

**WYDZIAŁ ELEKTROTECHNIKI, AUTOMATYKI, INFORMATYKI I INŻYNIERII BIOMEDYCZNEJ**

Praca dyplomowa

*Opracowanie heurystyk do optymalizacji problemów dyskretnych z wieloma ograniczeniami*

*Metaheuristics for multi-constrain discrete optimization problems*

Autor: *Maciej Morgalla*

Kierunek studiów: Automatyka i Robotyka

Opiekun pracy: *dr hab. inż. Wojciech Chmiel*

Kraków, 2023

# Spis treści

[1.Wstęp 3](#_Toc143081183)

[1.1. Wprowadzenie 3](#_Toc143081184)

[1.2. Cel i zakres pracy 3](#_Toc143081185)

[2. Rozważany problem i metody jego rozwiązania 4](#_Toc143081186)

[2.1. Produkcja i jej ograniczenia 4](#_Toc143081187)

[2.2. Metody optymalizacji produkcji 4](#_Toc143081188)

[3. Opis przyjętego modelu matematycznego 5](#_Toc143081189)

[3.1. Procesy produkcyjne 5](#_Toc143081190)

[3.2. Operacje 7](#_Toc143081191)

[3.3. Maszyny 10](#_Toc143081192)

[3.4. Symulacja 11](#_Toc143081193)

[3.5 Funkcja kary 12](#_Toc143081194)

[4. Stosowane algorytmy 14](#_Toc143081195)

[4.1. Algorytm genetyczny 14](#_Toc143081196)

[4.2. Schemat działania algorytmu genetycznego 16](#_Toc143081197)

[4.3. Algorytm Johnsona 16](#_Toc143081198)

[4.4. Wykorzystanie algorytmu Johnsona 17](#_Toc143081199)

[4.5. Utworzone heurystyki 18](#_Toc143081200)

[5. Opis aplikacji 21](#_Toc143081201)

[5.1. Środowisko programistyczne 21](#_Toc143081202)

[5.2. Proces powstawania i testowania aplikacji 22](#_Toc143081203)

[5.3 Struktura aplikacji 23](#_Toc143081204)

[6. Wybrane scenariusze testowe 23](#_Toc143081205)

[6.1. Scenariusz 1 23](#_Toc143081206)

[7. Podsumowanie 23](#_Toc143081207)

[8. Bibliografia 23](#_Toc143081208)

# 1.Wstęp

## 1.1. Wprowadzenie

Rozwój cywilizacji i napędzający ją postęp technologiczny stwarza każdemu człowiekowi szansę, na życie lepsze niż życie jego przodków. Postępująca industrializacja i mechanizacja, a w ostatnim czasie automatyzacja procesów produkcyjnych pozwoliła na zaoszczędzenie materiałów, czasu, energii i pieniędzy przy produkcji wyrobów znanych od dekad, oraz na wytwarzanie zupełnie nowych, o wyższej złożoności, jak na przykład mikroprocesory. Niestety przy tej okazji należy podkreślić, że często obniżenie kosztów produkcji towarzyszy obniżeniu jakości towarów. Kiedyś produkty AGD były droższe i produkowane w mniejszej ilości, ale jednocześnie były bardziej trwałe niż te produkowane w dzisiejszych czasach. Być może należałoby podnieść jakość obecnie produkowanych urządzeń, a ich ceny zmniejszać poszukując innych sposobów.

Proces produkcyjny zazwyczaj składa się choć z kilku operacji, które wykonuje się w określonym czasie, miejscu i kroku produkcji. Wobec tego usprawnienie przebiegu produkcji mogłoby ograniczyć znacząco czas, koszty, straty energii i inne jego parametry. Nie jest to jednak łatwe, szczególnie w przypadku dużej elastyczności parku maszynowego i dużej ilości procesów.

## 1.2. Cel i zakres pracy

# 2. Rozważany problem i metody jego rozwiązania

## 2.1. Produkcja i jej ograniczenia

## 2.2. Metody optymalizacji produkcji

# 3. Opis przyjętego modelu matematycznego

## 3.1. Procesy produkcyjne

## 

W celu wytworzenia danego produktu lub półproduktu konieczne jest przeprowadzenie odpowiedniego procesu, a co za tym idzie składających się na niego kroków. Wobec tego, kiedy wytwórca otrzymuje zamówienie na określone towary, bądź konieczne jest uzupełnienie asortymentu, zleca się wykonanie określonych procesów produkcyjnych. W przyjętym modelu matematycznym procesy te są elementami zbioru .

Każdy proces ma inny charakter, gdyż każdy skutkuje wytworzeniem towaru o innych właściwościach. W zależności od poziomu skomplikowania produkcji mogą się składać z zaledwie kilku lub z wielu operacji, o określonym czasie trwania. Z uwagi na różnorodność czynności wytwórczych, niektóre procesy mogłyby być wykonywane przez nieskończenie długi czas, gdyż upływ czasu nie wpływa na jakość wyrobów. Inne jednak, jak na przykład procesy w których kluczową rolę odgrywa obróbka cieplna, muszą zostać wykonane w skończonym czasie, aby zachować parametry wyrobu końcowego. Dlatego niezwykle ważne jest zachowanie narzuconych limitów czasowych. Ze względu na dużą korelację czasu wykonywania procesu z czasem trwania poszczególnych operacji i przerw między nimi, model matematyczny nie uwzględnia parametru uwzględniającego maksymalny czas trwania procesu, gdyż zawiera się on w limitach nałożonych na kroki niższego poziomu.

Każdy proces składa się z wielu operacji , które zgodnie z założeniami muszą zostać wykonane w ściśle ustalonej kolejności (rys. 3.1). W rzeczywistych procesach istnieją takie operacje, które mogą być wykonywane jednocześnie, jednak ze względu na duże komplikacje wynikające z takiej możliwości, model nie przyjmuje takich rozwiązań. Mimo to możliwa jest taka implementacja, przy czym proces taki należałoby podzielić na rozpatrywane osobno podprocesy o priorytetach skłaniających algorytm do umieszczenia ich w planie produkcji w wymaganej kolejności.

*Rysunek 3.1 Przykładowa kolejność wykonywania operacji w ramach procesu.*

Ze względu na różne właściwości procesów tj. ich długość trwania, miejsce w strategii firmy, odbiorca itp. mogą one posiadać określony priorytet , który odzwierciedla ich wagę wobec innych procesów. Jeśli proces posiada wyższy priorytet od innych, powinien zgodnie z założeniami zostać wykonany w pierwszej kolejności. Tutaj należy jednak zwrócić również uwagę na to, że w założeniu istnieje ograniczenie czasowe symulacji wynoszące jedną dobę, bowiem zakłada się, iż zlecenia wykonania procesów napływają raz dziennie, co w większości przypadków nie powoduje problemów, gdyż zamówienia wykonywane są na określony dzień. Wobec tego zachowanie priorytetu nie należy do najważniejszych celów do osiągnięcia przez rozwiązanie końcowego. Mimo to zostało zachowane w modelu, gdyż może być okolicznością zbliżającą rozwiązanie do perfekcyjnego.

*Tabela 3.1 Przykładowe parametry procesu.*

|  |  |  |  |
| --- | --- | --- | --- |
| Nazwa parametru | Oznaczenie | Przykładowa wartość | Uwagi |
| numer identyfikacyjny |  | 8 | - |
| priorytet |  | 4 | , przy czym większa wartość oznacza większy priorytet |
| zbiór operacji |  | {5,2,6,8} | , kluczowa jest kolejność wykonywania operacji |

## 3.2. Operacje

Jak wspomniano w poprzednim rozdziale, pojedyncza operacja stanowi część procesu. Tak jak wynikiem całego procesu jest wytworzenie danego produktu lub półproduktu, tak wynikiem wykonania operacji jest półprodukt o określonej strukturze i określonych właściwościach m.in. cieplnych i fizycznych. Przykładem operacji może być nalewanie soku do butelek, czy też podgrzewanie cieczy w kadzi do określonej temperatury

W niektórych przypadkach wymagane jest wykonanie operacji przed lub po pewnym czasie od zakończenia poprzedniej operacji. Może to być czas potrzebny np. na ostudzenie półproduktu, bądź czas, w którym półprodukt wciąż zachowuje swoje właściwości potrzebne w kolejnej operacji, czy też jakość półproduktu nie pogarsza się. Zachowanie limitów czasowych może być kluczowe w produkcji, dlatego należy zawsze je spełniać, gdyż ich niedotrzymanie może skutkować wytworzeniem wadliwego towaru. Model matematyczny w przypadku, gdy konieczne jest odczekanie przed rozpoczęciem kolejnej operacji, wymaga powiększenia o wymaganą wartość czas wykonywania operacji na wymagających tego maszynach. Jeśli zaś chodzi o maksymalny czas, w którym kolejna operacja ma zostać wykonana, to wartość tą można ustawić w parametrach operacji, zarówno dla okna czasowego przed jak i po.

Kluczowym parametrem wykonywania operacji jest czas. O ile ważne jest zachowanie limitów czasowych pomiędzy operacjami, tak i same operacje wykonywane są w określonym czasie. Ze względów praktycznych czas może się różnić w zależności od maszyny czy też stanowiska, na którym jest wykonywana. Wiąże się to z różnymi właściwościami maszyn (na przykład mocą), poziomem doświadczenia pracowników na stanowisku itp. (patrz rozdz. 3.3).

Każda operacja do prawidłowego wykonania potrzebuje określony surowiec, produkt lub półprodukt w odpowiedniej ilości, zaś po jej wykonaniu otrzymuje się inne produkty lub półprodukty przeznaczone do kolejnych operacji lub do dystrybucji. Dlatego niezwykle ważnym elementem symulowania procesu produkcji jest sprawdzanie ciągłości dostaw wymaganych surowców w odpowiedniej ilości. Pewną komplikacją może być sposób wytwarzania nowych produktów, gdyż może być przeprowadzony zgodnie z dwoma scenariuszami. W pierwszym scenariuszu wszystkie produkty operacji są gotowe do dalszego użycia dopiero w momencie jej zakończenia. Przykładem takiej operacji może być podgrzewanie cieczy. Dzięki odpowiedniemu mieszaniu ciecz powinna osiągnąć zadaną wartość temperatury w całym naczyniu dopiero po upływie pewnego czasu. Nawet jednak jeśli weźmie się pod uwagę nierównomierne nagrzewanie się cieczy, to oddzielenie cieczy o mniejszej temperaturze od tej o odpowiedniej temperaturze w większości przypadków jest niemożliwe, a w dodatku nie zapewnia, że nie zostanie pobrana niewłaściwa ilość. Przeciwnie, w drugim scenariuszu pewna ilość produktów jest gotowa do dalszej obróbki jeszcze podczas trwania operacji. Doskonałym przykładem takich operacji są działania wykonywane na taśmach produkcyjnych, gdzie gotowe produkty zjeżdżają z taśmy co określony czas. Stworzony model matematyczny zapewnia ustawienie nieregularnego wypuszczania produktów, co pozwala na symulowanie produkcji nie opierających się na maszynach (które zazwyczaj działają z taką samą częstotliwością), a na stanowiskach, obsługiwanych przez pracowników ludzkich, oraz na dowolne pobieranie gotowych produktów niezależnie od postępu produkcji.

Dodatkowo do modelu matematycznego zaimplementowana została możliwość pauzowania wykonywania operacji. Niestety jej używanie jest niepraktyczne z powodu wykorzystywanych algorytmów, bowiem zaimplementowany algorytm Johnsona opiera się w na przydzielaniu operacji do maszyn w odpowiednim czasie i kolejności, bez uwzględniania ewentualnych pauz, zaś algorytm genetyczny opiera się na harmonogramowaniu operacji i przerw pomiędzy operacjami. Dodanie możliwości pauzowania pozwoliłoby na ogromny bałagan w potencjalnych rozwiązaniach i choć zapewne dążyłoby do optymalnych obszarów, to utrudniałoby poszukiwanie, nie uwzględniając już o ich utraty przy korzystaniu z algorytmu Johnsona.

*Tabela 3.2 Przykładowe parametry operacji.*

|  |  |  |  |
| --- | --- | --- | --- |
| Nazwa parametru | Oznaczenie | Przykładowa wartość | Uwagi |
| numer identyfikacyjny |  | 3 | - |
| maksymalny czas rozpoczęcia po poprzedniej operacji [s] |  | 24 | - |
| maksymalny czas od zakończenia operacji do rozpoczęcia kolejnej [s] |  | 9.2 | W przypadku podania parametru  w kolejnej operacji model ignoruje ten parametr. |
| macierz potrzebnych surowców |  |  | Każda kolumna oznacza inny surowiec. Ostatnia kolumna oznacza % czasu, w którym następuje pobranie surowców w podanej liczbie. |
| lista produktów |  |  | Każda kolumna oznacza inny produkt. Ostatnia kolumna oznacza % czasu, w którym możliwe jest odebranie produktów w podanej liczbie. |
| możliwość pauzowania\* |  | prawda (true) | - |
| maksymalny czas pauzy [s]\* |  | 100 | - |
| maksymalna liczba pauz\* |  | 5 | Nadanie wartości -1 oznacza możliwość nielimitowanego pauzowania |

\*zmienna nie ma zastosowania przy stosowanych algorytmach

## 3.3. Maszyny

Choć operacja i jej właściwości są niezwykle ważne w kontekście poszukiwania rozwiązań problemu, to jednak dużo istotniejsze są obsługujące je maszyny. Pod pojęciem maszyny kryje się wiele sposobów przetwarzania produktów, od stacji roboczej, na której obróbkę wykonują pracownicy ludzcy, przez zautomatyzowane stanowiska, aż po produkcję taśmową. Pomimo tak dużej różnorodności, na potrzeby tej pracy, używa się tylko ogólnego pojęcia maszyna.

Maszyny mogą różnić się między sobą, chociażby wykonywalnymi operacjami. Przykładowo proces toczenia czy gwintowania byłby niemożliwy do zrealizowania na stanowisku przystosowanym tylko do malowania. Dlatego każda maszyna posiada swój własny zbiór operacji, które jest w stanie wykonać. Ponadto czas wykonywania tej samej operacji na dwóch różnych maszynach może się różnić. Powodem takiego stanu rzeczy może być różnica w konstrukcji maszyn, czy też doświadczenia kadry pracowniczej. Wobec tego możliwe jest optymalizowanie czasu produkcji, poprzez przydzielanie operacji do maszyn, które wykonują ją najszybciej. Jednocześnie pojawia się zagrożenie, bowiem może istnieć jedna supermaszyna, która może wykonać wszystkie operacje i w dodatku w najkrótszym czasie, jednakże wykonanie wszystkich operacji może zająć jej więcej czasu, niż gdyby niektóre działania przejęły inne maszyny. Tym ważniejsze staje się korzystanie z algorytmów.

Inną właściwością maszyny jest możliwa konieczność jej przekalibrowania, w momencie zakończenia operacji jednego typu, aby przystosować ją do wykonania kolejnej operacji lecz innego typu. Przykładem może być maszyna nalewająca soki do kartonów. Oczywiste jest, że gdy zmienia się nalewany sok z jabłkowego na porzeczkowy, konieczna jest zmiana opakowań, do których nalewa się soki, oraz oczyszczenie maszyny z resztek poprzedniego soku, co wymaga przerwy w produkcji. Czas ten może się znacząco różnić nawet przy przejściu z tej samej operacji, gdyż następujące operacje mogą posiadać zupełnie inny charakter i mogą mieć zupełnie inne wymagania produkcyjne.

*Tabela 3.3 Przykładowe parametry maszyny.*

|  |  |  |  |
| --- | --- | --- | --- |
| Nazwa parametru | Oznaczenie | Przykładowa wartość | Uwagi |
| numer identyfikacyjny |  | 0 | - |
| macierz przekalibrowania [s] |  |  | Każda operacja jest reprezentowana przez jedną kolumnę i jeden wiersz. Kolumny oznaczają operację poprzedzającą, a rzędy następującą. |
| lista czasów wykonywania poszczególnych operacji [s] |  | {0,0,15,28,0,56} | Zera reprezentują operacje, które nie mogą być wykonywane. Algorytm, przed zleceniem maszynie operacji, każdorazowo sprawdza czy czas na liście jest niezerowy. |

## 3.4. Symulacja

Kluczowym krokiem w badaniu rozpatrywanego rozwiązania jest wykonanie symulacji zgodnie z jego harmonogramem. Do jej przeprowadzenia konieczne jest stworzenie list zawierających informacje o wszystkich procesach, operacjach i maszynach, zgodnie z tabelami 3.1-3.3. Ponadto należy utworzyć listę zawierającą początkową ilość wszystkich surowców, półproduktów i produktów. Informacje zawarte we wszystkich czterech listach, wraz z rozwiązaniem proponowanym przez algorytm, są wystarczające, aby obliczyć parametry potrzebne do oceny rozwiązania.

W ramach symulacji model matematyczny rozpatruje kolejno każdą maszynę, dodając do czasu jej pracy wszystkie zawarte w harmonogramie czasy operacji, czasy przerw, oraz czasy na przeregulowanie. We wszystkich kluczowych momentach tworzona jest nowa struktura zawierająca informacje konieczne do oceny symulacji.

Wynikiem działania symulacji są trzy listy zawierające: czasy rozpoczęcia i zakończenia poszczególnych symulacji wraz ze szczegółowymi informacjami dotyczącymi procesu i maszyny, zapotrzebowania operacji pod postacią czasu zaistnienia potrzeby wraz z id i ilością surowca oraz produkty operacji w takiej samej postaci jak lista potrzeb.

## 3.5 Funkcja kary

Wyniki symulacji w postaci trzech list są kluczem do oceny rozwiązania. Analiza list pod kątem poprawności wraz z odnotowaniem w odpowiedniej postaci wszelkich nieprawidłowości pozwala na sprawdzenie przydatności rozwiązania poprzez użycie funkcji kary.

W założeniu funkcja kary powiększa się znacznie przy zaistnieniu każdego zaburzenia w procesie produkcyjnym. Jej podstawowym składnikiem jest czas trwania całej produkcji. Jest on szczególnie ważny w przypadku istnienia dwóch rozwiązań o takim samym stopni zgodności z założeniami, lecz o innych czasach, bowiem wtedy mniejsza wartość tej zmiennej wskazuje na lepszą alternatywę.

*x* – oceniane rozwiązanie

*a, b, c, d, e, f, g* – wagi poszczególnych zmiennych

– liczba procesów, w których kolejność operacji została zaburzona

*t* – czas potrzebny do wykonania wszystkich procesów [s]

– liczba nadmiarowych pauz

– nadmiarowy czas pauz [s]

– liczba procesów o niższym priorytecie, które zostały wykonane przed procesami o wyższym priorytecie

– bezwzględna różnica pomiędzy granicznym czasem trwania przerwy pomiędzy operacjami, a osiągniętym czasem

– ilość surowców niedostarczonych na czas do wykonania operacji

Wzór 3.1 przedstawia sposób przełożenia jakości rozpatrywanego rozwiązania na wartość liczbową. Zgodnie z założeniami im wynik funkcji jest mniejszy, tym lepsze jest rozwiązanie.

Należy przypomnieć, iż heurystyki używane do optymalizacji rozwiązań nie uwzględniają istnienia pauz, przez co z nich nie korzystają. Wobec tego zmienne oraz we wszystkich prawidłowych obliczeniach wynoszą zero.

# 4. Stosowane algorytmy

## 4.1. Algorytm genetyczny

Algorytm genetyczny (GA) należy do rodziny algorytmów ewolucyjnych. W 1975 roku John Henry Holland wydał książkę *Adaptation in Natural and Artificial Systems*, którą uważa się za przełom w zakresie badań nad algorytmami tego typu. To właśnie ten naukowiec przedstawił w latach dziewięćdziesiątych jego ideę. [4.1]

Podobnie jak wiele innych algorytmów przybliżonych, pomysł na GA zrodził się z inspiracji naturą. Tak jak ewolucja dąży do najlepszych możliwych kombinacji genetycznych, aby pozwolić na przetrwanie gatunków, poprzez przystosowanie się do zmieniającego się środowiska, tak algorytm genetyczny stara się łączyć pozytywne cechy najlepszych rozwiązań z danego zbioru, aby każda kolejna generacja rozwiązań zbliżała się do globalnego optimum.

Praca algorytmu rozpoczyna się od stworzenia losowej początkowej populacji. Następnie ocenia się każdego z osobników za pomocą określonej miary np. funkcji kary. Jeśli jeden z osobników okazuje się najlepszym spośród wszystkich dotychczasowych osobników, to zapamiętuje się jego genotyp, gdyż jest to potencjalnie najlepsze rozwiązanie. Dalej oddziela się określony odgórnie odsetek najlepszych osobników, w celu krzyżowania ich genotypu i otrzymania kolejnego pokolenia. Ponadto możliwe jest wykonanie odgórnie narzuconej liczy mutacji, zmieniających nieznacznie genotypy nowych osobników. Mutacje mają na celu losowe poszukiwanie nowych rozwiązań, a ponadto umożliwiają wydostanie się z potencjalnych minimów lokalnych. [4.2]

Na potrzeby tej pracy algorytm genetyczny został przystosowany do przedstawionego problemu (rozdział 3). Trzymając się terminologii biologicznej genotyp pojedynczego rozwiązania problemu składa się z chromosomów o liczbie równej liczbie maszyn w modelu matematycznym. Każdy chromosom to nic innego jak lista wszystkich operacji i przerw (lista genów), które maszyna ma za zadanie wykonać [4.3]. Należy podkreślić, że w stworzonym modelu matematycznym dopuszcza się sytuację, w której chromosom pozostaje pusty tj. na maszynie nie będzie wykonywana żadna operacja, a jedynym działaniem będzie realizacja przerwy w działaniu.

Proces tworzenia nowej populacji polega na iteracji po zbiorze wszystkich operacji tworzących poszczególne procesy oraz na przydzielaniu ich do losowych maszyn. Proces krzyżowania zaś polega na przekazywaniu losowo przez jednego z rodziców całych chromosomów (a więc całego harmonogramów dla określonych maszyn). Ponadto, aby uchronić się przed powieleniem oraz zanikaniem pewnych operacji, prowadzona jest ich kontrola, która uniemożliwia przekazanie przez jednego z rodziców genu, który znajduje się już w innym chromosomie. Gdy zaś zostanie wykryty gen, który nie znalazł się w żadnym chromosomie, to zostaje on przekazany przez jednego z rodziców do chromosomu pochodzącego od drugiego rodzica.

Aby uniknąć pojawienia się niepożądanych genów na chromosomie, sprawdza się czy chromosom (maszyna) przyjmuje taki gen (operację). Dzięki temu w populacji nie pojawiają się „mutanty”, które mogłyby wprowadzić zamieszanie w całej kolejnych generacjach. Byłoby to problematyczne szczególnie przy dużych rozmiarach problemu, gdyż większa liczba maszyn i operacji zwiększałaby prawdopodobieństwo przydzielenia maszynie operacji, która nie jest w stanie jej wykonać.

Jako, że pula genów w populacji jest ograniczona konieczne jest dokonywanie dodatkowych mutacji genów. Ilość mutacji w perspektywie całej populacji jest bardzo mała, bowiem ich duża ilość mogłaby oddalać większość rozwiązań w generacji od minimum lokalnego w niepożądane rejony.

Mutacje wykonują się na losowej części nowej populacji poprzez dodanie nowej pozycji przerwania pracy na losowy czas do harmonogramu lub poprzez zmianę lokalizacji genu na inny chromosom (tj. zlecenie wykonania operacji na innej maszynie).Tutaj także sprawdza się czy gen może znaleźć się na chromosomie, aby nie dopuścić do pojawienia się zupełnie niedopuszczalnych rozwiązań.

## 4.2. Schemat działania algorytmu genetycznego (na podstawie [4.3])

Krok 1: Stworzenie populacji początkowej o zadanej liczności,

Krok 2: Ocena z osobna każdego osobnika z populacji za pomocą funkcji kary,

Krok 3: Zapamiętanie najlepszego osobnika, jeśli jest globalnie najlepszy,

Krok 4: Wybór zadanego odsetku najlepszych osobników do krzyżowania,

Krok 5: Losowe krzyżowanie ze sobą najlepszych osobników,

Krok 6: Wykonanie losowych mutacji dla określonego odsetku nowej generacji,

Krok 7: Sprawdzenie warunku zatrzymania algorytmu. Jeśli warunek pozostaje niespełniony to powraca do kroku 2,

Krok 8: Zatrzymanie algorytmu

## 4.3. Algorytm Johnsona

Algorytm Johnsona (JA) został przedstawiony przez S. M. Johnsona w pierwszej połowie lat 50-tych XX wieku. Jest to algorytm dokładny umożliwiający ułożenie zadań wykonywanych przez dwie maszyny produkcyjne w takiej kolejności, aby wykonanie ich wszystkich trwało jak najkrócej. [4.4]

Zgodnie z założeniami każde zadanie ma zostać wykonane najpierw na jednej, a później na drugiej maszynie. Ponadto każde zadanie jest wykonywane na danej maszynie w określonym stałym czasie, który może się różnić pomiędzy maszynami. [4.5]

Działanie algorytmu rozpoczyna się od wybrania tych zadań, które są wykonywane szybciej na pierwszej maszynie. Następnie zadania te sortuje się rosnąco w zależności od czasu potrzebnego na realizację. Pozostałe zadania sortuje się malejąco i umieszcza na końcu utworzonej listy. Tak oto powstaje harmonogram, zgodnie z którym działanie rozpoczyna maszyna pierwsza. Po każdorazowym zakończeniu zadania na pierwszej maszynie, to samo zadanie zostaje przekazane do drugiej maszyny, która wykonuje je gdy tylko jest wolna. [4.2]

Niestety niewiele problemów dotyczących harmonogramowania opiera się na tylko dwóch maszynach. Istnieje jednak możliwość teoretycznego łączenia maszyn w taki sposób, że dodaje się wszystkie czasy wykonywania zadań dla wszystkich maszyn poza ostatnią (supermaszyna 1), oraz wszystkich poza pierwszą (supermaszyna 2). Dalsze procedowanie jest identyczne, przy czym maszyny wykonują zadania zgodnie z numeracją, czyli rozpoczynając od pierwszej, a kończąc na ostatniej. [4.5]

## 4.4. Wykorzystanie algorytmu Johnsona

Przyjęty model matematyczny nie wydaje się spełniać założeń algorytmu Johnsona. Na wstępie należy podkreślić, że każdy proces posiada niezamienialną kolejność wykonywania operacji. Wobec tego niewskazane jest tworzenie harmonogramu zgodnie z kolejnością maszyn. Co gorsza, próba zmiany kolejności maszyn zgodnie z jednym procesem może całkowicie nie zgadzać się z innym. W związku z tym, procesy wykonuje się według kolejności wynikającej z działania algorytmu Johnsona, ale zgodnie z ich wewnętrzną kolejnością operacji i przydzielonymi do nich maszynami. Zmiana ta przekreśla pewność co do optymalności rozwiązania wynikającego z działania algorytmu.

Dodatkowo należy podkreślić, że w większości przypadków operacje należące do jednego procesu nie mogą być wykonywane na wszystkich maszynach ze względu na ich ilość oraz na ograniczenia maszyn co do typów wykonywanych operacji (rozdział 3.3). Wobec tego uznaje się, że gdy żadna operacja należąca do procesu nie jest wykonywana na danej maszynie, to czas trwania procesu na tej maszynie wynosi zero sekund. W odwrotnym przypadku tj. gdy jedna maszyna wykonuje dwie lub więcej operacji z tego samego procesu, czasy te sumuje się.

Główną zaletą tak przyjętego wariantu algorytmu Johnsona jest jego pewność co do kolejności wykonywania procesów, jako, iż w przeciwieństwie do algorytmu genetycznego, opiera harmonogramowanie właśnie o ten czynnik, a nie o kolejność numeracji maszyn. Ponadto uwzględnia wszystkie przerwy wymagane do przekalibrowania maszyn, wymagane okna czasowe i przerwy wynikające z oczekiwania na ukończenie wcześniejszych zadań przez inne maszyny. Dlatego użycie tego algorytmu gwarantuje otrzymanie rozwiązań, które z dużym prawdopodobieństwem będą rozwiązaniami dopuszczalnymi.

Ważną kwestią pozostaje również charakter działania algorytmu Johnsona. W przeciwieństwie do algorytmu genetycznego nie pozwoliłby na wskazanie lepszego rozwiązania z inną kombinacją operacji wykonywanych przez maszynę, bowiem nie służy do przeszukiwania przestrzeni rozwiązań.. Wobec tego konieczna jest kooperacja z algorytmem genetycznym i opieranie się na rozwiązaniach znalezionych przez niego.

## 4.5. Utworzone heurystyki

Użycie dwóch algorytmów o zupełnie innych charakterach pozwoliło na stworzenie kilku wariantów heurystyk. Głównym algorytmem do przeszukiwania przestrzeni rozwiązań pozostaje algorytm genetyczny. Dzięki czynnikowi losowości pozwala na znajdywanie rozwiązań w różnych minimach lokalnych. Natomiast algorytm Johnsona, jako że służy jako do optymalizacji gotowej kombinacji, porządkuje znalezione przez algorytm genetyczne kombinacje przydzielonych do maszyn operacji.

Kooperacja algorytmów z uwagi na dzielące je różnice nie pozwala na pełne wykorzystanie ich potencjałów. Jak już wspomniano w rozdziale 4.4 ograniczenia modelu matematycznego nie pozwalają algorytmowi Johnsona na działanie zgodne z wszystkimi założeniami. Ponadto przetworzenie rozwiązania uzyskanego w GA przez JA powoduje utratę części rozwiązania, gdyż przerwy w pracy maszyn są planowane tak, aby zapewnić ciągłość produkcji.

Pierwszą heurystyką, która ma raczej charakter porównawczy, jest samodzielnie działający algorytm genetyczny. Jej działanie jest tożsame z opisem z rozdziałów 4.1 i 4.2. Ze względu na dużo większą elastyczność w porównaniu z innymi heurystykami, heurystyka ta może znajdować wiele rozwiązań dużo lepszych parametrach, ale niekoniecznie dopuszczalnych.

Druga heurystyka używa głównie algorytmu genetycznego, z wyjątkiem procesu tworzenia populacji początkowej. Po jej utworzeniu każde rozwiązanie jest przetwarzane przez algorytm Johnsona i dopiero w takiej postaci są poddawane ocenie. Dalsza praca algorytmu jest tożsama z pierwszą heurystyką, a JA nie jest już używany w kolejnych etapach.

Trzecia heurystyka opiera się na wspólnym działaniu algorytmów GA i JA. Każda nowa generacja jest „poprawiana” algorytmem Johnsona, a następnie jest poddawana ocenie oraz krzyżowaniu i mutacji. Wart uwagi jest sam proces krzyżowania, bowiem tylko w tej heurystyce wszystkie rozwiązania potomne pochodzą z rozwiązań przetwarzanych przez JA. Wobec tego algorytm genetyczny jest bardzo mocno ograniczony.

Czwarta heurystyka jest nieco bardziej skomplikowana. Każda nowa generacja jest przetwarzana przez algorytm Johnsona, a następnie oceniana. Jednakże do krzyżowania i mutacji używa się pierwotnych wyników uzyskanych z algorytmu genetycznego. W skrócie można opisać to działanie jako samotne poszukiwanie rozwiązania przez GA, a z pomocą algorytmu Johnsona rozwiązania są poprawiane do prawdopodobnie dopuszczalnej formy. Heurystyka ta ma na celu ograniczenie wpływu JA na GA i umożliwienie sprawniejszego przeszukiwania przestrzeni rozwiązań.

# 5. Opis aplikacji

## 5.1. Środowisko programistyczne

Zarówno model matematyczny jak i heurystyki zostały zaimplementowane w języku C#. Jest to język stworzony przez Andersa Hejlsberga, Scotta Wiltamutha i Petera Golde’a dla firmy Microsoft. Pierwsza szeroko dystrybuowana implementacja języka C# ukazała się w lipcu 2000 roku, jako część struktury .NET. Głównym założeniem przy tworzeniu było stworzenie języka programowania prostego, nowoczesnego i obiektowego, oszczędnego w użyciu mocy obliczeniowej, przystępnego dla użytkowników języków C i C++ oraz wspierającego zasady inżynierii oprogramowania. [5.1]

Do stworzenia programu realizującego symulowanie procesu produkcji oraz optymalizowanie jego harmonogramu za pomocą utworzonych heurystyk użyty został program Microsoft Visual Studio. Jest to zintegrowane środowisko programistyczne pozwalające na programowanie w wielu językach jak na przykład: C, C++, C#, Python itd.. Jest to podstawowe środowisko do obsługi języka C#, dlatego było ono pierwszym i prawdopodobnie najlepszym wyborem, jako że posiada ogromną gamę narzędzi ułatwiających procesy programowania, debugowania, zapisu aktualnych wersji i wielu innych. Ponadto jest to darmowy program jeśli jest używany w celach niekomercyjnych. [5.2]

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

Opis wygenerowany automatycznie

*Rysunek 5.1 Zrzut ekranu programu Microsoft Visual Studio*

## 5.2. Proces powstawania i testowania aplikacji

Proces powstawania kodu aplikacji, ze względu na dużą złożoność problemu składał się z kilku etapów. W pierwszym etapie stworzona została struktura umożliwiająca wprowadzenie do aplikacji wartości poszczególnych zmiennych. Może się to odbyć poprzez interakcję z oknem komend, poprzez wprowadzenie danych do odpowiednich tabel w formacie CSV lub poprzez losowy dobór oparty na określonych przedziałach.

W kolejnym etapie utworzone zostały klasy zawierające informacje o procesach, operacjach i maszynach oraz symulacji i funkcji kary. Pełnią one niezwykle ważną rolę w procesie testowania rozwiązania. Etap ten był niezwykle pracochłonny ze względu na możliwe pojawienie się błędów w wielu miejscach kodu, co zmuszało do kompleksowego sprawdzania jego poprawności.

W ostatnim etapie zakodowane zostały wszystkie opracowane heurystyki. Umożliwiony został wybór heurystyki w zależności od potrzeb i scenariuszy testowych. Ponadto zaimplementowano kod umożliwiający zapis wyników symulacji i działania algorytmów pozwalające na ocenę poprawności działania aplikacji oraz na opracowanie wyników. Najtrudniejszymi zadaniami okazały się być tworzenie nowych rozwiązań (losowych i na podstawie genotypu rodziców) oraz mutowanie. Głównym powodem tego stanu rzeczy była konieczność kontrolowania poprawności i kompletności genotypu nowych osobników, aby ani żadna operacja nie została pominięta, ani żadna maszyna nie otrzymała do wykonania operacji, której nie jest w stanie wykonać.

Po ukończeniu procesu kodowania wykonany został szereg testów działania zarówno symulacji jak i heurystyk. Pozwolił on na detekcję kilku błędów całkowicie zaburzających wyniki i działanie. Zostały one jednak szybko naprawione i wielokrotnie testowane pod kątem ich pojawiania się. W momencie, w którym wszystkie testy zostały przeprowadzone z pozytywnym skutkiem, proces tworzenia i testowania kodu został uznany za ukończony.

## 5.3 Struktura aplikacji

Aplikacja zawierająca model matematyczny oraz heurystyki składa się z kliku plików. Poza plikami generowanymi automatycznie przez program Microsoft Visual Studio, które pozwalają na zapis i otwarcie programu oraz użycie narzędzi, stworzonych zostało pięć plików zawierających kod.

Plik o nazwie *Classes.sc* zawiera struktury klas podstawowych parametrów symulacji, a więc procesów, operacji i maszyn, jak również klasy pozwalającej na uniwersalny zapis poszczególnych rozwiązań. Dzięki tym klasom możliwe jest przełożenie danego problemu na formę, którą aplikacja jest w stanie zaaplikować.

Plik *FromCSV.cs* zawiera specjalne klasy umożliwiające odczyt plików w formacie CSV za pomocą biblioteki CsvHelper. Klasy te przetwarzają tabele z plików CSV na tabele tworzone w języku C#. Dzięki odpowiedniej interpretacji takich tabel, możliwe jest zapisanie wprowadzanych danych w format obowiązujący w aplikacji.

Ważnym plikiem w kontekście informacji wyjściowych z działania aplikacji jest plik *SaveResults.cs*. Zawiera on klasę i metodę umożliwiającą zapis wyników symulacji, obowiązujących ustawień oraz kluczowych cech działania heurystyk w postaci plików tekstowych i plików w formacie CSV, które umożliwiają proste i sprawne weryfikowanie oraz analizowanie.

Rdzeniem symulacji jest plik *Model.cs*, gdzie oprócz klas odpowiedzialnych za symulację i obliczanie funkcji kary znajdują się fragmenty kodu umożliwiające wprowadzenie danych na trzy sposoby (patrz rozdział 5.2). Dzięki temu bardzo łatwo można sprawdzić działanie aplikacji za pomocą scenariuszy losowych oraz gotowych w postaci tabel lub w postaci danych wpisywanych w okno komend. To również tutaj wywoływane są komendy zapisujące wyniki.

Najważniejszym plikiem jeśli chodzi o optymalizację jest plik *Algorithm.cs*. To tutaj zawarte są wszystkie heurystyki, używany przez część z nich Algorytm Johnsona, metody krzyżujące i mutujące rozwiązania oraz fragmenty kodu oceniające i sortujące za pomocą funkcji kary z pliku *Model.cs* poszczególne rozwiązania, również zapisujące najlepsze z nich.

# 6. Wybrane scenariusze testowe

## 6.1. Scenariusz 1

# 7. Podsumowanie

# 8. Bibliografia

(numeracja bibliografii będzie poprawiona po ukończeniu części teoretycznej na format [1],[2] itd.)

**[4.1]** Inthachot, M., Boonjing, V., & Intakosum, S. (2016). Artificial neural network and genetic algorithm hybrid intelligence for predicting Thai stock price index trend. Computational Intelligence and Neuroscience, 2016, Article 3045254.

**[4.2]** Y. Xiong, S. Huang, M. Wu, J. She and K. Jiang, "A Johnson's-Rule-Based Genetic Algorithm for Two-Stage-Task Scheduling Problem in Data-Centers of Cloud Computing," in IEEE Transactions on Cloud Computing, vol. 7, no. 3, pp. 597-610, 1 July-Sept. 2019, doi: 10.1109/TCC.2017.2693187.

**[4.3]** R. Singh and S. K. Gupta, "Distributed process scheduling using genetic algorithm," Confluence 2013: The Next Generation Information Technology Summit (4th International Conference), Noida, 2013, pp. 48-54, doi: 10.1049/cp.2013.2292.

**[4.4]** Johnson, S. M. (1954). ["Optimal Two- and Three-Stage Production Schedules With Set-up Time Included"](http://www.rspq.org/pubs/j2.pdf) . Naval Research Logistics Quarterly. 1: 61–68.

**[4.5]** Okwu M., Emovon I. Application of Johnson’s algorithm in processing jobs through two-machine system. Journal of Mechanical and Energy Engineering, Vol. 4(44), No. 1, 2020, pp. 33-38.

**[5.1]** ECMA International Standardizing Information and Comunication Systems *C# Language Specification* Standard ECMA-334 2nd edition – December 2002.

**[5.2] (online)** https://visualstudio.microsoft.com/pl/vs/.