

**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

*Pragnę złożyć serdeczne podziękowania promotorowi dr. hab. inż. Wojciechowi Chmielowi prof. AGH za poświęcony czas i bezcenne rady przydatne w implementacji aplikacji oraz procesie pisania pracy i jej doskonalenia.*

# Spis treści

[1.Wstęp 5](#_Toc145579748)

[1.1. Wprowadzenie 5](#_Toc145579749)

[1.2. Cel i zakres pracy 5](#_Toc145579750)

[2. Rozważany problem i metody jego rozwiązania 6](#_Toc145579751)

[2.1. Produkcja i jej ograniczenia 6](#_Toc145579752)

[2.2. Znane metody optymalizacji produkcji 8](#_Toc145579753)

[3. Opis przyjętego modelu matematycznego 9](#_Toc145579754)

[3.1. Procesy produkcyjne 9](#_Toc145579755)

[3.2. Operacje 11](#_Toc145579756)

[3.3. Maszyny 14](#_Toc145579757)

[3.4. Symulacja 16](#_Toc145579758)

[3.5 Funkcja kary 16](#_Toc145579759)

[4. Stosowane algorytmy 18](#_Toc145579760)

[4.1. Algorytm genetyczny 18](#_Toc145579761)

[4.2. Schemat działania algorytmu genetycznego (na podstawie [4.3]) 20](#_Toc145579762)

[4.3. Algorytm Johnsona 21](#_Toc145579763)

[4.4. Wykorzystanie algorytmu Johnsona 22](#_Toc145579764)

[4.5. Utworzone heurystyki 23](#_Toc145579765)

[5. Opis aplikacji 25](#_Toc145579766)

[5.1. Środowisko programistyczne 25](#_Toc145579767)

[5.2. Proces powstawania i testowania aplikacji 26](#_Toc145579768)

[5.3 Struktura aplikacji 27](#_Toc145579769)

[5.4 Przykładowe wyniki działania aplikacji 28](#_Toc145579770)

[6. Wybrane scenariusze testowe 31](#_Toc145579771)

[6.1. Scenariusz 1 31](#_Toc145579772)

[6.2 Scenariusz 2 32](#_Toc145579773)

[6.3 Scenariusz 3 37](#_Toc145579774)

[6.4 Scenariusz 4 40](#_Toc145579775)

[6.5 Porównanie czasu działania heurystyk 54](#_Toc145579776)

[7. Podsumowanie 56](#_Toc145579777)

[8. Bibliografia 56](#_Toc145579778)

# 1.Wstęp

## 1.1. Wprowadzenie

Rozwój cywilizacji i napędzający ją postęp technologiczny stwarza każdemu kolejnemu pokoleniu szansę, na łatwiejsze życie. Postępująca industrializacja i mechanizacja, a w ostatnim czasie automatyzacja i informatyzacja 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 częstą praktyką jest obniżanie ceny produktów kosztem jakości towarów. Przykładowo kiedyś sprzęty AGD były droższe i produkowane w mniejszej ilości, ale jednocześnie były bardziej trwałe niż te produkowane w dzisiejszych czasach. Dziś producentom bardziej opłaca się sprzedawanie mniej trwałych urządzeń, które równocześnie trzeba częściej wymieniać na nowe tworząc niepotrzebny popyt. W czasach zwiększającej się publicznej świadomości ekologicznej należałoby powrócić do dobrych praktyk, zaś okazji do oszczędności i zysku poszukiwać w innych aspektach produkcji.

Proces produkcyjny zazwyczaj składa się przynajmniej z kilku operacji, które wykonuje się w określonym czasie, miejscu i etapie produkcji. Wobec tego odpowiednio dobrany harmonogram produkcji mógłby ograniczyć znacząco czas i koszty produkcji, straty energii oraz inne kluczowe parametry. Jego stworzenie nie jest jednak łatwe, szczególnie w przypadku dużej elastyczności parku maszynowego i dużej ilości wykonywanych procesów. Dziś przydatne w rozwiązaniu tego problemu mogą być znane algorytmy, a w niedalekiej przyszłości zapewne wciąż rozwijana sztuczna inteligencja.

W poszukiwaniu dzisiaj dostępnych rozwiązań, na potrzeby niniejszej pracy i stworzonego modelu matematycznego, opracowanych zostało kilka heurystyk, których zadaniem jest poszukiwanie optymalnych harmonogramów procesów produkcyjnych, umożliwiających podniesienie efektywności i jakości.

## 1.2. Cel i zakres pracy

Celem niniejszej pracy dyplomowej jest stworzenie aplikacji pozwalającej na symulowanie wybranego problemu dyskretnego z wieloma ograniczeniami, jakim jest harmonogramowanie procesu produkcyjnego, oraz opracowanie i implementacja heurystyk poszukujących optymalnego rozwiązania tego problemu.

Zakres pracy obejmuje:

* zbadanie i określenie problemów związanych z procesami produkcyjnymi,
* weryfikację modeli, metod i algorytmów używanych powszechnie do rozwiązywania tego typu problemów,
* formalizację modelu matematycznego reprezentującego najistotniejsze aspekty produkcji,
* opracowanie heurystyk poszukujących optymalnego rozwiązania zdefiniowanego problemu,
* implementację aplikacji realizującej model matematyczny oraz heurystyki w ramach symulacji,
* testowanie jakości działania poszczególnych heurystyk w określonych scenariuszach,
* sformułowanie wniosków dotyczących specyfiki niektórych możliwych scenariuszy produkcyjnych oraz działania heurystyk w ich ramach.

# 2. Rozważany problem i metody jego rozwiązania

## 2.1. Produkcja i jej ograniczenia

Jak podaje słownik online PWN *„Produkcja to zorganizowana działalność mająca na celu wytwarzanie jakichś towarów, usług lub dóbr kultury; też: to, co zostało wytworzone”* [1]. Populacja ludzka produkuje przeróżne towary od co najmniej miliona lat, czyli początku epoki kamienia. Zaczynając od prymitywnego uderzania kamienia o kamień, nasi przodkowie tworzyli pierwsze wyroby, „dzieła sztuki” oraz narzędzia umożliwiające łatwiejszą pracę. Następnie po wielu wiekach odkryty został brąz czyli stop miedzi i cyny, który pozwolił na powstanie i rozwój metalurgii, a finalnie doprowadził ludzkość do epoki żelaza. Czasy prehistoryczne charakteryzowały się niewielką ilością wykonywanych towarów oraz niską dokładnością ich wykonania, ze względu na prymitywność produkcji. Taki stan rzeczy trwał przez wiele dziesięcioleci, aż kolejne odkrycia pozwoliły na poprawę wydajności produkcji i jakości towarów. Poprzez średniowieczne manufaktury, aż do rewolucji przemysłowej udało się cywilizacji ludzkiej usprawnić ten proces do masowego wytwarzania wszelkich dóbr. [2]

Ludzkość nadal stara się udoskonalać wszystkie aspekty życia. Tak jest i w przypadku procesów produkcyjnych, które na przestrzeni wielu wieków ewoluowały. Dzięki nowym odkryciom możliwe jest podniesienie jakości i efektywności produkcji na wyższy poziom. Niestety wciąż istnieją pewne ograniczenia, których często nie da się rozwiązać w prosty sposób. Mogą je stwarzać niewystarczające: zasoby materiałowe, zasoby produkcyjne, zasoby energetyczne, zasoby ludzkie, zasoby czasowe i zasoby pieniężne. Niezaspokojenie tych potrzeb może uniemożliwiać prawidłowe wykonanie procesu produkcyjnego. Dlatego skuteczna produkcja może odbywać się tylko w odpowiednich warunkach.

Poza ogólnymi warunkami umożliwiającymi produkcję istnieją również ograniczenia dotyczące samego przebiegu produkcji. Są to m.in. ograniczenia czasowe i fizyczne wynikające z typu surowców i materiałów, z którego wykonuje się dany produkt, terminy dostaw towarów do klienta oraz priorytetyzacja ważniejszych procesów. Ich istnienie praktycznie uniemożliwia wykonywanie produkcji w sposób dowolny przy dużej jej skali i niewielkiej elastyczności. Niespełnienie jednego z ograniczeń może powodować straty zasobów. Dlatego przy tworzeniu harmonogramów określających przebieg produkcji w danej jednostce konieczne jest zwracanie uwagi na istotne ograniczenia. Na potrzeby niniejszej pracy wyodrębnione zostały te, które są najbardziej powiązane z samym procesem produkcji (rozdział 3).

## 2.2. Problemy planowania produkcji w literaturze

Planowanie produkcji jest rozpatrywane w literaturze zazwyczaj w oparciu o ogólnie zdefiniowane problemy. Jednym z nich jest *Job Shop Scheduling Problem* (JSSP). Zgodnie z [3] w ramach tego problemu istnieje zbiór maszyn oraz zbiór procesów , które są tworzone przez elementy ze zbioru operacji . Operacje mogą być wykonywane tylko na określonych maszynach w określonym czasie i kolejności zgodnej z założeniami procesu oraz nie mogą być przerywane. Poszukiwanym rozwiązaniem problemu jest odnalezienie takiego harmonogramu wykonywania operacji w ramach procesu, aby przy zachowaniu wszystkich założeń, minimalizować czas od rozpoczęcia pierwszej operacji do zakończenia ostatniej.

Nieco innym modelem problemu jest *Open Shop Scheduling Problem* (OSSP). W odróżnieniu od JSSP, kolejność wykonywania operacji nie jest istotna, jednakże operacje w ramach tego samego procesu nie mogą być wykonywane w tym samym czasie. Dodatkowo dana operacja nie musi zostać wykonana przez konkretną maszynę, ale przez jedną z maszyn, która jest do tego zdolna. Podobnie jak w przypadku JSSP oczekiwanym rozwiązaniem problemu jest znalezienie harmonogramu minimalizującego czas pełnej produkcji. [4]

W literaturze pojawia się również *Flow Shop Scheduling Problem* (FSSP). W odróżnieniu od powyższych problemów, w tym każdy proces ze zbioru składa się z operacji ze zbioru , a każda z nich musi zostać wykonana na innej maszynie ze zbioru , który liczy maszyn. Operacje z każdego procesu muszą być wykonywane w odpowiedniej kolejności. Nie jest jednak istotna kolejność wykonywania procesów. Tak samo jak w przypadku pozostałych problemów, poszukiwane jest rozwiązanie w postaci harmonogramu o najkrótszym czasie wykonania. [5]

Przytoczone powyżej modele problemów są najczęściej analizowanymi problemami dotyczącymi produkcji. Niestety ich konstrukcja, choć jest niezwykle prosta do wdrożenia, nie pozwala na przeniesienie każdego rzeczywistego problemu w postać odpowiadającą któremuś z modeli. Dlatego model matematyczny użyty w niniejszej pracy (rozdział 3) nie można bezpośrednio zakwalifikować do żadnego z wyżej przytoczonych problemów.

## 2.3. Znane metody optymalizacji produkcji

W zależności od opracowywanego modelu problemu do jego rozwiązania używa się różnych metod i algorytmów. W źródle [3] przedmiotem badań jest JSSP. Optymalizacja w tym problemie była dokonywana za pomocą algorytmu przybliżonego Fast Taboo Search. Algorytmy przybliżone charakteryzują się przeszukiwaniem przestrzeni w poszukiwaniu najlepszych rozwiązań zgodnie z przyjętą metodyką. Niestety ich użycie nie gwarantuje znalezienia optimum. Jako, że w przyjętym problemie mogły pojawić się rozwiązania niedopuszczalne, użycie algorytmu dokładnego, czyli algorytmu dającemu tylko jedno, optymalne rozwiązanie, mogłoby zakończyć się fiaskiem. Wobec tego bardziej sensowne staje się użycie właśnie algorytmów przybliżonych. W [6], gdzie rozpatrywano FSSP oraz w [7], gdzie rozważano niepasujący do wymienionych w rozdziale 2.2 problem, używano innego algorytmu przybliżonego – algorytmu genetycznego.

W mniej skomplikowanych JSSP, w które nie są zbyt skomplikowane ze względu na niewielką liczbę maszyn (dwie lub trzy) stosuje się algorytmy dokładne. W wielu z nich (na przykład [8] i [9]) używa się algorytmu Johnsona. W przeciwieństwie do algorytmów przybliżonych algorytm ten szereguje operacje w ramach produkcji zgodnie z kryteriami, aby zwrócić optymalną kolejność ich wykonywania. Wobec tego, ten sam problem, o tych samych parametrach, po użyciu algorytmu dokładnego da to samo rozwiązanie. Przy spełnieniu wszystkich założeń, zgodnie z dowodem autora algorytmu, jest to rozwiązanie optymalne.

Podsumowując, w zależności od skomplikowania przyjętego problemu należy używać odpowiedniego typu algorytmu (dokładnego lub przybliżonego). W rzeczywistej produkcji praktycznie zawsze występują pewne ograniczenia wynikające z wielu czynników (rozdział 2.1). Dlatego w modelach lepiej oddających rzeczywiste warunki konieczne jest użycie algorytmów przybliżonych, które niekoniecznie mogą znaleźć rozwiązanie optymalne, a często nawet dopuszczalne.

# 3. Opis przyjętego modelu matematycznego

## 3.1. Procesy produkcyjne

## 

W celu wytworzenia danego produktu 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 lub typie. W zależności od poziomu skomplikowania procesu 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, jedna po drugiej (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, odbiorcę 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. Wprowadzone zostało ograniczenie czasowe symulacji wynoszące jedną dobę, wynikające z założenia, iż zlecenia na produkcję są wprowadzane do systemu raz w ciągu dnia. Wobec tego zachowanie priorytetu nie należy do najważniejszych celów do osiągnięcia przez rozwiązanie końcowego. Mimo to zostało zachowane w modelu, gdyż może być parametrem zbliżającym rozwiązanie do optimum.

*Tabela 3.1 Przykładowe parametry klasy procesu.*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nazwa parametru | Identyfikator | Przykładowa wartość | Format | Uwagi |
| numer identyfikacyjny |  | 8 | liczba naturalna | - |
| priorytet |  | 4 | liczba naturalna | , przy czym większa wartość oznacza większy priorytet |
| zbiór operacji |  | (5,2,6,8) | lista | , 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 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 być to 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 czasu wykonywania operacji na wymagających tego maszynach. Jeśli zaś chodzi o maksymalny czas, w którym kolejna operacja ma zostać wykonana, to wartość tę 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 określonego surowca, produktu lub półproduktu 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 jednym z dwóch scenariuszy. 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 jeśli jednak weźmie się pod uwagę nierównomierne nagrzewanie się cieczy, to oddzielenie cieczy o mniejszej temperaturze od tej o odpowiedniej temperaturze w większości przypadków jest niemożliwe. Zupełnie inaczej jest w drugim scenariuszu, gdzie 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 opuszczają taśmę co określony czas. Stworzony model matematyczny pozwala zarówno na pobieranie wszystkich surowców przed rozpoczęciem operacji i dystrybucję gotowych produktów po jej ukończeniu, jak i na nieregularne pobieranie i dystrybucję już podczas trwania operacji.

Dodatkowo do modelu matematycznego zaimplementowana została możliwość zatrzymania wykonywania operacji. Niestety jej używanie jest niepraktyczne z powodu wykorzystywanych algorytmów, bowiem zaimplementowany algorytm Johnsona opiera się 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. Dlatego dodanie możliwości pauzowania całkowicie traci sens, bowiem JA z nich nie korzysta, a GA mógłby tworzyć harmonogramy, które ze względu na dodatkowy czas potrzebny na zmianę czasu ustawień, wynikający z większej liczby zmian wykonywanych operacji, nie byłyby wystarczająco dobre do tworzenia kolejnych pokoleń rozwiązań.

*Tabela 3.2 Przykładowe parametry klasy operacji.*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nazwa parametru | Identy-fikator | Przykładowa wartość | Format | Uwagi |
| numer identyfikacyjny |  | 3 | liczba naturalna | - |
| maksymalny czas rozpoczęcia po poprzedniej operacji [s] |  | 24.0 | liczba wymierna | - |
| maksymalny czas od zakończenia operacji do rozpoczęcia kolejnej [s] |  | 9.2 | liczba wymierna | W przypadku podania parametru  w kolejnej operacji model ignoruje ten parametr. |
| macierz potrzebnych surowców |  |  | macierz / tabela | Każda kolumna oznacza inny surowiec. Ostatnia kolumna oznacza % czasu, w którym następuje pobranie surowców w liczbie podanej w rzędzie. |
| macierz produktów |  |  | macierz / tabela | Każda kolumna oznacza inny produkt. Ostatnia kolumna oznacza % czasu, w którym możliwe jest odebranie produktów w podanej liczbie. |
| możliwość pauzowania\* |  | prawda (true) | wartość binarna | - |
| maksymalny czas pauzy [s]\* |  | 100 | liczba wymierna | - |
| maksymalna liczba pauz\* |  | 5 | liczba naturalna | 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, 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. Mogą się jednak zdarzać scenariusze, w których jedna z maszyn może wykonać każdą z operacji szybciej niż pozostałe maszyny. Wtedy intuicyjnym działaniem byłoby przekazanie wszystkich operacji do realizacji na tej maszynie, co w większości przypadków prowadziłoby do zatoru. Dlatego pewniejsze jest użycie algorytmów, które same będą poszukiwać optimum.

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. Kalibracje maszyn uznawane są za operacje na maszynie, lecz ze względu na wyjątkowy charakter nie są one wykorzystywane przez heurystyki, lecz są wykonywane automatycznie.

*Tabela 3.3 Przykładowe parametry klasy maszyny.*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nazwa parametru | Identyfikator | Przykładowa wartość | Format | Uwagi |
| numer identyfikacyjny |  | 0 | liczba naturalna | - |
| macierz czasów przekalibrowania [s] |  |  | macierz / tabela | 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) | lista | Czas wynoszący 0 oznacza brak możliwości wykonania danej operacji. Algorytmy sprawdzają tę wartość przed przydzieleniem operacji. |

## 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.

*Tabela 3.4 Wygląd struktury*

|  |  |  |
| --- | --- | --- |
| Parametr | Identyfikator | Przykładowa wartość |
| Numer maszyny |  | 2 |
| Numer procesu |  | 3 |
| Numer operacji |  | 7 |
| Globalny czas [s] |  | 13.5 |
| Stan (początek operacji - 1 / koniec operacji - 9 ) |  | 9 |

Wynikiem działania symulacji są trzy listy zawierające: czasy rozpoczęcia i zakończenia poszczególnych symulacji wraz ze szczegółowymi informacjami dotyczącymi procesu i maszyny (tabela 3.4), surowców na potrzeby operacji pod postacią czasu zaistnienia potrzeby wraz z id i ilością surowca (tabela 3.5) oraz produkty operacji w takiej samej postaci jak lista potrzebnych surowców (tabela 3.5).

*Tabela 3.5 Struktura opisująca potrzeby/produkty operacji w czasie*

|  |  |  |
| --- | --- | --- |
| Parametr | Identyfikator | Przykładowa wartość |
| Czas [s] |  | 11.5 |
| Id towaru |  | 3 |
| Ilość [szt/kg] |  | 10 |

## 3.5 Funkcja kary

Wyniki symulacji w postaci trzech list opisanych w rozdziale 3.4 są wystarczające do oceny rozwiązania. Analiza list pod kątem poprawności wraz z odnotowaniem w odpowiedni sposób 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 , czyli wykonania operacji zanim zakończyła się poprzednia, według ustalonej kolejności. 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,

– proces, w którym 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 maksymalnym czasem trwania przerwy pomiędzy operacjami, a osiągniętym czasem ,

– ilość surowców niedostarczonych na czas do wykonania operacji,

– lista zawierająca aktualne dostępne zasoby surowców, aktualizowana z biegiem symulacji.

Wzór 3.2 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. [10]

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 kryterium oceny rozwiązania. 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. [6]

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 [7]. 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 iterowaniu 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. Liczba mutacji w perspektywie całej populacji jest bardzo mała, bowiem ich duża liczba 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 [7])

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 również globalnie najlepszy,

Krok 4: Wybór zadanego odsetka 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. [8]

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. [9]

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. [6]

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. [9]

## 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. 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 prawidłowej kolejności wykonywania procesów, gdyż w przeciwieństwie do algorytmu genetycznego, opiera harmonogramowanie na kolejności wykonywania operacji w ramach procesu, a nie na kolejności 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. [11]

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. [12]

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.cs* 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.

## 5.4 Przykładowe wyniki działania aplikacji

Dane dotyczące modelu:

Wyniki testu o kryptonimie: b49

Maszyny:

Maszyna id=0

Czas trwania kolejnych operacji: 3 5 5 5 5

Czas trwania dostosowania maszyny z jednej operacji na drugą:

0 1 1 1 1

1 0 1 1 1

1 1 0 1 1

1 1 1 0 1

1 1 1 1 0

Maszyna id=1

Czas trwania kolejnych operacji: 5 3 5 5 5

Czas trwania dostosowania maszyny z jednej operacji na drugą:

0 1 1 1 1

1 0 1 1 1

1 1 0 1 1

1 1 1 0 1

1 1 1 1 0

Maszyna id=2

Czas trwania kolejnych operacji: 5 5 3 5 5

Czas trwania dostosowania maszyny z jednej operacji na drugą:

0 1 1 1 1

1 0 1 1 1

1 1 0 1 1

1 1 1 0 1

1 1 1 1 0

Maszyna id=3

Czas trwania kolejnych operacji: 5 5 5 3 5

Czas trwania dostosowania maszyny z jednej operacji na drugą:

0 1 1 1 1

1 0 1 1 1

1 1 0 1 1

1 1 1 0 1

1 1 1 1 0

Maszyna id=4

Czas trwania kolejnych operacji: 5 5 5 5 3

Czas trwania dostosowania maszyny z jednej operacji na drugą:

0 1 1 1 1

1 0 1 1 1

1 1 0 1 1

1 1 1 0 1

1 1 1 1 0

Procesy:

Proces o id= 0

Priorytet: 1

Operacje wchodzące w skład procesu:

Operacja o id=4

Czas po którym musi się rozpocząć po zakończeniu poprzedniej: 100

Czas po którym musi się rozpocząć kolejna operacja: 100

Czy można pauzować?: False

Potrzebne materiały (ostatnia kolumna oznacza % postępu, w którym są potrzebne):

0 0 0 0 0 6 0

Produkowane materiały (ostatnia kolumna oznacza % postępu, w którym są produkowane):

0 0 0 0 5 0 0

Operacja o id=0

Czas po którym musi się rozpocząć po zakończeniu poprzedniej: 100

Czas po którym musi się rozpocząć kolejna operacja: 100

Czy można pauzować?: False

Potrzebne materiały (ostatnia kolumna oznacza % postępu, w którym są potrzebne):

0 2 0 0 0 0 0

Produkowane materiały (ostatnia kolumna oznacza % postępu, w którym są produkowane):

1 0 0 0 0 0 0

Operacja o id=1

Czas po którym musi się rozpocząć po zakończeniu poprzedniej: 100

Czas po którym musi się rozpocząć kolejna operacja: 100

Czy można pauzować?: False

Potrzebne materiały (ostatnia kolumna oznacza % postępu, w którym są potrzebne):

0 0 3 0 0 0 0

Produkowane materiały (ostatnia kolumna oznacza % postępu, w którym są produkowane):

0 2 0 0 0 0 0

Operacja o id=3

Czas po którym musi się rozpocząć po zakończeniu poprzedniej: 100

Czas po którym musi się rozpocząć kolejna operacja: 100

Czy można pauzować?: False

Potrzebne materiały (ostatnia kolumna oznacza % postępu, w którym są potrzebne):

0 0 0 0 5 0 0

Produkowane materiały (ostatnia kolumna oznacza % postępu, w którym są produkowane):

0 0 0 4 0 0 0

Operacja o id=2

Czas po którym musi się rozpocząć po zakończeniu poprzedniej: 100

Czas po którym musi się rozpocząć kolejna operacja: 100

Czy można pauzować?: False

Potrzebne materiały (ostatnia kolumna oznacza % postępu, w którym są potrzebne):

0 0 0 4 0 0 0

Produkowane materiały (ostatnia kolumna oznacza % postępu, w którym są produkowane):

0 0 3 0 0 0 0

Liczba produktów na początku procesu produkcyjnego: 100 100 100 100 100 100

Ustawienia algorytmów:

Populacja początkowa: 10

Maksymalna populacja: 100

Procent najlepszych osobników do reprodukcji: 10

Procent mutowanych osobników: 2

Populacja początkowa: 1

Wagi: a:1000,c:1,d:1,e:1,f:1,g:1,t:1

Dane dotyczące najlepszego wyniku:

Wyniki rozwiązania testu o kryptonimie: b49

Liczba zaburzeń procesów: 0

Ilość przypadków zbyt wielu pauz: 0

Czas przekraczający maksymalny czas pauz: 0

Całkowity czas produkcji: 15

Różnica czasu ukończenia procesów z wyższym priorytetem i niższym (nieuwzględnienie priorytetów): 0

Czas wykraczający poza maksymalną długość okna czasowego: 0

Ilość nieotrzymanych materiałów: 0

Struktura najlepszej kombinacji:

Proces 0

START O:4 M:4, Czas:0,

STOP O:4 M:4, Czas:3,

START O:0 M:0, Czas:3,

STOP O:0 M:0, Czas:6,

START O:1 M:1, Czas:6,

STOP O:1 M:1, Czas:9,

START O:3 M:3, Czas:9,

STOP O:3 M:3, Czas:12,

START O:2 M:2, Czas:12,

STOP O:2 M:2, Czas:15,

# 6. Wybrane scenariusze testowe

## 6.1. Scenariusz 1

Na początek przetestowano działanie heurystyk dla bardzo prostego problemu. Według scenariusza jedna maszyna musi wykonać jeden proces składający się z pięciu operacji, o tym samym czasie trwania (5s), zaś kalibracja maszyny do wykonywania kolejnej operacji trwa jedną sekundę. Nie przewiduje się żadnych przerw ani konieczności wykonania operacji w określonym czasie po zakończeniu poprzedniej. Ilość produkowanych dóbr i surowców na początku symulacji są w zupełności wystarczające, aby wykonanie procesu przebiegło bez zakłóceń.

Jeśli zaś chodzi o wagi składników funkcji kary, to wszystkie zostały ustawione na wartość 1 poza wagą kryterium sprawdzającego prawidłową kolejność wykonywania operacji w ramach procesu, która otrzymała wartość 1000. Taka kombinacja parametrów wejściowych powinna umożliwić algorytmom poszukiwanie rozwiązań tylko na podstawie czasu realizacji procesu.

*Tabela 6.1 Liczba iteracji potrzebnych do znalezienia najlepszego rozwiązania w poszczególnych testach*

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Typ heurystyki | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | Najlepszy wynik funkcji kary |
| Heurystyka I GA | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 29 |
| Heurystyka II GA + JA na początku | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 29 |
| Heurystyka III GA + JA | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 29 |
| Heurystyka IV GA + JA bez wpływu na obszar poszukiwań | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 29 |

Jak się okazało, tak łatwy problem był rozwiązywany przez wszystkie heurystyki już w pierwszej iteracji. Prawdopodobnie działo się tak przez implementację tworzenia pierwszych osobników w GA, bowiem kolejność przypisywania operacji do maszyn jest tożsama z kolejnością ich wykonywania w ramach procesu. Co warto zauważyć, wszystkie heurystyki poradziły sobie równie dobrze i odnalazły takie samo najlepsze rozwiązanie.

## 6.2 Scenariusz 2

Wobec prawidłowego działania heurystyk w przypadku prostego scenariusza, w kolejnym scenariuszu zwiększono liczbę maszyn do 5; przy czym każda maszyna może wykonać każdą operację. Każda maszyna posiada jedną operację, która jest wykonywana w znacznie krótszym czasie niż pozostałe (3s). Dzięki temu każda z operacji może zostać wykonana w znacznie krótszym czasie pod warunkiem, że zostanie zlecona odpowiedniej maszynie. Wszystkie pozostałe parametry są identyczne jak w scenariuszu 1.

*Tabela 6.2 Wyniki testów Heurystyki 1*

|  |  |  |
| --- | --- | --- |
| L. P. testu | Uzyskany najlepszy wynik | Liczba iteracji potrzebnych do uzyskania najlepszego wyniku |
| 1 | 1017 | 1 |
| 2 | 29 | 20 |
| 3 | 1005 | 3 |
| 4 | 1010.1710503116874 | 37 |
| 5 | 1005 | 2 |
| 6 | 27 | 18 |
| 7 | 1003 | 2 |
| 8 | 2005 | 1 |
| 9 | 1005 | 9 |
| 10 | 1011 | 2 |

Heurystyka używająca jedynie GA nie poradziła sobie najlepiej z tym zadaniem. Na 10 symulacji tylko dwie znalazły rozwiązania dopuszczalne. Po dodatkowych próbach na większej populacji z częstszym mutowaniem najlepszy wynik nadal wynosił 27, ale pojawiał się znacznie częściej. Mimo to zaimplementowany algorytm genetyczny nie był w stanie znaleźć idealnego rozwiązania.

*Tabela 6.3 Wyniki testów Heurystyki 2*

|  |  |  |
| --- | --- | --- |
| L. P. testu | Uzyskany najlepszy wynik | Liczba iteracji potrzebnych do uzyskania najlepszego wyniku |
| 1 | 24 | 1 |
| 2 | 23 | 1 |
| 3 | 26 | 1 |
| 4 | 23 | 1 |
| 5 | 22 | 1 |
| 6 | 22 | 1 |
| 7 | 22 | 1 |
| 8 | 23 | 1 |
| 9 | 22 | 1 |
| 10 | 23 | 1 |

Jak można zauważyć, w przeciwieństwie do pierwszej heurystyki, heurystyka używająca tylko w początkowej fazie algorytmu Johnsona, bardzo szybko znajdywała dobre i dopuszczalne rozwiązania. Niestety w żadnym z testów nie uzyskano idealnego wyniku wynoszącego 15, choć w porównaniu z pierwszymi testami tego scenariusza są znacznie bardziej zadowalające.

Dodatkowe testy na znacznie większej populacji pozwoliły na uzyskanie idealnej wartości 15, lecz nadal większość testów osiągała rozwiązanie o wyniku 18. Należy przy tym podkreślić, że wszystkie najlepsze wyniki były osiągane w pierwszych iteracjach, co wskazuje na potrzebę korzystania z większej populacji, a niekoniecznie większej liczby iteracji.

*Tabela 6.4 Wyniki testów Heurystyki 3*

|  |  |  |
| --- | --- | --- |
| L. P. testu | Uzyskany najlepszy wynik | Liczba iteracji potrzebnych do uzyskania najlepszego wyniku |
| 1 | 18 | 60 |
| 2 | 18 | 54 |
| 3 | 19 | 44 |
| 4 | 20 | 2 |
| 5 | 18 | 42 |
| 6 | 15 | 67 |
| 7 | 22 | 34 |
| 8 | 20 | 91 |
| 9 | 18 | 2 |
| 10 | 18 | 4 |

Trzecia heurystyka bardzo dobrze poradziła sobie w stworzonych warunkach. Nie tylko osiągnęła bardzo dobre rozwiązania, ale w jednym z testów znalazła optymalne rozwiązanie. Ponadto należy podkreślić, że najlepsze wyniki znajdywane były w całym zakresie iteracji, (niektóre na początku, inne bliżej 50 iteracji, a jedna w 91 iteracji) co wskazuje na zdolność algorytmu do odszukiwania lepszych rozwiązań nie tylko w pierwszych iteracjach.

Dodatkowe testy, które charakteryzowały się zwiększoną liczbą iteracji pokazały ogromny potencjał tej heurystyki. W 8 na 10 testach rozwiązanie optymalne było odszukiwane, co więcej zawsze w maksymalnie 300 iteracji. W pozostałych przypadkach algorytm osiągał rozwiązanie o wartości 19, które prawdopodobnie w jakiś sposób nie pozwalało na odnalezienie wyjścia z minimum lokalnego, co mogło się zdarzyć przy tak małej populacji rozwiązań.

*Tabela 6.5 Wyniki testów Heurystyki 4*

|  |  |  |
| --- | --- | --- |
| L. P. testu | Uzyskany najlepszy wynik | Liczba iteracji potrzebnych do uzyskania najlepszego wyniku |
| 1 | 21 | 2 |
| 2 | 15 | 49 |
| 3 | 21 | 67 |
| 4 | 15 | 13 |
| 5 | 20 | 70 |
| 6 | 19 | 3 |
| 7 | 21 | 1 |
| 8 | 15 | 83 |
| 9 | 15 | 53 |
| 10 | 15 | 3 |

Ostatnia heurystyka wykazała się największą skutecznością w znajdywaniu optimum. Dokładnie połowa testów zakończyła się wskazaniem optymalnego rozwiązania. To wskazuje na dużą skuteczność tej heurystyki w niesprzyjających warunkach małej liczności populacji rozwiązań i liczby iteracji.

## 6.3 Scenariusz 3

W celu sprawdzenia działania poszczególnych heurystyk w modelu, gdzie istnieje więcej niż jeden proces, na bazie poprzednich scenariuszy został utworzony nowy scenariusz zawierający 3 procesy produkcyjne. Każde z zadań składa się z dwóch operacji, które są wykonywane na dwóch takich samych maszynach. Czasy wykonywania poszczególnych operacji ponownie zostały wyrównane.

*Tabela 6.6 Wyniki testów Heurystyki 1*

|  |  |  |
| --- | --- | --- |
| L. P. testu | Uzyskany najlepszy wynik | Liczba iteracji potrzebnych do uzyskania najlepszego wyniku |
| 1 | 17 | 1 |
| 2 | 17 | 1 |
| 3 | 17 | 1 |
| 4 | 17 | 9 |
| 5 | 28 | 7 |
| 6 | 23 | 2 |
| 7 | 18.668891757449334 | 64 |
| 8 | 22 | 49 |
| 9 | 17 | 1 |
| 10 | 22 | 5 |

Choć w połowie przypadków pierwsza heurystyka osiągnęła optymalne rozwiązanie, to w pozostałych znalazła nieco gorsze. Jednakże po zwiększeniu populacji do 100 osobników optymalne rozwiązanie było odszukiwane za każdym razem w pierwszej iteracji. Ponownie ustawienia parametrów algorytmu genetycznego okazały się kluczowe w skuteczności tej heurystyki.

*Tabela 6.7 Wyniki testów Heurystyki 2*

|  |  |  |
| --- | --- | --- |
| L. P. testu | Uzyskany najlepszy wynik | Liczba iteracji potrzebnych do uzyskania najlepszego wyniku |
| 1 | 27 | 1 |
| 2 | 17 | 9 |
| 3 | 22 | 1 |
| 4 | 17 | 8 |
| 5 | 22 | 1 |
| 6 | 22 | 1 |
| 7 | 17 | 2 |
| 8 | 22 | 3 |
| 9 | 17 | 92 |
| 10 | 22 | 1 |

Druga heurystyka poradziła sobie porównywalnie dobrze jak pierwsza. Skuteczność i prędkość dotarcia do najlepszego znalezionego rozwiązania są bardzo podobne. Dodatkowe testy wykazały, że niemal pewną skuteczność heurystyka ta osiągała, gdy populacja rozwiązań wynosiła 1000 osobników. Jest to dużo większa wartość, niż w przypadku pierwszej heurystyki, gdzie do pewności w znalezieniu optymalnego rozwiązania problemu wystarczyło 100 osobników.

*Tabela 6.8 Wyniki testów Heurystyki 3*

|  |  |  |
| --- | --- | --- |
| L. P. testu | Uzyskany najlepszy wynik | Liczba iteracji potrzebnych do uzyskania najlepszego wyniku |
| 1 | 22 | 1 |
| 2 | 22 | 1 |
| 3 | 22 | 1 |
| 4 | 22 | 1 |
| 5 | 22 | 1 |
| 6 | 22 | 3 |
| 7 | 22 | 1 |
| 8 | 22 | 1 |
| 9 | 22 | 1 |
| 10 | 22 | 1 |

Niezwykle ciekawe okazały się wyniki trzeciej heurystyki. Poza jednym wyjątkiem we wszystkich testach najlepsze znalezione rozwiązanie o wartości 22 zostało znalezione w populacji początkowej. Wobec tego nie można uznać tej heurystyki za pomocną w poszukiwaniu optymalnego rozwiązania dla tego problemu. Nawet znaczne zwiększenie liczby osobników i iteracji nie zmieniło uzyskiwanych wyników.

*Tabela 6.9 Wyniki testów Heurystyki 4*

|  |  |  |
| --- | --- | --- |
| L. P. testu | Uzyskany najlepszy wynik | Liczba iteracji potrzebnych do uzyskania najlepszego wyniku |
| 1 | 22 | 12 |
| 2 | 22 | 1 |
| 3 | 22 | 1 |
| 4 | 22 | 21 |
| 5 | 22 | 1 |
| 6 | 22 | 12 |
| 7 | 22 | 1 |
| 8 | 22 | 1 |
| 9 | 22 | 1 |
| 10 | 22 | 1 |

Podobnie jak trzecia heurystyka, również czwarta nie potrafiła znaleźć lepszego rozwiązania niż to o wartości 22. Co gorsza, i w tym przypadku zwiększenie wspominanych wcześniej parametrów nie polepszyło uzyskiwanych wyników.

Podsumowując ten scenariusz należy zwrócić uwagę na kłopoty algorytmu Johnsona w poszukiwaniu najlepszego rozwiązania. Prawdopodobnie jego zaimplementowany sposób działania uniemożliwia dotarcie do optymalnego rozwiązania.. Okazuje się więc, że obawy co do tego czy JA w tej formie zawsze będzie wskazywał optimum były słuszne.

## 6.4 Scenariusz 4

Po wykonaniu testów na scenariuszach, których optymalne rozwiązanie było doskonale widoczne, w kolejnym scenariuszu postanowiono sprawdzić działanie heurystyk dla dużo bardziej skomplikowanego problemu. Ze względu na jego złożoność większość parametrów została dobrana losowo. W poniższej tabeli zapisane zostały wszystkie najważniejsze parametry tego scenariusza.

*Tabela 6.10 Najważniejsze parametry zadania Scenariusza 4*

|  |  |
| --- | --- |
| Parametr | Wartość |
| Liczba maszyn | 10 |
| Liczba procesów | 5 |
| Liczba operacji | 20 |

*Tabela 6.11 Najważniejsze parametry instancji Scenariusza 4*

|  |  |
| --- | --- |
| Parametr a funkcji kary | 1000 |
| Parametr b funkcji kary | 1 |
| Parametr c funkcji kary | 0.5 |
| Parametr d funkcji kary | 0.5 |
| Parametr e funkcji kary | 1 |
| Parametr f funkcji kary | 1000 |
| Parametr g funkcji kary | 1000 |
| Liczba iteracji algorytmu | 1000 |
| Początkowa liczność populacji rozwiązań | 10000 |
| Liczność populacji w kolejnych generacjach | 10000 |
| Procent osobników wybieranych do stworzenia następnego pokolenia | 2 |
| Procent rozwiązań w pokoleniu poddanych mutacji | 1 |
| Średni czas przerwania działania przez mutację [s] | 5 |

*Tabela 6.12 Wyniki testów Heurystyki 1*

|  |  |  |  |
| --- | --- | --- | --- |
| L. P. testu | Uzyskany najlepszy wynik | Liczba iteracji potrzebnych do uzyskania najlepszego wyniku | Najlepszy wynik w pierwszej iteracji |
| 1 | 397.75337313 | 81 | 5329.79844675 |
| 2 | 471.912564094 | 96 | 5314.702354668 |
| 3 | 438.30540673987304 | 63 | 4620.920542957 |
| 4 | 478.60471497699996 | 94 | 4500.228323396 |
| 5 | 306.27988924000005 | 157 | 5354.935765108 |
| 6 | 485.561978706 | 414 | 5190.69922775 |
| 7 | 357.05165467899997 | 932 | 5232.297363623999 |
| 8 | 490.018965442 | 536 | 5391.544476909 |
| 9 | 402.28881597399993 | 996 | 5375.976940273 |
| 10 | 400.49804540299994 | 358 | 5435.342012327 |

*Wykres 6.1 Wartość najlepszego znalezionego globalnie rozwiązania w poszczególnych etapach testów Heurystyki 1*

Najlepsze znalezione rozwiązanie (test 5):

(O – numer operacji, M – numer maszyny)

Proces 0

START O:19 M:0, Czas:0,

STOP O:19 M:0, Czas:25,29969438,

START O:0 M:1, Czas:28,590642834,

STOP O:0 M:1, Czas:106,02549560400001,

START O:18 M:1, Czas:114,55969175500002,

STOP O:18 M:1, Czas:166,622453725,

START O:1 M:1, Czas:175,253594173,

STOP O:1 M:1, Czas:179,5243422,

Proces 1

START O:17 M:1, Czas:0,

STOP O:17 M:1, Czas:26,77684159,

START O:2 M:0, Czas:30,534041447999996,

STOP O:2 M:0, Czas:93,949688548,

START O:16 M:4, Czas:112,112835758,

STOP O:16 M:4, Czas:122,839786378,

START O:3 M:0, Czas:189,70265691900002,

STOP O:3 M:0, Czas:279,905593269,

Proces 2

START O:15 M:4, Czas:0,

STOP O:15 M:4, Czas:73,35891451,

START O:4 M:4, Czas:77,52675843700001,

STOP O:4 M:4, Czas:109,359053947,

START O:14 M:4, Czas:126,772789426,

STOP O:14 M:4, Czas:209,60364736600002,

START O:5 M:4, Czas:218,20388333600002,

STOP O:5 M:4, Czas:219,65065662,

Proces 3

START O:13 M:6, Czas:0,

STOP O:13 M:6, Czas:9,17063634,

START O:6 M:0, Czas:103,399003511,

STOP O:6 M:0, Czas:128,19354543100002,

START O:12 M:0, Czas:133,58554759300003,

STOP O:12 M:0, Czas:181,58415883300003,

START O:7 M:0, Czas:285,353138509,

STOP O:7 M:0, Czas:293,203415961,

Proces 4

START O:11 M:2, Czas:0,

STOP O:11 M:2, Czas:91,8583142,

START O:8 M:1, Czas:187,891078265,

STOP O:8 M:1, Czas:251,822448385,

START O:10 M:1, Czas:252,155532559,

STOP O:10 M:1, Czas:289,481958349,

START O:9 M:0, Czas:296,62013100900003,

STOP O:9 M:0, Czas:306,27988924000005,

Wyniki uzyskane przez pierwszą heurystykę, w porównaniu z kolejnymi, są mierne. Zaledwie jedno z uruchomień algorytmu pozwoliło na zbliżenie się w poszukiwaniu optymalnego rozwiązania do wartości równej 300. Pozostałe najlepsze wyniki osiągają wartości pomiędzy 350 a 500. Należy również zwrócić uwagę na bardzo złe rozwiązania osiągane w pierwszych iteracjach. W przeciwieństwie do heurystyk używających JA, które już w pierwszej iteracji odszukiwały rozwiązania o zbliżonej jakości do najlepszych znalezionych, wyniki pierwszej heurystyki dopiero po kilku iteracjach zbliżały się do najlepszych znalezionych przez nie rozwiązań. Mimo to trzeba przyznać, że poprawa jakości w kilku pierwszych iteracjach jest znaczna.

*Tabela 6.13 Wyniki testów Heurystyki 2*

|  |  |  |
| --- | --- | --- |
| L. P. testu | Uzyskany najlepszy wynik | Liczba iteracji potrzebnych do uzyskania najlepszego wyniku |
| 1 | 300.58946462500006 | 1 |
| 2 | 474.16946022000013 | 1 |
| 3 | 554.8304716330001 | 1 |
| 4 | 429.6404718349999 | 1 |
| 5 | 440.81902705300007 | 126 |
| 6 | 439.7390903880002 | 1 |
| 7 | 411.381679897 | 41 |
| 8 | 521.825670287 | 131 |
| 9 | 1218.3857500380002 | 1 |
| 10 | 642.1879904589238 | 21 |

*Wykres 6.2 Wartość najlepszego znalezionego globalnie rozwiązania w poszczególnych etapach testów Heurystyki 2*

Najlepsze znalezione rozwiązanie (test 1):

(O – numer operacji, M – numer maszyny)

Proces 0

START O:19 M:5, Czas:0,

STOP O:19 M:5, Czas:24,41876376,

START O:0 M:4, Czas:24,41876376,

STOP O:0 M:4, Czas:87,14088281000001,

START O:18 M:6, Czas:87,14088281000001,

STOP O:18 M:6, Czas:163,72066579,

START O:1 M:6, Czas:169,791330147,

STOP O:1 M:6, Czas:196,34669979699999,

Proces 1

START O:17 M:7, Czas:94,134434155,

STOP O:17 M:7, Czas:189,661414765,

START O:2 M:7, Czas:193,58694226699998,

STOP O:2 M:7, Czas:205,25697986699998,

START O:16 M:3, Czas:217,173652354,

STOP O:16 M:3, Czas:276,44228406400003,

START O:3 M:6, Czas:277,576620794,

STOP O:3 M:6, Czas:288,876824944,

Proces 2

START O:15 M:1, Czas:0,

STOP O:15 M:1, Czas:92,57832714,

START O:4 M:3, Czas:92,57832714,

STOP O:4 M:3, Czas:175,33325691,

START O:14 M:2, Czas:183,796445616,

STOP O:14 M:2, Czas:192,45646019,

START O:5 M:3, Czas:193,277993995,

STOP O:5 M:3, Czas:215,828310915,

Proces 3

START O:13 M:4, Czas:95,807487694,

STOP O:13 M:4, Czas:187,423509994,

START O:6 M:8, Czas:224,051088118,

STOP O:6 M:8, Czas:258,150183928,

START O:12 M:8, Czas:259,187960398,

STOP O:12 M:8, Czas:270,020605328,

START O:7 M:7, Czas:272,29693224500005,

STOP O:7 M:7, Czas:300,58946462500006,

Proces 4

START O:11 M:8, Czas:0,

STOP O:11 M:8, Czas:68,61940247,

START O:8 M:7, Czas:68,61940247,

STOP O:8 M:7, Czas:93,0150575,

START O:10 M:2, Czas:93,0150575,

STOP O:10 M:2, Czas:129,18173371,

START O:9 M:8, Czas:132,89119588,

STOP O:9 M:8, Czas:222,92955833,

Wyniki testów drugiej heurystyki okazały się niewiele lepsze od testów pierwszej. Choć trzeba odnotować osiągnięcie dopuszczalnych rozwiązań już w pierwszej iteracji, to ich poprawa miała miejsce w zaledwie czterech na dziesięć testów. Znając wyniki kolejnych testów można zauważyć, że najlepsze znalezione przez drugą heurystykę rozwiązania są dużo gorsze od tych, które były osiągalne. Należy jednak podkreślić rolę algorytmu Johnsona, który pozwolił algorytmowi genetycznemu zacząć poszukiwania od całkiem dobrych rozwiązań. Jak się jednak okazało algorytm genetyczny w dalszej części nie poradził sobie najlepiej. Być może powodem tego stanu rzeczy było wpadanie GA w minima lokalne.

*Tabela 6.14 Wyniki testów Heurystyki 3*

|  |  |  |
| --- | --- | --- |
| L. P. testu | Uzyskany najlepszy wynik | Liczba iteracji potrzebnych do uzyskania najlepszego wyniku |
| 1 | 143.476273934 | 101 |
| 2 | 170.51322176099998 | 381 |
| 3 | 218.069875686 | 483 |
| 4 | 182.79324661 | 168 |
| 5 | 164.31250476600002 | 63 |
| 6 | 188.31287557000002 | 146 |
| 7 | 168.55645665900002 | 67 |
| 8 | 227.566780404 | 401 |
| 9 | 183.625194819 | 278 |
| 10 | 230.02746508600003 | 17 |

*Wykres 6.3 Wartość najlepszego znalezionego globalnie rozwiązania w poszczególnych etapach testów Heurystyki 3*

Najlepsze znalezione rozwiązanie (test 1):

(O – numer operacji, M – numer maszyny)

Proces 0

START O:19 M:0, Czas:0,

STOP O:19 M:0, Czas:25,29969438,

START O:0 M:9, Czas:25,29969438,

STOP O:0 M:9, Czas:26,653115658999997,

START O:18 M:0, Czas:33,350339250999994,

STOP O:18 M:0, Czas:39,813769994999994,

START O:1 M:0, Czas:44,00930984399999,

STOP O:1 M:0, Czas:99,10843609399998,

Proces 1

START O:17 M:5, Czas:0,

STOP O:17 M:5, Czas:4,924905342,

START O:2 M:7, Czas:4,924905342,

STOP O:2 M:7, Czas:16,594942942,

START O:16 M:5, Czas:26,069001483999998,

STOP O:16 M:5, Czas:80,91077479399999,

START O:3 M:5, Czas:87,432723504,

STOP O:3 M:5, Czas:121,618637554,

Proces 2

START O:15 M:3, Czas:0,

STOP O:15 M:3, Czas:21,08019984,

START O:4 M:6, Czas:21,08019984,

STOP O:4 M:6, Czas:32,55726903,

START O:14 M:9, Czas:42,278669586,

STOP O:14 M:9, Czas:100,198961206,

START O:5 M:0, Czas:108,74588813199999,

STOP O:5 M:0, Czas:135,29251058199998,

Proces 3

START O:13 M:1, Czas:0,

STOP O:13 M:1, Czas:22,30092538,

START O:6 M:8, Czas:22,30092538,

STOP O:6 M:8, Czas:56,400021190000004,

START O:12 M:8, Czas:57,43779766,

STOP O:12 M:8, Czas:68,27044259,

START O:7 M:1, Czas:68,918767714,

STOP O:7 M:1, Czas:143,476273934,

Proces 4

START O:11 M:2, Czas:0,

STOP O:11 M:2, Czas:91,8583142,

START O:8 M:7, Czas:94,775255047,

STOP O:8 M:7, Czas:119,170910077,

START O:10 M:7, Czas:126,207424899,

STOP O:10 M:7, Czas:130,661836706,

START O:9 M:7, Czas:131,760673502,

STOP O:9 M:7, Czas:141,470802653,

Trzecia heurystyka odznaczyła się dużo lepszymi wynikami od poprzednich dwóch. Nawet wynik testu o najgorszym wyniku dla tej heurystyki jest lepszy, niż rezultaty testów o najlepszych wynikach dla dwóch pierwszych heurystyk.

Dodatkowo należy zwrócić uwagę na wykres 6.3, gdzie można zauważyć, że już w pierwszej iteracji znajdywane były dosyć dobre rozwiązania, które były bardzo szybko zastępowane jeszcze lepszymi. Wszystkie najlepsze rozwiązania w skali całych testów odkrywane były przed pięćsetną iteracją, a większość przed dwusetną, co wskazuje na zdolność heurystyki do szybkiego znajdywania lepszych rozwiązań.

*Tabela 6.15 Wyniki testów Heurystyki 4*

|  |  |  |
| --- | --- | --- |
| L. P. testu | Uzyskany najlepszy wynik | Liczba iteracji potrzebnych do uzyskania najlepszego wyniku |
| 1 | 198.81011502899997 | 34 |
| 2 | 170.44807121399998 | 40 |
| 3 | 177.52771093599998 | 167 |
| 4 | 166.69052617500003 | 775 |
| 5 | 168.085328482 | 510 |
| 6 | 162.557808391 | 525 |
| 7 | 156.545973025 | 605 |
| 8 | 241.06562173100002 | 985 |
| 9 | 200.46281788 | 870 |
| 10 | 181.63046771199998 | 335 |

*Wykres 6.4 Wartość najlepszego znalezionego globalnie rozwiązania w poszczególnych etapach testów Heurystyki 4*

Proces 0

START O:19 M:2, Czas:0,

STOP O:19 M:2, Czas:27,16448184,

START O:0 M:9, Czas:27,16448184,

STOP O:0 M:9, Czas:28,517903119,

START O:18 M:2, Czas:32,086747164,

STOP O:18 M:2, Czas:76,14388960400001,

START O:1 M:2, Czas:80,76667933,

STOP O:1 M:2, Czas:111,29542949,

Proces 1

START O:17 M:5, Czas:0,

STOP O:17 M:5, Czas:4,924905342,

START O:2 M:7, Czas:4,924905342,

STOP O:2 M:7, Czas:16,594942942,

START O:16 M:7, Czas:25,831116058,

STOP O:16 M:7, Czas:113,310319338,

START O:3 M:5, Czas:113,739755723,

STOP O:3 M:5, Czas:147,925669773,

Proces 2

START O:15 M:3, Czas:0,

STOP O:15 M:3, Czas:21,08019984,

START O:4 M:6, Czas:21,08019984,

STOP O:4 M:6, Czas:32,55726903,

START O:14 M:1, Czas:32,55726903,

STOP O:14 M:1, Czas:104,80300994,

START O:5 M:1, Czas:106,69306734599999,

STOP O:5 M:1, Czas:122,37643034599999,

Proces 3

START O:13 M:9, Czas:33,359600459999996,

STOP O:13 M:9, Czas:79,9288799,

START O:6 M:3, Czas:83,926663832,

STOP O:6 M:3, Czas:122,973514652,

START O:12 M:2, Czas:124,89946719900001,

STOP O:12 M:2, Czas:133,057728652,

START O:7 M:0, Czas:148,695695573,

STOP O:7 M:0, Czas:156,545973025,

Proces 4

START O:11 M:0, Czas:0,

STOP O:11 M:0, Czas:16,38885996,

START O:8 M:0, Czas:21,737115438,

STOP O:8 M:0, Czas:43,365574028,

START O:10 M:0, Czas:52,871821857,

STOP O:10 M:0, Czas:132,580572537,

START O:9 M:0, Czas:132,63835287199998,

STOP O:9 M:0, Czas:142,298111103,

Wyniki testów dotyczące czwartej heurystyki okazały się być podobne do wyników trzeciej. Co prawda żaden z testów nie poprawił najlepszego wyniku pierwszego testu trzeciej heurystyki (wynik 143.476273934), ale średni najlepszy uzyskany wynik jest w przypadku tej heurystyki nieco lepszy.

W przeciwieństwie do poprzedniej heurystyki, testy wykazały mniejszą zdolność dążenia do lepszych rozwiązań. Większość najlepszych rozwiązań było odszukiwanych dopiero po pięćsetnej iteracji, przez co średni numer iteracji, po którym nie odnaleziono lepszego rozwiązania (484,6) jest ponad dwukrotnie gorszy ten sam parametr trzeciej heurystyki (210,5).

## 6.5 Porównanie czasu działania heurystyk

Kluczowym parametrem wszystkich heurystyk jest czas ich działania. Jest to niezwykle ważny parametr, bowiem użycie długo liczącego oprogramowania nie byłoby opłacalne w przedsiębiorstwie, które potrzebuje harmonogramu co okres krótszy niż czas obliczeń. Nawet najlepsze heurystyki, które potrafiłyby z bardzo dużym prawdopodobieństwem odnaleźć optymalne rozwiązanie stają się bezużyteczne, gdy czas potrzebny na ich obliczenie oraz obciążenie jednostek obliczeniowych przekracza dopuszczalne granice.

Testy czasowe wykonywane były jeden po drugim na tym samym komputerze, który poza procesami systemu operacyjnego i paroma procesami w tle, które pracowały niezmiennie, miał za zadanie jedynie realizować działanie programu. Model matematyczny, który został użyty pochodził ze scenariusza 4. W kliku przypadkach dopatrzono się błędów grubych, które zostały zastąpione wynikami dodatkowych testów. Za błędy grube uznano czasy, które różniły się od pozostałych o więcej niż 16%.

*Tabela 6.16 Porównanie czasu działania zaimplementowanego programu w zależności od używanej heurystyki*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Heurystyka | Pomiar 1 [s] | Pomiar 2 [s] | Pomiar 3 [s] | Średnia |
| 1 | 757,0693123 | 785,8484276 | 782,9454813 | 775,2877404 |
| 2 | 870,4368423 | 777,6475243 | 811,4160547 | 819,8334738 |
| 3 | 1340,1116387 | 1252,6073122 | 1305,2586538 | 1299,3258682 |
| 4 | 1027,8385089 | 1008,2863408 | 1023,6941771 | 1019,9396756 |

Wyniki pomiarów czasowych wskazały na większe obciążenie w przypadku heurystyk, które używają JA. Zauważalna jest różnica pomiędzy drugą heurystyką, która używa algorytmu Johnsona tylko przy tworzeniu populacji początkowej, czwartą, która używa go tylko do oceny rozwiązań, oraz trzecią, która używa go łącznie z algorytmem genetycznym.

Porównując czasy potrzebne do wykonania obliczeń z jakością znajdywanych rozwiązań ze scenariusza 4, można stwierdzić, że najlepszym połączeniem jakości i czasu działania charakteryzuje się czwarta heurystyka. Jeśli jednak liczba iteracji trzeciej heurystyki została zmniejszona do 500 (co zgodnie z wynikami testów nie wpływa na jakość wyników), to właśnie ta heurystyka była najlepsza pod względem czasu i jakości.

Nie należy zapominać o scenariuszu 2, który wykazał brak zdolności trzeciej i czwartej heurystyki do znalezienia optimum. Wobec tego pierwsze dwie heurystyki w niektórych przypadkach mogą być znacznie lepszym wyborem niż ostatnie dwie.

# 7. Podsumowanie

# 8. Bibliografia

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

**[1]** (online) <https://sjp.pwn.pl/sjp/produkcja;2572515.html>

**[2]** N. Davies *Europa. Rozprawa historyka z historią* Wydawnictwo Znak 2010.

**[3]** E. Nowicki and C. Smutnicki *A Fast Taboo Search Algorithm for the Job Shop Problem* Management Science , Jun., 1996, Vol. 42, No. 6 (Jun., 1996), pp. 797-813

**[4]** M. M. Ahmadian, M. Khatami, A. Salehipour, T.C.E. Cheng*, Four decades of research on the open-shop scheduling problem to minimize the makespan*, European Journal of Operational Research, Volume 295, Issue 2, 2021, pp. 399-426, ISSN 0377-2217,

**[5]** A. Baskar, M. Anthony Xavior, *A Simple Model to Optimize General Flow-Shop Scheduling Problems With Known Break Down Time And Weights Of Jobs*, Procedia Engineering, Volume 38, 2012, pp. 191-196, ISSN 1877-7058,

**[6]** 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.

**[7]** 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.

**[8]** 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.

**[9]** 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.

**[10]** 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.

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

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