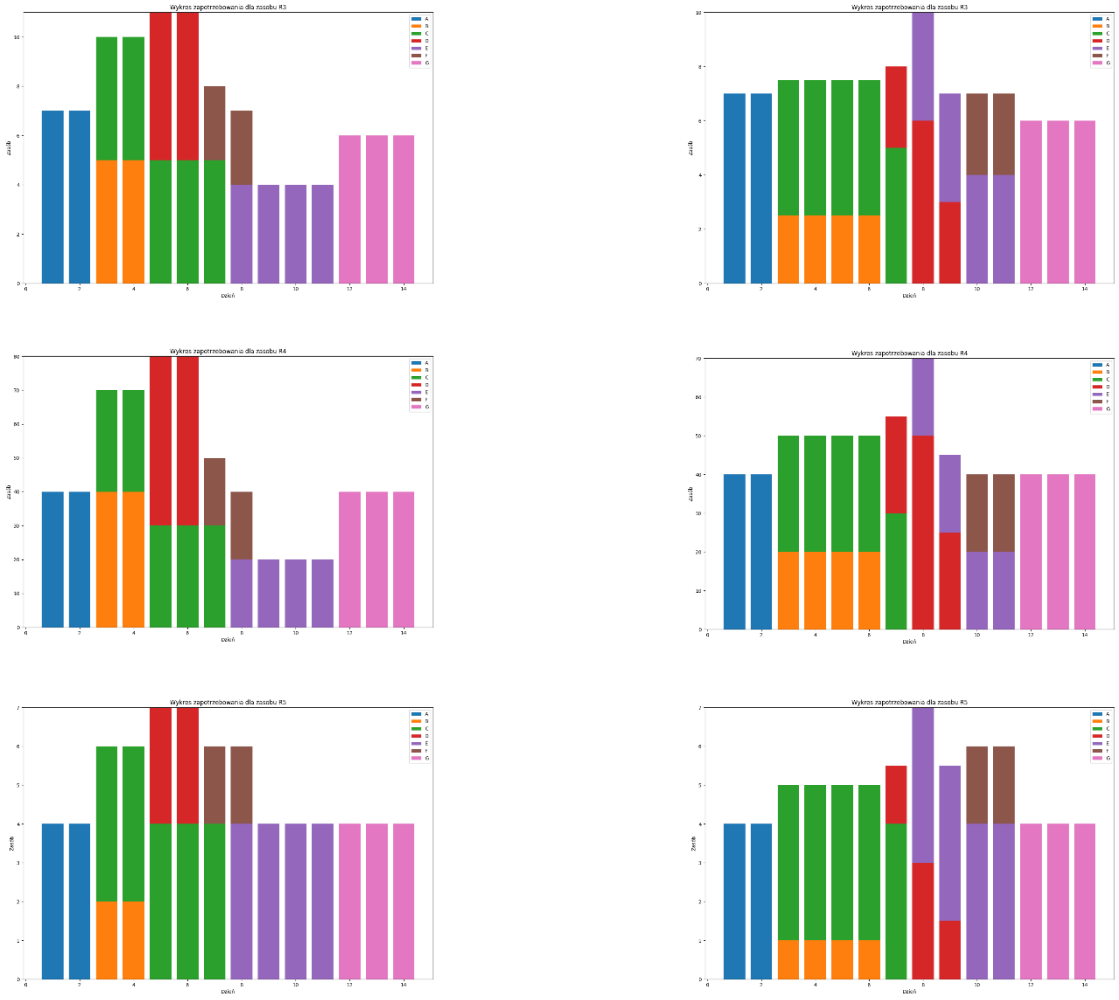
Scenariusz nr 2

Tabela 3 Dane dla scenariusza nr 2

Description	Previous_task	Time	R1	R2	R3	R4	R5
Zadanie 1		2	4	2	7	40	4
Zadanie 2	A	2	3	1	5	40	2
Zadanie 3	A	5	3	2	5	30	4
Zadanie 4	B	2	4	2	6	50	3
Zadanie 57	C	4	2	1	4	20	4
Zadanie 6	D	2	4	2	3	20	2
Zadanie 7	E,F	3	4	2	6	40	4

Zapotrzebowanie początkowe Zapotrzebowanie końcowe





Rysunek 19 Wyrównanie zasobów scenariusza drugiego

Dla scenariusza drugiego także możemy zaobserwować zmniejszenie maksymalnego zapotrzebowania dla konkretnych zasobów, a także wyrównania w stopniu większym niż było ono na początku.

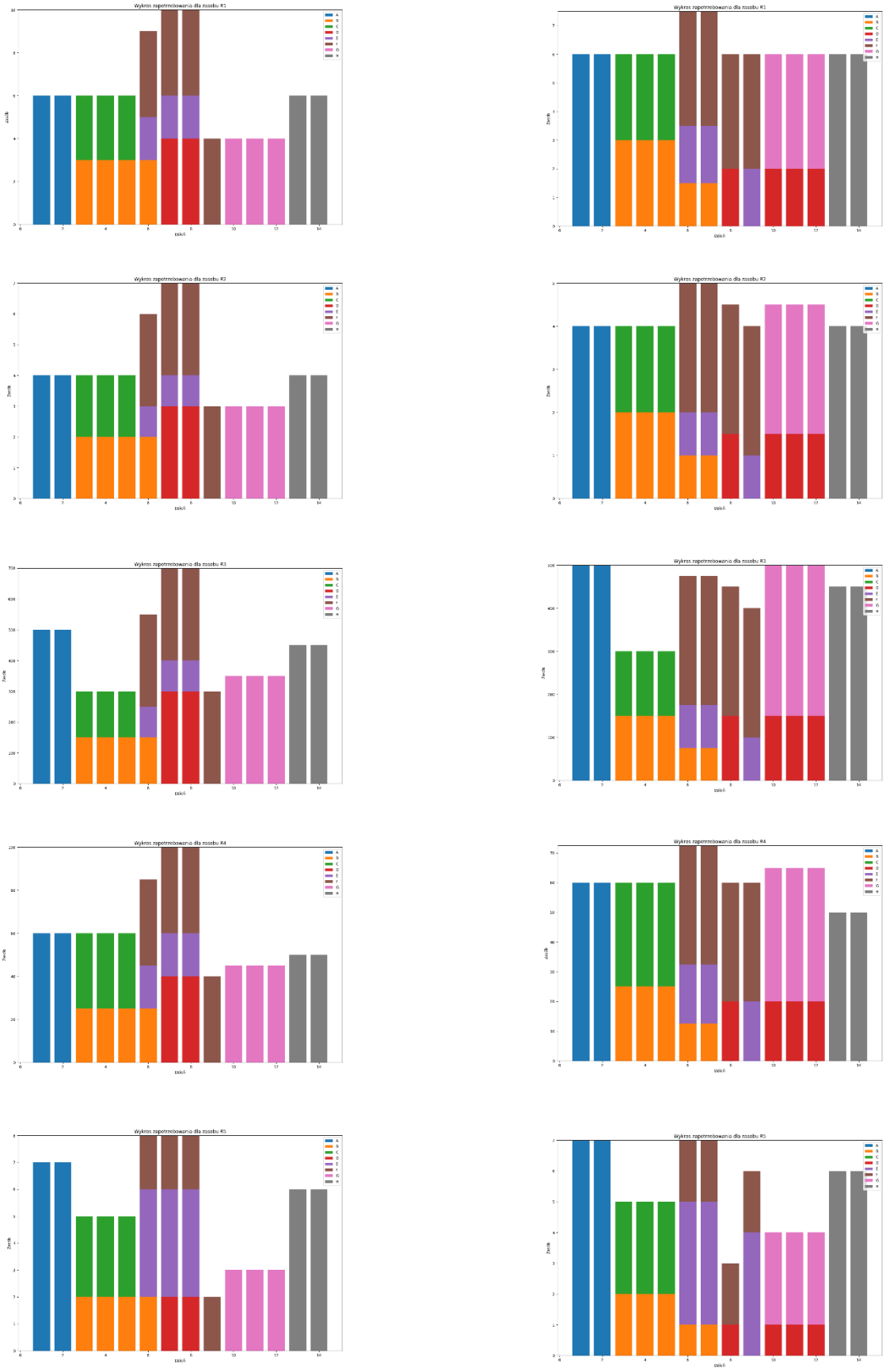
Scenariusz nr 3

Tabela 4 Dane dla scenariusza nr 3

Task	Description	Previous_task	Time	R1	R2	R3	R4	R5
A	Zadanie 1		2	6	4	500	60	7
B	Zadanie 2	A	4	3	2	150	25	2
C	Zadanie 3	A	3	3	2	150	35	3
D	Zadanie 4	B	2	4	3	300	40	2
E	Zadanie 5	C	3	2	1	100	20	4
F	Zadanie 6	C	4	4	3	300	40	2
G	Zadanie 7	E,F	3	4	3	350	45	3
H	Zadanie 8	D,G	2	6	4	450	50	6

Zapotrzebowanie początkowe

Zapotrzebowanie końcowe



Rysunek 20 Wyrównanie zasobów scenariusza trzeciego

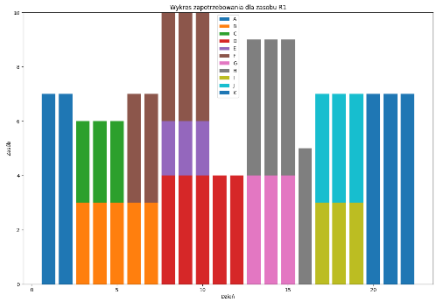
Dla scenariusza trzeciego, gdzie algorytm mamy doczynienia z relacjami równoległymi, oraz algorytm musi zwracać uwagę na kolejność następujących po sobie czynności, możemy zauważyć, że algorytm poradził sobie z wyrównanie wszystkich możliwych zasobów, oraz relacje (zaznaczonym kolorem brązowym E oraz fioletowym F) zostały wykonanie przed rozpoczęciem kolejnej czynności (różowe G).

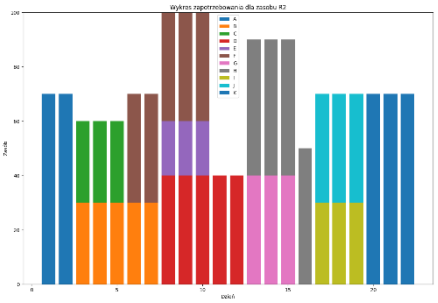
Scenariusz nr 4

Tabela 5 Dane dla scenariusza nr 4

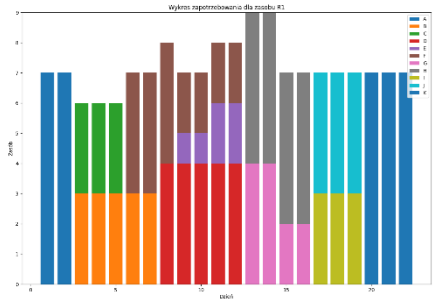
Task	Description	Previous_task	Time	R1	R2	R3	R4	R5
A	Zadanie 1		2	7	70	4	8	4
B	Zadanie 2	A	5	3	30	4	7	1
C	Zadanie 3	A	3	3	30	5	5	3
D	Zadanie 4	B	5	4	40	6	9	2
E	Zadanie 5	B,C	3	2	20	3	2	3
F	Zadanie 6	C	5	4	40	4	5	1
G	Zadanie 7	D,F	3	4	40	6	6	1
H	Zadanie 8	D,E	4	5	50	6	3	3
I	Zadanie 9	D,H	3	3	30	4	5	2
J	Zadanie 10	H,G	3	4	40	4	2	3
K	Zadanie 11	I,J	3	7	70	5	8	4

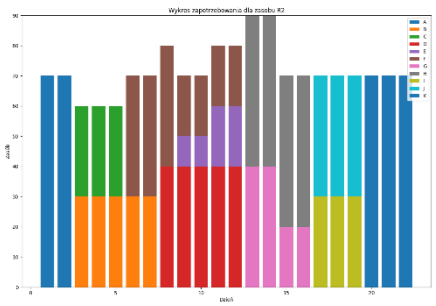
Zapotrzebowanie początkowe

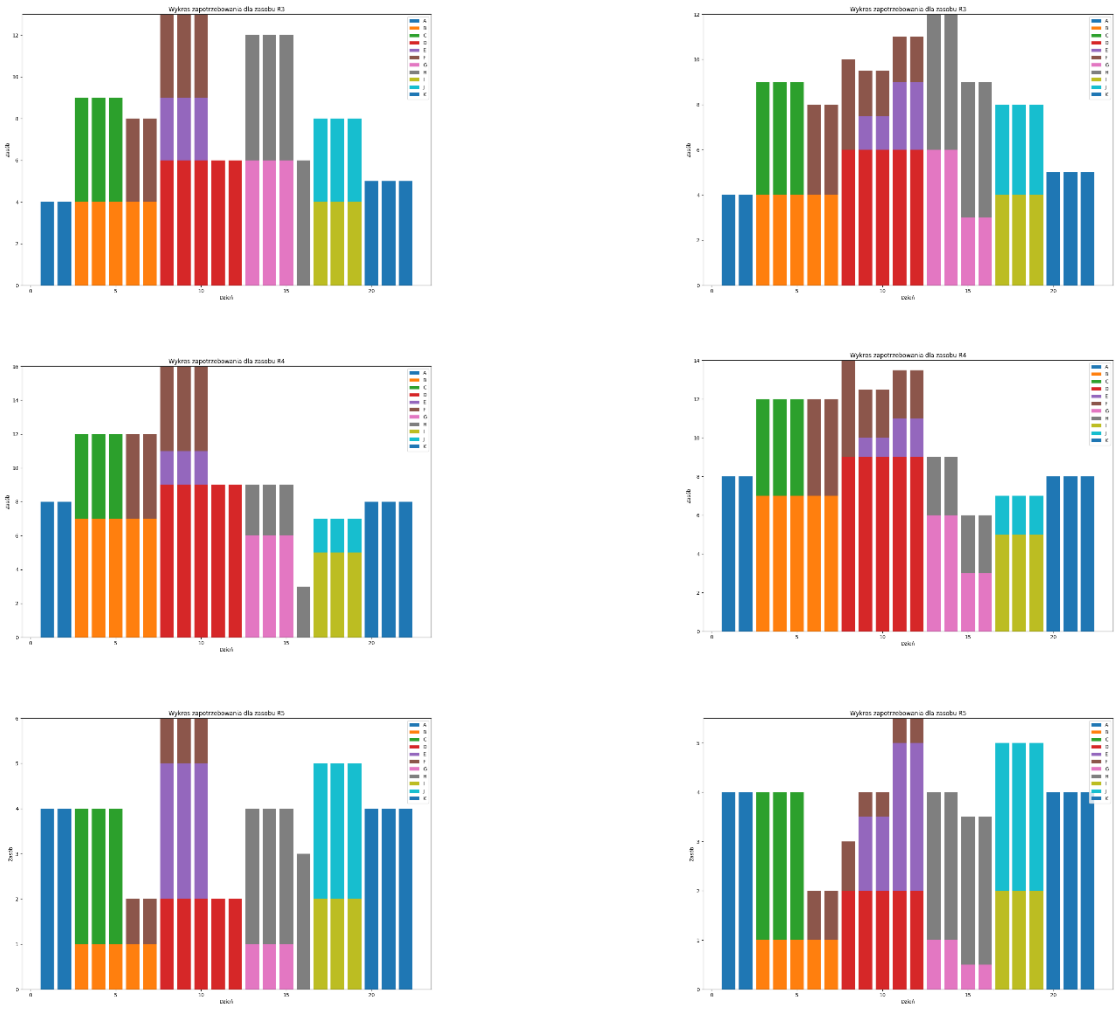




Zapotrzebowanie końcowe







Rysunek 21 Wyrównanie zapotrzebowania scenariusza czwartego

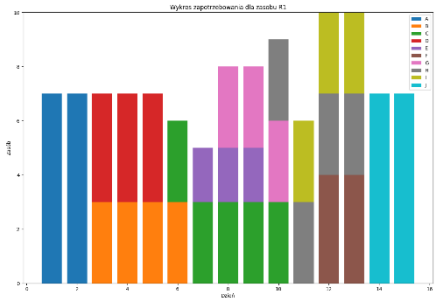
W scenariuszu czwartym algorytm musiał dbać o wiele relacji kolejnościowych i mimo ich poszukać miejsca, w których można byłoby wykonać wyrównanie. Jak możemy zaobserwować na powyższych rezultatach poradził sobie z tym zadaniem całkiem dobrze. Dobranie odpowiednich danych miałoby tutaj największy wpływ, ponieważ niektórych zadań nie jesteśmy wstanie ruszyć. Należą one do ścieżki krytycznej, natomiast niektóre są blokowane przez czynności poprzednie i następne.

4.3 Wpływ stosowanych metod mutacji na rozwiązanie

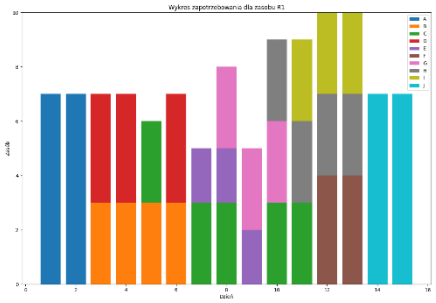
W tej części eksperymentu za pośrednictwem pierwszego scenariusza sprawdzono jakie znaczenie w poszukiwaniu optimum ma każda z czterech metod mutacji.

	Możliwe metody mutacji	Wartość funkcji celu
1	Alokacja	1719
2	Rozdzielanie	1625
3	Alokacja, Podział	1413
4	Rozdzielanie, Podział	1052

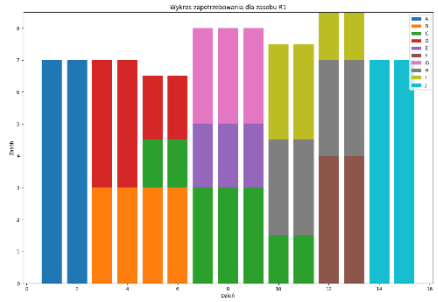
5	Alokacja, Dodanie dnia	6140
6	Rozdzielanie, Dodanie dnia	3766
7	Alokacja, Podział, Dodanie dnia	2266
8	Rozdzielanie, Podział, Dodanie dnia	2106



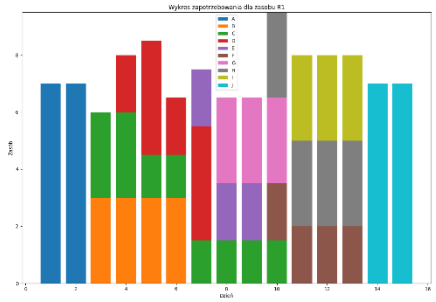
1



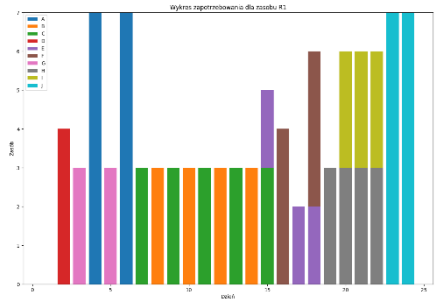
2



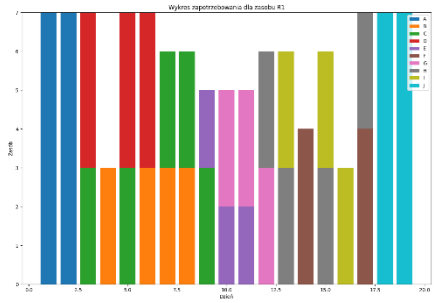
3



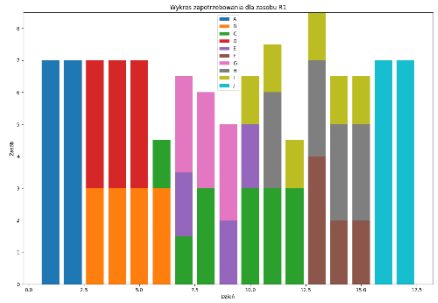
4



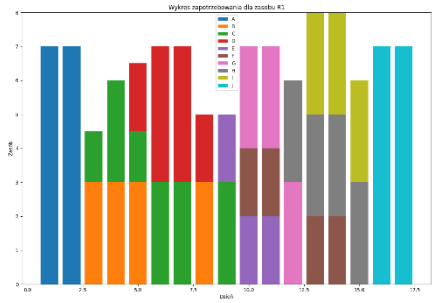
5



6



7



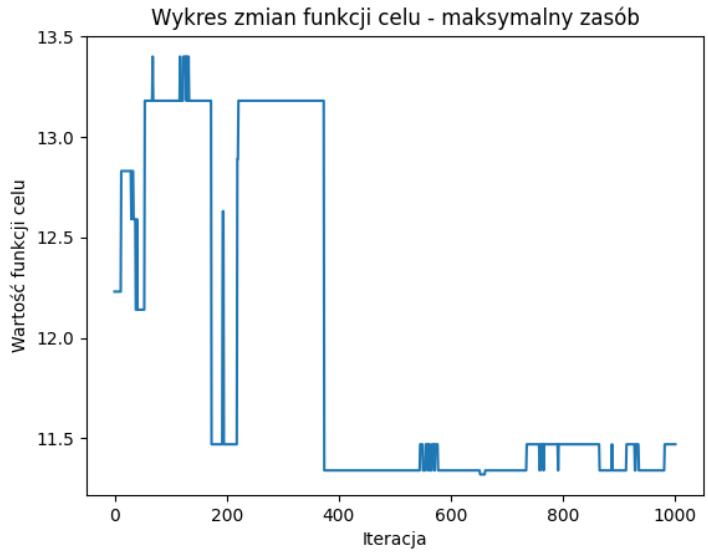
8

Rysunek 22 Wpływ mutacji na wyrównanie zasobów

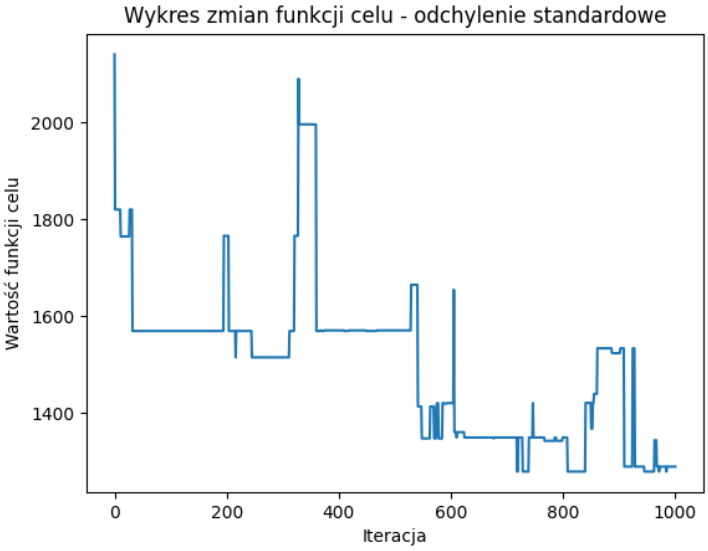
Jak można zauważyć najlepszy wpływ na poprawę rozwiązania ma w pierwszej kolejności podział zadania, a następnie rozdzielanie, czego połącznie daje najlepsze efekty. Duże znaczenie na poprawę lub pogorszenie wyniku ma także dodanie kolejnego dnia, natomiast jego waga zależy przede wszystkim od tego, jak dużą karą jest nakładana na funkcje celu za przeprowadzenie mutacji. W powyższych testach najprawdopodobniej została dobrana zbyt mała kara za dodanie dnia czego efekty możemy zauważyć na *wykresie nr 6*.

4.4 Wpływ zastosowanej funkcji celu na rozwiązanie

W tej części skupiono się na sprawdzeniu jak duży wpływ na znalezione rozwiązanie ma wybrana funkcja celu. W programie mamy do wyboru dwie funkcję minimalizację maksymalnego zasobu oraz odchylenie standardowe. Skorzystano ze scenariusza pierwszego oraz pobierano wyniki funkcji celu dla najlepszego osobnika po każdej iteracji algorytmu:



Rysunek 24 Wykres zmian funkcji celu dla minimalizacji maksymalnego zasobu

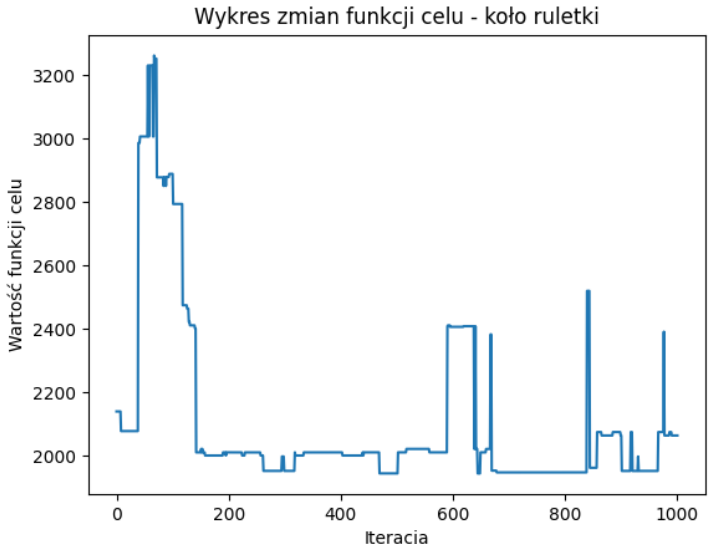


Rysunek 23 Wykres zmian funkcji celu dla odchylenia standardowego

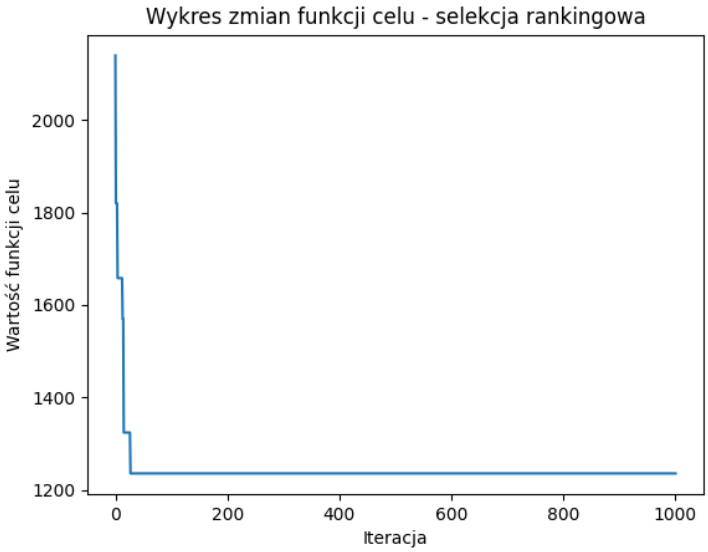
Jak możemy zaobserwować funkcja druga zatrzymuje się głownie na dwóch poziomach co spowodowane jest tym, że jeśli dla rozwiązania nie da się już bardziej zminimalizować maksimum, to wynik funkcji celu też nie ulegnie poprawie. Nie ma dla niej znaczenia ilość tych maksimów, ponieważ skupia się tylko na najwyższym punkcie. Natomiast funkcja pierwsza poprzez to, że sprawdza punkty najbardziej odległe w obie strony od średniej może znajdować lepsze wyniki, gdzie wynik maksymalny czy minimalny pojawia się rzadziej. Dlatego dla ogólnych przypadków lepiej zastosować drugą funkcję celu, natomiast pierwsza sprawdzi się lepiej w momencie chęci zejścia jakiegoś parametru poniżej ustalonego pułapu.

4.5 Wpływ zastosowanej metody selekcji na rozwiązanie

Kolejnym elementem godnym uwagi jest dobranie odpowiedniej metody selekcji. W algorytmie mamy doczynienia z dwoma metodami selekcji, dla których przeprowadzono test, aby sprawdzić którą lepiej wybierać w jakim przypadku. Testy zostały przeprowadzone dla scenariusza pierwszego:



Rysunek 25 Wykres zmian funkcji celu dla selekcji metodą koła ruletki

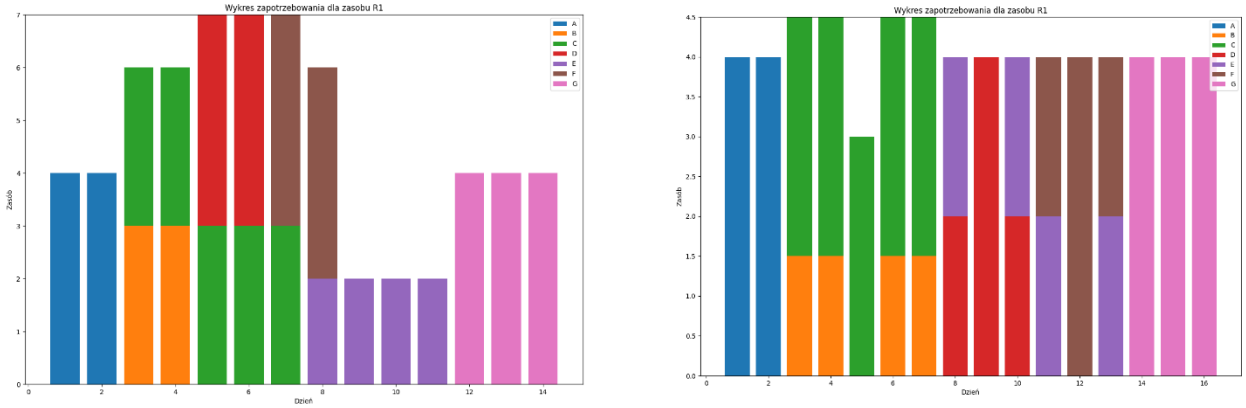


Rysunek 26 Wykres zmian funkcji celu dla selekcji rankingowej

Jak można zauważyć selekcja rankingowa dużo szybciej znalazła wynik suboptymalny, oraz kolejne iteracje nie spowodowały przesadnych zmian na funkcji celu. Metoda koła ruletki przez to, że do kolejnej iteracji może wybrać osobniki, które mogą być gorzej przystosowane pozwala na większą różnorodność w wynikach, co powoduje możliwość znalezienia lepszego rozwiązania. Dlatego przy małej liczbie iteracji programu lepiej wybrać selekcję rankingowo, gdzie zastosowanie selekcji koła ruletki, może przynieść lepsze efekty kiedy uruchamiamy program dla większej liczby iteracji.

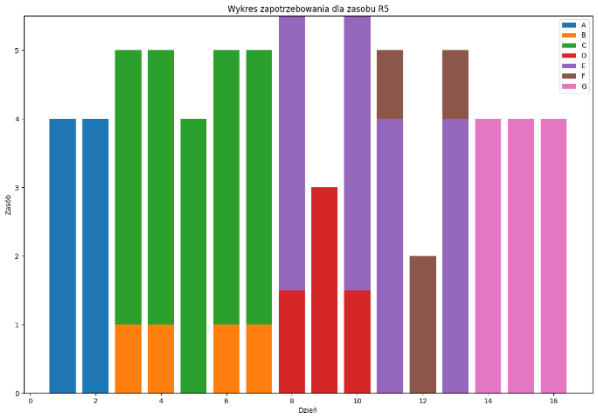
4.6 Wpływ priorytetów dla zasobu na rozwiązanie

Jedną z opcji programu jest nadanie priorytetu dla odpowiedniego zasobu przez co algorytm w dużo większym stopniu skupia się na optymalizacji tego właśnie zasobu. Na podstawie scenariusza drugiego postanowiono sprawdzić działanie nadając zadaniu numer 1 priorytet równy 100, który oznaczał, że optymalizacja tego zasobu nastąpi 100 razy częściej niż innych, oraz że funkcja celu będzie brała pod uwagę ten zasób z powyższą wagą.



Rysunek 27 Wykresy zmian zapotrzebowania przed i po wyrównaniu dla funkcji z priorytetem

Jak można zauważyć algorytm znalazł rozwiązanie optymalny dla zasobu pierwszego, gdzie każda zmiana pogorszyłaby tylko wyrównanie tego zasobu, ponieważ czynność zaznaczona na zielono należy do ścieżki krytycznej. Natomiast nie jest to rozwiązanie optymalne dla pozostałych zasobów jak możemy zobaczyć np. dla zasobu piątego:



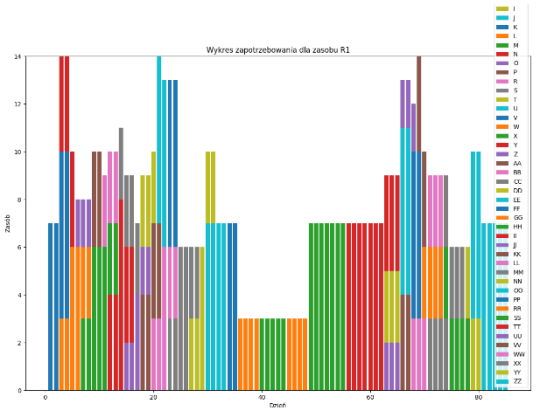
Rysunek 28 Wykres zapotrzebowania na zasób nr 5

Stosowanie więc priorytetów powinno zostać dobrane odpowiednie do potrzeb danego projektu, gdzie sam kierownik wie na wyrównaniu jakiego zasobu najbardziej mu zależy.

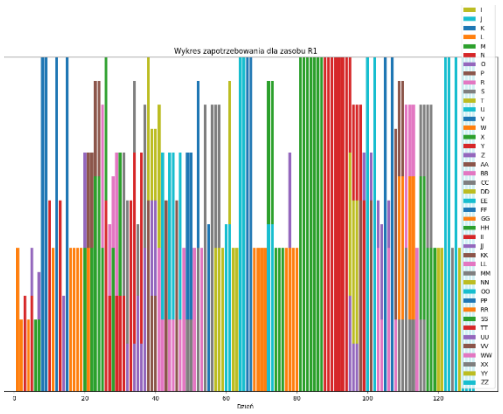
4.7 Działanie algorytmu dla dużego przypadku rzeczywistego

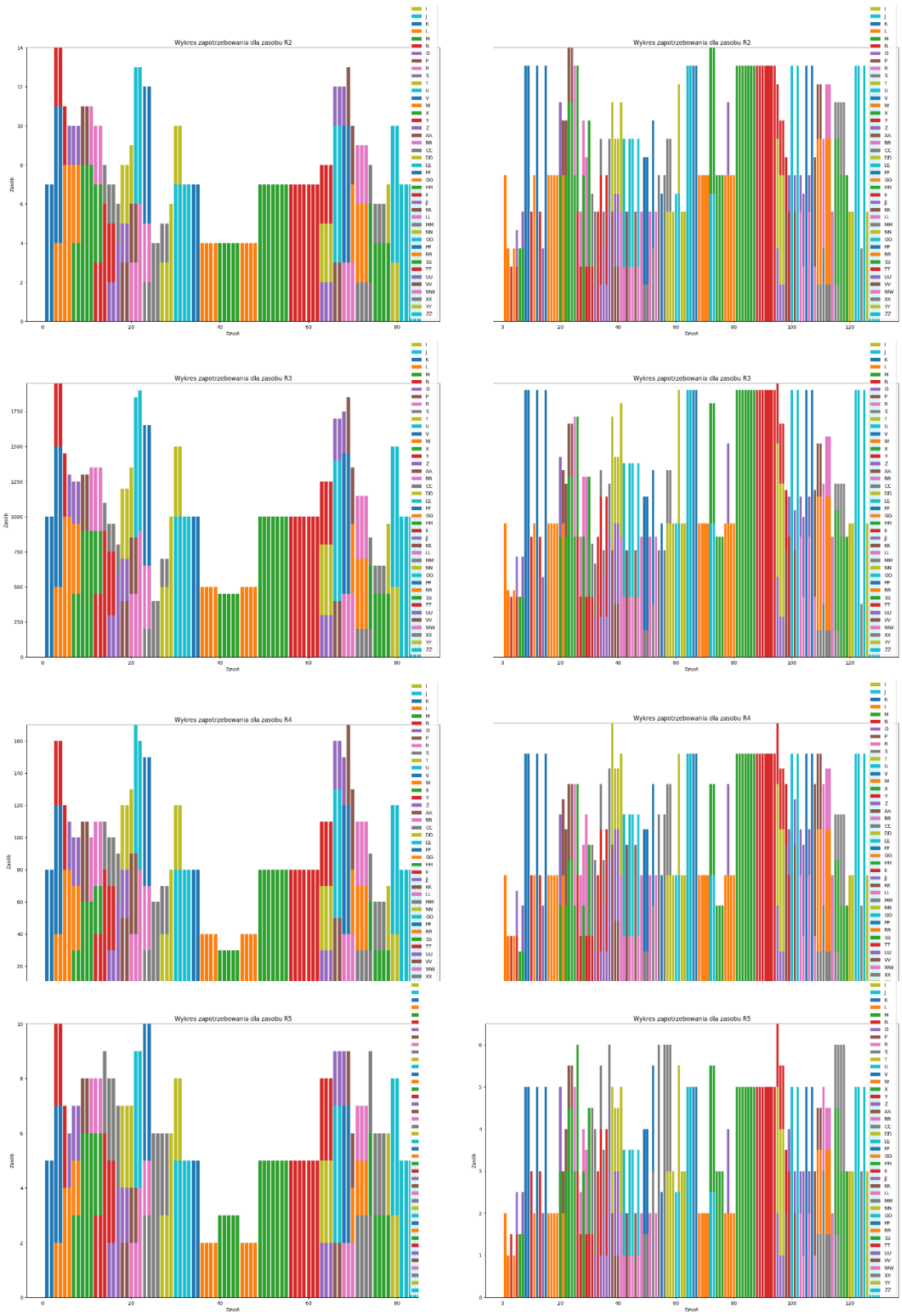
W rzeczywistych projektach zazwyczaj mamy doczynienia z dużą ilością czynności różnie połączonych ze sobą relacjami. Ostatni eksperyment miał na celu sprawdzenie jak program poradzi sobie w sytuacji, kiedy będzie miał do optymalizacji 50 zadań. Testy wykonano dla 500 iteracji przy włączonych wszystkich metodach mutacji. Otrzymano wyniki:

Zapotrzebowanie początkowe



Zapotrzebowanie końcowe





Rysunek 29 Wyrównanie dla scenariusza rzeczywistego

Jak możemy zauważyć zapotrzebowanie zostało wyrównane w dużo większym stopniu niż było ono na początku. Zostały dodane nowe dni, a także podzielone zadania. Najprawdopodobniej dla większej ilości iteracji wyrównanie byłoby jeszcze lepsze natomiast wykonanie takiego zadania przez program jest bardzo czasochłonne, dlatego też zastosowano tylko 500 iteracji. Natomiast można uznać wynik takiego wyrównania za sukces i można puszczać, że wywołanie takiego algorytmu na mocniejszym komputerze dla większej liczby iteracji przyniosłoby oczekiwany rezultat i optymalne rozwiązanie problemu.

Podsumowanie

Założeniem pracy była implementacja programu wspomagającego zarządzanie zasobami przy planowaniu projektów. Wspiera on użytkownika w wyrównaniu zasobów oraz stworzeniu harmonogramu prac.

Zakładane cele pracy zostały osiągnięte. Program spełnia swoje zadanie, proponując użytkownikowi harmonogram pracy dla wyrównanych zasobów, z uwzględnieniem relacji między czynnościami. Opracowany algorytm jest elastyczny i przystosowany do każdego rodzaju przedsięwzięcia. Zapewnia również możliwość manipulacji przez użytkownika w celu dodania odpowiednich ograniczeń.

Przeprowadzone eksperymenty obliczeniowe pokazały, że algorytm genetyczny jest skuteczną metodą do rozwiązanie problemu zarządzania zasobami. Pozwala on na generację rozwiązania zarówno dla małych jak i dużych projektów. Czas jego realizacji zależy głównie od wielkości problemu oraz liczby iteracji jaką ma wykonać. Można zauważyć, że funkcja celu stosująca odchylenie standardowe w lepszym stopniu radzi sobie ze znajdowaniem poprawnych wyników. Wybór odpowiednich ograniczeń przez użytkownika ma wpływ na rozwiązanie końcowe.

Praca pozostawia możliwość rozwoju poprzez dodania nowych funkcji celu, metod selekcji, mutacji czy krzyżowania. Mogłaby także zostać rozbudowana logika wyboru zasobu do optymalizacji oraz wyboru odpowiedniej mutacji, w celu zmniejszenia losowości przy wyborze odpowiednich metod. Opcją przydaną byłoby dodanie funkcji, która by sama proponowała odpowiednie ustawienia poszukiwania rozwiązania dla konkretnego przedsięwzięcia. Nie została wykonana aplikacja, która pozwoliłaby użytkownikowi w prosty sposób bez jakiejkolwiek wiedzy korzystać z programu, czego realizacja mogłaby być kolejnym etapem rozwoju.

Problem zarządzania zasobami jest ciągle aktualny, a idealne rozwiązanie nie istnieje, dlatego warto kontynuować badania związane z jego tematyką w celu znalezienia rozwiązania, które przyniosłoby najlepszy efekt i można byłoby je wykorzystać komercyjnie.

Dodatek

Całość projektu składa się z:

- Projektu Dyplomowego – niniejszy dokument.

- Dwóch plików Python (CPM.py i main.py) zawierających kod źródłowy wraz z opisami

Bibliografia

[1] Just Great DataBase, „Resource management,” [Online]. Available: https://jgdb.com/dictionary/resource-management. [Data uzyskania dostępu: 04 01 2023].

[2] B. Standards, Project management. Vocabulary. Nr BS 6079-2:2000, BSI, 2000.

[3] J. Węglarz, Sterowanie w systemach typu kompleks operacji, Warszawa–Poznań: PWN, 1981.

[4] N. Mingus, Zarządzanie projektami, Gliwice: Helion, 2001.

[5] M. Trocki i P. Wyrozębski, Planowanie przebiegu projektów, Warszawa: Szkoła Główna Handlowa w Warszawie, 2015.

[6] M. Park, „Model-based dynamic resource management for construction projects,” *Automation in Construction ,* tom 14, nr 5, pp. 585-598, 2005.

[7] S. Easa, „Resource leveling in construction by optimization,” *Journal of Construction Engineering and Management,* tom 115, nr 2, pp. 302-316, 1989.

[8] M. Bandelloni, M. Tucci i R. Rinaldi, „Optimal resource leveling using non-serial dynamic programming,” *European Journal of Operational Research,* tom 78, nr 2, pp. 162-177, 1994.

[9] J. Rieck, J. Zimmermann i T. Gather, „Mixed-integer linear programming for resource leveling problems,” *European Journal of Operational Research,* tom 221, nr 1, pp. 27-37, 2012.

[10] H. Ahuja, Construction Perfomance Control by Networks, New York: Wiley, 1976.

[11] K. Neumann i J. Zimmermann, „Procedures for resource leveling and net present value problems in project scheduling with general temporal and resource constraints,” *European Journal of Operations Research,* tom 127, nr 2, pp. 425-443, 2000.

[12] A. Burguess i J. Killebrew, „Variation in activity level on a cyclic arrow diagram,” *Journal of Industrial Engineering,* tom 13, pp. 76-83, 1962.

[13] P. Burman, Precedence Networks for Project Planning and Control, London: McGraw Hill, 1972.

[14] R. Harris, Precedence and Arrow Networking Techniques for Construction, New York : Wiley, 1978.

[15] R. Harris, „Packing method for resource leveling (PACK),” *Journal of Construction Engineering and Management,* tom 116, nr 2, pp. 331-350, 1990.

[16] M. Hiyassat, „Modification of minimum moment approach in resource leveling,” *Journal of Construction Engineering and Management,* tom 126, nr 4, pp. 278-284, 2000.

[17] V.A. Jeentra, O.V. Ksishnaiah Chetty i J. P. Redy, „Petri nets for project management and resource leveling,” *International Journal of Advanced Manufacturing Technology,* tom 16, pp. 516-520, 2000.

[18] J. Meridith i S. Mantel, Project Management with Approach, New York: Wiley, 1995.

[19] P. Jaśkowski i A. Sobotka, „Scheduling construction projects using evolutionary algorithm,” *Journal of Construction Engineering and Management,* tom 132, nr 8, pp. 861-870, 2006.

[20] V. K. Koumousis i P. G. Georgiou, „Genetic algorithms in discrete optimization of steel truss roofs.,” *J. Comput. Civ. Eng.,* tom 8, nr 3, pp. 309-325, 1994.

[21] A. Haidar, S. Naoum, R. Howes i J. Tah, „Genetic algorithms application and testing for equipment selection,” *J. Constr. Eng. Manage.,* tom 125, nr 1, pp. 32-38, 1999.

[22] H. Li i P. Love, „Using improved genetic algorithms to facilitate time-cost optimization,” *J. Constr. Eng. Manage.,,* tom 123, nr 3, p. 233–237, 1997.

[23] T. Hegazy, „Optimization of resource allocation and leveling using genetic algorithms,” *J. Constr. Eng. Manage,* tom 125, nr 3, p. 167–175, 1999.

[24] J. Son i M. Skibniewski, „Multiheuristic approach for resource leveling problem in construction engineering: hybrid approach,” *Journal of Construction Engineering and Management,* tom 125, nr 1, pp. 23-31, 1999.

[25] H. Alsayegh i M. Hariga, „Hybrid meta-heuristic methods for the multi-resource leveling problem with activity splitting,” *Automation in Construction,* tom 27, pp. 89-98, 2012.

[26] H. Adeli i A. Karim, „Scheduling/cost optimization and neural dynamics model for construction,” *J. Constr. Eng. Manage.,* tom 123, nr 4, p. 450–458, 1997.

[27] J. Kanet i H. Adelsberger, „Expert systems in production scheduling,” *Eur. J. Oper. Res.,* tom 29, nr 1, p. 51–59, 1987.

[28] H. Portman, „CHAOS 2020: Beyond Infinity,” *The Standish Group.*

[29] Encyklopedia Zarządzania, , „Algorytm genetyczny,” [Online]. Available: https://mfiles.pl/pl/index.php/Algorytm_genetyczny, https://sound.eti.pg.gda.pl/student/isd/isd03-algorytmy_genetyczne.pdf. [Data uzyskania dostępu: 04 01 2023].