

Jolanta KRYSTEK, Kacper WACH
Politechnika Śląska

ANALIZA WYBRANYCH ALGORYTMÓW DO SZEREGOWANIA ZADAŃ W WIELOMASZYNOWYM PRZEPŁYWOWYM SYSTEMIE PRODUKCYJNYM

Streszczenie. Celem pracy był wybór, implementacja oraz przetestowanie działania wybranych algorytmów umożliwiających rozwiązanie problemów szeregowania zadań w wielomaszynowym systemie przepływowym oraz optymalizacji wybranych wskaźników jakości. Zaimplementowano takie algorytmy jak algorytm genetyczny, algorytm iteracyjny zachłanny, poszukiwanie Tabu oraz symulowane wyżarzanie. Do badań wykorzystano autorską aplikację umożliwiającą generowanie problemów szeregowania zadań, dobór parametrów algorytmów oraz wyświetlanie otrzymanych wyników. Badania przeprowadzono na różnych zestawach instancji problemów, a do oceny jakości otrzymanych rozwiązań wykorzystano odpowiednie kryteria.

ANALYSIS OF SELECTED ALGORITHMS FOR SCHEDULING JOBS IN A MULTI-MACHINE FLOW SHOP PRODUCTION SYSTEM

Summary. The aim of the work was to select, implement and test the operation of selected algorithms enabling the solution of task scheduling problems in a multi-machine flow system and the optimization of selected quality indicators. The following algorithms were selected and implemented in the study: Genetic Algorithm, Iterated Greedy, Tabu Search, and Simulated Annealing. The research used a proprietary application that allows for generating task scheduling problems, configuration of algorithm parameters, and presentation the obtained results. The research was conducted on various sets of problem instances, and appropriate evaluation criteria were used to assess the obtained results.

1. Wprowadzenie

Współczesne przedsiębiorstwa produkcyjne funkcjonują w warunkach wysokiej konkurencyjności rynku, co wymaga ciągłego zwiększania efektywności procesów technologicznych oraz jak najlepszego wykorzystania dostępnych zasobów. Rozwój automatyzacji produkcji oraz koncepcji Przemysłu 4.0 powoduje, że efektywne harmonogramowanie procesów staje się jednym z kluczowych elementów zarządzania produkcją. Odpowiednio dobrane metody szeregowania umożliwiają redukcję kosztów operacyjnych, skrócenie czasu realizacji zamówień, zmniejszenie przestojów maszyn oraz poprawę terminowości wykonywania zleceń.

Jednym z podstawowych problemów rozważanych w teorii szeregowania jest szeregowanie zleceń w systemie przepływowym (ang. *flow shop scheduling problem*). W systemie idealnym przepływowym każde zlecenie jest wykonywane na wszystkich maszynach w tej samej kolejności, rozpoczynając od pierwszej maszyny i kończąc na ostatniej. Można również rozpatrywać system ogólny, w którym nie wszystkie zadania poszczególnych zleceń są wykonywane na wszystkich maszynach. Do oceny jakości otrzymanych rozwiązań najczęściej wykorzystuje się dwa kryteria: C_{max} (ang. *makespan*), czyli czas zakończenia ostatniego zadania na ostatniej maszynie, będący jednocześnie całkowitym czasem realizacji procesu oraz F_{sr} , oznaczający średni czas przepływu wszystkich zleceń. Wybór odpowiedniego kryterium zależy od specyfiki rozważanego problemu oraz jego zastosowania praktycznego.

2. Opis zaimplementowanych algorytmów

Problemy szeregowania zadań w systemie przepływowym, w których liczba maszyn jest większa niż dwie, należą do klasy problemów NP-trudnych. Oznacza to, że dla bardziej złożonych instancji problemu nie istnieją znane algorytmy pozwalające na wyznaczenie rozwiązania optymalnego w czasie wielomianowym. Z tego powodu w praktyce często stosuje się algorytmy heurystyczne [14,2,12,8], które pozwalają na znalezienie rozwiązania bliskiego optymalnemu w akceptowalnym czasie obliczeniowym.

W pracy zaimplementowano pięć różnych algorytmów: metoda wykorzystująca algorytm Johnsona, algorytm genetyczny (ang. *Genetic Algorithm*), algorytm iteracyjny zachłanny (ang. *Iterated Greedy*), poszukiwanie Tabu (ang. *Tabu Search*) oraz symulowane wyżarzanie (ang. *Simulated Annealing*).

2.1. Algorytm Johnsona

Algorytm Johnsona pozwala na wyznaczenie optymalnego rozwiązania problemu szeregowania zadań w dwumaszynowym systemie przepływowym [1] ze względu na C_{max} , czyli długość uszeregowania. W pracy analizowano następujące przypadki, w których stosowano algorytm Johnsona:

1. System dwumaszynowy – w takim przypadku można od razu wykorzystać algorytm Johnsona. Otrzymany harmonogram pracy maszyn jest identyczny na obu maszynach.
2. Systemy wielomaszynowe (z parzystą lub nieparzystą liczbą maszyn), w których otrzymywane harmonogramy pracy maszyn (w zależności od sposobu zastosowania algorytmu Johnsona) były identyczne lub różne na wszystkich maszynach.

2.2. Algorytm genetyczny

Algorytm genetyczny jest heurystyczną metodą optymalizacji, która w [3] została zaproponowana do rozwiązywania problemów w systemie przepływowym. Zaimplementowana metoda składa się z następujących kroków:

1. Tworzenie populacji osobników – osobnikiem jest dana kolejność wykonywania zleceń na maszynach.
2. Obliczanie wskaźnika jakości (C_{max}) dla każdego osobnika w populacji.
3. Selekcja – wybór osobników (na podstawie na wyliczonych wskaźników jakości) które będą brały udział w procesie krzyżowania. Jako metodę selekcji wybrano metodę ruletki [3] w której osobniki do krzyżowania wybierane są losowo jednak im lepsza jest

wartość wskaźnika jakości danego osobnika (w tym przypadku im niższa) tym większe jest prawdopodobieństwo wylosowania danego osobnika.

4. Krzyżowanie – powstają nowe osobniki, czyli nowe harmonogramy wykonywania zadań na maszynach (wykorzystano metodę piramidową zaproponowaną w [3]).

5. Mutacja – metoda zapobiegania utknięciu algorytmu w lokalnym minimum. Można ją wykonać na trzy sposoby: pierwszy z nich polega na przesunięciu pierwszego zlecenia harmonogramu na jego koniec. Druga metoda polega na przeniesieniu ostatniego zlecenia na początek kodu genetycznego. W ostatniej metodzie mutacji losowane są dwa zlecenia, które zostają zamienione miejscami.

6. Utworzenie nowej populacji – kiedy mamy już nowo powstałe osobniki tworzymy z nich nową populację, która składa się w połowie z osobników z poprzedniej populacji (posiadających najlepsze wskaźniki jakości), a druga połowa z potomków. Dla nowo powstałej populacji powtarzane są powyższe kroki od 2 do 5.

7. Kryterium stopu – jako kryterium stopu przyjmuje się zadaną liczbę iteracji oraz brak poprawy wskaźnika jakości przez określoną liczbę iteracji.

2.3. Algorytm iteracyjny zachłanny

Algorytm zachłanny (ang. *Iterated Greedy – IG*) [11]. generuje kolejne sekwencje rozwiązań poprzez stosowanie zachłannych heurystyk powtarzających się iteracyjnie, czyli destrukcji i odbudowie. Składa się z następujących kroków:

1. Inicjalizacja – tworzenie pierwszej sekwencji wykonywania zadań na maszynach. przy użyciu klasycznego algorytmu heurystycznego NEH [9] do szeregowania zadań w systemie przepływowym. Algorytm ten działa w następujący sposób: najpierw dla każdego zlecenia sumuje się czasy wykonywania zadań na każdej maszynie, następnie sortuje się zlecenia w kolejności malejącej ze względu na obliczone wcześniej sumy. Wybierane są dwa pierwsze zlecenia (posiadające największe sumy czasów) dla których sprawdza się, które z uszeregowanych tych dwóch zleceń posiada lepszy wskaźnik jakości. Te procedurę powtarza się dla wszystkich zleceń z wcześniej otrzymanej sekwencji.

2. Destrukcja – usuwanie zleceń. Liczba zleceń do usunięcia (d) jest definiowana przez użytkownika jako jeden z parametrów algorytmu (musi być mniejsza niż liczba zleceń w uszeregowaniu). Indeksy zleceń do usunięcia są wybierane losowo bez powtórzeń.

3. Odbudowa – podczas tego kroku zlecenia usunięte z uszeregowania zostają ponownie do niego wprowadzone z wykorzystaniem części algorytmu NEH. Zadania usunięte w fazie destrukcji są ustawiane według malejących sum czasów wykonywania, a następnie kolejno wstawiane we wszystkie możliwe miejsca uszeregowania. Dla każdego wstawienia wyznaczana jest wartość wskaźnika jakości, po czym wybierane jest rozwiązanie najlepsze. Proces ten jest powtarzany aż do wyczerpania listy wszystkich usuniętych zleceń.

4. Lokalne poszukiwanie – w celu poprawy otrzymanego w fazie odbudowy rozwiązania można skorzystać z wybranego wariantu algorytmu poszukiwania lokalnego.

5. Kryterium akceptacji – po otrzymaniu nowego rozwiązania należy ustalić, czy otrzymane rozwiązanie jest lepsze od początkowego i zostanie zaakceptowane i wykorzystane w następnej iteracji. Najprostszym kryterium akceptacji jest poprawa wskaźnika jakości C_{max} . Alternatywą zaproponowaną w artykule [11] jest kryterium akceptacji zastosowane w algorytmie symulowanego wyżarzania.

6. Kryterium stopu – jako kryterium stopu można przyjąć wybraną liczbę iteracji oraz brak poprawy wskaźnika jakości po wykonaniu zadanej liczby iteracji.

2.4. Algorytm Tabu Search

Algorytm Tabu Search [4, 5] jest heurystyczną metodą opartą na lokalnym poszukiwaniu oraz unikaniu powracania do poprzednio otrzymanych rozwiązań. Jako wskaźnik jakości określający skuteczność rozwiązań wybrano długość uszeregowania. Algorytm składa się z następujących kroków:

1. Generowanie początkowego rozwiązania – w sposób losowy lub przy użyciu heurystyki konstrukcyjnej (na przykład heurystyki NEH).
2. Generowanie rozwiązań sąsiednich – wykonywane jest poprzez proste przekształcenia początkowego rozwiązania, na przykład przez zamianę kolejności dwóch zleceń lub przeniesienie danego zlecenia w inne miejsce w sekwencji. Istotnym elementem algorytmu Tabu Search jest lista tabu, która przechowuje informacje o niedawno wykonanych ruchach, które stają się na pewną liczbę iteracji zabronione w celu zapobiegania powracania do wcześniej otrzymanych rozwiązań (*tabu tenure*). Warto dodać, że w algorytmie stosowany jest mechanizm nazywany kryterium aspiracji, który pozwala w szczególnych przypadkach na wykonanie zabronionego ruchu z listy tabu. Jest to możliwe wtedy, kiedy ten ruch pozwoli na otrzymanie lepszego rozwiązania niż rozwiązanie aktualnie najlepsze.
3. Wybór kolejnego początkowego rozwiązania – spośród wszystkich rozwiązań sąsiednich wygenerowanych w poprzednich iteracjach. Jako nowe rozwiązanie początkowe przyjmowane jest to, które ma najniższą wartość funkcji celu. Należy jednak zaznaczyć, że jego wartość funkcji celu nie musi być lepsza od wartości rozwiązania bazowego z poprzedniej iteracji. Oznacza to, że nowe rozwiązanie początkowe nie musi być najlepszym rozwiązaniem znalezionym do tej pory w całym przebiegu algorytmu.
4. Kryterium stopu – algorytm działa do momentu jego spełnienia. Kryterium stopu może być maksymalna liczba iteracji lub też brak poprawy rozwiązania przez określoną liczbę iteracji.

2.5. Algorytm symulowanego wyżarzania

Zaimplementowana wersja algorytmu symulowanego wyżarzania (ang. *Simulated Annealing*) [7], składa się z następujących kroków:

1. Inicjalizacja – wybór pierwszego początkowego uszeregowania zadań na maszynach. Może być to losowa kolejność lub kolejność otrzymana za pomocą algorytmu heurystycznego NEH. Przy uruchomieniu algorytmu należy zdefiniować takie parametry jak T_i (początkowa temperatura), α (współczynnik zmniejszenia temperatury w kolejnych iteracjach) oraz L (liczba wykonanych iteracji dla danej temperatury T_i).
2. Generowanie nowego rozwiązania – nowe rozwiązania mogą być generowane na różne sposoby, na przykład przez zamianę losowo wyznaczonych dwóch różnych zleceń lub przez przeniesienie losowo wyznaczonego zlecenia w inne miejsce uszeregowania.
3. Wyznaczanie wskaźnika jakości – dla nowo wyznaczonego rozwiązania (π') obliczany jest wskaźnik jakości $C_{max}(\pi')$ i jest on porównywany ze wskaźnikiem jakości początkowego rozwiązania (π), $C_{max}(\pi)$, przez różnicę $\Delta T = C_{max}(\pi') - C_{max}(\pi)$.
4. Akceptacja nowego rozwiązania – nowe rozwiązanie jest akceptowane, jeżeli $\Delta T \leq 0$ lub jest akceptowane z prawdopodobieństwem wyliczonym jako $e^{-\Delta T/T_i}$. Warto dodać,

że przyjęcie rozwiązania nie jest równoznaczne z uznaniem go za najlepsze rozwiązanie. Tylko w przypadku, kiedy otrzymana wartość wskaźnika jakości nowego rozwiązania jest lepsza od poprzedniej najlepszej znalezionej wartości wskaźnika jakości, rozwiązanie przyjmuje się jako aktualnie najlepsze.

5. Powtarzanie dla danej temperatury T_i – kroki od 2 do 4 są powtarzane L -krotnie.

6. Przejście do następnej iteracji – przy przejściu do nowej iteracji obliczana jest nowa temperatura $T_{i+1} = \alpha \cdot T_i$, po czym powtarza się kroki od 2 do 5.

7. Kryterium stopu – jako kryterium stopu najczęściej stosuje się liczbę iteracji oraz spadek temperatury poniżej zadanego progu ϵ ($T_i < \epsilon$).

3. Metodyka badań

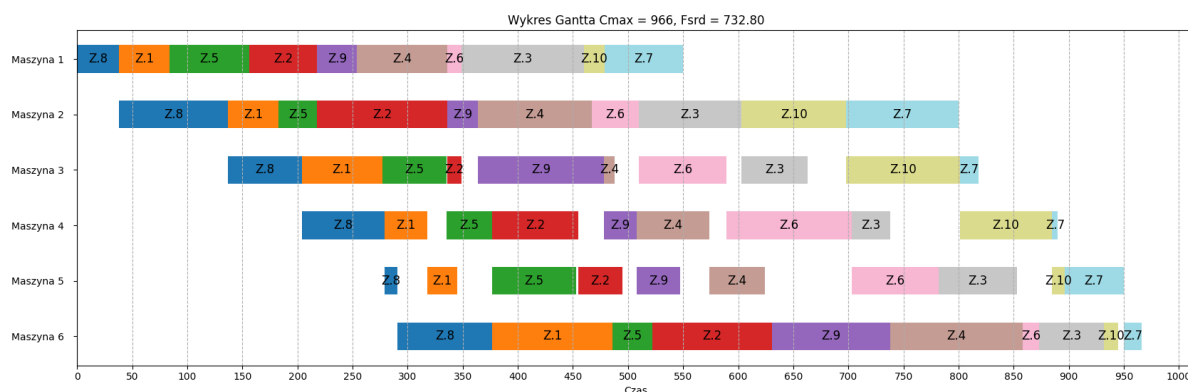
Zaimplementowane algorytmy zostały porównane zarówno między sobą, jak i każdy z algorytmów był testowany dla różnych zestawów parametrów. Badania algorytmów dla konkretnych problemów były powtarzane wielokrotnie w celu sprawdzenia powtarzalności otrzymanych wyników a otrzymane wartości wskaźników jakości i czasów wykonywania obliczeń były uśredniane. Dokonano również porównania działania algorytmów z najlepszymi znanymi rozwiązaniami dla wybranych instancji ze zbioru Taillarda [13]. W niniejszej pracy przedstawiono tylko część badań i wyników. Jako kryteria porównawcze w tych badaniach zastosowano wskaźniki C_{max} oraz F_{sr} , jak również czas działania algorytmów. Badania były przeprowadzane z wykorzystaniem komputera z systemem Windows 11, procesorem Intel Core i5-10310U, 16 GB pamięci RAM.

Przykład testowy (obejmujący 6 maszyn i 10 zleceń) zamieszczono poniżej jako ilustrację prawidłowego działania testowanych algorytmów.

Tabela 1

Dane testowe: czasy zadań

	Zlecenie 1	Zlecenie 2	Zlecenie 3	Zlecenie 4	Zlecenie 5	Zlecenie 6	Zlecenie 7	Zlecenie 8	Zlecenie 9	Zlecenie 10
Maszyna 1	46	62	111	82	72	13	71	38	36	19
Maszyna 2	46	118	93	103	35	43	102	99	28	95
Maszyna 3	73	13	60	10	58	79	17	67	114	103
Maszyna 4	39	78	35	66	42	114	5	75	30	84
Maszyna 5	27	40	71	50	76	79	54	12	39	11
Maszyna 6	109	109	59	120	36	15	16	86	107	13



Rys. 1. Harmonogram pracy maszyn – minimalizacja długości uszeregowania C_{max}

Tabela 1
Wyniki dla minimalizacji C_{max}

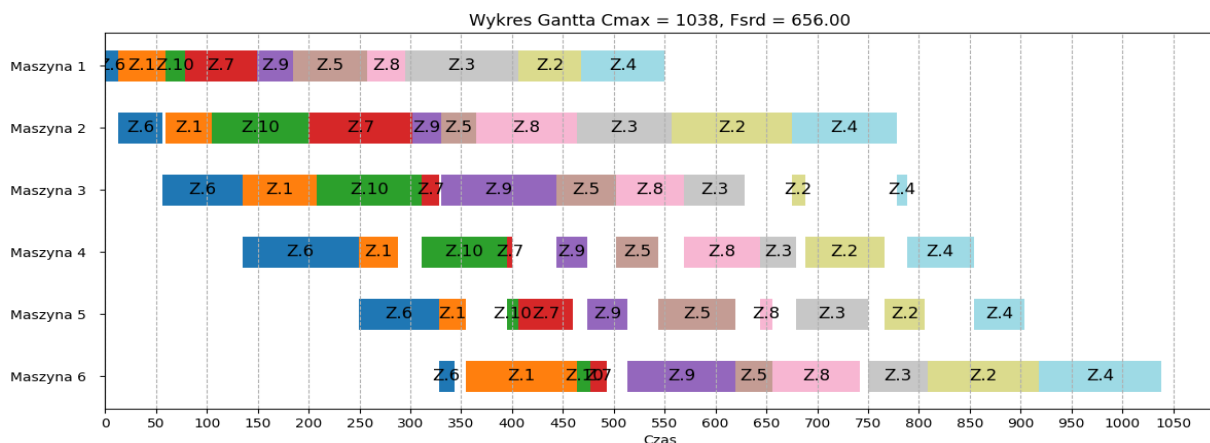
algorytm	czas	C_{max}
genetyczny	0,2779	966
alg. IG	0,0458	966
Tabu Search	0,0310	966
alg. SA	0,0266	966

Tabela 2
Wyniki dla minimalizacji F_{sr}

algorytm	czas	F_{sr}
genetyczny	0,3848	656
alg. IG	0,2087	656
Tabu Search	0,4833	656
alg. SA	0,3221	656

Parametry zaimplementowanych algorytmów:

- genetyczny: liczebność populacji=100; liczba populacji =150; prawdopodobieństwo mutacji = 35%.
- Iterated Greedy: liczba iteracji = 30; $d = 4$; liczba iteracji bez poprawy wskaźnika jakości = 10; początkowe rozwiązanie otrzymane z algorytmu NEH.
- Tabu Search: *tabu tenure* = 10; liczba sąsiadów = 20; liczba iteracji = 30; pierwsze rozwiązanie otrzymane z algorytmu NEH.
- Simulated Annealing: temperatura $T=100$; $\alpha = 0.25$; liczba iteracji = 100, $L = 10$.



Rys. 2. Harmonogram pracy maszyn – minimalizacja średniego czasu przepływu F_{sr}

Badania porównawcze wykorzystujące zbiór Taillarda. W tabelach (3-5) oznaczenia instancji są zgodne ze zbiorem Taillarda, gdzie *ta* oznacza zbiór Taillarda, a numer wskazuje konkretną instancję problemu. Zastosowano następujące skróty: *naj.* oznacza najlepszą otrzymaną wartość wskaźnika jakości C_{max} dla danej instancji, *śr.* oznacza średnią wartość wskaźnika C_{max} otrzymaną podczas dwudziestu uruchomień danej instancji. Natomiast *najlepsze* oznacza najlepszą znaną wartość wskaźnika jakości dla tej instancji.

Wyniki przedstawione w tabelach zostały otrzymane dla tych samych zestawów parametrów każdego z algorytmów wybranych na podstawie badań przeprowadzonych dla problemów generowanych losowo. Zastosowane parametry były następujące:

- Algorytm genetyczny: liczebność populacji=400, liczba generacji=500, prawdopodobieństwo mutacji = 0,55%.
 - Iterated Greedy: $d=4$, liczba iteracji=300, liczba iteracji bez poprawy=40,
 - Tabu Search: *tabu tenure*=20, liczba sąsiadów=60, liczba iteracji=300.
 - Simulated Annealing: $T = 1000$, wsp. $\alpha = 0,98$, $L = 80$, liczba iteracji=5000.
- Jako rozwiązanie początkowe (dla trzech algorytmów: alg. IG, alg. Tabu, alg. SA.) zastosowano rozwiązanie uzyskane za pomocą algorytmu NEH.

Tabela 3

Porównanie algorytmów dla problemów ta001–ta010

instancja	alg. gen		alg. IG		alg. Tabu		alg. SA		najlepsze
	naj.	śr.	naj.	śr.	naj.	śr.	naj.	śr.	
ta001	1294	1296,65	1278	1278,4	1278	1279,15	1278	1278,65	1278
ta002	1359	1362,75	1359	1362	1359	1362,85	1359	1359,85	1359
ta003	1081	1091,1	1081	1086,95	1081	1094,75	1081	1088,35	1081
ta004	1309	1309,4	1297	1300	1293	1298,8	1293	1298,4	1293
ta005	1243	1251,05	1235	1237,8	1235	1247,25	1235	1241,85	1235
ta006	1210	1218,64	1195	1205,05	1195	1209,25	1195	1205,5	1195
ta007	1251	1252,85	1249	1250,9	1251	1251	1239	1250,4	1234
ta008	1206	1212,1	1206	1208	1206	1208,75	1206	1209,85	1206
ta009	1230	1236,3	1230	1238,7	1230	1244,45	1230	1235,6	1230
ta010	1108	1120,3	1108	1108,15	1108	1110,6	1108	1108	1108

Tabela 4

Porównanie algorytmów dla problemów ta031–ta040

instancja	alg. gen		alg. IG		alg. Tabu		alg. SA		najlepsze
	naj.	śr.	naj.	śr.	naj.	śr.	naj.	śr.	
ta031	2724	2729,95	2724	2726,25	2729	2729	2724	2726,85	2724
ta032	2836	2861,4	2836	2837,8	2838	2838	2838	2840	2834
ta033	2621	2635,9	2621	2621,6	2621	2621,7	2621	2623,85	2621
ta034	2751	2776,15	2753	2722,1	2768	2774,85	2751	2766,1	2751
ta035	2863	2863,95	2863	2863,9	2864	2864	2863	2863,9	2863
ta036	2832	2835,85	2829	2833,6	2832	2835,8	2829	2832,75	2829
ta037	2736	2750,7	2725	2725,45	2725	2730,55	2725	2726,05	2725
ta038	2700	2705,25	2683	2686,75	2686	2688,7	2683	2701,6	2683
ta039	2564	2570,85	2561	2562,5	2561	2564,45	2561	2564,95	2552
ta040	2782	2782	2782	2782,1	2782	2783,85	2782	2782,75	2782

Tabela 5

Porównanie algorytmów dla problemów ta061–ta070

instancja	alg. gen		alg. IG		alg. Tabu		alg. SA		najlepsze
	naj.	śr.	naj.	śr.	naj.	śr.	naj.	śr.	
ta061	5495	5509,6	5493	5493,9	5493	5502,9	5493	5493,4	5493
ta062	5284	5289,1	5268	5282,5	5270	5282,5	5283	5288,35	5268
ta063	5193	5212,3	5175	5186,35	5175	5211,85	5175	5198,6	5175
ta064	5021	5034,15	5018	5022,25	5021	5022,9	5021	5022,9	5014
ta065	5255	5265,85	5252	5253,55	5253	5255,8	5250	5253,9	5250
ta066	5135	5142,3	5135	5138,8	5139	5139	5135	5138,8	5135
ta067	5251	5286,1	5251	5251,6	5252	5243,7	5246	5257,3	5246
ta068	5098	5126,6	5094	5101,05	5094	5097,4	5094	5108,8	5094
ta069	5448	5476	5448	5458,65	5448	5460,95	5448	5458,65	5448
ta070	5342	5349,4	5322	5325,05	5322	5338,55	5328	5337,5	5322

Dla eksperymentów przeprowadzonych na instancjach ze zbioru Taillarda najlepsze rozwiązania najczęściej znajdował algorytm Simulated Annealing, natomiast najlepsze średnie wartości wskaźnika jakości uzyskiwał zazwyczaj algorytm Iterated Greedy. Najsłabsze wyniki zarówno pod względem jakości otrzymywanych rozwiązań, jak i ich powtarzalności, uzyskiwał algorytm genetyczny.

4. Podsumowanie

Zaimplementowane algorytmy różnią się zarówno sposobem przeszukiwania przestrzeni rozwiązań, jak i złożonością obliczeniową. Algorytm Johnsona umożliwia szybkie wyznaczanie rozwiązań dla problemów dwumaszynowych, jednak jego zastosowanie dla większej liczby maszyn jest ograniczone. Metaheurystyki, takie jak Tabu Search, Simulated Annealing, Iterated Greedy oraz algorytm genetyczny, pozwalają uzyskać rozwiązania o wysokiej jakości również dla dużych instancji problemu. Ich skuteczność zależy jednak od odpowiedniego doboru parametrów oraz czasu działania algorytmu.

Praca została zrealizowana przy wsparciu finansowym Politechniki Śląskiej w ramach dotacji na utrzymanie i rozwój potencjału badawczego 2026 02/040/BK_24/1056

LITERATURA

1. Alshorman R., Alrossan H., Smadi S.: Toward Proving the Extension of Johnson's Rule to a Three-Machine Flow Shop Scheduling Problem. W: *Tikrit Journal of Engineering Sciences* 32, 2025, s. 1–10. doi: 10.25130/tjes.sp1.2025.48.
2. Arik O., A.: Population-based Tabu search with evolutionary strategies for permutation flow shop scheduling problems under effects of position-dependent learning and linear deterioration. W: *Soft Computing* 25, 2020, s. 1501–1518.
3. Cubukcuoglu A., Karacan I., Ceylan Z., Bulkan S.: Minimizing Makespan in Ordered Flow Shop Scheduling Using a Robust Genetic Algorithm. W: *Processes* 13.4, 2025.
4. Dodu C.E., Ancău M.: A Tabu Search Approach for Permutation Flow Shop Scheduling. W: *Studia Universitatis Babes-Bolyai Informatica* 65.1, 2020, s. 104–115. issn: 2065-9601. doi: 10.24193/subbi.2020.1.08.
5. Glover F.: Tabu Search—Part I. W: *ORSA Journal on Computing* 1.3, 1989, s. 190–206.
6. Gmys J.: *Permutation Flow-shop: best-known makespans and schedules for Taillard and VFR benchmarks*. 2021. doi: 10.5281/zenodo.4542886.
7. Kai M.: Research On the No-Wait Flow Shop Scheduling Problem Based on Simulated Annealing Algorithm. W: *MEIMIE* 2023. T. 71. 2023, s. 436–442.
8. Kurdi M., Mzili T., Steef A., M Shaheen M.: A New Hybrid Social Spider Optimization and Tabu Search for the Permutation Flow Shop Scheduling Problem". W: *Journal of Electrical and Computer Engineering* 2026. doi: 10.1155/jece/6022369.
9. Nawaz M., Ensore E.E. Jr. Ham I.: A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. W: *Omega* 11.1, 1983, s. 91–95.
10. Pinedo M.: *Theory, Algorithms, and Systems*. New York: Springer, 2008.
11. Ruiz R., Stützle T.: A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem". W: *European Journal of Operational Research* 177.3, 2007, s. 2033–2049.
12. Sedoud A., Ahmed H.: Genetic Algorithm, Simulated Annealing, and Hybrid SA-GA Algorithm for Flow Shop Scheduling: A Comparative Performance Evaluation. 5th International Conference on Recent Academic Studies December 02-03, 2025, Turkey.
13. Taillard E.: Benchmarks for basic scheduling problems. W: *European Journal of Operational Research* 64.2, 1993, s. 278–285. doi: 10.1016/0377-2217(93)90182-M.
14. Zeidan M.A., Al-Abaji M.A., Ahmed M.H.: Improved Particle Swarm Algorithm for Permutation Flow Shop Scheduling Problems. W: *Revista Investigación Operacional* 42.2, 2021, s. 195–203.