Andrzej GNATOWSKI, Radosław IDZIKOWSKI, Jarosław RUDY, Teodor NIŻYŃSKI
Politechnika Wrocławska

METODA UCZENIA PRZEZ WZMOCNIENIE DLA HYBRYDOWEGO PRZEPŁYWOWEGO PROBLEMU SZEREGOWANIA ZADAŃ Z EFEKTEM UCZENIA

Streszczenie. W pracy przeanalizowano specyficzne, występujące w literaturze, wykorzystanie uczenia ze wzmocnieniem dla pewnego wariantu przepływowego problemu szeregowania zadań. Rozważana metoda stosuje nietypowy mechanizm Q-learningu podczas działania algorytmu w celu wyboru w danej iteracji operatora perturbacji. Na podstawie dokonanej analizy i eksperymentów numerycznych wskazano cechy metody oraz jej możliwe usprawnienia wskazujące na jej potencjał oraz konieczność dalszych badań.

ON A REINFORCEMENT LEARNING APPROACH TO A HYBRID FLOW SHOP SCHEDULING PROBLEM WITH LEARNING EFFECTS

Summary. In the paper an analysis of a specific literature use of reinforcement learning for a certain variant of flow shop scheduling problem is presented. The considered method uses an atypical Q-learning mechanism during running of the algorithm in order to choose a perturbation operator in a given iteration. Based on the performed analysis and numerical experiments, the features and possible improvement areas of the method are highlighted, which point to the potential of the method and need of further research.

1. Wstęp

Szeregowanie zadań stanowi jedno z kluczowych wyzwań w dziedzinie badań operacyjnych, mające istotne znaczenie w wielu sektorach przemysłu. Aby rozwiązywać problemy w dynamicznych i złożonych środowiskach produkcyjnych, konieczne staje się tworzenie coraz bardziej rozbudowanych modeli matematycznych. W tym kontekście, dużym wyzwaniem jest identyfikowanie własności problemu które mogłyby posłużyć do jego efektywniejszego rozwiązywania, przykładowo tak jak to pokazano w pracach [7, 14]. Jednym z obiecujących trendów, który może stanowić odpowiedź na ta sformułowane wyzwanie, jest zastosowanie uczenia maszynowego [2]. W szczególności dynamicznie rozwijające się uczenie ze wzmocnieniem (RL, z ang. *Reinforcement Learning*) może pozwalać na automatyczną adaptację algorytmu do rozwiązywania konkretnej instancji problemu [4] lub nawet samego problemu jako takiego [12].

Chociaż bezpośrednie zastosowanie RL do rozwiązywania problemów optymalizacyjnych często napotykało na trudności, to użycie RL do poprawy lub uzupełnienia istniejących metod, zwłaszcza metaheurystyk, osiągnęło większe sukcesy [9]. Przy-

kładem jest praca [8], gdzie zaproponowano algorytm poszukiwania z zakazami (Tabu Search) wykorzystujący Q-learning (jedną z technik RL) do dynamicznego wyboru sąsiedztwa i uzyskujący dobrej jakości wyniki w porównaniu do istniejących algorytmów. Zbliżone podejście zastosowano w pracy [10], dla algorytmu Iterated Greedy Search, gdzie wariant wykorzystujący RL do wyboru operatora perturbacji osiągnął lepszy wynik niż wariant standardowy. Z kolei w pracy [11] zastosowano RL oparte o głębokie sieci neuronowe dla algorytmu symulowanego wyżarzania, pozwalające na dostosowanie struktury otoczenia na podstawie historii działania algorytmu. Powyższe metody dotyczą metaheurystyk poszukiwania lokalnego, istotnych w kontekście tej pracy. Jednakże techniki RL wykorzystywane są także dla innych metaheurystyk, takich jak algorytmy genetyczne. Przykładem jest praca [6], gdzie Q-learning został wykorzystany do adaptacji prawdopodobieństwa krzyżowania dla przepływowego problemu szeregowania zadań, uzyskując lepsze wyniki niż wariant standardowy. Z kolei w pracy [17] wykorzystano głębokie sieci neuronowe do poprawy skuteczności algorytmu genetycznego dla problemu szeregowania linii produkcyjnych w przemyśle ciężkim.

Interesującym podejściem, wykorzystanym między innymi w pracach [18] oraz [16], jest nietypowe zastosowanie Q-learningu w algorytmach metaheurystycznych do wyboru metody modyfikacji rozwiązania w każdej iteracji. Zbiór tych modyfikacji (na przykład operatorów perturbacji) stanowi jednocześnie zarówno zbiór możliwych stanów, jak i zbiór dozwolonych akcji. Takie wykorzystanie RL dostarcza agentowi bardzo mało informacji o faktycznym stanie środowiska. Fakt ten, w połączeniu ze zmieniającym się charakterem procesu rozwiązywania problemu, tworzy unikalne wyzwania na polu zastosowania RL w badaniach operacyjnych. Pomimo że we wspomnianych pracach podejście to okazało się skuteczne, brakuje w literaturze szerszej dyskusji na temat jego własności i ograniczeń. Niniejsza praca stanowi wstęp do takich rozważań, na przykładzie problemu i metody jego rozwiązywania opisanej w [18].

2. Sformułowanie problemu

W pracy rozważamy hybrydowy (permutacyjny) przepływowy problem szeregowania zadań z efektem uczenia zaproponowanym przez Biskupa [3] (ang. Hybrid Flow Shop Scheduling Problem with Learning Effects, HFSSP-LE). Rozważany problem należy do klasy problemów NP-trudnych, ze względu na to, że podstawowy hybrydowy problem przepływowy (będący szczególnym przypadkiem HFSSP-LE) jest NP-trudny [5]. Poniżej prezentujemy opis problemu w zwięzłej formie. Pełny model matematyczny dostępny jest w pracy Zhang i in. [18]. Danych jest n zadań (detali), które muszą zostać wytworzone w systemie produkcyjnym składającym się z s etapów. Dla każdego etapu j przez $m_j \leqslant n$ oznaczamy liczbę identycznych stanowisk (przy czym $m_j > 1$ dla co najmniej jednego j), zaś przez x_j określamy czy stanowisko składa się z maszyny ($x_j = 0$) czy pracownika z efektem uczenia ($x_j = 1$). Innymi słowy, etap j składa się z m_j identycznych maszyn, albo m_j identycznych pracowników z efektem uczenia.

Przez $P_{i,j}$ oznaczamy podstawowy czas wykonania zadania i na etapie j. Rzeczywisty czas wykonania zadania $\hat{P}_{i,j}$ zależy od typu stanowiska. Dla stanowisk z maszynami $\hat{P}_{i,j} = P_{i,j}$. Dla stanowisk z pracownikami należy uwzględnić efekt uczenia pracownika, zaś rzeczywisty czas wykonywania zadania obliczany jest jako:

$$\hat{P}_{i,j} = [P_{i,j} \cdot q^{\tau}], \tag{1}$$

gdzie q jest pozycją zadania w kolejce zadań pracownika, $\tau \leqslant 0$ jest tzw. indeksem uczenia, zaś $[\cdot]$ oznacza zaokrąglenie. Przykładowo, dla $P_{i,j} = 40$ oraz $\tau = -0.2$, rzeczywisty czas $\hat{P}_{i,j}$ wynosi 40, gdy jest to pierwsze zadanie wykonywane przez pracownika ($q = 1$), 35 gdy jest to drugie zadanie wykonywane przez pracownika ($q = 2$), 32 gdy jest trzecie itd.

Przed rozpoczęciem wykonywania zadania i na etapie j należy przeprowadzić przezbrojenie, którego czas oznaczamy przez $ST_{i,i',j}$, gdzie i' jest poprzednim zadaniem wykonanym na stanowisku przed wykonaniem zadania i. Jeśli i jest pierwszym zadaniem wykonywanym na stanowisku, to $i' = i$.

Celem jest określenie takiego harmonogramu produkcji S^*, który zminimalizuje czas zakończenia wszystkich zadań $C_{\max}(S)$:

$$C_{\max}(S^*) = \min_{S \in \mathbb{S}} C_{\max}(S) = \min_{S \in \mathbb{S}} \max_{i,j} (S_{i,j} + \hat{P}_{i,j}), \tag{2}$$

gdzie $\mathbb{S}$ jest zbiorem dopuszczalnych harmonogramów, zaś $S_{i,j}$ jest czasem rozpoczęcia zadania i na etapie j. Dodatkowo aby być dopuszczalny, harmonogram musi spełniać kilka ograniczeń. Zadanie może być wykonywane naraz tylko na jednym stanowisku, a stanowisko może na raz wykonywać tylko jedno zadanie. Wykonywania zadań nie można przerywać. Zakładamy również nieograniczone bufory pomiędzy stanowiskami.

Ze względu na obecność harmonogramów niedopuszczalnych, posłużymy się, podobnie do pracy [18], reprezentacją harmonogramu S w postaci (pojedynczej) permutacji $\pi = (\pi(1), \pi(2), \ldots, \pi(n))$, gdzie $\pi(i)$ jest zadaniem, które zostanie wykonane w etapie pierwszym jako i-te. Kolejność wykonywania zadań w pozostałych etapach określa zasada FIFO (First-In First-Out) — w etapie j jako pierwsze zostanie uszeregowane zadanie, które pierwsze zakończyło wykonywanie w etapie $j - 1$. Ostatnią kwestią jest wybór stanowiska dla zadania, do czego posłuży zasada FAM (First Available Machine) — zadanie zostanie uszeregowane na maszynie, która będzie dostępna najwcześniej. W przypadku jeżeli opisane reguły nie wystarczają do ustalenia kolejności, decydujące są numery zadań i stanowisk. Przy takiej reprezentacji problem HFSSP-LE można zapisać jako:

$$C_{\max}(\pi^*) = \min_{\pi \in \Pi} C_{\max}(\pi), \tag{3}$$

gdzie π^* jest permutacją optymalną, zaś Π jest zbiorem wszystkich permutacji zbioru $\{1, 2, \ldots, n\}$. Każda $\pi \in \Pi$ jednoznacznie reprezentuje pewien dopuszczalny, dosunięty w lewo harmonogram S, który można uzyskać z π używając zasad FIFO i FAM. Wartość funkcji celu można wyznaczyć w czasie $O(sn \log n)$. Należy zaznaczyć, że wprowadzenie π jest uproszczeniem, w związku z czym nie ma gwarancji, że $C_{\max}(\pi^*) = C_{\max}(S^*)$. Jest to analogiczne do różnicy pomiędzy permutacyjnym problemem przepływowym, a niepermutacyjnym problemem przepływowym. Szerzej o wpływie FIFO/FAM i podobnych reprezentacji na złożoność przestrzeni i jakość rozwiązania można znaleźć we wspomnianej wcześniej pracy [5].

3. Metoda rozwiązywania

Rozważany problem HFSSP-LE jest NP-trudny, stąd był rozwiązywany w [18] za pomocą przybliżonego, a konkretnie hybrydowego algorytmu MRLM (z ang. *Meta-*

Algorytm 1: Faza 3 algorytmu MRLM

Input: Q, π, $T_{\max}$, γ i ω
Output: π

$s_0 \leftarrow$ Losowo wybierz stan początkowy;
$t \leftarrow 0$;
repeat
  $a_t \leftarrow$ Wybierz operator na podstawie Q i s_t;
  $\pi' \leftarrow$ Użyj operatora a_t na π;
  $\pi' \leftarrow$ Użyj LSO aby poprawić π';
  $r_t \leftarrow$ Wyznacz nagrodę;
  if *LosowaLiczba* $\sim \mathcal{U}(0,1) <$ *KryteriumAkceptacji*(π, π') **then**
    $\pi \leftarrow \pi'$;
  $s_{t+1} \leftarrow$ Określ następny stan;
  $\alpha_t \leftarrow$ Zmodyfikuj współczynnik uczenia;
  $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha_t\Big(r_t + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)\Big)$;
  $t \leftarrow t + 1$;
until *osiągnięto czas trwania* $T_{\max}$;

Reinforcement Learning-based Metaheuristic). Algorytm ten łączy cechy metaheurystyki oraz techniki opartej o RL. Proces rozwiązywania instancji problemu za pomocą MRLM można podzielić na 3 fazy:

1. Generowanie rozwiązania początkowego π za pomocą konstrukcyjnej heurystyki wzorowanej na algorytmie NEH [13].
2. Meta trening, mający na celu przede wszystkim inicjalizację tablicy Q, ale także potencjalnie prowadzący do poprawy rozwiązania π.
3. Poprawa rozwiązania π za pomocą metaheurystyki wzorowanej na symulowanym wyżarzaniu, nawigowanej przez mechanizm oparty o Q-learning.

Wyniki raportowane w [18] wskazują, że dla niewielkich instancji $n = 40$ oraz $s \in \{5, 10, 15, 20\}$ faza 2 może zostać pominięta. W ten sposób uzyskano lepsze jakościowo rozwiązania niż przy rozdzieleniu budżetu obliczeniowego po równo pomiędzy fazy 2 i 3. Stąd, z uwagi na skupienie się w niniejszej pracy na instancjach o ograniczonym rozmiarze, przybliżone zostaną jedynie fazy 1 i 3.

Faza 1 to konstrukcyjna heurystyka wzorowana na algorytmie NEH. Algorytm tworzy na początku losową sekwencję zadań. Następnie kolejno wstawia je do częściowego rozwiązania na pozycji pozwalającej na minimalizowanie $C_{\max}$. W przeciwieństwie do tradycyjnego NEH, algorytm ten wprowadza element losowości, co jest uzasadnione specyfiką fazy 2.

Pseudokod fazy 3 pokazano w Algorytmie 1, gdzie α jest współczynnikiem (tempem) uczenia, zaś γ opisuje wagę przyszłych nagród względem teraźniejszych (dla $\gamma = 1$ przyszła nagroda jest równoważna obecnej, dla $\gamma = 0$ istotna jest tylko nagroda obecna); z kolei ω jest parametrem jednego z operatorów opisanych poniżej. Faza zaczyna się od losowego ustalenia stanu początkowego s_0. Tablica Q o rozmiarze 3×3 (dla 3 możliwych akcji i stanów), może pochodzić z fazy 2, być inicjalizowana loso-

wo lub zerami. Następnie, wykonywane są kolejne iteracje algorytmu, aż do osiągnięcia kryterium stopu — wykorzystania czasowego budżetu obliczeniowego $T_{\max}$. Każda z iteracji $t = 0, 1, \ldots$, zaczyna się od wyboru akcji a_t na podstawie stanu s_t i tablicy Q. W MRLM decyzja jest zawsze podejmowana zachłannie, maksymalizując oczekiwaną nagrodę. Akcje odpowiadają zastosowaniu na π jednego z trzech operatorów: SO, IO, lub DCO, gdzie:

Zamień (SO) Zamienia w π miejscami dwa losowe zadania, w czasie $O(sn \log n)$.

Wstaw (IO) Wykonuje sekwencyjnie ω losowych ruchów typu wstaw, zwraca najlepsze z uzyskanych rozwiązań, w czasie $O(\omega \cdot sn \log n)$.

Destrukcji-Konstrukcji (DCO) Z rozwiązania π usuwane jest d zadań. Następnie, w tej samej kolejności w której występowały w oryginalnym rozwiązaniu, umieszczane są w częściowym rozwiązaniu na pozycjach minimalizujących $C_{\max}$, w czasie $O(dn \cdot sn \log n)$.

Następnie rozwiązanie to jest modyfikowane przez przeszukiwanie lokalne (LSO), uzyskując π'. W LSO każde z zadań, w losowej kolejności, umieszczane jest na każdej innej pozycji rozwiązania π. Jeżeli w wyniku takiej zamiany uzyskany $C_{\max}$ jest niższy, zamiana staje się permanentna. W sumie rozważane jest $n(n-1)$ rozwiązań, a więc złożoność czasowa LSO jest w $O(sn^3 \log n)$. Na podstawie różnicy wartości funkcji celu pomiędzy π, a π' określana jest nagroda r_t. Jeżeli nastąpiła poprawa $C_{\max}(\pi') < C_{\max}(\pi)$, to akcja jest nagradzana $r_t = 1$, w przeciwnym jest karana $r_t = -1$. W liniach 8 i 9 rozwiązanie π' zastępuje π zgodnie z kryterium akceptacji wykorzystywanym w symulowanym wyżarzaniu. W MRLM brak jasno wskazanego kryterium, w pracach cytowanych przez [18] prawdopodobieństwo akceptacji jest dane przez

$$\exp\left(\frac{C_{\max}(\pi) - C_{\max}(\pi')}{C_{\max}(\pi)} \cdot \frac{1}{1 - T/T_{\max}}\right), \tag{4}$$

gdzie T to czas który upłynął od rozpoczęcia działania algorytmu, a $T_{\max}$ to całkowity czas przeznaczony na obliczenia. W następnej kolejności określany jest kolejny stan s_{t+1} na podstawie akcji a_t i stanu s_t. W MRLM stan wybierany jest na podstawie turnieju, w którym dla dwóch losowo wybranych stanów s^1 i s^2, wybierany jest ten dla którego wartość $Q(s_t, s^y),\ y \in \{1, 2\}$ jest mniejsza. Dalej, ustalana jest wartość współczynnika uczenia, ze wzoru

$$\alpha = 1 - (0{,}9 \cdot T/T_{\max}). \tag{5}$$

Ostatecznie, na podstawie nowego stanu, akcji i nagrody, aktualizowana jest tablica Q.

4. Badania numeryczne

Celem eksperymentów było lepsze zrozumienie poszczególnych elementów algorytmu MRLM, tak aby wyciągnąć wnioski na temat wykorzystania Q-learningu w rozważanym kontekście. W badaniach jako algorytm odniesienia wykorzystano podejście MRLM z pracy [18], gdzie hiperparametry ustalono na $\gamma = 0{,}7$, $\omega = 10$ oraz $T_{\max} = 60 \cdot n \cdot s$ milisekund. Testy obliczeniowe zostały przeprowadzone z użyciem stacji roboczej wyposażonej w procesor AMD Ryzen Threadripper 3990X 2,9 z 64 fizycznymi rdzeniami oraz 64GB pamięci RAM; z systemem Ubuntu 22.04.4 LTS. Algorytmy zostały zaimplementowane z użyciem języka Julia w wersji 1.10.0. Instancje

Tabela 1

Wpływ wybranego zbioru operatorów na działanie algorytmu.

zbiory operatorów	śr. popr.	śr. liczba iter.	śr. wykorzystanie operatorów [%]
[SO, IO, DCO1]	106,78	3576,7	[52, 33, 15]
[SO, IO, DCO3]	107,69	3951,6	[42, 26, 31]
[SO, IO, DCO5]	106,25	3717,6	[35, 26, 38]
[SO, IO, DCO7]	104,14	3229,9	[33, 27, 40]
[SO, IO, DCO1, DCO3]	108,49	3914,3	[34, 25, 11, 30]
[SO, IO, DCO1, DCO3, DCO5]	108,26	3682,8	[22, 19, 10, 21, 27]
[SO, IO, DCO1, DCO3, DCO5, DCO7]	106,03	3628,9	[16, 16, 9, 17, 20, 21]

testowe zostały wygenerowane zgodnie z założeniami przedstawionymi w pracy [18]. Jako parametry wejściowe generatora przyjęto liczbę zadań, etapów, maszyn oraz ziarno. Zgodnie z opisem w Rozdziale 3, skupiono się na instancjach o $n \leqslant 50$. Rozważono następujące rozmiary instancji $(n \times s)$: 5×2, 10×2, 5×3, 10×3, 15×3, 20×3, 15×5, 20×5, 30×5, 20×7, 30×7, 40×7, 50×5, 30×10, 40×10 oraz 50×10. Łącznie wygenerowano 160 instancji, po 10 na każdy rozmiar.

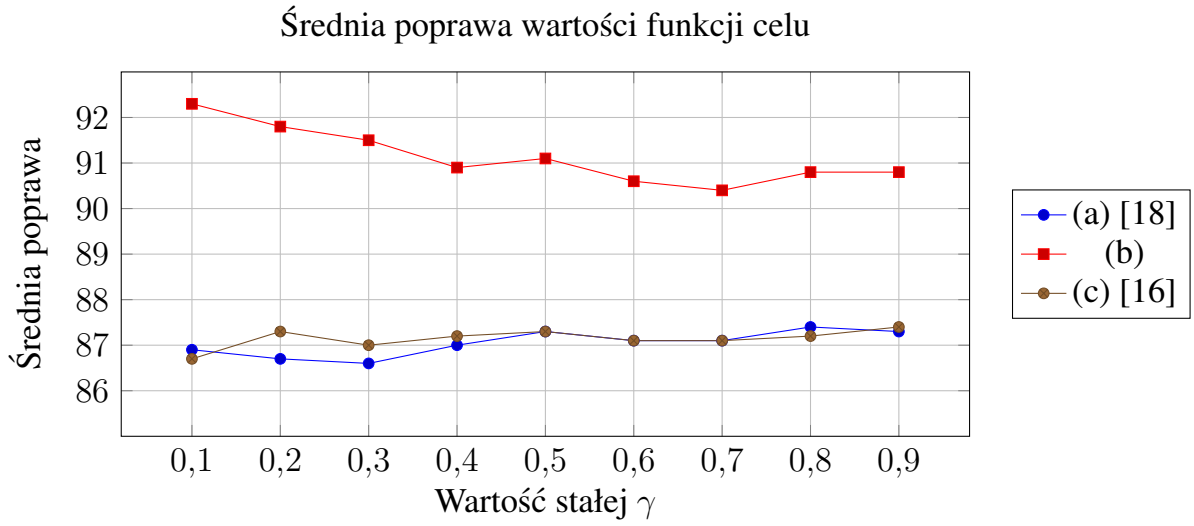
4.1. Konstrukcja portfolio operatorów

W pierwszej kolejności zbadano wpływ portfolio operatorów na uzyskiwane rezultaty. W pracy [18] nie podano użytej wartości parametru d dla operatora DCO. Stąd, zdecydowano się zbadać warianty operatora DCO: DCO1, DCO3, DCO5 oraz DCO7, dla odpowiednio: $d = 1, 3, 5, 7$. Skonstruowano następujące zbiory operatorów: [SO, IO, DCO1], [SO, IO, DCO3], [SO, IO, DCO5], [SO, IO, DCO7], [SO, IO, DCO1, DCO3], [SO, IO, DCO1, DCO3, DCO5], [SO, IO, DCO1, DCO3, DCO5, DCO7]. Pozostałe parametry algorytmu pozostawiono bez zmian. Algorytm dla każdego zbioru operatorów oraz dla każdej instancji uruchomiono 10-krotnie. Ponieważ operator DCO7 nie mógł zostać wykonany dla instancji z najmniejszą liczbą zadań $n = 5$, instancje te pominięto w badaniach w tym podrozdziale. Następnie obliczono dla każdej kombinacji parametrów proporcję wykorzystania operatorów, t.j. jak często wykorzystywany był każdy z nich oraz średnią poprawę wartości funkcji celu względem rozwiązania z fazy 1

$$\frac{1}{X}\sum_{i=1}^{X}\Big(C_{\max}(\pi_i^I) - C_{\max}(\pi_i)\Big), \tag{6}$$

gdzie X to liczba rozważanych instancji, π_i^I to rozwiązanie z 1 fazy algorytmu dla instancji i, zaś π_i to ostateczne rozwiązanie uzyskane przez algorytm. Rezultaty badania przedstawiono w tabeli 1.

Analizując wyniki można zauważyć, że operator DCO1 jest relatywnie rzadko wybierany. Jest to uzasadnione, ponieważ dla $d = 1$ następuje pojedyncza perturbacja, która ma i tak miejsce dalej w ramach LSO. Jeszcze mniej skuteczny okazał się DCO7, którego wprowadzenie prowadziło do zmniejszenia średniej liczby wykonanych iteracji, a co za tym idzie jakości rezultatów. Prawdopodobnie konieczne byłoby uwzględnienie nakładu obliczeniowego operatorów w nagrodzie, jak to zrobiono na przykład w [16]. Obserwacja wskazuje na potencjał metody do automatycznego identyfikowania i ignorowania nieefektywnych operatorów. Równie interesujący wydaje się efekt zwiększenia



Rys. 1. Wykres średniej poprawy wartości funkcji celu dla różnych wartości γ i wariantów (a), (b), (c) strategii ustalania α

liczby operatorów – ze względu na cechy charakterystyczne algorytmu, dokonywana jest eksploracja wszystkich możliwych akcji, co może przynieść korzyści w postaci większej poprawy jak dla kombinacji [SO, IO, DCO1, DCO3]. Tym niemniej, w dalszych badaniach wykorzystano głównie najlepszy wariant 3 operatorów ([SO, IO, DCO3]), ze względu na utrzymanie spójności z oryginalną pracą.

4.2. Dobór hiperparametrów Q-learningu

W drugim etapie zbadano wpływ parametrów γ oraz α na skuteczność algorytmu. Przetestowano 3 warianty modyfikacji α: (a) domyślna metoda ze wzoru (5), (b) stała $\alpha \in \{0{,}1,\ 0{,}2, \ldots, 0{,}9\}$ oraz (c) metoda z [16]:

$$(\alpha_{\text{initial}} + \alpha_{\text{final}})/2 - (\alpha_{\text{initial}} - \alpha_{\text{final}})/2 \cdot \cos(\pi \cdot (1 - T/T_{\max})), \tag{7}$$

gdzie odpowiednio $\alpha_{\text{initial}} = 0{,}1$ oraz $\alpha_{\text{final}} = 0{,}9$. Zbadano również różne wartości $\gamma \in \{0{,}1,\ 0{,}2, \ldots, 0{,}9\}$. Pozostałe parametry algorytmu pozostawiono bez zmian. Algorytm dla każdej kombinacji parametrów oraz dla każdej instancji uruchomiono 10-krotnie. Następnie obliczono dla każdej kombinacji parametrów średnią poprawę wartości funkcji celu względem rozwiązania z fazy 1. Wykres dla wszystkich trzech wariantów w funkcji γ przedstawiono na Rysunku 1. Ponadto wynik dla wariantu (b) pokazano na mapie ciepła na Rysunku 2. Relatywnie niższe wartości poprawy w porównaniu z poprzednim podrozdziałem są spowodowane uwzględnieniem instancji $n = 5$.

Rezultaty wskazują, że metody zmniejszające α w czasie są względnie mniej skuteczne niż odpowiednio dobrane stałe wartości. Obserwacja ta stoi w sprzeczności z ukonstytuowaną praktyką w dziedzinie RL. Może być to spowodowane przez fakt ciągle zmieniającej się specyfiki środowiska, czym rozważany przypadek odróżnia się od typowego. Tym samym może nie istnieć jedna optymalna polityka zadana przez Q dla całego procesu optymalizacji i tablica Q musi ulegać zmianom przez cały czas jego trwania. Podobnie, relatywnie lepsze rezultaty dla niskich wartości γ sugerują konieczność podejmowania decyzji biorących pod uwagę krótki horyzont czasowy.

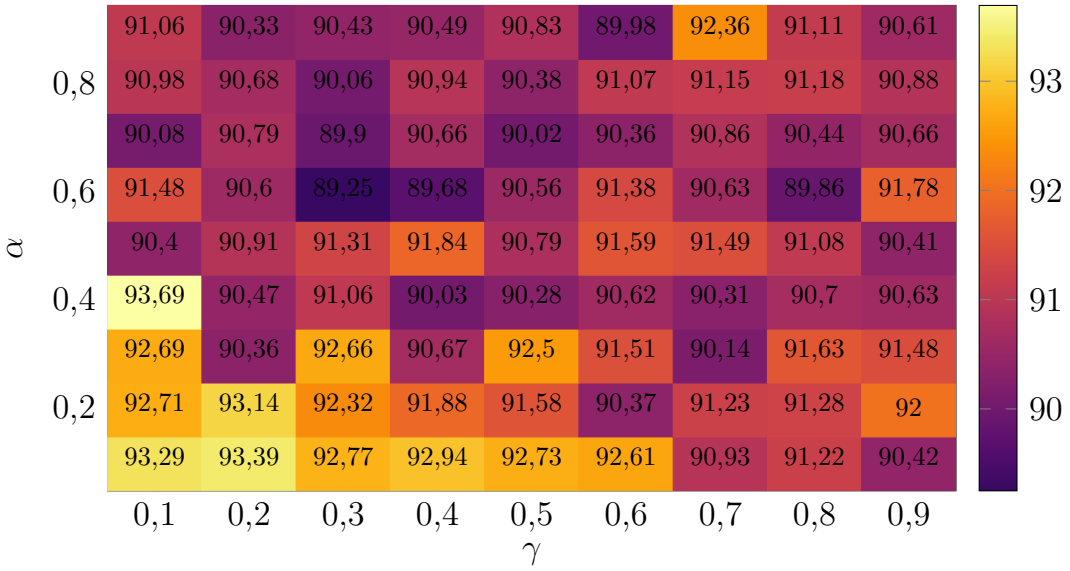
Tabela 2

Średnia poprawa dla różnych metod wyboru akcji i stanu

Operatory	Metoda wyboru akcji	Metoda wyboru stanu	Średnia poprawa
[SO, IO, DCO1, DCO3]	losowa	losowa	92,48
[SO, IO, DCO1, DCO3]	zachłanna	zachłanna	93,44
[SO, IO, DCO1, DCO3]	zachłanna	losowa	94,45
[SO, IO, DCO1, DCO3]	zachłanna	turniej	96,32
[SO, IO, DCO3]	losowa	losowa	91,91
[SO, IO, DCO3]	zachłanna	zachłanna	93,94
[SO, IO, DCO3]	zachłanna	losowa	94,38
[SO, IO, DCO3]	zachłanna	turniej	95,62

4.3. Metoda wyboru akcji i stanu

Kolejnym badanym elementem heurystyki był sposób wyboru akcji i stanu. Oryginalnie, w algorytmie MRLM zastosowano zachłanny wybór akcji i niedeterministyczną zmianę stanu opartą o turniej. Przeprowadzono następujące eksperymenty. Dla wyboru akcji, przetestowano 2 metody: losowa oraz zachłanna (wartość domyślna). Dla wyboru stanu, przetestowano 3 metody: losowa, zachłanna (wybór stanu odpowiadającego akcji z najwyższą wartością w tablicy Q, $s_{t+1} \leftarrow \arg\max_a Q(s_t, a)$) oraz turniej (wartość domyślna). Dodatkowo wykorzystano dwa najbardziej obiecujące zbiory operatorów. Wykorzystano stałe wartości $\alpha = 0{,}1$ i $\gamma = 0{,}1$ na podstawie wyników przedstawionych w podrozdziale 4.2., co odpowiada zauważalnemu trendowi poprawy jakości rozwiązań dla mniejszych wartości tych parametrów. Algorytm dla każdej kombinacji parametrów oraz dla każdej instancji uruchomiono 10-krotnie. Następnie obliczono dla każdej kombinacji parametrów średnią poprawę wartości funkcji celu względem rozwiązania z fazy 1. Rezultaty pokazano w tabeli 2.



Rys. 2. Mapa ciepła, obrazująca średnią poprawę wartości funkcji celu dla wariantu (b), dla różnych ustalonych wartości α i γ.

Tabela 3

Średnia poprawa dla różnych metod akceptacji rozwiązań niepoprawiających

Wariant	Nazwa	Wzór	Średnia poprawa
a)	domyślna	$\exp\left(\frac{C_{\max}(\pi)-C_{\max}(\pi')}{C_{\max}(\pi)} \cdot \frac{1}{1-T/T_{\max}}\right)$	95,23
b)	Boltzmann	$\exp\left(\frac{C_{\max}(\pi)-C_{\max}(\pi')}{C_{\max}(\pi)} \cdot \frac{1}{(1-T/T_{\max})^2}\right)$	97,76
c)	Cauchy	$\left(1+\frac{C_{\max}(\pi')-C_{\max}(\pi)}{C_{\max}(\pi)} \cdot \frac{1}{1-T/T_{\max}}\right)^{-1}$	95,32
d)	logarytmiczna	$\exp\left(\frac{C_{\max}(\pi)-C_{\max}(\pi')}{C_{\max}(\pi)} \cdot \frac{1}{\ln(T_{\max}+1)-\ln(T+1)}\right)$	95,71
e)	kwadratowa	$\exp\left(\frac{C_{\max}(\pi)-C_{\max}(\pi')}{C_{\max}(\pi)} \cdot \frac{T_{\max}^2}{(T_{\max}-T)^2}\right)$	95,89
f)	liniowa	$1-\frac{T}{T_{\max}}$	99,01

Rezultaty wskazują przewagę mechanizmu opartego o Q-learning nad losowym wyborem operatora. Jeżeli chodzi o metodę wyboru stanu, najlepszą okazała się domyślna, wykazując średnią poprawę powyżej metody losowej. Z kolei metoda zachłanna, kierująca się przeciwnym kryterium niż metoda domyślna (maksymalizacja, zamiast minimalizacji), okazała się gorsza od metody losowej. Sugeruje to na przewagę kierowania się minimalizacją, co w [18] było interpretowane jako działanie intensyfikujące eksplorację.

4.4. Metoda akceptacji

Ostatnim przebadanym aspektem algorytmu było kryterium akceptacji. Oprócz kryterium opisanego wzorem (4) i oznaczonego jako wariant a), wykorzystano również cztery inne metody akceptacji uwzględniające jakość akceptowanego rozwiązania. Są to kolejno: b) adaptujące kryterium akceptacji Boltzmanna, c) kryterium wykorzystujące rozkład Cauchy'ego, d) kryterium oparte o logarytmiczne zmniejszanie prawdopodobieństwa akceptacji oraz e) kryterium oparte o kwadratowe zmniejszanie prawdopodobieństwa. Dodatkowo uwzględniono kryterium f) z liniowym spadkiem prawdopodobieństwa akcentacji w oparciu o pozostały czas działania algorytmu. Wybór metod był inspirowany przez źródła literaturowe [1, 15]. Pozostałe parametry badania dobrane zostały w oparciu o najlepsze wyniki z wcześniejszych podrozdziałów: wybrano najbardziej obiecujący wariant trzech operatorów ([SO, IO, DCO3]); dobrano stałe wartości $\alpha = 0,1$ i $\gamma = 0,1$ oraz zastosowano zachłanną metodę wyboru akcji i turniejową metodę wyboru stanu. Pozostałe parametry pozostawiono bez zmian. Ponownie algorytm uruchamiano 10-krotnie i obliczono średnią poprawę. Wyniki przeprowadzonych eksperymentów oraz szczegóły kryteriów akceptacji zostały przedstawione w tabeli 3.

Analizując wyniki eksperymentów można zauważyć przewagę linowej metody akceptacji, która odróżnia się od pozostałych kryteriów ignorowaniem względnej różnicy jakości rozwiązań π oraz π'. Prawdopodobnie dla uzyskania lepszych rezultatów dla pozostałych kryteriów konieczne byłoby wprowadzenie współczynnika skalującego wspomnianą różnicę. Niemniej badania tego typu wykraczałyby poza zakres tej pracy,

skupiającej się na aspektach RL analizowanej metody. Drugie w kolejności skuteczności znalazło się kryterium Boltzmanna. Pozostałe 4 kryteria okazały się wyraźnie słabsze. Interesującym jest fakt, że domyślna metoda akceptacji stosowana przez MRLM uzyskała najgorszy wynik (wartość prawie 4% mniej w stosunku do metody liniowej i $2{,}66\%$ mniej niż adaptacyjne kryterium Boltzmanna). Zaobserwowane różnice podkreślają celowość przeprowadzenia dogłębnych badań nad technikami użytymi w MRLM, zarówno aby stworzyć skuteczniejszy algorytm, ale także aby lepiej zrozumieć ich własności.

5. Wnioski

W pracy przeanalizowano działanie zaproponowanej w literaturze metody MRLM, uczenia ze wzmocnieniem dla hybrydowego przepływowego problemu szeregowania zadań z efektem uczenia pracowników. Rozważana metoda wykorzystuje Q-learning do wyboru operatora perturbacji w każdej iteracji. W ramach badań własnych, część elementów algorytmu (np. wykorzystywane operatory perturbacji) zmodyfikowano, uzyskując poprawę otrzymywanych rozwiązań dla testowanych instancji w stosunku do oryginalnej pracy. Należy nadmienić, że wiele z zaproponowanych zmian poprawiających wyników zasadniczo różni się od typowych zaleceń stosowanych w Q-learningu do poprawy skuteczności. Aspekt ten wymaga dalszych badań.

Otrzymane wyniki wskazują na celowość dalszych badań. Dalsze planowane kierunki badawcze obejmują: (a) wartość nagrody, a w szczególności uwzględnienie nakładów obliczeniowych przy wyborze danego operatora (czego nie umożliwia obecna wersja, traktując lekkie i ciężkie operatory w sposób równoważny), (b) rozważenie innych operatorów, (c) poprawa działania funkcji akceptacji (d) rozważenie instancji o większym rozmiarze, z uwzględnioną fazą meta-uczenia oraz (e) rozważenie innych problemów optymalizacji dyskretnej. Przeprowadzone badania pokazują potencjał metody MRLM, będącej swoistym połączeniem sztucznej inteligencji oraz metaheurystyki lokalnego poszukiwania, zapewniającej unikalny sposób eksploracji przestrzeni problemu. Metoda ta, po dalszych modyfikacjach, zapewne z powodzeniem mogłaby znaleźć zastosowanie w ramach wyspecjalizowanego portfolia dedykowanych problemu dla rozważanego problemu szeregowania zadań.

LITERATURA

1. Aarts E., Korst J.:. *Simulated annealing and Boltzmann machines: a stochastic approach to combinatorial optimization and neural computing*,. John Wiley & Sons, Inc. 1989.
2. Bengio Y., Lodi A.,, Prouvost A.: Machine learning for combinatorial optimization: A methodological tour d'horizon, *European Journal of Operational Research*, 4 2021, vol. 290 p. 405–421.
3. Biskup D. Single-machine scheduling with learning considerations, *European Journal of Operational Research*, vol. 115(1) p. 173–178.
4. Fakoor R., Chaudhari P., Soatto S.,, Smola A. J.:. Meta-Q-learning 2020.
5. Fernandez-Viagas V., Perez-Gonzalez P.,, Framinan J. M.: Efficiency of the solu-

tion representations for the hybrid flow shop scheduling problem with makespan objective, *Computers & Operations Research*, 2019, vol. 109 p. 77–88.

6. Gao X., Yang S.,, Li L.: Optimization of flow shop scheduling based on genetic algorithm with reinforcement learning, *Journal of Physics: Conference Series*, apr 2022, vol. 2258(1) p. 012019.
7. Gnatowski A., Rudy J.,, Idzikowski R.: Scheduling disjoint setups in a single-server permutation flow shop manufacturing process, *Processes*, 2022, vol. 10(9) p. 1837.
8. Gu X., Zhao S.,, Wang Y.: Reinforcement learning enhanced multi-neighborhood tabu search for the max-mean dispersion problem, *Discrete Optimization*, 2022, vol. 44 p. 100625. Quadratic Combinatorial Optimization Problems.
9. Karimi-Mamaghan M., Mohammadi M., Meyer P., Karimi-Mamaghan A. M.,, Talbi E.-G.: Machine learning at the service of meta-heuristics for solving combinatorial optimization problems: A state-of-the-art, *European Journal of Operational Research*, 1 2022, vol. 296 p. 393–422.
10. Karimi-Mamaghan M., Mohammadi M., Pasdeloup B.,, Meyer P.: Learning to select operators in meta-heuristics: An integration of q-learning into the iterated greedy algorithm for the permutation flowshop scheduling problem, *European Journal of Operational Research*, 2023, vol. 304(3) p. 1296–1330.
11. Kosanoglu F., Atmis M.,, Turan H. H.: A deep reinforcement learning assisted simulated annealing algorithm for a maintenance planning problem, *Annals of Operations Research*, 3 2022, vol.
12. Lee J.-H., Kim H.-J.: Reinforcement learning for robotic flow shop scheduling with processing time variations, *International journal of production research*, 2022, vol. 60(7) p. 2346–2368.
13. Nawaz M., Enscore Jr E. E.,, Ham I.: A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem, *Omega*, 1983, vol. 11(1) p. 91–95.
14. Nowicki E., Smutnicki C.: A fast taboo search algorithm for the job shop problem, *Management science*, 1996, vol. 42(6) p. 797–813.
15. Orsila H., Salminen E.,, Hämäläinen T. D.: Best practices for simulated annealing in multiprocessor task distribution problems, *Simulated annealing*, 2008, vol. 16 p. 321–342.
16. Qu C., Gai W., Zhong M.,, Zhang J.: A novel reinforcement learning based grey wolf optimizer algorithm for unmanned aerial vehicles (uavs) path planning, *Applied Soft Computing*, 4 2020, vol. 89 p. 106099.
17. Zeng D., Zhan J., Peng W.,, Zeng Z.: Evolutionary job scheduling with optimized population by deep reinforcement learning, *Engineering Optimization*, 2023, vol. 55(3) p. 494–509.
18. Zhang Z., Shao Z., Shao W., Chen J.,, Pi D.: Mrlm: A meta-reinforcement learning-based metaheuristic for hybrid flow-shop scheduling problem with learning and forgetting effects, *Swarm and Evolutionary Computation*, 3 2024, vol. 85 p. 101479.