Wydział Informatyki i Telekomunikacji
Politechnika Wrocławska

Rozprawa doktorska

Algorytmy szeregowania zadań z nieustalonymi terminami gotowości na maszynach dowolnych

mgr inż. Mirosław Ławrynowicz

Promotor:
prof. dr hab. inż. Jerzy Józefczyk

Wrocław 2023

Składam serdeczne podziękowania mojemu promotorowi prof. dr. hab. inż. Jerzemu Józefczykowi za wyrozumiałość, cierpliwość oraz pomoc przy realizacji niniejszej rozprawy.

Streszczenie

Zasadniczym celem rozprawy jest opracowanie i przebadanie algorytmów rozwiązania problemów szeregowania zadań na maszynach dowolnych z nieustalonymi terminami gotowości w celu minimalizacji długości uszeregowania. Nieustalony charakter terminu gotowości może wynikać z faktu, że jest on konsekwencją poprzednich decyzji albo z braku znajomości jego precyzyjnej wartości. W pierwszym przypadku odniesieniem jest łączny problem szeregowania zadań i rozmieszczenia maszyn (ang. Scheduling and Location, w skrócie ScheLoc), gdzie terminy gotowości dla zadań są związane z czasami przemieszczania się zadań do podlegających wyborowi miejsc rozmieszczenia maszyn. Badana w rozprawie nieprecyzyjność terminów gotowości jest reprezentowana przez dane przedziały niepewności, a jakość szeregowania jest oparta na funkcji maksymalnego żalu, co prowadzi do zagadnień optymalizacji odpornej. W konsekwencji sformułowano i rozpatrzono dwa rodzaje problemów optymalizacyjnych. Po pierwsze, przebadano problem deterministyczny z kryterium długości uszeregowania i precyzyjnie określonymi terminami gotowości dla zadań. Następnie rozważono niedeterministyczny problem optymalizacji odpornej z kryterium maksymalnego żalu opartym na długości uszeregowania oraz z przedziałowymi terminami gotowości.

Deterministyczny problem szeregowania sformułowano przy założeniu, że precyzyjnie określone terminy gotowości zadań zależą od maszyn. Opracowano pięć metod oszacowania wartości kryterium oraz wskazano szczególne przypadki, które można rozwiązać optymalnie w czasie wielomianowym lub pseudowielomianowym. Algorytmy rozwiązania problemu wykorzystują zachłanną strategię podejmowania decyzji z planowaniem długoterminowym dla decyzji o takiej samej jakości, metaheurystykę Simulated Annealing lub dekompozycję z przeglądem wyczerpującym. Stosując badania statystyczne wykazano, że podejście zachłanne z planowaniem długoterminowym dla decyzji o takiej samej jakości gwarantuje statystycznie najmniejszą długość uszeregowania spośród opracowanych algorytmów.

Rozróżniono dwa przypadki problemu optymalizacji odpornej, w których przedziały niepewności dla terminów gotowości są określone tylko dla zadań lub mają szerszą interpretację i zależą również od maszyn. Analitycznie wykazano różnicę w interpretacji kryterium maksymalnego żalu dla obydwu przypadków. Istotną kwestią jest ograniczenie liczby przeglądanych scenariuszy, tak aby efektywnie oszacować wartość kryterium maksymalnego żalu. Udowodniono, że scenariusz skrajny i scenariusz pośredni mogą jednocześnie reprezentować najgorszy scenariusz. Opracowano algorytmy rozwiązania

problemów optymalizacji odpornej, wykorzystujące zachłanną strategię podejmowania decyzji, dekompozycję kryterium maksymalnego żalu, metaheurystykę Tabu Search lub determinizację problemu. Stosując badania statystyczne wykazano, że algorytm zachłanny gwarantuje statystycznie najmniejszą wartość oszacowanego maksymalnego żalu, spośród opracowanych algorytmów, dla obydwu rozważanych przypadków problemów optymalizacji odpornej.

Rezultaty prac nad problemami szeregowania wykorzystano do rozwiązania złożonego problemu ScheLoc z kryterium długości uszeregowania i maszynami dowolnymi. W tym celu zaadaptowano algorytm genetyczny oraz algorytm oparty na metaheurystyce Simulated Annealing. Zbadano wpływ optymalizacji dwuetapowej (zwanej podejściem sekwencyjnym) na długość uszeregowania w problemie Scheloc poprzez porównanie z rezultatami optymalizacji łącznej (zwanej podejściem holistycznym lub systemowym). W podejściu dwuetapowym rozmieszczenie maszyn jest realizowane w oparciu o rozwiązania problemów m-median oraz m-center. Oryginalnym elementem rozprawy jest wykorzystanie algorytmów opracowanych dla niedeterministycznych problemów szeregowania do rozwiązania deterministycznego problemu ScheLoc. Wskazano przypadki problemu ScheLoc, które można rozwiązać efektywniej, zakładając nieprecyzyjność terminów gotowości zadań.

W ostatniej części rozprawy omówiono obszary potencjalnych zastosowań opracowanych algorytmów. Zaprezentowano możliwość wykorzystania algorytmów opracowanych dla niedeterministycznych problemów szeregowania w planowaniu produkcji obwodów drukowanych PCB (ang. Printed Circuit Board) wraz z alokacją zasobów. Dodatkowo przedyskutowano teoretyczny problem harmonogramowania realizacji naukowych badań symulacyjnych.

Summary

The fundamental aim of this dissertation is the development and study of algorithms for solving the job scheduling problems with indeterminate release dates on unrelated machines. The notion of „indeterminateness" may arise from the fact that a job's release date depends on the decisions, or it is due to the lack of knowledge of its exact value. In the first case, the reference is an integrated problem of machine location and job scheduling (Scheduling and Location, in short ScheLoc), where job's release dates are related to the time required to move each job to selectable machine locations. The considered imprecision of release dates is represented by well-defined uncertainty intervals, and the quality of scheduling is based on a regret function, leading to robust optimization problem. In consequence, the two types of optimization problems are taken into account. Firstly, a deterministic problem with the makespan criterion and precisely defined release dates of jobs is studied. Next, a non-deterministic (robust) problem with the minmax regret criterion for the makespan and interval (uncertain) release dates is investigated.

The deterministic scheduling problem is formulated under the assumption that the release dates are machine-dependent. The five methods for assessing the makespan value are developed and special cases, which can be solved optimally or in pseudopolynomial time, are identified. The algorithms for solving the underlying scheduling problem use a greedy decision-making strategy with long-term planning for the same quality decisions, a semi-random search (an algorithm based on Simulated Annealing metaheuristic) and a decomposition with exhaustive search. The statistical tests showed that a greedy algorithm guarantees statistically the lowest makespan value in comparison with the other methods.

The two cases of the robust optimization problem, in which the uncertainty intervals of release dates are job-dependent or machine-dependent, are distinguished. The difference in the interpretation of the maximum regret criterion for both cases is proved analytically. It is important to limit the number of scenarios in order to efficiently assess the minmax regret criterion value. It is shown that an extreme scenario and an intermediate scenario can simultaneously represent the worst-case scenario. Algorithms for solving the non-deterministic problems use a greedy decision-making strategy, a decomposition of the robust criterion, a semi-random search (an algorithm based on Tabu Search metaheuristic) or a reformulation to the deterministic case. The statistical showed that the greedy algorithm

guarantees statistically the lowest value of estimated maximum regret value for both robust optimization problems.

The results of research on scheduling problems are used to solve the ScheLoc problem with the makespan criterion and unrelated machines. For this purpose, a genetic algorithm and a metaheuristic-based algorithm (Simulated Annealing) are adapted. The impact of a two-stage optimization (called the sequential approach) on the makespan criterion is investigated by comparing it with the results of a joint optimization (called the holistic approach or the systemic approach). In the two-stage optimization, m-median and m-center problems are used to select the machine locations. A distinctive feature of this dissertation is the use of algorithms developed for the non-deterministic scheduling problems to solve the deterministic ScheLoc problem. It is shown that there exist instances of the ScheLoc problem that can be solved more efficiently assuming interval release dates of jobs.

The final section of this dissertation discusses application areas of the developed algorithms. It is shown how to apply the algorithms developed for the non-deterministic scheduling problems to production planning of Printed Circuit Boards (PCBs) with resource allocation. In addition, a theoretical example of scheduling scientific simulation studies is also discussed in detail.

Spis treści

Podstawowe oznaczenia

Parametry problemów

J – zbiór zadań
j – indeks zadania, $j \in$ J (również t, l, g, h)
M – zbiór maszyn
i – indeks maszyny, $i \in$ M (również s)
k – indeks pozycji w sekwencji (również d)
n – liczba zadań, $|\mathrm{J}| = n$
n_i – liczba zadań przypisanych do maszyny i (długość sekwencji przypisanej do maszyny i)
m – liczba maszyn, $|\mathrm{M}| = m$
i_m – indeks najdłużej pracującej maszyny
w – liczba dostępnych lokalizacji dla maszyn
r – macierz reprezentująca terminy gotowości wszystkich zadań
r_j – termin gotowości zadania j
$r_{i,j}$ – termin gotowości zadania j na maszynie i
p – macierz reprezentująca czasy przetwarzania zadań
$p_{i,j}$ – czas przetwarzania zadania j na maszynie i
L – zbiór współrzędnych lokalizacji zadań
$l_j = (l_{j,1}, l_{j,2}) \in \mathbb{R}^2$ – współrzędne lokalizacji zadania j, $l_j \in$ L
$\tilde{\mathrm{L}}$ – zbiór współrzędnych dostępnych lokalizacji dla maszyn
$\tilde{l}_g = (\tilde{l}_{g,1}, \tilde{l}_{g,2}) \in \mathbb{R}^2$ – współrzędne lokalizacji, w której można ulokować maszynę, $\tilde{l}_g \in \tilde{\mathrm{L}}$
U – zbiór wszystkich scenariuszy dla problemu $Rm|r_{i,j}^- \le r_{i,j} \le r_{i,j}^+|Reg(C_{\max})$
u – macierz reprezentująca scenariusz, $u \in$ U
$\hat{u}$ – najgorszy scenariusz, $\hat{u} \in$ U
$u_{i,j}$ – przedział niepewności dla terminu gotowości zadania j na maszynie i, $u_{i,j} = [r_{i,j}^-, r_{i,j}^+]$
$\tilde{\mathrm{U}}$ – zbiór wszystkich scenariuszy skrajnych dla problemu $Rm|r_{i,j}^- \le r_{i,j} \le r_{i,j}^+|Reg(C_{\max})$, $\tilde{\mathrm{U}} \subseteq$ U
$r_{i,j}^z$ – granica przedziału niepewności dla terminu gotowości zadania j na maszynie i, $z \in \{-, +\}$
$\tilde{u}^{i,j}$ – scenariusz skrajny (macierz), $\tilde{u}^{i,j} \in \tilde{\mathrm{U}}$
$\bar{\mathrm{U}}$ – zbiór scenariuszy skrajnych dla problemu $Rm|r_j^- \le r_j \le r_j^+|Reg(C_{\max})$, $\bar{\mathrm{U}} \subseteq \tilde{\mathrm{U}}$
$\bar{u}^j$ – scenariusz skrajny (wektor), $\bar{u}^j \in \bar{\mathrm{U}}$
u_j – przedział niepewności dla terminu gotowości zadania j, $\bar{u}_j = [r_j^-, r_j^+]$
r_j^z – granica przedziału niepewności dla terminu gotowości zadania j, $z \in \{-, +\}$

Zmienne optymalizacyjne

x – uszeregowanie dla deterministycznego problemu $Rm|r_{i,j}|C_{\max}$
$x_{i,k,j}$ – binarna decyzja o przypisaniu zadania j na pozycji k w sekwencji przypisanej do maszyny i
$\tilde{x}$ – uszeregowanie dla niedeterministycznego problemu $Rm|r_{i,j}^- \le r_{i,j} \le r_{i,j}^+|Reg(C_{\max})$
$\bar{x}$ – uszeregowanie dla niedeterministycznego problemu $Rm|r_j^- \le r_j \le r_j^+|Reg(C_{\max})$
y – macierz binarna reprezentująca decyzje o wyborze lokalizacji dla maszyn
$y_{g,i}$ – binarna decyzja o przypisaniu maszyny i do lokalizacji $\tilde{l}_g$

Kryteria oceny

$C_{\max}(x; \mathrm{S})$ – długość uszeregowania dla problemu $Rm|r_{i,j}|C_{\max}$, $\mathrm{S} = \{p, r\}$

$C_{\max}(\tilde{x}, u; p)$ – długość uszeregowania dla problemu $Rm|r_{i,j}^{-} \le r_{i,j} \le r_{i,j}^{+}|Reg(C_{\max})$

$C_{\max}(\bar{x}, \bar{u}; p)$ – długość uszeregowania dla problemu $Rm|r_j^{-} \le r_j \le r_j^{+}|Reg(C_{\max})$

$C_{\max}(x, y; \mathrm{S}_l)$ – długość uszeregowania dla problemu ScheLoc, $\mathrm{S}_l = \{\tilde{\mathrm{L}}, \mathrm{L}, p, r^{\mathrm{SL}}\}$

$Q(\tilde{x}, u; p)$ – wartość żalu przy scenariuszu u dla problemu $Rm|r_{i,j}^{-} \le r_{i,j} \le r_{i,j}^{+}|Reg(C_{\max})$

$Z(\tilde{x})$ – wartość maksymalnego żalu dla problemu $Rm|r_{i,j}^{-} \le r_{i,j} \le r_{i,j}^{+}|Reg(C_{\max})$

$\tilde{Z}(\tilde{x})$ – oszacowana wartość maksymalnego żalu dla problemu $Rm|r_{i,j}^{-} \le r_{i,j} \le r_{i,j}^{+}|Reg(C_{\max})$

$\hat{Z}(\bar{x})$ – wartość maksymalnego żalu dla problemu $Rm|r_j^{-} \le r_j \le r_j^{+}|Reg(C_{\max})$

$\bar{Z}(\bar{x})$ – oszacowana wartość maksymalnego żalu dla problemu $Rm|r_j^{-} \le r_j \le r_j^{+}|Reg(C_{\max})$

Tabela 1. Zestawienie opracowanych algorytmów

Problem optymalizacyjny	Akronimy algorytmów rozwiązania
$Rm\|r_{i,j}\|C_{\max}$	1. D_Brute-Force (BF) – przegląd wyczerpujący z dekompozycją 2. S_Annealing (SA) – algorytm oparty na metaheurystyce Simulated Annealing 3. Greedy (GD) – algorytm konstrukcyjny, zachłanny
$Rm\|r_{i,j}^{-} \le r_{i,j} \le r_{i,j}^{+}\|Reg(C_{\max})$	1. D_Regret (DR) – algorytm konstrukcyjny wykorzystujący dekompozycję 2. G_Regret (GR) – algorytm konstrukcyjny, zachłanny 3. T_Search (TS) – algorytm oparty na metaheurystyce Tabu Search
$Rm\|r_j^{-} \le r_j \le r_j^{+}\|Reg(C_{\max})$	1. S_Regret (SR) – algorytm konstrukcyjny z determinizacją 2. algorytmy DR i GR
ScheLoc – podejście łączne	1. Genetic (GN) – algorytm genetyczny 2. Hybrid (HS) – połączenie algorytmów SA i GD 3. ExGreedy(EG)–połączenie przeglądu wyczerpującego EX z algorytmem GD
ScheLoc – podejście sekwencyjne	1. EX+GD – połączenie przeglądu wyczerpującego EX z algorytmem GD 2. S_Annealing_Loc (SA_L)+GD – połączenie algorytmów SA_L (algorytm opracowany dla podproblemu rozmieszczenia maszyn; oparty na metaheurystyce Simulated Annealing) i GD (z przeglądem zupełnym)
ScheLoc – podejście niepewne	1. T+EX, T∈{DR,GR,SR} – dostosowanie, wykorzystując przegląd wyczerpujący EX, uszeregowań uzyskanych przez algorytmy DR, GR i SR do ewaluacji w problemie ScheLoc

1. Wstęp

1.1. Wprowadzenie do tematyki pracy

Problematyka szeregowania zadań jest rozwijana od wielu dekad i stanowi ważną część badań operacyjnych, a także podstaw podejmowania i wspomagania decyzji w różnych zastosowaniach, np. w systemach produkcyjnych, usługowych, transportowych lub obliczeniowych. Rozwój metod i algorytmów optymalizacji oraz wzrost możliwości obliczeniowych współczesnych środków informatyki sprawiają, że jest możliwe badanie i rozwijanie trudniejszych problemów szeregowania zadań, ale bliższych zastosowaniom praktycznym. Jednym z aspektów tego typu badań jest rozpatrywanie problemów złożonych, w których należy łącznie podejmować wiele decyzji, a szeregowanie zadań może być tylko częścią takiego złożonego problemu. Innym aspektem jest uwzględnianie braku determinizmu danych zagadnienia szeregowania zadań, co jest często spotykane w zastosowaniach praktycznych. Wspomniane dwie cechy, czyli złożoność problemów oraz brak determinizmu danych w odniesieniu do wybranego zagadnienia szeregowania zadań, są podstawowymi wyróżnikami tej pracy.

Według Pinedo (2012), szeregowanie zadań jest procesem podejmowania decyzji o alokacji zasobów wymaganych do przetworzenia zadań, które należy uszeregować względem wybranego kryterium lub wielu kryteriów. W niniejszej pracy rozwiązanie problemu szeregowania jest nazywane „uszeregowaniem” lub „harmonogramem” oraz ograniczono się wyłącznie do zagadnień optymalnego podejmowania decyzji (w skrócie: do zagadnień optymalizacji). W rozprawie doktorskiej skoncentrowano się na stosunkowo nowym zagadnieniu łącznego problemu rozmieszczenia maszyn i szeregowania zadań, który w literaturze funkcjonuje pod nazwą ScheLoc (ang. Scheduling and Location). Przykładową i jedną z pierwszych prac ilustrujących takie podejście jest Hennes & Hamacher (2002), gdzie zdefiniowano problem szeregowania zadań, których terminy gotowości zależą od rozmieszczenia (lokalizacji) wybranego dla jednej maszyny. W tym przypadku położenie maszyny jest decyzją strategiczną, ponieważ wpływa bezpośrednio na terminy gotowości zadań i nie może być korygowane samym uszeregowaniem. Analogiczny problem dla co najmniej dwóch identycznych maszyn został sformułowany między innymi przez Rajabzadeh et al. (2016) i Heßler & Deghdak (2017). Wówczas alokacja zasobów, wymieniona w przytoczonej definicji szeregowania, jest wynikiem rozwiązania łącznego problemu rozmieszczenia maszyn i szeregowania zadań.

W problemie ScheLoc terminy gotowości zadań wynikają z decyzji o rozmieszczeniu maszyn, ponieważ dowolne zadanie jest gotowe do przetwarzania na maszynie w momencie przemieszczenia się do jej wskazanej lokalizacji. Castillo-Salazar, Landa-Silva & Qu (2016) podali przegląd prac łączących podproblemy przemieszczania się osób do wyznaczonych miejsc oraz harmonogramowania zadań, które muszą wykonane we wskazanych miejscach, względem arbitralnie wybranego kryterium. Natomiast podstawy teoretyczne dla problemu, w którym to maszyny mogą się przemieszczać w środowisku przemysłowym podał Józefczyk (2001). Badania literaturowe nad złożonymi problemami optymalizacyjnymi, łączącymi podproblemy szeregowania zadań i alokacji zasobów (wybór ścieżek w grafie, przypisanie dźwigów do statków), wpływających na terminy gotowości zadań, obejmują:

1. Wprowadzanie dodatkowych ograniczeń. Paraskevopoulos et al. (2019) wykonali przegląd publikacji podejmujących łączne problemy szeregowania i marszrutyzacji z ograniczonymi zasobami (np. wymuszenie ograniczeń wynikających z problemów logistycznych, brakiem zasobów w postaci maszyn lub osób mających kompetencje do wykonywania założonych zadań itp.).
2. Integrację dodatkowych podproblemów. Cappanera, & Scutellà (2015) zintegrowali podproblem przypisania pracowników opieki medycznej o odpowiednich kompetencjach z podproblemem wyznaczania tras dojazdów do odbiorców usług oraz podproblemem planowania kolejności realizacji usług.
3. Projektowanie rozwiązań dla szczególnych przypadków zastosowań. Bierwirth & Meisel (2010) podali przegląd literatury dotyczącej alokacji zasobów i harmonogramowaniu oraz ich użycia w problemach planowania wykorzystania suwnic nabrzeżnych.
4. Uwzględnianie niepewności danych. Özarık et al. (2021) sformułowali problem dystrybucji i harmonogramowania dostaw produktów, w którym uwzględnili niepewność dotyczącą obecności odbiorcy pod wskazanym adresem w momencie dotarcia kuriera.
5. Opracowywanie nowych algorytmów dla znanych problemów (np. Markowski & Józefczyk, 2008; Jamili, 2017).

Cechą każdego z wymienionych wyżej problemów optymalizacji łącznej jest zależność decyzji w podproblemach, tzn. decyzje podjęte w jednym podproblemie wpływają na jakość lub przestrzeń dopuszczalnych decyzji w pozostałych podproblemach.

Rozważane złożone problemy optymalizacyjne można również sklasyfikować ze względu na sposób sformułowania funkcji celu:

1. Problem optymalizacyjny z jednym kryterium, którego wartość uwzględnia specyfikę problemu dominującego. Lyu et al. (2019) zdefiniowali łączny problem szeregowania oraz marszrutyzacji zadań w elastycznym systemie produkcyjnym, oceniając jakość rozwiązania przez pryzmat długości uszeregowania.
2. Problem optymalizacyjny z jednym kryterium, którego wartość uwzględnia specyfikę wszystkich problemów składowych. Chen & Vairaktarakis (2005) zdefiniowali łączny problem szeregowania zadań oraz dystrybucji efektów ich realizacji. Jako obszar zastosowań wskazano usługi gastronomiczne (przygotowanie posiłków w restauracji oraz ich dostarczenie do klienta) lub przemysł (wyprodukowanie i dystrybucja części). Celem jest minimalizacja funkcji celu, której postać jest wynikiem skalaryzacji dwóch cząstkowych kryteriów oceny: jakości obsługi klienta (wyrażonej w klasycznych miarach jakości uszeregowania: najdłuższa długość uszeregowania lub średnia długość uszeregowania) oraz kosztu dystrybucji.
3. Problem optymalizacyjny zdekomponowany do sekwencji zależnych od siebie podproblemów. Wang et al. (2020) sformułowali dwuetapowy problem alokacji zasobów i szeregowania zadań. Praca porusza temat administrowania kosztami funkcjonowania placówki medycznej. Celem jest z minimalizacja kosztów utrzymania personelu medycznego i sal operacyjnych dla planowanych zabiegów (pierwszy etap) oraz minimalizacji kosztów nadgodzin wynikających z harmonogramu pracy ułożonego dla przydzielonych zasobów finansowych (drugi etap).
4. Wielokryterialny problem optymalizacyjny. Siddiqui & Verma (2015) zaproponowali dwukryterialny problem planowania tras tankowców i budowania harmonogramów dostaw ropy w taki sposób, aby minimalizować ryzyko i koszt transportu jednocześnie zaspokajając popyt na surowiec.

Modelowanie rzeczywistości w praktyce wymaga przyjmowania założeń o pewnym stopniu niepewności zdefiniowanych parametrów. Pod pojęciem niepewności rozumiemy nieprecyzyjnie określoną wartość parametru, która może być reprezentowana na przykład przez przedział lub zbiór wartości oraz dodatkowo opisywana przez rozkład prawdopodobieństwa albo funkcję przynależności. Problemy zakładające niepewności

parametrów wejściowych nazywamy niedeterministycznymi a proces ich rozwiązywania nazywamy optymalizacją w warunkach niepewności (w szczególności optymalizacją odporną, która jest rozpatrywana w rozprawie). Szczegółowa analiza teoretyczna oraz projektowanie algorytmów dla takich problemów pozwalają decydentowi uzyskać rozwiązanie, które toleruje, w założonym zakresie, ryzyko związane z jakością podjętych decyzji. Prace nad optymalizacją w warunkach niepewności są prowadzone regularnie od końca lat sześćdziesiątych (np. Gupta & Rosenhead, 1968; Rosenhead, Elton & Gupta, 1972; Mulvey, Vanderbei & Zenios, 1995; Daniels & Kouvelis, 1995; Kouvelis & Yu 2013; Gabrel, Murat & Thiele, 2014; Keith & Ahner, 2021). Zastosowanie koncepcji niepewności w problemach optymalizacyjnych, integrujących co najmniej dwa różne podproblemy, umożliwia rozpatrywanie zaawansowanych przypadków optymalnego podejmowania decyzji, które oddają rzeczywistość znacznie bardziej precyzyjnie niż ich odpowiedniki implementujące podejście deterministyczne. Literatura na temat niedeterministycznych problemów optymalizacji łącznej nie jest tak obszerna, jak dla ich deterministycznych odpowiedników. Jednak w ostatnich latach szczególnie widoczna jest intensyfikacja prac nad omawianym zagadnieniem. Na przykład Rahbari et al. (2019) sformułowali łączny problem wyznaczania tras dostaw produktów o krótkim terminie ważności oraz szeregowania ruchu (wejściowego i wyjściowego) ciężarówek w centrach dystrybucji towaru. Przyjęto niepewne terminy ważności produktów oraz momenty opuszczenia centrum dystrybucji przez ciężarówki. Rodrigues & Agra (2021) zdefiniowali problem przypisania miejsc do statków przy nabrzeżu, gdy ich terminy przybycia są niesprecyzowane oraz, w następnym etapie, przydzielają żurawie portowe i projektują harmonogramy obsługi statków. Łączny problem wyznaczania tras dojazdów opiekunów medycznych oraz harmonogramowania ich pracy przy założeniu niepewnych czasów realizacji usług medycznych sformułowali Shahnejat-Bushehri et al. (2021).

Genezą szczegółowych zagadnień rozpatrywanych w rozprawie jest rozwój wspomnianego już zagadnienia ScheLoc (Hennes & Hamacher, 2002; Heßler & Deghdak, 2017) oraz jego podproblemów. W niniejszej rozprawie przeprowadzono szczegółową analizę problemu ScheLoc w wersji z kryterium długości uszeregowania oraz z maszynami dowolnymi. Obszerna część badań została poświęcona analizie problemu szeregowania zadań z terminami gotowości i kryterium długości uszeregowania (wersja deterministyczna) oraz analizie problemu szeregowania zadań z przedziałowymi terminami gotowości i kryterium maksymalnego żalu dla długości uszeregowania (wersja niedeterministyczna). Algorytmy

optymalizacyjne opracowane dla wymienionych problemów szeregowania są wykorzystane do rozwiązania deterministycznej i niedeterministycznej (odpornej) wersji problemu ScheLoc. Celem dodatkowym dla rozpatrywanego problemu ScheLoc jest porównanie efektywności optymalizacji łącznej (podejście systemowe zwane również łącznym lub holistycznym) i optymalizacji dwuetapowej, wykorzystującej dekompozycję i kryterium pośrednie dla rozmieszczenia maszyn (podejście sekwencyjne) (Józefczyk & Hojda, 2021). W szczególności, zakładając arbitralne kryterium pośrednie, sprawdzono jak różne strategie rozmieszczenia wpływają na wartość kryterium długości uszeregowania.

Przedmiotem opracowanej rozprawy doktorskiej jest szczegółowa analiza problemów szeregowania zadań z nieustalonymi terminami gotowości. Określenie „nieustalony" może mieć jedno z trzech znaczeń:

1. Terminy gotowości zadań nie są znane bez ustalonego harmonogramu, ponieważ przyjęto terminy gotowości zależne od maszyn.
2. Terminy gotowości zadań są nieprecyzyjne i są reprezentowane w postaci ciągłych przedziałów wartości.
3. Terminy gotowości zadań wynikają z rozwiązania innego problemu optymalizacyjnego.

Ważnym celem niniejszej rozprawy jest zbadanie, czy harmonogramowi stanowiącemu rozwiązanie niedeterministycznego problemu szeregowania z nieprecyzyjnymi (przedziałowymi) terminami gotowości zadań, w którym niepewności parametrów problemu zostały obliczone na podstawie parametrów problemu ScheLoc, może odpowiadać mniejsza długość uszeregowania niż harmonogramowi wyznaczonemu przez algorytmy dedykowane.

Przed bardziej szczegółowym opisem literatury problemu oraz rozwiązywanych zadań badawczych, w następnych dwóch podrozdziałach 1.2 i 1.3 zostaną scharakteryzowane: wykorzystane w rozprawie techniki optymalizacji oraz wybrane reprezentacje niepewności wraz z kryteriami dedykowanymi optymalizacji odpornej.

1.2. Techniki optymalizacji wykorzystane w pracy

Złożoność obliczeniowa problemu decyzyjnego lub optymalizacyjnego determinuje wybór metody jego rozwiązania. W pracy zajmujemy się problemami szeregowania na co najmniej dwóch maszynach, które są co najmniej NP-trudne. Jeżeli można dla problemu „trudnego" zaprojektować algorytm aproksymacyjny, to rozwiązanie gwarantuje

określoną stratę jakości względem rozwiązania optymalnego. W innym przypadku, aby dysponować efektywnym czasowo algorytmem, pozostaje wykorzystanie technik przeszukiwania losowego (np. za pomocą losowych heurystyk) lub wyczerpującego (dla małych instancji). W niniejszej rozprawie wykorzystano następujące techniki optymalizacji:

- algorytmy oparte na metaheurystykach Simulated Annealing oraz Tabu Search,
- algorytm genetyczny (ang. genetic algorithm),
- algorytm hybrydowy (ang. hybrid algorithm),
- algorytm zachłanny (ang. greedy algorithm),
- algorytm przeszukiwania wyczerpującego (ang. exhaustive search).

Metaheurystyka Simulated Annealing jest wygodnym i elastycznym narzędziem, które zaprojektowane zgodnie z regułami przedstawionymi przez (Kirkpatrick, Gelatt & Vecchi, 1983; Ingber, 1993) można wykorzystać do efektywnego rozwiązywania trudnych problemów optymalizacyjnych. Idea działania metaheurystyki wywodzi się z procesu wyżarzania (obróbki cieplnej rozłożonej w czasie) stali w metalurgii. W rezultacie czas działania algorytmu opartego na metaheurystyce Simulated Annealing jest kontrolowany za pomocą wybranej przez projektanta strategii zmniejszania temperatury. Bazując na literaturze można wyróżnić wiele strategii chłodzenia: liniową, kwadratową, logarytmiczną (Nourani & Andresen, 1998). Funkcja temperatury w każdej iteracji zmniejsza jej wartość do momentu osiągnięcia założonego kryterium zatrzymania obliczeń (np. czas obliczeń lub konkretna liczba iteracji bez poprawy jakości rozwiązania). Ten proces zależy od wysokości temperatury początkowej, której dobór może być arbitralny (podyktowany czasem obliczeń lub tempem zbieżności do lokalnego/globalnego ekstremum) lub zależeć od instancji (Ben-Ameur, 2004). Józefczyk & Ławrynowicz (2018) wykorzystali niewielkie, intencjonalne podnoszenie temperatury do poprawy jakości rezultatów w problemie optymalizacji decyzji zakupowych. Cechą charakterystyczną algorytmów opartych na Simulated Annealing jest możliwość zastąpienia (z pewnym prawdopodobieństwem zależnym od rozkładu Boltzmanna) najlepszego znalezionego rozwiązania przez nowe, które zapewnia gorszą jakość z punktu widzenia rozważanego kryterium. Takie zachowanie pozwala uniknąć zbieżności w lokalnym optimum lub zachłannego podejmowania decyzji (Michalewicz & Fogel, 2013). Metoda generowania nowych (sąsiednich) rozwiązań stanowi kluczowy element każdej metaheurystyki. Przyjmuje się, że sąsiednie rozwiązania powinny być zmodyfikowane w małym stopniu, w porównaniu do rozwiązania podstawowego,

aby zachować rozsądny kompromis pomiędzy intensyfikacją i dywersyfikacją w przeglądzie przestrzeni dopuszczalnych rozwiązań. Przykład adaptacji metaheurystyki Simulated Annealing w problemach szeregowania można znaleźć w Józefowska et al. (2001), Wu et al. (2021).

Mataheursytyka Tabu Search intensyfikuje przegląd przestrzeni dopuszczalnych rozwiązań poprzez wykluczenie pewnego ich podzbioru. Ograniczanie przestrzeni wymaga umieszczania pełnych rozwiązań lub pojedynczych decyzji w zbiorze rozwiązań/decyzji zakazanych (lista tabu), którego rozmiar jest parametrem metaheurystyki (Glover, 1989, 1990). Dodatkowym parametrem jest czas (lub liczba iteracji) utrzymania rozwiązania/decyzji na liście. Te dwa parametry determinują czas przeglądu listy, który bezpośrednio rzutuje na czas obliczeń, ponieważ jest wykonywany w każdej iteracji. Element listy tabu może być sprawdzony jako składowa rozwiązania problemu, jeżeli spełnione jest przyjęte kryterium aspiracji. Kryterium aspiracji stanowi warunek dopuszczający wykorzystanie zakazanego rozwiązania/decyzji. Warunek zakończenia obliczeń jest tożsamy z Simulated Annealing z wyłączeniem funkcji temperatury. Efektywną technika, wykorzystywaną do eksploracji przestrzeni możliwych rozwiązań, jest zrównoleglanie obliczeń. Bożejko, Pempera & Smutnicki (2013) zaproponowali metodę zrównoleglania generowania rozwiązań sąsiednich w złożonym problemie przepływowym. Przykład adaptacji metaheurystyki Tabu Search, realizującej obliczenia sekwencyjnie, podali Hindi & Toczyłowski (1995), Harbaoui & Khalfallah (2020).

Algorytmy genetyczne, realizujące obliczenia ewolucyjne, są bardzo często wykorzystywane w obliczeniach inżynierskich (Michalewicz et al. 1996). Podstawową kwestią w projektowaniu algorytmu genetycznego jest wybór sposobu reprezentacji rozwiązania (Michalewicz & Fogel, 2013). Rozwiązanie reprezentowanie w postaci wektora lub macierzy jest poddawane modyfikacjom w kolejnych iteracjach algorytmu. Warto podkreślić, że nieefektywna reprezentacja rozwiązania może skutkować długim czasem ewaluacji rozwiązania, co jest spowodowane transformacją zmiennych decyzyjnych do postaci umożliwiającej wyliczenie wartości funkcji celu. Standardowy algorytm genetyczny realizuje proces optymalizacji w trzech fazach: generowanie nowych rozwiązań za pomocą operatora krzyżowania (faza rekombinacji), losowa modyfikacja rozwiązań (faza mutacji) oraz wybór rozwiązań, które będą podlegały modyfikacjom w kolejnych iteracjach algorytmu (faza selekcji). Każda operacja powinna być dobierana odpowiednio do formy

rozwiązania, aby uniknąć niedopuszczalnych rozwiązań (niepoprawnie zakodowanych). Przykład adaptacji algorytmu genetycznego można znaleźć w Zhang et al. (2020).

Podejście hybrydowe, czyli łączące różne techniki optymalizacji, jest stosowane do rozwiązywania skomplikowanych problemów optymalizacji kombinatorycznej. Jako dobry przykład można wskazać algorytm memetyczny, który wykorzystuje operator przeszukiwania bazujący na metaheurystyce (np. El Fallahi, Prins & Calvo, 2008). W literaturze można spotkać różne zintegrowane metody np. Cuckoo Search+Diffetential Evolution (Zhang, Ding & Jia, 2019), Particle Swarm Optimization+Artificial Bee Colony (Li. et al. 2015).

Zachłanne podejście, czyli opierające się na krótkowzrocznych decyzjach zapewniających najlepszą wartość kryterium z punktu widzenia konkretnego stanu rozwiązania, jest często stosowaną techniką optymalizacji. Algorytm zachłanny może być opracowany dla konkretnego problemu (np. Yu & Jacobson, 2020) lub być składową algorytmu hybrydowego (greedy algorithm+differential evolution w Myszkowski et al., 2018).

Algorytm przeszukiwania wyczerpującego (algorytm realizujący przegląd zupełny wszystkich możliwych rozwiązań) można efektywnie zastosować jedynie dla małych instancji problemu optymalizacyjnego lub decyzyjnego. Wydajność przeglądu zupełnego można poprawić stosując algorytm z nawrotami (ang. backtracking). Wówczas ograniczenia wynikające ze sformułowania problemu mogą stanowić podstawę do wykluczania nieefektywnych rozwiązań. Dla większych instancji takie algorytmy są niepraktyczne z powodu konieczności alokacji dużych zasobów obliczeniowych.

1.3. Niepewność przedziałowa oraz kryteria oceny problemów optymalizacji odpornej

Problemy optymalizacyjne uwzględniające proces podejmowania decyzji w warunkach niepewności stały się obiektem zainteresowania badaczy w latach czterdziestych dwudziestego wieku (Wald, 1945; Hurwicz, 1951; Savage, 1951; Dantzig, 1955). Pierwsze prace zorientowane w tematyce problemów szeregowania zakładających pewien stopień niepewności (nieprecyzyjności) parametrów wejściowych były prowadzone w latach pięćdziesiątych (Dannerstedt, 1955; Schild, 1959), trzydzieści lat od pierwszych badań nad problemami deterministycznymi (Gantt, 1919; Raymond, 1929). Z biegiem czasu matematyczne modele systemów ze względów praktycznych zaczęły w większym wymiarze

uwzględniać nieprecyzyjność parametrów (Rockafellar & Wets, 1991). Najbardziej ogólną podstawą podziału źródeł niepewności jest teoretyczny i praktyczny aspekt badawczy. Prace teoretyczne w większości skupiają się na formułowaniu nowych pojęć w dziedzinie, opracowywaniu nowych podejść (Averbakh, 2000; Ermoliev & Wets, 1988; Lodwick & Kacprzyk, 2010), lub na badaniu niedeterministycznych wersji dobrze zbadanych deterministycznych problemów optymalizacyjnych (Averbakh, 2006; Aissi, Bazgan & Vanderpooten, 2009). Podejścia wypracowane we wcześniej wymienionych pracach stanowią podstawę modelowania rzeczywistych problemów, w których reprezentacja niepewności jest motywowana obszarem zastosowania. Kwestią rozstrzygającą o wyborze formy reprezentacji może być ilość informacji jakimi dysponuje decydent lub sama charakterystyka problemu. Warto zauważyć, że problem planowania wykorzystujący dane historyczne, którego rozwiązanie ma skutek długoterminowy (np. Bagheri, Wang & Zhao, 2016), znacznie różni się od problemów, które wymagają częstych zmian decyzji podjętych dla krótkiego horyzontu czasowego realizacji, bazując na małej ilości informacji (np. Chen et al. 2018).

Omówimy niepewność przedziałową, która jest stosowana w sformułowaniach problemów rozważanych w niniejszej rozprawie doktorskiej oraz kryteria oceny dedykowane dla takiej postaci parametrów niepewnych. Zdefiniujmy deterministyczny problem optymalizacyjny

$$\min_{x \in \mathrm{X}} F(x; a), \tag{1.1}$$

gdzie x jest zmienną optymalizacyjną dowolnej postaci oraz $a = \left[a_1, a_2, \dots, a_j, \dots, a_n\right]^{\mathrm{T}}$ jest wektorem precyzyjnie określonych parametrów, $a_j \in \mathbb{R}$ charakteryzuje obiekt podejmowania decyzji j. Przedziałowa reprezentacja niepewności jest wykorzystywana w przypadku, gdy decydent dysponuje najmniejszą możliwą wiedzą na temat parametru, którego wartość jest nieprecyzyjna. W takim przypadku wystąpienie dowolnej wartości parametru z ustalonego przedziału jest równie prawdopodobne lub niewielka ilość danych uniemożliwia estymację parametrów rozkładów prawdopodobieństwa. W kolejnych rozważaniach przyjęto, że wartość a_j należy do określonego przedziału wartości

$$a_j \in \left[a_j^-, a_j^+\right],\ a_j^- < a_j^+. \tag{1.2}$$

Dowolny wektor a ustalonych wartości parametrów dla obiektów $1,2,\dots,j,\dots,n$ jest nazywany „scenariuszem". Zbiór wszystkich możliwych scenariuszy odpowiada iloczynowi kartezjańskiemu danych przedziałów

$$\Phi = [a_1^-, a_1^+] \times [a_2^-, a_2^+] \times \dots \times \left[a_j^-, a_j^+\right] \times \dots \times [a_n^-, a_n^+]. \tag{1.3}$$

Można łatwo zauważyć, że wektor parametrów wejściowych problemu deterministycznego (1.1) jest jednym z możliwych scenariuszy $a \in \Phi$. Niewątpliwą zaletą apriorycznie zdefiniowanych (1.2), (1.3) jest brak konieczności estymacji rozkładów wartości parametrów, co może generować duży błąd w przypadku niewystarczającej ilości danych (Kasperski, 2008). W zagadnieniach optymalizacji coraz częściej wykorzystywany jest koncept γ-odporności (ang. γ-robustness; Bertsimas & Sim, 2003, 2004), zgodnie z którym na przedziały wartości nakładane jest ograniczenie

$$\sum_{j \in \mathrm{J}} \left(a_j - a_j^-\right) \leq \Gamma, \Gamma \in \mathbb{R}_+. \tag{1.4}$$

Ograniczenie (1.4) znajduje coraz większe zastosowanie w problemach szeregowania (Bachler, Krumke & Le, 2020; Krumke & Le, 2020), ponieważ pozwala generować instancje z założonym apriorycznie rozsądnym marginesem niepewności w odniesieniu do wszystkich przedziałów.

Kryteria oceny jakości w problemach z niepewnością przedziałową bazują na determinizacji parametru niepewnego. Jest to koncept, który wymaga założeń przyjętych przez decydenta lub jest skutkiem samej charakterystyki problemu, gdy skończona liczba scenariuszy pozwala uzyskać rozwiązanie optymalne.

Jako problem podstawowy (niedeterministyczny) z kryterium uwzględniającym scenariusze parametrów wejściowych można przyjąć zdefiniowany przez Walda (1945)

$$\min_{x \in \mathrm{X}} \max_{a \in \Phi} F(x, a). \tag{1.5}$$

Podejmowanie decyzji w oparciu o (1.5) implikuje determinizację przedziałowych wartości parametrów do postaci wektora a. Konsekwencją wyboru kryterium (1.5) jest rozważanie najgorszego przypadku po to, aby wprowadzone ograniczenia zostały spełnione bez względu na zrealizowany scenariusz (Rockafellar, 2007).

Innym podejściem zaproponowanym w literaturze do oceny jakości decyzji podjętych w warunkach niepewności jest kryterium maksymalnego żalu (Savage, 1951). Kryterium to jest oparte na różnicy wartości kryterium, obliczonej na podstawie podjętych decyzji przy

ustalonym scenariuszu, i optymalnej wartości tego kryterium przy tym samym scenariuszu. Niedeterministyczny problem optymalizacyjny wykorzystujący kryterium maksymalnego żalu definiuje się w postaci

$$\min_{x\in \mathrm{X}} Q(x) = \min_{x\in \mathrm{X}} \max_{a\in\Phi}\big(F(x,a) - F(x_a^*)\big), \tag{1.6}$$

gdzie x_a^* jest optymalnym rozwiązaniem problemu deterministycznego (1.1) przy zrealizowanym scenariuszu a, wartość $Q(x)$ nazywamy maksymalnym żalem dla zmiennej decyzyjnej x a $\arg\max_{a\in\Phi}(F(x,a) - F(x_a^*))$ najgorszym scenariuszem. Podejmowanie decyzji angażujące kryteria (1.5) i (1.6) jest w literaturze nazywane optymalizacją odporną. Równanie (1.6) jest bardziej kompleksowym sformułowaniem z punktu widzenia decydenta, w porównaniu do (1.5), gdyż angażuje wszystkie możliwe scenariusze. Kluczową kwestią związaną z wyliczeniem wartości (1.6) jest liczba scenariuszy, którą należy sprawdzić, aby wyznaczyć ekstremum globalne, i trudność problemu deterministycznego. Jeżeli liczba scenariuszy w sfomułowaniu (1.6) jest nieskończona lub problem (1.1) jest co najmniej NP-trudny, to wartość kryterium można jedynie oszacować. Problemy optymalizacji odpornej sformułowane w postaci (1.6) są popularnym obszarem badań z powodu minimalizacji awersji do ryzyka w procesie podejmowania decyzji. Analiza złożoności obliczeniowej problemów sformułowanych w sposób taki jak (1.6), które można rozwiązać w czasie wielomianowym w przypadku deterministycznej wersji, ułatwia opracowanie efektywnych rozwiązań. Aissi, Bazgan & Vanderpooten (2005) wykazali, że klasyczny problem przypisania na grafie dwudzielnym z przedziałowymi wartościami reprezentującymi długości krawędzi jest silnie NP-trudny zarówno w wersji (1.5) jak i (1.6). Problem przypisania na grafie dwudzielnym w wersji deterministycznej można rozwiązać w czasie wielomianowym. W literaturze kontynuowane są również prace nad złożonością obliczeniową problemów niedeterministycznych, które są co najmniej NP-trudne w wersji deterministycznej. Drwal & Rischke (2016) wykazali, że problem szeregowania zadań na identycznych maszynach równoległych sformułowany w postaci (1.6) i z przedziałowymi czasami przetwarzania zadań dla sumy terminów zakończeń realizacji zadań jest silnie NP-trudny.

Zmniejszenie liczby scenariuszy, będące efektem przyjęcia ograniczonego zbioru wartości reprezentujących parametr niepewny, ułatwia wyliczenie wartości (1.6). Jeżeli a_j jest parametrem niepewnym, który może być reprezentowany (z równym prawdopodobieństwem) przez dowolną wartość ze zbioru $\left\{a_j^{(1)}, a_j^{(2)}, \dots, a_j^{(e_j)}\right\}$, gdzie e_j

liczbą dopuszczalnych reprezentacji parametru a_j to zbiór wszystkich scenariuszy dla n parametrów niepewnych można przedstawić jako

$$\widetilde{\Phi} = \left\{a_1^{(1)}, a_1^{(2)}, \dots, a_1^{(e_1)}\right\} \times \left\{a_2^{(1)}, a_2^{(2)}, \dots, a_2^{(e_2)}\right\} \times \dots \times \left\{a_j^{(1)}, a_j^{(2)}, \dots, a_j^{(e_j)}\right\} \times \dots \times \left\{a_j^{(1)}, a_j^{(2)}, \dots, a_n^{(e_n)}\right\}. \tag{1.7}$$

Wybór takiego podejścia może być podyktowany preferencją decydenta, charakterystyką problemu lub przyjętymi założeniami. Preferencja może wynikać z faktu, że opracowany algorytm operuje na arbitralnie założonym scenariuszu, który nie jest najgorszym scenariuszem, lub być konsekwencją ograniczenia wartości kryterium. Jako odpowiedni przykład można podać pracę Kasperski & Zieliński, (2008), w której autorzy wykazali, że heurystyka środka przedziałów jest algorytmem 2-aproksymacyjnym dla problemu szeregowania w wersji z kryterium (1.6) i przedziałowymi czasami przetwarzania zadań dla sumy terminów zakończeń zadań na jednej maszynie. Dyskretna reprezentacja niepewności może gwarantować znalezienie najgorszego scenariusza. Pereira (2016) wykazał tę własność w niedeterministycznej wersji problemu szeregowania zadań na jednej maszynie z kryterium (1.6) dla ważonej sumy terminów zakończeń zadań z niepewnymi czasami ich przetwarzania. Sformułowania klasycznych problemów optymalizacyjnych (problem plecakowy, najkrótszej ścieżki, minimalnego drzewa rozpinającego) w wersjach niedeterministycznych z arbitralnie założonymi dyskretnymi przedziałami niepewności można znaleźć w Kouvelis & Yu (2013).

Kryterium scalającym podejścia w (1.5), (1.6) jest relatywna odporność $\tilde{Q}(x)$ (ang. relative robustness), która stanowi podstawę problemu minimalizacji

$$\min_{x \in \mathrm{X}} \tilde{Q}(x) = \min_{x \in \mathrm{X}} \max_{a \in \Phi} \left(\frac{F(x,a) - F(x_a^*)}{F(x_a^*)} \right). \tag{1.8}$$

Yang & Yu (2002) zastosowali kryteria (1.5)-(1.6) i (1.8) w problemie szeregowania zadań dla sumy terminów zakończeń zadań na jednej maszynie. Autorzy wykazali NP-zupełność niedeterministycznej wersji $1||\sum C_j$ postaci (1.3) dla (1.7).

Popularnym kierunkiem prac nad formułowaniem kryteriów w problemach optymalizacji odpornej jest parametryzacja. Kryterium zakładające kompromis pomiędzy skrajnymi wartościami funkcji celu zaproponował Hurwicz (1951). W literaturze można zidentyfikować różne metody wykorzystania takiego podejścia w kontekście badań operacyjnych (Gaspars-Wieloch, 2014). Aby wykorzystać kryterium Hurwicza

w optymalizacji odpornej, wystarczy wyznaczyć maksima i minima lokalne dla (1.5)-(1.6), (1.8) i wówczas

$$\begin{aligned} z_{\mathrm{H1}}(x) &= \beta \max_{x\in \mathrm{X}} \max_{a\in\Phi} F(x;a) - (1-\beta) \min_{x\in \mathrm{X}} \min_{a\in\Phi} F(x;a), \\ z_{\mathrm{H2}}(x) &= \beta \max_{x\in \mathrm{X}} Q(x) - (1-\beta) \min_{x\in \mathrm{X}} Q(x), \\ z_{\mathrm{H3}}(x) &= \beta \max_{x\in \mathrm{X}} \tilde{Q}(x) - (1-\beta) \min_{x\in \mathrm{X}} \tilde{Q}(x), \end{aligned} \tag{1.9}$$

gdzie $\beta \in [0,1]$ jest parametrem dobieranym przez decydenta. Yager (1988) zaproponował kryterium zakładające preferowanie scenariuszy poprzez dobór odpowiednich wag dla skończonego zbioru scenariuszy $\sum_{a\in\widetilde{\Phi}} w_a = 1$ i

$$\mathrm{OWA}(x) = \sum_{a\in\widetilde{\Phi}} w_a F(x,a). \tag{1.10}$$

Wartość (1.10) może podlegać (w zależności od sformułowania problemu) minimalizacji, maksymalizacji, uśrednieniu itp. Problematyczną kwestią jest dobór wag dla wszystkich scenariuszy, który może być arbitralny lub wykonany zaawansowanymi metodami (np. Majdan & Ogryczak, 2012). Adaptacje kryterium Hurwicza i (1.10) można znaleźć w Kasperski & Zieliński (2016), gdzie podjęto temat różnych problemów szeregowania zadań na jednej maszynie z niepewnymi czasami przetwarzania, terminami zakończeń lub wagami. Podejścia bazujące na koncepcji (b,w)-odporności (ang. (b,w)-robustness), które pośrednio bazują na (1.5)-(1.6) i (1.8) zaproponował Roy (2010). Parametry b i w stanowią ograniczenia dla wartości kryterium. Wartość b jest ograniczeniem dla funkcji celu przy zrealizowanym scenariuszu. Parametr $w \le \max_{x\in \mathrm{X}} \min_{a\in\Phi} F(x,a)$ jest wskaźnikiem rozstrzygającym o odrzuceniu rozwiązania, jeżeli dla co najmniej jednego scenariusza wartość funkcji celu jest mniejsza niż w. Jako przykład takiego podejścia można podać (b,w)-bezwzględną odporność

$$z_{\mathrm{BW}}(x) = \max_{x\in \mathrm{X}} \sum_{a\in\widetilde{\Phi}} 1(b - F(x,a)), \tag{1.11}$$

gdzie $1(x) = 1(0)$ dla $x \ge 0 (x < 0)$ oraz $F(x,a) < w$. Podobne koncepcje nazwane „(b,w)-absolute deviation" i „(b,w)-relative deviation" można znaleźć w Roy (2010). Aissi & Roy (2010) zdefiniowali również inne wskaźniki oceny

$$z_{\mathrm{BW1}}(x) = \frac{1}{|\widetilde{\Phi}|} \sum_{a\in\widetilde{\Phi}} \max_{i=1,2,\ldots,m} |b_i - F(x,a)|, \tag{1.12}$$

$$z_{\mathrm{BW2}}(x) = \frac{1}{m|\widetilde{\Phi}|}\sum_{i=1}^{m}\sum_{a\in\widetilde{\Phi}}|b_i - F(x,a)|,$$

$$z_{\mathrm{BW3}}(x) = \frac{1}{m|\widetilde{\Phi}|}\sum_{i=1}^{m}\sum_{a\in\widetilde{\Phi}}|b_i - F(x,a)|^2,$$

gdzie m jest parametrem problemu. Snyder (2006) zdefiniował pojęcie p-odporności (ang. p-robustness) do akceptacji rozwiązań gwarantujących odchylenie o wartość p od rozwiązania optymalnego. W artykule wartość p wyrażono w procentach. Zbiór rozwiązań spełniających założenia p-odporności można zdefiniować jako

$$\mathrm{B}(\mathrm{X}, p) = \left\{x \in \mathrm{X}: \underset{a\in\Phi}{\forall} F(x,a) - F(x_a^*) \leq p\right\}. \tag{1.13}$$

W (1.13) nie rozważamy jedynie rozwiązania minimalizującego odchylenie przy najgorszym scenariuszu jak w (1.6), (1.8) ale również agregujemy rozwiązania zapewniające różny stopień odchylenia od wartości optymalnej. Kalaı, Lamboray & Vanderpooten (2012) zdefiniowali leksykograficzną α-odporność (ang. lexicographic α-robustness), która stanowi modyfikację (1.13). Przyjmowane są rozwiązania, które spełniają dopuszczalny margines odchylenia α a porządek leksykograficzny odnosi się do agregacji rozwiązań posortowanych względem wartości odchyleń.

1.4. Przegląd literatury

Badania literaturowe przeprowadzono dla szerokiej gamy problemów mogących być składowymi łącznego problemu szeregowania zadań i rozmieszczenia maszyn dowolnych (ScheLoc). Chronologia omawiania poszczególnych pozycji odpowiada stopniowi zaawansowania wybranych zagadnień. Dokonano przeglądu wszystkich dostępnych publikacji (w momencie składania rozprawy doktorskiej) z zakresu szeregowania zadań z terminami gotowości i kryterium długości uszeregowania, odpornego szeregowania zadań z przedziałowymi terminami gotowości na maszynach równoległych z kryterium maksymalnego żalu dla długości uszeregowania oraz dotyczących problemu ScheLoc. Wyniki przeglądu są przestawione w kolejnych częściach tego podrozdziału. Pozostałe pozycje powiązane z treścią pracy wybrano arbitralnie, z powodu obszernej literatury, aby przekrojowo omówić badania dotyczące wybranych zagadnień. Do opisu deterministycznych problemów szeregowania wykorzystano notację Grahama. W niedeterministycznych problemach szeregowania oraz w problemie ScheLoc zastosowano zmodyfikowaną notację.

1.4.1. Problemy szeregowania zadań z terminami gotowości i kryterium długości uszeregowania

Przegląd literatury rozpoczniemy od bazowego (najprostszego) deterministycznego problemu szeregowania na m równoległych maszynach, w którym terminy gotowości wszystkich zadań są równe. Problem bazowy można opisać w notacji grahama jako $Rm||C_{\max}$. Pozycje literaturowe podano w tabelach 1.1 i 1.2.

Tabela 1.1. Zestawienie algorytmów aproksymacyjnych opracowanych dla $Rm||C_{\max}$. W lewej kolumnie tabeli podano wartości współczynników aproksymacji

$1+\varepsilon$*	Horowitz & Sahni (1976)
$m;(1+\sqrt{5})/2$ dla $m=2$	Ibarra & Kim (1977)
$2\sqrt{m}$	Davis & Jaffe (1981)
$2;\ 3/2$ dla $m=2$	Potts (1985)
2	Lenstra et al. (1990)
$2-m^{-1}$	Shchepin &Vakhania (2005)

* Wielomianowy schemat aproksymacji (ang. Polynomial-Time Approximation Scheme). Autorzy opracowali algorytm podejmujący decyzje w czasie $O(n^{2m}/\varepsilon)$, gdzie n jest liczbą zadań do uszeregowania.

Tabela 1.2. Zestawienie literatury problemów szeregowania zadań z terminami gotowości w wersjach: deterministycznej (kryterium długości uszeregowania) i niedeterministycznej (kryterium maksymalnego żalu dla długości uszeregowania)

$Rm\|\|C_{\max}$*	Cytowania umieszczone w tabeli 1.1; Srivastava, 1998; Ghirardi & Potts, 2005
$1\|r_j\|C_{\max}$	Baker (1984)
$Rm\|r_j\|C_{\max}$	Yang-Kuei & Chi-Wei (2013); Lin (2013); Avdeenko & Mezentsev (2016); Mezentsev, Estraykh & Chubko (2019)
$1\|r_j^- \le r_j \le r_j^+\|C_{\max}$**	Bachler, Krumke & Le (2020)
$Rm\|r_j^- \le r_j \le r_j^+\|Reg(C_{\max})$***	Ławrynowicz & Józefczyk (2021)
$Rm\|r_{i,j}^- \le r_{i,j} \le r_{i,j}^+\|Reg(C_{\max})$***	Józefczyk & Ławrynowicz (2021)
$Rm\|r_{i,j}\|C_{\max}$	Ławrynowicz & Józefczyk (2022a; 2022b)

* Z powodu obszernej literatury wybrano pozycje w taki sposób, aby przekrojowo omówić opublikowane metody rozwiązania problemu.

** Zaproponowano modyfikację notacji Grahama na przypadek problemu z niepewnością przedziałową. Niepewność parametru β jest wyrażona w postaci nierówności $\beta^- \le \beta \le \beta^+$.

*** Oznaczenie $Reg(C_{\max})$ dotyczy kryterium maksymalnego żalu dla długości uszeregowania.

Problem szeregowania na m identycznych maszynach równoległych $Pm||C_{\max}$ jest NP-trudny (dla dwóch maszyn) i silne NP-trudny (dla co najmniej trzech maszyn). Dowody przeprowadzono wykorzystując redukcję wielomianową problemu podziału (ang. partition problem) do $P2||C_{\max}$ oraz redukcję pseudowielomianową problemu podziału na trzyelementowe podzbiory liczb o tej samej sumie (ang. 3-partition problem) do $P3||C_{\max}$ (Garey & Johnson, 1979). Dodatkowo wskazano, że problem $Pm||C_{\max}$ można rozwiązać w czasie pseudowielomianowym dla arbitralnie ustalonej liczby maszyn. W konsekwencji $Rm||C_{\max}$ nie może cechować się mniejszą złożonością, ponieważ $Pm||C_{\max}$ jest jego szczególnym przypadkiem. Z tego powodu wszystkie prace podejmujące problemy szeregowania zadań na co najmniej dwóch maszynach, wskazane w przeglądzie literaturowym w niniejszej rozprawie, są problemami co najmniej NP-trudnymi.

Złożoność problemu $Rm||C_{\max}$ wymusza opracowywanie algorytmów znajdujących przybliżone rozwiązania w czasie wielomianowym. Obszerna część badań dotyczy projektowania algorytmów aproksymacyjnych. Pozycje literaturowe podano w tabeli 1.1. Algorytm o najbardziej efektywnym współczynniku aproksymacji został opracowany przez Shchepin & Vakhania (2005). Autorzy podczas pierwszego etapu podejmowania decyzji wykorzystują algorytm konstrukcyjny, bazujący na zmodyfikowanym podejściu zachłannym opracowanym przez Lenstra et al. (1990), do budowy harmonogramu, który jest modyfikowany w następnym kroku przy pomocy techniki zaokrąglania (ang. rounding; Williamson & Shmoys, 2011). Na podstawie analizy wartości współczynników aproksymacji podanych tabeli 1.1 można zauważyć, że część autorów uzyskało współczynnik aproksymacji $\alpha(I)$ zależny od instancji problemu I. Dominująca część prac uwzględnia zależność jakości aproksymacji od liczby maszyn m. Inne podejścia (wymienione w tabeli 1.2) zakładają wykorzystanie znanych metaheurstyk np. Tabu Search (Srivastava, 1998) lub Recovering Beam Search (Ghirardi & Potts, 2005). Prace nad rozwinięciem podstawowych założeń $Rm||C_{\max}$ są nadal kontynuowane w literaturze. Na przykład Azar et al. (2018) zdefiniowali problem $Rm||C_{\max}$, w którym każde zadanie musi być przetworzone na różnych maszynach oraz opracowano algorytm 2-aproksymacyjny bazujący na technikach opisanych w Lenstra et al. (1990) i Azar et al. (2004).

Problemy szeregowania zadań z terminami gotowości i kryterium długości uszeregowania nie stały się dotychczas przedmiotem tak szczegółowych badań jak $Rm||C_{\max}$. Najprostszym przypadkiem jaki można wyróżnić jest problem szeregowania zadań na jednej maszynie $1|r_j|C_{\max}$. Optymalnym rozwiązaniem $1|r_j|C_{\max}$ jest przypisanie

zadań w kolejności odpowiadającej ich terminom gotowości. Taki algorytm (oparty na sortowaniu) w literaturze nazywany jest metodą (lub regułą) ERD (ang. Earliest Release Date rule; Baker, 1984). Yang-Kuei & Chi-Wei (2013) jako pierwsi zdefiniowali mieszany model programowania całkowitoliczbowego dla $Rm|r_j|C_{\max}$ oraz zaproponowali metodę rozwiązania opartą na zachłannej strategii podejmowania decyzji. Autorzy posługują się terminem „dispatching rule" zamiast klasycznego określenia „scheduling algorithm", co jest związane z intencją implementacji rozwiązań w przemyśle. Procedura optymalizacji jest dwuetapowa. Podczas pierwszego (zachłannego) etapu algorytm projektuje harmonogram minimalizując w każdej iteracji termin zakończenia przypisywanego zadania zachowując wcześniej podjęte decyzje. Następnie harmonogram jest modyfikowany poprzez zmianę terminów wykonania zadań, które zostały przypisane do najdłużej pracującej maszyny, ale mogą się zakończyć wcześniej na innych maszynach, poprawiając jednocześnie wartość kryterium. Lin (2013) wykorzystał metaheurystykę Particle Swarm Optimization (PSO) oraz zdefiniował dolne ograniczenia dla $Rm|r_j|C_{\max}$, aby oszacować jakość przeszukiwania opartego na roju cząstek. Zaproponowano dwie metody szacowania: uśrednienie najkrótszych czasów przetwarzania zadań dodane do najwcześniejszego terminu gotowości oraz najpóźniejszy z najwcześniejszych możliwych terminów zakończenia jednego zadania (podejście „maxmin"). Wykazano efektywność algorytmu opartego na metaheurystyce PSO, porównując wyniki optymalizacji z rezultatami uzyskanymi przez Yang-Kuei & Chi-Wei (2013). Algorytm PSO zwrócił mniejszą długość uszeregowania zarówno dla małych (4 maszyny i 18 zadań) jak i dużych instancji (10 maszyn i 100 zadań). Avdeenko & Mezentsev (2016) opracowali algorytm wykorzystujący technikę programowania dynamicznego. Dodatkowo zaimplementowano przegląd wyczerpujący dla podproblemów do eliminacji rozwiązań nieefektywnych celem skalowania algorytmu dla dużych instancji problemów (100 zadań, 30 maszyn). Mezentsev, Estraykh & Chubko (2019) poprawili wyniki poprzedniej pracy, stosując zaawansowane konstrukcyjne heurystyki. W podanych wcześniej pozycjach literaturowych terminy gotowości są parametrami zadań. Ławrynowicz & Józefczyk (2022a) założyli, że terminy gotowości zadań w problemie $Rm|r_{i,j}|C_{\max}$ zależą od maszyn. Wówczas rozbieżności w terminach gotowości zadań na różnych maszynach mogą wykluczać możliwość równoległego przetwarzania zadań w systemie. Autorzy opracowali zachłanny algorytm dla $Rm|r_{i,j}|C_{\max}$ oraz wykazali, że długość uszeregowania jest ograniczona względem oszacowania optymalnej wartości kryterium. W procesie podejmowania decyzji wykorzystano planowanie

długoterminowe w przypadku, gdy jakość decyzji przypisania wielu zadań z określonego podzbioru zadań nieuszeregowanych była jednakowa. Porównano efektywność algorytmu zachłannego z rezultatami osiąganymi przez algorytm oparty na przeglądzie zupełnym oraz zaproponowano metody oszacowania wartości kryterium. Opracowanie algorytmu gwarantującego stały współczynnik aproksymacji dla problemu $Rm|r_{i,j}|C_{\max}$ nadal pozostaje kwestią otwartą. Badania nad wyselekcjonowaniem instancji problemu $Rm|r_{i,j}|C_{\max}$, które można rozwiązać w czasie wielomianowym przeprowadzili Ławrynowicz & Józefczyk (2022b). Ponadto wskazano instancję oryginalnego problemu szeregowania $Rm|r_{i,j}, d_i|C_{\max}$, której optymalne rozwiązanie w czasie pseudowielomianowym gwarantuje również rozwiązanie szczególnej instancji problemu $Rm|r_{i,j}|C_{\max}$ w tym samym czasie.

1.4.2. Problemy szeregowania zadań z przedziałowymi (niepewnymi) terminami gotowości

Niniejsza rozprawa stanowi kontynuację prac nad tematyką problemów szeregowania zadań z niepewnością przedziałową, które zostały zrealizowane na Wydziale Informatyki i Zarządzania Politechniki Wrocławskiej w Katedrze Informatyki i Inżynierii Systemów przez Siepak (2013) i Ćwik (2017).

Relatywnie nieduża liczba pozycji literaturowych poddających analizie problemy szeregowania zadań z nierównymi terminami gotowości i kryterium długości uszeregowania przekłada się bezpośrednio na niewielką liczbę prac poruszających temat ich niedeterministycznych odpowiedników (tabela 1.2). Wynika to z faktu, że terminy gotowości zadań są często zakładane nie wprost (np. czas dotarcia zadania do miejsca przetwarzania definiuje termin gotowości) lub w postaci parametrów niepewnych. Bazowym problemem szeregowania z przedziałowymi terminami gotowości zadań r_j, którego sformułowanie obejmuje odporne podejście do kryterium długości uszeregowania, jest $1|r_j^- \le r_j \le r_j^+|C_{\max}$ (Bachler, Krumke and Le, 2020). Autorzy wskazali, że problem $1|r_j^- \le r_j \le r_j^+|C_{\max}$ z kryterium zdefiniowanym w postaci (1.5) jest równoważny $1|r_j|C_{\max}$, który można rozwiązać optymalnie za pomocą metody ERD (dla pesymistycznego scenariusza, w którym wszystkie zadania są dostępne do realizacji w najpóźniejszym możliwym terminie). Opracowane sformułowanie problemu uwzględnia założenia Gamma-odporności. Ciekawą cechą omawianego problemu jest możliwość jego optymalnego rozwiązania w czasie wielomianowym zarówno dla kryterium długości uszeregowania jak i dla kryterium maksymalnego żalu. Wynika to z faktu, że liczba istotnych scenariuszy jest

równa liczbie zadań oraz deterministyczną wersję problemu można rozwiązać w czasie wielomianowym, co pozwala obliczyć dokładną wartość kryterium. Rozwiązanie problemu sprowadza się do podejmowania zachłannych decyzji bazujących na obliczaniu wartości kryterium dla każdego istotnego scenariusza. Ławrynowicz & Józefczyk (2021) rozszerzyli niedeterministyczny problem $1|r_j^- \leq r_j \leq r_j^+|Reg(C_{\max})$ na przypadek m równolegle przetwarzających maszyn dowolnych $Rm|r_j^- \leq r_j \leq r_j^+|Reg(C_{\max})$. Wykonano badanie efektywności algorytmów zachłannych dla różnych form dekompozycji kryterium. Autorzy wykazali w badaniach eksperymentalnych, że bazując na deterministycznym kryterium długości uszeregowania można uzyskać rozwiązania o jakości zbliżonej do rozwiązań uzyskanych przez algorytmy dedykowane problemowi niedeterministycznemu. Efektywność algorytmu wykorzystującego kryterium deterministyczne wynikała z zastosowania metody wyboru kolejności zadań do przypisania (przed realizacją procedury szeregowania) na podstawie rezultatów szacowania liczby zmian w harmonogramie optymalnym w porównaniu do harmonogramu ustalonego przez decydenta. Dla każdego nieprzypisanego zadania w harmonogramie obliczano wartość wskaźnika, która wyraża oszacowanie jak wiele nieprzypisanych w harmonogramie zadań może się przed nim wykonać, gdyby było ono umieszczone w harmonogramie. Podkreślono, że wraz ze wzrostem wartości wskaźnika rośnie prawdopodobieństwo pozostawienia niewykorzystanych okien czasowych, co przekłada się na pogorszenie jakości rozwiązania. Józefczyk & Ławrynowicz (2021) sformułowali problem niedeterministyczny $Rm|r_{i,j}^- \leq r_{i,j} \leq r_{i,j}^+|Reg(C_{\max})$, w którym przedział niepewności dla terminu gotowości dowolnego zadania zależy od maszyny, na której jest ono przetwarzane. Zdefiniowana zależność determinuje zupełnie inną interpretację kryterium, w porównaniu do problemu $Rm|r_j^- \leq r_j \leq r_j^+|Reg(C_{\max})$, ponieważ w nowym problemie wartość maksymalnego żalu może być większa niż suma czasów przetwarzania zadań. Wartość kryterium uwzględnia również czasy oczekiwania na gotowość zadań. Autorzy zastosowali algorytm oparty na metaheurystyce Tabu Search do rozwiązania problemu.

W literaturze można również znaleźć sformułowania problemów szeregowania zadań z niepewnymi terminami gotowości, w których autorzy przyjmują inne kryteria do oceny jakości rozwiązań. Pierwsza praca w dziedzinie porusza problem gniazdowy (ang. job shop scheduling) z kryterium długości uszeregowania, w którym termin gotowości pojawienia się zadania na pierwszej maszynie jest dany przedziałem (Wu et al., 2005). Jako metody rozwiązania zaproponowano podejścia: *Expectation* (rozwiązanie problemu dla wszystkich

możliwych decyzji, przy możliwie największej liczbie scenariuszy, i wybór najlepszego rozwiązania), *Consensus* (rozwiązanie problemu dla losowych scenariuszy i wybór rozwiązania dla najczęstszego przypadku) i *Regret* (adaptacja podejścia *Consensus* z uwzględnieniem odchylenia od wartości optymalnej dla każdego wyniku). Szczególnym obszarem zastosowań szerokiej gamy problemów szeregowania zakładających niepewność parametrów jest przemysł. Yue et al. (2018) sformułowali silnie NP-trudny problem szeregowania zadań $1|r_j^- \leq r_j \leq r_j^+|Reg(WT_{\max})$ na jednej maszynie z przedziałowymi terminami gotowości zadań i z kryterium maksymalnego żalu dla maksymalnego czasu oczekiwania na zakończenie realizacji zadania. Jako obszar wdrożenia optymalizacji wskazano proces produkcji żelaza, podczas którego momenty wyciągania materiału z pieca są nieprecyzyjne określone. Do rozwiązania zaproponowano zmodyfikowaną heurystykę Gusfield'a oraz algorytm bazujący na metaheurystyce Variable Neighborhood Search. Złożony silnie NP-trudny wielokryterialny gniazdowy problem szeregowania $Jm|r_j^- \leq r_j \leq r_j^+|C_{\max}$ & $T_{\max}$ & $Robust$, którego celem jest minimalizacja długości uszeregowania, sumy ważonych opóźnień zakończeń zadań oraz zaproponowanego kryterium opisującego odporność na odchylenie od wartości dwóch poprzednich kryteriów sformułowali Shen et al. (2016). Celem pracy jest adaptacja algorytmu ewolucyjnego MOEA/D do rozwiązania problemu z niepewnymi terminami gotowości zadań i skończoną liczby scenariuszy. Terminy gotowości oraz czasy przetwarzania zadań dane w postaci rozkładów prawdopodobieństwa w silnie NP-trudnym problemie $Pm|r_j(\omega), p_j(\omega)|\text{Cost}$ & $\text{E}[\sum E_j + T_j]$ założyli Liu et al. (2021). Wskazano, że kształt rozkładów jest rezultatem agregacji i przetworzenia danych o historii dystrybucji i produkcji części w przemyśle. Celem jest minimalizacja kosztów przygotowania maszyn oraz wartości oczekiwanej kosztu kar związanych ze zbyt wczesnym lub zbyt późnym uszeregowaniem zadań. Rozwiązanie problemu jest dwuetapowe. Podczas pierwszego etapu zadania są przypisywane do konkretnych maszyn bez przyjęcia konkretnej realizacji parametrów niepewnych. Następnie dla ustalonych parametrów wyznaczany jest harmonogram. Autorzy wykorzystują symulację Monte Carlo i opracowany algorytm konstrukcyjny, który efektywnie ogranicza zbiór możliwych scenariuszy jako metody rozwiązania problemu.

1.4.3. Problemy rozmieszczenia maszyn i szeregowania zadań (ScheLoc)

Pełny przegląd literatury problemu ScheLoc zaprezentowano w tabeli 1.3.

Tabela 1.3. Przegląd literatury problemu rozmieszczenia maszyn i szeregowania zadań (ScheLoc)

$1\|r_j(y)\|C_{\max}\|\mathrm{D}$	Hennes & Hamacher (2002)
$1\|r_j(y)\|C_{\max}\|\mathrm{C}$	Elvikis et al. (2007); Elvikis et al. (2009); Kalsch and Drezner (2010); Scholz (2012)
$1\|r_j(y)\|C_{\max}\|\mathrm{Tree}$	Hennes & Hamacher (2002)
$1\|r_j(y)\|C_{\max}\|\mathrm{Graph}$	Hennes & Hamacher (2002)
$1\|r_j(y)\|\sum C_j\ \|\mathrm{C}$	Kalsch & Drezner (2010)
$1\|r_j(y),\mathrm{pmtn}\|\sum w_j\mathrm{F}(C_j)\ \|\mathrm{Graph}$	Kaufmann (2014)
$1\|r_j^- \le r_j \le r_j^+\|C_{\max}\|\mathrm{Tree}$	Krumke & Le (2020)
$Pm\|r_j(y)\|C_{\max}\|\mathrm{D}$	Rajabzadeh et al. (2016); Heßler & Deghdak (2017); Wang et al. (2020); Li et al. (2021); Kramer & Kramer (2021; 2022)
$Pm\|r_j(y)\|C_{\max}\|\mathrm{C}$	Rajabzadeh et al. (2016)
$Pm\|r_j(y)\|\sum C_j\ \|\mathrm{C}$	Piasecki (2018); Piasecki & Józefczyk (2018)
$Pm\|r_j(y)\|\mathrm{Cost}\|\mathrm{D}$	김동욱 (2021)
$Pm\|r_j(y),d_j,p_j(\omega)\|\mathrm{Cost}\|\mathrm{D}$	Liu & Liu (2019b)
$Pm\|r_j(y),p_j(\omega)\|\sum w_j\mathrm{Cost}_j\ \&\ \mathrm{E}\left[\sum C_j(\omega)\right]\|\mathrm{D}$	Liu et al. (2019); Liu & Liu (2019a)
$Pm\|r_j(y)\|\mathrm{Cost}\ \&\ \sum C_j\ \|\mathrm{D}$	Filcek et al. (2021); Ławrynowicz & Filcek (2021)
$Pm\|r_j(y)\|\mathrm{Cost}\ \&\ C_{\max}\|\mathrm{D}$	Wu et al. (2022)
$Pm\|r_j(y)\|\mathrm{E}\left[\sum C_j\ \&\ \sum r_j(y)\right]\|\mathrm{D}$	Fu et al. (2022)
$Pm\|r_j(y)\|C_{\max}\ \&\ \sum r_j(y)\ \|\mathrm{D}$	Li et al. (2022)
$Pm\|r_j(y),reject\|\mathrm{Multi}\|\mathrm{D}$	Zhang et al. (2022a)
$Pm\left\|r_j(y),d_j\right\|\sum w_j\mathrm{Cost}_j\ \&\ T_j\|\mathrm{D}$	Zhang et al. (2022b)
$Pm\|r_j(y),\tilde{p}_j(\omega),d_j,reject\|\mathrm{Profit}\|\mathrm{D}$	Liu et al. (2023)
$Rm\|r_j(y)\|C_{\max}\|\mathrm{D}\|\mathrm{m\_median}$	Ławrynowicz & Józefczyk (2019)
$Rm\|r_j(y)\|C_{\max}\|\mathrm{D}$	Ławrynowicz & Józefczyk (2019)
$Fm\|r_j(y)\|C_{\max}\|\mathrm{D}$	Buckhorst, Canto & Schmitt (2022)

Do opisu różnych wersji problemu ScheLoc wprowadzono zmodyfikują notację Grahama bazującą na wyrażeniu regularnym A|B|C|D|E, gdzie A – typ maszyn oraz liczba maszyn, B – parametry problemu, C – kryteria problemu, D – charakterystyka rozmieszczenia (D – dyskretne, C – ciągłe, Tree – na drzewie, Graph – na grafie), E – kryterium pośrednie dla podproblemu rozmieszczenia maszyn.

Podstawowy koncept łącznego problemu wyboru współrzędnych położenia dla jednej maszyny i szeregowania zadań z kryterium długości uszeregowania $1|r_j(y)|C_{\max}|\mathrm{D}$ został zaproponowany przez Hennes & Hamacher (2002). Autorzy sformułowali problem, w którym termin gotowości każdego zadania jest równy odległości pomiędzy jego daną lokalizacją i wybraną, spośród skończonej liczby potencjalnych pozycji, przez decydenta lokalizacją maszyny. Formuła $r_j(y)$ oznacza, że termin gotowości zadania zależy od zmiennej optymalizacyjnej y, która opisuje rozmieszczenie maszyn. Problem można rozwiązać w czasie wielomianowym korzystając z metody ERD oraz pełnego przeglądu dostępnych lokalizacji.

Hennes & Hamacher (2002) sformułowali problem $1|r_j(y)|C_{\max}|\mathrm{Graph}$, w którym zadania mogą przemieszczać się wzdłuż wybranych ścieżek na nieskierowanym grafie. Zaproponowano dwie wersje problemu: maszyna może być umieszczona na jednym z wierzchołków grafu (wersja *node*) lub w dowolnym miejscu na krawędzi grafu (wersja *absolute*). Wskazano, że problemy $1|r_j(y)|C_{\max}|\mathrm{Graph}$ (wersja *node*) oraz $1|r_j(y)|C_{\max}|\mathrm{Tree}$ można rozwiązać optymalnie stosując przegląd wszystkich możliwych wierzchołków, w których można ulokować maszynę oraz metodę ERD. Opracowano dwa algorytmy konstrukcyjne łączące wyszukiwanie najkrótszych ścieżek pomiędzy wierzchołkami grafu (lub drzewa) i identyfikację regionów Webera, aby uzyskać optymalne decyzje dla problemu $1|r_j(y)|C_{\max}|\mathrm{Graph}$ (wersja *absolute*) oraz analogicznego problemu $1|r_j(y)|C_{\max}|\mathrm{Tree}$ zorientowanego na drzewie. Elvikis et al. (2007) sformułowali problem $1|r_j(y)|C_{\max}|\mathrm{C}$ dopuszczający wybór dowolnych współrzędnych w obrębie ustalonego obszaru w dwuwymiarowej przestrzeni Euklidesowej. Opracowano algorytm optymalny dzielący dany obszar metodą bisekcji przy jednoczesnym uwzględnieniu współrzędnych rozmieszczenia zadań. Następnie, dla wskazanej pozycji, problem $1|r_j(y)|C_{\max}$ jest rozwiązywany metodą ERD. Pracę nad problemem $1|r_j(y)|C_{\max}|\mathrm{C}$ kontynuowali Elvikis et al. (2009). Autorzy zaproponowali algorytmy łączące rozwiązywanie problemów programowania liniowego dla podproblemów rozmieszczenia (sformułowania równań bazują na własnościach geometrycznych obszaru) oraz metodę ERD w procesie szeregowania. Kalsch & Drezner (2010) opracowali nowe algorytmy konstrukcyjne dla problemów $1|r_j(y)|C_{\max}|\mathrm{C}$ oraz $1|r_j(y)|\sum C_j\,|\mathrm{C}$. Lenstra, Kan & Brucker (1977) udowodnili, że problem $1|r_j|\sum C_j$ jest silnie NP-trudny dla jednej maszyny. Do rozwiązania obu problemów wykorzystano triangulacje Delaunay'a do identyfikacji obszarów, w których można

ulokować maszynę. W obszarze każdego wygenerowanego trójkąta decyzja o wyborze lokalizacji jest efektem działania algorytmu podziału i ograniczeń. Ostatecznie wybierana jest lokalizacja maszyny, która zapewnia najmniejszą długość uszeregowania po przypisaniu maszyn zgodnie z metodą ERD. Badania nad nowymi algorytmami rozwiązania problemu $1|r_j(y)|C_{\max}|\mathrm{C}$ i pod kątem analizy teoretycznej kontynuował Scholz (2012). Model z uniwersalnym kryterium $\sum w_j \mathrm{F}(C_j)$, gdzie F jest niemalejącą funkcją wprowadził Kaufmann (2014). Przeprowadzono analizę szczególnej wersji problemu $1|r_j(y)$,pmtn$|\sum w_j \mathrm{F}(C_j)|$Graph zorientowanego na grafie z możliwymi wywłaszczeniami zadań i wykazano, że problem tej postaci dla dowolnej funkcji F może być rozwiązany w czasie wielomianowym stosując regułę najkrótszego pozostałego czasu przetwarzania (ang. Shortest Remaining Processing Time Rule). Krumke & Le (2020) sformułowali niedeterministyczny problem $1|r_j^- \le r_j \le r_j^+|C_{\max}|$Tree, w którym zadania są umieszczone na wierzchołkach drzewa a długości krawędzi drzewa są dane w postaci niepewnych (przedziałowych) wartości. W rezultacie termin gotowości zadania, które musi dotrzeć (wybraną ścieżką) do maszyny ulokowanej przez decydenta na jednej pozycji na drzewie jest nieprecyzyjny. Założono, że ograniczenia nałożone na przedziały niepewności spełniają warunki Gamma-odporności. Autorzy opracowali algorytm konstrukcyjny, który dokonuje wyboru lokalizacji równoważących czasy przemieszczeń zadań do maszyny.

Założenie o rozmieszczeniu wielu identycznych maszyn równoległych w sformułowaniu problemu ScheLoc zostało po raz pierwszy przyjęte przez Rajabzadeh et al. (2016). Autorzy zdefiniowali zawansowane modele mieszanego programowania całkowitoliczbowego dla dyskretnego ($Pm|r_j(y)|C_{\max}|\mathrm{D}$) oraz ciągłego ($Pm|r_j(y)|C_{\max}|\mathrm{C}$) problemu rozmieszczenia maszyn oraz szeregowania zadań. Badania przeprowadzono solverem GAMS, aby zweryfikować jakość zdefiniowanych modeli dla bardzo małych instancji problemu (nie więcej niż 10 zadań, 3 maszyny, 6 potencjalnych pozycji). Heßler & Deghdak (2017) kontynuowali prace nad $Pm|r_j(y)|C_{\max}|\mathrm{D}$ pod kątem analizy teoretycznej, opracowano algorytmy heurystyczne oraz sformułowano wydajny model mieszanego programowania całkowitoliczbowego. Na problem ScheLoc składają się co najmniej dwa różne podproblemy optymalizacyjne: rozmieszczenie maszyn i szeregowanie zadań. Autorzy pracy przyjęli dwuetapowy (sekwencyjny lub iteracyjny) proces optymalizacji. Heßler & Deghdak (2017) opracowali algorytmy rozwiązania problemu wykorzystujące klasteryzację (dla problemu rozmieszczenia maszyn) oraz metodę ERD (dla problemu szeregowania zadań) z możliwą korektą harmonogramu po procesie optymalizacji lub zastosowaniem operatora

przeszukiwania lokalnego. Pierwszy algorytm wskazuje pozycje dla maszyn na podstawie arbitralnych kryteriów rozmieszczenia (uwzględniających czasy przetwarzania zadań, łączną lub sumę odległości pomiędzy maszynami i zadaniami), aby w kolejnym etapie zastosować zmodyfikowaną metodę ERD (zadania są przypisywane na maszynach równoległych według kryterium gotowości do realizacji). Drugi algorytm heurystyczny (konstrukcyjny) wyznacza klastry na podstawie środków ciężkości obszaru, wybierając losowy podzbiór współrzędnych rozmieszczenia zadań (liczba klastrów jest równa liczbie maszyn; wybiera się pozycję najbliższą środkowi klastra), oraz metodę ERD (niezależnie dla zadań przynależących do wyznaczonego klastra) do czasu braku poprawy rezultatu optymalizacji. Pozostałe algorytmy bazują na naprzemiennie uruchamianych, opisanych wyżej, dwóch algorytmach.

Wang et al. (2020) wykorzystali solver CPLEX do optymalnego rozwiązania problemu $Pm|r_j(y)|C_{\max}|$D. Dla założonej odgórnie długości uszeregowania (stanowiącej ograniczenie w sformułowaniu) uruchamiano solver. W przypadku braku możliwości otrzymania wyniku długość uszeregowania jest stopniowo zwiększana aż do momentu, kiedy uzyskanie rezultatu stanie się możliwe. Drugi algorytm bazuje na dekompozycji problemu na rozmieszczenie maszyn (z użyciem klasteryzacji) i szeregowanie zadań (zgodnie z zasadą najwcześniejszego terminu zakończenia przetwarzania). Kontynuacja badań nad problemem $Pm|r_j(y)|C_{\max}|$D, prowadzona przez Kramer & Kramer (2021; 2022) i Li et al. (2021), obejmuje sformułowanie problemu w postaci Arc-flow, opracowanie nowych modeli mieszanego programowania liniowego oraz propozycje kolejnych heurystyk. Piasecki & Józefczyk (2018) i Piasecki (2018) jako jedyni rozważyli problem $Pm|r_j(y)|\sum C_j|$C z m maszynami i ciągłym charakterem rozmieszczenia maszyn dla kryterium sumy terminów zakończeń realizacji zadań. Autorzy zaproponowali algorytmy optymalizacyjne wykorzystujące obliczenia ewolucyjne lub bazujące na dwuetapowym procesie podejmowania decyzji łączącym triangulację lub symetryczny podział regionu (do wskazania pozycji maszyn) i przypisanie według metody ERD. Ławrynowicz & Józefczyk (2019) jako pierwsi uwzględnili maszyny dowolne w sformułowaniu problemu ScheLoc oraz zaprojektowali algorytm memetyczny z operatorem lokalnej optymalizacji bazującym na metaheurystyce Tabu Search. Przeprowadzono badanie mające na celu porównanie sekwencyjnego podejścia do problemu, z kryterium pośrednim m-median dla rozmieszczenia maszyn, oraz podejścia łącznego (bez kryterium pośredniego). 김동욱 (2021) zorientował problem $Pm|r_j(y)|$Cost|D w obszarze dystrybucji paczek za pomocą dronów. Zadaniem decydenta jest wybór m

lokalizacji (centrów dystrybucji, z których wyruszają drony dostarczając paczki do klienta), wskazanie liczby dronów mających operować w uruchomionych centrach oraz opracowanie harmonogram dystrybucji paczek. Realizacja zadania jest równoznaczna z wysłaniem drona do wyznaczonego klienta, wykonaniem usługi oraz powrotem drona do centrum. Kryterium jest wyrażone jako suma kosztów uruchomienia centrów oraz łączna suma kosztów realizacji dostaw. 김동욱 (2021) wykorzystał dekompozycję Dantziga–Wolfa do rozwiązania problemu konstrukcyjną heurystyką w dwóch etapach: rozmieszczenia centrów oraz budowy harmonogramu dostaw.

Badania literaturowe w dziedzinie problemu ScheLoc z niepewnościami skupiają się na modelowaniu czasów realizacji zadań $p_j(\omega)$, które są dane w postaci rozkładów prawdopodobieństwa. Konsekwencją takiego założenia jest optymalizacja stochastyczna. Bazowy problem $Pm|r_j(y), d_j, p_j(\omega)|\mathrm{Cost}|\mathrm{D}$ sformułowali Liu et al. (2019b). Celem optymalizacji jest minimalizacja kosztu uruchomienia systemu. Aby rozwiązać problem solverem CPLEX, autorzy zaproponowali model, w którym wykorzystano nierówność Bonferroni'ego do oszacowania i dekompozycji wartości wybranych ograniczeń. Koncepcja czasów realizacji zadań modelowanych rozkładami prawdopodobieństwa została wykorzystana przez Liu et al. (2019) oraz Liu & Liu (2019a) w dwukryterialnym problemie $Pm\left|r_j(y), p_j(\omega)\right| \sum w_j \mathrm{Cost}_j$ & $\mathrm{E}\left[\sum C_j(\omega)\right]|\mathrm{D}$. Kryterium wyrażone z użyciem techniki skalaryzacji ma postać sumy ważonych kosztów wyboru lokalizacji i wartości oczekiwanej sumy terminów zakończeń realizacji zadań. W referacie konferencyjnym opracowanym przez Liu & Liu (2019a) model problemu sformułowano jako dwuetapowy problem optymalizacji stochastycznej. Opracowano dwa modele. W pierwszym uwzględniono awersję do ryzyka (adaptując koncept Expected shortfall; Conditional Value at Risk) natomiast drugi bazuje na technice *Sample average approximation* (SAA). Liu et al. (2019) zaproponowali nowe algorytmy rozwiązania problemu: zmodyfikowane podejście SAA, algorytm genetyczny, algorytm heurystyczny dekomponujący problem ScheLoc na podproblemy rozlokowania maszyn (rozwiązywany solverem CPLEX) i szeregowania zadań, które jest realizowane zmodyfikowaną regułą SPT (ang. shortest processing time). Liu et al. (2023) sformułowali złożony problem $Pm|r_j(y), \tilde{p}_j(\omega), d_j, reject|\mathrm{Profit}|\mathrm{D}$, w którym uwzględnili ryzyko związane z realizacją płatności przez klienta za wykonanie zadanie. Czasy przetwarzania zadań nie są precyzyjnie określone. Czas przetwarzania zadania $\tilde{p}_j(\omega)$ jest reprezentowany przez dwie wartości: kowariancję oraz średnią. Celem podejmowania decyzji jest

rozmieszczenie maszyn i przypisanie zadań w taki sposób, aby zmaksymalizować prawdopodobieństwo zysku przy uwzględnieniu ryzyka wypłacalności klientów oraz kar wynikających z niedotrzymania wymaganych terminów zakończeń przetwarzania zadań lub odrzucenia ich realizacji. Autorzy jako jedyni w literaturze problemu ScheLoc dopuścili możliwość umieszczania wielu maszyn w tej samej lokalizacji. Do rozwiązania problemu opracowano algorytm na technice *Sample average approximation* (SAA) oraz konstrukcyjną heurystykę.

Wielokryterialne podejście do problemu ScheLoc stało się popularnym przedmiotem badań w dziedzinie. Problem ScheLoc z kryterium kosztu rozmieszczenia maszyn oraz sumy terminów zakończeń przetwarzania zadań $Pm|r_j(y)|\mathrm{Cost}\ \&\ \sum C_j\ |\mathrm{D}$ został sformułowany przez Filcek et al. (2021). Do rozwiązania wykorzystano algorytm ewolucyjny NSGA-II, zaprojektowano nowy wskaźnik oceny jakości rozwiązań bazujący na punkcie nadir oraz szczegółowo omówiono studium przypadku ewakuacji ludzi z terenów zagrożonych. Ławrynowicz & Filcek (2021) przedstawili adaptację metaheurystyki Simulated Annealing do rozwiązania omawianego dwukryterialnego problemu. Wu et al. (2022) opracowali nowy model matematyczny dla $Pm|r_j(y)|\mathrm{Cost}\ \&\ \sum C_j\ |\mathrm{D}$ oraz zaproponowali nowe metody jego rozwiązania (metodę ε-ograniczeń oraz algorytm bazujący na logice rozmytej). Zhang et al. (2022a) sformułowali wielokryterialny (kryterium wyrażono w formie skalaryzacji) problem harmonogramowania wizyt na szczepienia $Pm|r_j(y), reject|\mathrm{Multi}|\mathrm{D}$. W artykule przyjęto następujące kryteria: koszt uruchomienia punktów szczepień, koszt dojazdów na szczepienia do wybranych punktów, koszt odwołania umówionych wizyt oraz ważone koszty kar za spóźnienia. Odwołanie wizyt, gdy osoby nie dotarły na szczepienie w określonym oknie czasowym, wymusza zmianę harmonogramu szczepień lub odrzucenie obsługi pacjenta w danym punkcie. Wprowadzenie kar za spóźnienia definiuje nie wprost wymagane okna czasowe realizacji zadań. Problem zdekomponowano wykorzystując metodę Bendera (ang. logic-based Bender's decomposition). Algorytm rozwiązania problemu, w podproblemie głównym, wybiera lokalizacje dla punktów szczepień oraz przypisuje im podzbiory osób do szczepień bazując na algorytmie branch-and-cut przy oszacowanej i ustalonej wartości dolnego ograniczenia dla kryterium ważonych kar. Następnie dla ustalonych wcześniej decyzji, zadania są szeregowane niezależnie w każdym punkcie, aby minimalizować ważone koszty kar za spóźnienia (taki problem oznaczamy w notacji Grahama jako $1|r_j(y)|\sum w_j T_j$). Do ustalenia sekwencji zadań wykorzystano programowanie dynamiczne. Dodatkowo opracowano algorytm hybrydowy łączący programowanie

matematyczne z podejściem heurystycznym. Algorytm rozwiązuje zrelaksowany problem programowania liniowego (np. solverem CPLEX) pozwalający wskazać punkty szczepień oraz przypisać im podzbiory zadań. Następnie zadania są sekwencjonowanie niezależnie we wskazanych lokalizacjach zgodnie z rozwiązaniem problemu $1|r_j(y)|\sum w_j T_j$. Zhang et al. (2022b) rozszerzyli poprzedni problem do $Pm|r_j(y), d_j|\sum w_j \mathrm{Cost}_j$ & $\sum T_j$ |D. Przyjęto kryterium ważonej sumy kosztów uruchomienia maszyn w wybranych lokalizacjach, kosztów transportu zadań oraz sumy kar doliczanych indywidualnie dla każdego zadania jedynie w przypadku niedotrzymania odgórnie wymaganego najpóźniejszego terminu dotarcia zadania do wybranej lokalizacji. Algorytm rozwiązania jest zmodyfikowaną wersją algorytmu dla problemu $Pm|r_j(y), reject|$Multi|D. Zaproponowano dekompozycję problemu na podproblemy rozmieszczenia maszyn oraz sekwencjonowania zadań. Sformułowano nowe modele programowania liniowego dla problemu rozmieszczenia oraz, w drugim etapie optymalizacji, harmonogramy są projektowane w oparciu o rozwiązanie problemu $1|r_j(y)|\sum T_j$. Li et al. (2022) wykorzystali rozwiązanie dwukryterialnego problemu $Pm|r_j(y)|C_{\max}$ & $\sum r_j(y)$ |D do zaprojektowania harmonogramu testowania obywateli w celu wykrycia osób chorych na COVID. Intencją autorów jest taki dobór miejsc, w których można ulokować punkty testowania, aby łączny czas dotarcia do tych punktów był najmniejszy oraz cały proces testowania zakończył się jak najszybciej. Autorzy opracowali trzy algorytmy rozwiązania problemu: algorytm bazujący na technice ε-ograniczeń (wartość jednego kryterium stanowi ograniczenie dla decyzji stanowiących o wartości drugiego kryterium), algorytm iteracyjnie rozwiązujący problem programowania matematycznego gwarantujący uzyskanie przybliżonego rozwiązania oraz algorytm bazujący na dwukryterialnej dekompozycji Bendera. Sformułowanie problemu $Pm|r_j(y)|\mathrm{E}\left[\sum C_j \,\&\, \sum r_j(y)\right]$|D zorientowanego w obszarze opieki zdrowotnej zaproponowali Fu et al. (2022). Wdrożono koncept maszyn (usługodawców zabiegów medycznych) przemieszczających się do miejsc realizacji zadań (odbiorcy usług medycznych). Przyjęto niepewne czasy przemieszczania się maszyn oraz wprowadzono minimalne wymagania, które musi spełnić maszyna, aby móc zrealizować powierzonej jej zadanie. Zaprojektowano hybrydowe podejście do rozwiązania problemu łączące algorytm genetyczny oraz symulację stochastyczną. Buckhorst, Canto & Schmitt (2022) sformułowali przepływowy problem $Fm|r_j(y)|C_{\max}$|D dedykowany realizacji zadań w fabrykach, których linie produkcyjne charakteryzują się częstą zmianą struktury. Elastyczność w produkcji jest wymagana do efektywnej produkcji różnych części w obrębie tej samej fabryki. Na każde zadanie składa

się sekwencja operacji do wykonania w obrębie wybranego systemu. Wykorzystano podstawowe sformułowanie problemu ScheLoc z kryterium długości uszeregowania do określenia tras realizacji zadań w systemie oraz ich uszeregowanie. Autorzy pracy opracowali bardzo zaawansowany model matematyczny problemu, który został zaimplementowany w solverze Gurobi do rozwiązania wygenerowanych instancji.

1.5. Motywacja, cele cząstkowe oraz tezy pracy

Przegląd literatury uwidocznił brak badań nad problemami szeregowania zadań z nieustalonymi terminami gotowości na maszynach dowolnych w wersji deterministycznej z kryterium długości uszeregowania i w wersji niedeterministycznej z kryterium maksymalnego żalu dla długości uszeregowania. W literaturze nie przeprowadzono jak dotąd analizy własności i nie opracowano algorytmów dla deterministycznego problemu szeregowania z kryterium długości uszeregowania i terminami gotowości zależnymi od maszyn dowolnych. W takim przypadku parametry dowolnego zadania tzn. termin gotowości i czas przetwarzania wynikają z decyzji o uszeregowaniu. Kluczowym aspektem badań nad rozważanymi problemami optymalizacji odpornej jest porównanie efektywności zachłannych strategii podejmowania decyzji, dekompozycji kryterium oraz przeglądu losowego (bazującego na metaheurystykach). Ponadto przeprowadzenie analizy najgorszego scenariusza ułatwia zrozumienie natury rozważanych problemów optymalizacji odpornej. Ciekawą kwestią jest zbadanie czy algorytm wykorzystujący deterministyczne kryterium długości uszeregowania oraz własności problemu niedeterministycznego może zwracać uszeregowania, które są w stanie zapewnić mniejszą wartość maksymalnego żalu niż algorytmy podejmujące decyzje w oparciu o kryterium maksymalnego żalu.

Opracowane algorytmy szeregowania stanowią podstawę dla algorytmów rozwiązania złożonego łącznego i nierozpatrywanego dotąd w literaturze problemu rozmieszczenia maszyn i szeregowania zadań (ScheLoc) z kryterium długości uszeregowania dla maszyn dowolnych. Innowacyjnym aspektem badawczym jest wykorzystanie nieustaloności terminów gotowości zadań do rozwiązania problemu ScheLoc. Porównanie podejścia sekwencyjnego (dwuetapowego) z kryterium pośrednim dla podproblemu rozmieszczenia maszyn i systemowego (łącznego) do rozwiązania problemu Scheloc stanowi również innowacyjny aspekt badawczy. Wówczas nieustaloność terminów gotowości wynika z konieczności rozwiązania podproblemu pośredniego. W literaturze nie pojawiły się jak dotąd pozycje, w których intencjonalnie byłyby przyjęte niepewności parametrów wejściowych problemu ScheLoc celem zbadania jakości uszeregowań w problemie

deterministycznym z precyzyjnie określonymi parametrami wejściowymi. Jest to cel cząstkowy pracy, który pozwala określić jaki wpływ na wartość kryterium długości uszeregowania ma podejmowanie decyzji w warunkach niepewności. W rozważanym przypadku nieustaloność terminów gotowości jest konsekwencją nieprecyzyjności momentów gotowości, reprezentowanej w postaci przedziałów wartości. Przeprowadzenie takich badań wymaga opracowania metod definiowania niepewności przedziałowych na podstawie precyzyjnie określonych parametrów wejściowych problemu ScheLoc.

Celem podstawowym rozprawy jest opracowanie i przebadanie algorytmów szeregowania dla problemów $Rm|r_{i,j}|C_{\max}$, $Rm|r_j^- \leq r_j \leq r_j^+|Reg(C_{\max})$, $Rm|r_{i,j}^- \leq r_{i,j} \leq r_{i,j}^+|Reg(C_{\max})$ oraz dla problemu ScheLoc z kryterium długości uszeregowania i dyskretnym podproblemem rozmieszczenia maszyn dowolnych. Cel podstawowy został osiągnięty poprzez realizację następujących celów cząstkowych:

1. Zbadano deterministyczny problem szeregowania zadań $Rm|r_{i,j}|C_{\max}$.
   1.1. Przeanalizowano podstawowe własności oraz wskazano przypadki, które można rozwiązać w czasie wielomianowym i pseudowielomianowym.
   1.2. Zaproponowano metody oszacowania wartości dolnego ograniczenia kryterium długości uszeregowania.
   1.3. Opracowano trzy algorytmy rozwiązania problemu, implementujące różne strategie podejmowania decyzji: zachłanną z planowaniem długoterminowym dla decyzji o takiej samej jakości, wyczerpującą dla podproblemów oraz opartą na metaheurystyce Simulated Annealing.
2. Zbadano dwa niedeterministyczne problemy szeregowania zadań z przedziałowymi terminami gotowości: $Rm|r_{i,j}^- \leq r_{i,j} \leq r_{i,j}^+|Reg(C_{\max})$ i $Rm|r_j^- \leq r_j \leq r_j^+|Reg(C_{\max})$.
   2.1. Wykazano analitycznie różnicę w interpretacji obu problemów oraz wskazano, że najgorszy scenariusz może być jednocześnie scenariuszem skrajnym i scenariuszem pośrednim.
   2.2. Opracowano dwa algorytmy konstrukcyjne oraz algorytm oparty na metaheurystyce Tabu Search dla problemu $Rm|r_{i,j}^- \leq r_{i,j} \leq r_{i,j}^+|Reg(C_{\max})$.
   2.3. Opracowano algorytm konstrukcyjny dla problemu $Rm|r_j^- \leq r_j \leq r_j^+|Reg(C_{\max})$, który podejmuje decyzje w oparciu o zastępcze kryterium długości szeregowania.
3. Zbadano łączny problem rozmieszczenia maszyn dowolnych i szeregowania zadań na maszynach dowolnych z kryterium długości uszeregowania (ScheLoc).

3.1. Opracowano algorytm hybrydowy (łączący strategię zachłanną oraz metaheurystykę Simulated Annealing) oraz algorytm genetyczny.

3.2. Zdefiniowano dwa modele zdekomponowanego problemu Scheloc z kryterium pośrednim dla podproblemu rozmieszczenia maszyn dowolnych.

3.3. Zbadano możliwość zastąpienia deterministycznego problemu ScheLoc algorytmami niedeterministycznymi opracowanymi w ramach celów cząstkowych.

4. Dla celów cząstkowych wskazanych w punktach 1, 2 i 3 przeprowadzono badania symulacyjne w celu eksperymentalnej oceny działania algorytmów oraz ich porównania, a także badania statystyczne dla większości przypadków.

Wymienione cele cząstkowe stanowią o oryginalności pracy i nie zostały podjęte w literaturze.

Wyniki realizacji wskazanych celów cząstkowych uzasadniają następującą złożoną tezę pracy:

1. **Zastosowanie zachłannej strategii podejmowania decyzji w problemie $Rm|r_{i,j}|C_{\max}$ gwarantuje statystycznie mniejszą długość uszeregowania niż dedykowane algorytmy oparte na przeglądzie wyczerpującym lub metaheurystyce Simulated Annealing.**
2. **Zastosowanie zachłannej strategii podejmowania decyzji w problemach $Rm|r_j^- \le r_j \le r_j^+|Reg(C_{\max})$ i $Rm|r_{i,j}^- \le r_{i,j} \le r_{i,j}^+|Reg(C_{\max})$ gwarantuje statystycznie mniejszą wartość oszacowanego maksymalnego żalu niż algorytmy bazujące na dekompozycji kryterium lub determinizacji problemu.**
3. **Rozmieszczenie maszyn, które jest rozwiązaniem podproblemu m-median gwarantuje statystycznie mniejszą długość uszeregowania w sekwencyjnym (dwuetapowym) podejściu do rozwiązania problemu ScheLoc niż rozmieszczenie maszyn, które jest rozwiązaniem podproblemu m-center.**
4. **Istnieją szczególne przypadki rozpatrywanego w rozprawie a nierozważanego w literaturze problemu rozmieszczenia maszyn i szeregowania zadań (ScheLoc), w których algorytmy opracowane dla niedeterministycznych problemów szeregowania $Rm|r_j^- \le r_j \le r_j^+|Reg(C_{\max})$ i $Rm|r_{i,j}^- \le r_{i,j} \le r_{i,j}^+|Reg(C_{\max})$ w połączeniu z przeglądem zupełnym dla podproblemu rozmieszczenia maszyn zwracają mniejszą długość uszeregowania niż algorytmy opracowane dla wersji deterministycznej (łącznej) problemu ScheLoc.**

Taka rozbudowana forma tezy jest spowodowana wielowątkowością rozprawy doktorskiej. Formułowanie jednej syntetycznej tezy uniemożliwiłoby ukazanie w jej ramach specyfiki podjętych badań. Należy jeszcze raz podkreślić, że pierwotną motywacją pracy było przekonanie o potrzebie sprawdzenia, czy jest uzasadnione wykorzystanie narzędzi optymalizacji odpornej z niepewnością przedziałową do rozwiązywania złożonego problemu optymalizacyjnego ScheLoc. Teza w formie syntetycznej dla rozpatrywanego problemu ScheLoc, brzmiałaby następująco: **W szczególnych przypadkach jest uzasadnione wykorzystanie narzędzi optymalizacji odpornej z niepewnością przedziałową do rozwiązywania złożonego problemu optymalizacyjnego ScheLoc.** Takie sformułowanie nawiązuje wprost do tezy szczegółowej nr 4. Jej uzasadnienie wymagało jednak rozwiązania szeregu problemów szczegółowych, które mają samoistne znaczenie i są powiązane z pozostałymi tezami cząstkowymi.

1.6. Przegląd treści pracy

Praca jest podzielona na dwie logicznie powiązane ze sobą części. Na pierwszą część pracy składają się rozdziały 2 i 3, w których sformułowano problemy szeregowania zadań z nieustalonymi terminami gotowości zadań na maszynach dowolnych (w wersji deterministycznej z kryterium długości uszeregowania oraz w wersji niedeterministycznej z kryterium maksymalnego żalu dla długości uszeregowania), przeprowadzono analizę teoretyczną własności problemów oraz przedstawiono opracowane algorytmy ich rozwiązania. Rezultaty prac nad problemami szeregowania z części pierwszej wykorzystano w drugiej części pracy doktorskiej, czyli w rozdziale 4, gdzie szczegółowo przebadano wybrany problem rozmieszczenia maszyn i szeregowania zadań (ScheLoc). Ponadto w rozdziałach 5 i 6 wskazano praktyczne obszary implementacji omawianych zagadnień, podsumowano uzyskane rezultaty oraz nakreślono możliwe kierunki dalszych prac.

2. Problem szeregowania zadań z terminami gotowości zależnymi od maszyn

2.1. Wprowadzenie

Celem prac, których wyniki są prezentowane w tym rozdziale, było przebadanie oryginalnego problemu szeregowania zadań $Rm|r_{i,j}|C_{\max}$ z terminami gotowości zależnymi od maszyn dowolnych. Rozważamy deterministyczną wersję problemu (terminy gotowości zadań są ustalone) z kryterium długości uszeregowania. Rozdział rozpoczyna się od wprowadzenia podstawowej notacji, terminologii oraz sformułowania modelu mieszanego programowania nieliniowego. Badania teoretyczne uwzględniają szczegółową analizę złożoności obliczeniowej problemu. W konsekwencji zaproponowano warunki gwarantujące możliwość optymalnego rozwiązania problemu. Dodatkowo zaproponowano pięć metod dolnego oszacowania wartości kryterium. W celu rozwiązania problemu $Rm|r_{i,j}|C_{\max}$ opracowano następujące algorytmy:

1. algorytm wykorzystujący przegląd zupełny (wyczerpujący) dla arbitralnie przyjętych niezależnych podzbiorów decyzji,
2. algorytm oparty na metaheurystyce Simulated Annealing,
3. algorytm konstrukcyjny wykorzystujący zachłanną strategię podejmowania decyzji z planowaniem długoterminowym dla decyzji o takiej samej jakości.

Ewaluacja wyników obejmuje porównanie efektywności optymalizacji opracowanych algorytmów oraz analizę jakości najlepszych rozwiązań w porównaniu do dolnego oszacowania wartości kryterium $C_{\max}$. Fragmenty badań umieszczonych w niniejszym rozdziale zostały opublikowane w Ławrynowicz & Józefczyk (2022a) oraz Ławrynowicz & Józefczyk (2022b).

2.2. Sformułowanie problemu $Rm|r_{i,j}|C_{\max}$

Niech $\mathrm{J} = \{1,2,\dots,j,\dots,n\}$ będzie zbiorem n niezależnych i niewywłaszczalnych zadań, które należy przetworzyć na m maszynach dowolnych danych zbiorem $\mathrm{M} = \{1,2,\dots,i,\dots,m\}$. W opracowanym sformułowaniu problemu termin gotowości $r_{i,j} \geq 0$ oraz czas przetwarzania dla zadania $p_{i,j} > 0$ zależą od maszyny i. Zarówno $r_{i,j}$ jak i $p_{i,j}$ są dodatnimi liczbami całkowitymi (wyrażonymi w jednostkach czasu), będącymi elementami macierzy $p = \left[p_{i,j}\right]_{\substack{i=1,2,\dots,m \\ j=1,2,\dots,n}}$ oraz $r = \left[r_{i,j}\right]_{\substack{i=1,2,\dots,m \\ j=1,2,\dots,n}}$. Rozpatrujemy przypadek,

gdy wartość terminu gotowości zadania j wyraża moment, w którym zadanie staje się możliwe do wykonania (przetworzenia) na maszynie i.

Prezentowane sformułowanie zakłada wprowadzenie n wirtualnych pozycji na każdej maszynie. Liczba n odpowiada skrajnemu przypadkowi, w którym wszystkie zadania są uszeregowane na tej samej maszynie. Binarna zmienna decyzyjna $x_{i,k,j} \in \{0,1\}$ wskazuje, czy zadanie j jest przypisane do pozycji k, $k = 1,2,\dots,n$, na maszynie i. Pełny harmonogram jest reprezentowany przez macierz binarną $x = \left[x_{i,k,j}\right]_{\substack{i=1,2,\dots,m \\ k,j=1,2,\dots,n}}$. Termin zakończenia zadania przypisanego do pozycji k w sekwencji zadań uszeregowanych na maszynie i można wyliczyć z wykorzystaniem równania rekurencyjnego

$$C_{i,k}(x;\mathrm{S}) = \sum_{j=1}^{n} x_{i,k,j}\left(p_{i,j} + \max\left\{C_{i,k-1}(x;\mathrm{S}), r_{i,j}\right\}\right), \quad i = 1,2,\dots,m, \quad k = 1,2,\dots,n, \quad C_{i,0}(x;\mathrm{S}) = 0 \tag{2.1}$$

oraz przyjmujemy, że na niewykorzystanej pozycji termin zakończenia zadania (nieprzypisanego) wynosi zero, czyli $\underset{\substack{i=1,2,\dots,m \\ k=1,2,\dots,n}}{\forall} C_{i,k}(x,r) = 0$ jeżeli $x_{i,k,j} = 0$, $\mathrm{S} = \{p,r\}$.

Długość uszeregowania definiujemy jako

$$C_{\max}(x;\mathrm{S}) = \max_{i=1,2,\dots,m} C_{i,n_i}(x;\mathrm{S}), \tag{2.2}$$

gdzie $n_i = \sum_{k=1}^{n}\sum_{j=1}^{n} x_{i,k,j}$ jest liczbą zadań przypisanych do maszyny i (długość sekwencji jedynek).

Sformułowanie problemu można przedstawić w następujący sposób: Dla danych J, M, p, r, deterministyczny problem szeregowania wymaga znalezienia optymalnej macierzy x^* minimalizującej długość uszeregowania

$$C_{\max}(x^*;\mathrm{S}) = \min_{x} C_{\max}(x;\mathrm{S}). \tag{2.3}$$

Na zmienną decyzyjną x nałożono następujące ograniczenia:

$$\sum_{i=1}^{m}\sum_{k=1}^{n} x_{i,k,j} = 1, \quad j = 1,2,\dots,n, \tag{2.4}$$

$$\sum_{j=1}^{n} x_{i,k,j} \le 1, \quad i = 1,2,\dots,m,\ k = 1,2,\dots,n, \tag{2.5}$$

$$x_{i,k+1,j} - x_{i,k,t} \le 0, \quad i = 1,2,\dots,m,\ k = 1,2,\dots,n-1, \tag{2.6}$$

$$x_{i,k,j} \in \{0,1\}. \tag{2.7}$$

Ograniczenie (2.4) gwarantuje, że każde zadanie zostanie uszeregowane. Na dowolnej pozycji może znajdować się co najwyżej jedno zadanie (2.5). Ograniczenie (2.6) ma charakter techniczny. Wymusza poprawne wyliczenie wartości (2.1)-(2.3) oraz umożliwia reprezentowanie harmonogramu w postaci ciągłych sekwencji jedynek. Dziedzina wartości pojedynczej decyzji jest dana w (2.7).

Krumke & Le (2020) podkreślili fakt, że długość uszeregowania w problemie $1|r_j|C_{\max}$ można wyliczyć wykorzystując czas zadania (nazywanego krytycznym), rozpoczynającego się najpóźniej w najwcześniejszym możliwym terminie jego gotowości oraz czasów realizacji zadań uszeregowanych po zadaniu krytycznym. W artykule nie przeprowadzono dowodu tej własności. Poniżej udowodniono, że problem $Rm|r_{i,j}|C_{\max}$ cechuje się tożsamą własnością.

Własność 2.1. *Kryterium długości uszeregowania w problemie* $Rm|r_{i,j}|C_{\max}$ *można zapisać w postaci*

$$C_{i_m,n_{i_m}}(x;\mathrm{S}) = \hat{r}_{i_m,j} + \sum_{t\in \mathrm{A}_{i_m,j}(x)} p_{i_m,t}, \tag{2.8}$$

gdzie $i_m = \arg\max_{i=1,2,\dots,m} C_{i,n_i}(x;\mathrm{S})$ *jest najdłużej pracującą maszyną,* $\hat{r}_{i_m,j}$ *jest najpóźniejszym terminem gotowości spełniającym równość*

$$\hat{r}_{i_m,j} = \max\left\{C_{i_m,k-1}(x;\mathrm{S}), \hat{r}_{i_m,j}\right\} \text{ przy decyzji } x_{i_m,k,j} = 1 \tag{2.9}$$

oraz zbiór

$$\mathrm{A}_{i_m,j}(x) = \left\{t \in \mathrm{J} \mid C_{i_m,k}(x;\mathrm{S}) > \hat{r}_{i_m,j} \wedge x_{i_m,k,t} = 1 \wedge (2.9), k = 1,2,\dots,n_{i_m}\right\} \tag{2.10}$$

zawiera zadania, które są uszeregowane od momentu $\hat{r}_{i_m,j}$.

Dowód. Zauważmy, że musi istnieć co najmniej jeden termin gotowości spełniający (2.9), ponieważ, przy dowolnym uszeregowaniu, przetwarzanie zadania na pierwszej pozycji w sekwencji zawsze rozpoczyna się w momencie jego gotowości. Warunek (2.9) gwarantuje zachowanie nierówności

$$r_{i_m,j_{l+1}} < C_{i_m,l}(x;\mathrm{S}), \qquad l = k, k+1, \dots, n_{i_m} - 1, \tag{2.11}$$

gdzie $r_{i_m,j_{k+1}}$ jest terminem gotowości zadania przypisanego na pozycji $k+1$ w sekwencji na maszynie i_m ($x_{i_m,k+1,j_{k+1}} = 1$). Zatem wszystkie zadania uszeregowanie od terminu $\hat{r}_{i_m,j}$ są gotowe do realizacji bez oczekiwania na ich terminy gotowości, co umożliwia zapisanie kryterium długości uszeregowania w postaci (2.8). Q.E.D.

Sformułowanie problemu (2.3) w postaci $C_{\max}(x^*;\mathrm{S}) = \hat{r}_{i_m^*,j^*} + \sum_{t\in \mathrm{A}_{i_m^*,j^*}(x^*)} p_{i_m^*,t}$, gdzie i_m^* oraz j^* są parametrami wynikającymi z optymalnego uszeregowania x^*, uzasadnia możliwość istnienia wielu równoważnych rozwiązań optymalnych. Przed terminem $\hat{r}_{i_m^*,j^*}$ mogą zostać uszeregowane zadania, których kolejność jest dowolna np. jeżeli suma czasów realizacji tych zadań i terminów ich gotowości nie przekracza $\hat{r}_{i_m^*,j^*}$.

2.3. Analiza złożoności obliczeniowej

Analiza złożoności obliczeniowej dużej liczby problemów szeregowania jest przeprowadzana z wykorzystaniem znanych problemów decyzyjnych na zbiorach liczb (ang. 3-partition problem, 2-partition problem, subset sum problem itp.; Garey & Johnson, 1979). Złożoność obliczeniową problemu (2.3) można uzasadnić metodą definiowania ograniczeń. Przyjęcie założeń $\underset{i\neq s, j\neq t}{\forall} r_{i,j} = r_{s,t}$ i $\underset{j\neq t}{\forall} p_{i,j} = p_{i,t}$ sprowadza (2.3) do problemu szeregowania na m identycznych maszynach równoległych $Pm||C_{max}$ (również, jeżeli $\underset{i,j}{\forall} r_{i,j} \neq 0$), który jest silnie NP-trudny dla trzech maszyn (specjalny przypadek 3-partition problem) oraz NP-trudny dla co najmniej dwóch maszyn (specjalny przypadek 2-partition problem) (Garey & Johnson, 1979; Pinedo, 2012). Ustalenie relacji pomiędzy czasami przetwarzania zadań a ich terminami gotowości umożliwia głębsze zrozumienie złożoności obliczeniowej problemu $Rm|r_{i,j}|C_{\max}$ oraz pomaga wyselekcjonować pewne instancje, które można rozwiązać w czasie wielomianowym. Niezależność macierzy r i p powoduje, że pozornie nieistotna zmiana niektórych wartości może mieć wpływ na przestrzeń optymalnych decyzji. Odnosząc się do wartości składowych macierzy r i p, omówimy trzy warunki, których spełnienie umożliwia optymalne rozwiązanie problemu $Rm|r_{i,j}|C_{\max}$ w czasie wielomianowym lub pseudowielomianowym.

1) *Wpływ nieproporcjonalnie szybko przetwarzającej maszyny na możliwość optymalnego rozwiązania problemu $Rm|r_{i,j}|C_{\max}$ w czasie wielomianowym.*

Nieproporcjonalnie szybko przetwarzająca maszyna oferuje najwcześniejsze terminy gotowości oraz najszybsze czasy przetwarzania wszystkich zadań. Jeżeli przyjmiemy, że maszyna i przetwarza zadania nieproporcjonalnie szybko, to spełnione są warunki

$$\begin{aligned} &\underset{j=1,2,\ldots,n}{\forall}\left(i = \arg\min_{s=1,2,\ldots,m} p_{s,j} \ \wedge \ i = \arg\min_{s=1,2,\ldots,m} r_{s,j}\right) \\ &\wedge \ \underset{j\neq t \wedge\, r_{i,j}<r_{i,t}}{\forall} \left(\left[r_{i,j}, r_{i,j}+p_{i,j}\right] \cap \left[r_{i,t}, r_{i,t}+p_{i,t}\right] = \emptyset\right). \end{aligned} \tag{2.12}$$

Koniunkcja w pierwszym kwantyfikatorze gwarantuje, że każde zadanie jest gotowe do realizacji w najwcześniejszym możliwym terminie na maszynie i oraz jest na niej przetwarzane w najkrótszym możliwym czasie. Rozłączność przedziałów w drugim warunku umożliwia przypisanie zdań w terminie ich gotowości. Weryfikację spełnialności warunków w (2.12) można zrealizować w czasie $\mathcal{O}(\max\{nm, n\log n\})$, gdzie przegląd parametrów zadań względem parametrów maszyn można wykonać w czasie $\mathcal{O}(nm)$, natomiast sprawdzenie rozłączności przedziałów na osi czasu, w których zadania są przetwarzane, wymaga realizacji operacji sortowania. Rozwiązaniem instancji problemu spełniającej warunki podane w (2.12) jest ustanowienie sekwencji na najwydajniejszej maszynie, której kolejność jest zdeterminowana przez metodę ERD. Złożoność czasowa metody ERD, wykorzystującej algorytm sortowania przez scalanie, wynosi $\mathcal{O}(n\log n)$. Efektem wystąpienia bardzo szybko przetwarzającej maszyny jest ignorowanie pozostałych maszyn w procesie podejmowania decyzji, co prowadzi do niezrównoważonego obciążenia poszczególnych maszyn podczas przetwarzania zadań. W rezultacie problem (2.3) można traktować jako szeregowanie na co najwyżej m maszynach.

2) *Wpływ nieproporcjonalnie późnych terminów gotowości jednego zadania na możliwość optymalnego rozwiązania problemu* $Rm|r_{i,j}|C_{\max}$ *w czasie wielomianowym*

Terminy gotowości jednego zadania w problemie $Rm|r_{i,j}|C_{\max}$ mogą decydować o trudności instancji. Niech parametry zadań i maszyn spełniają nierówność

$$\sum_{s\in \mathrm{M}} \sum_{j\in \mathrm{J}\setminus\{t\}} \left(r_{s,j} + \max_{s'=1,2,\dots,m} p_{s',j}\right) \leq r_{i,t}, \qquad i = 1,2,\dots,m, \tag{2.13}$$

to optymalną długość uszeregowania można uzyskać w czasie wielomianowym, wybierając i przypisując zadanie t na ostatniej pozycji w sekwencji ustanowionej na maszynie $i' = \arg\min_{i=1,2,\dots,m} r_{i,t} + p_{i,t}$. Rozwiązanie optymalne można uzyskać w czasie $\mathcal{O}(n^2m)$, ponieważ spełnialność (2.13) wymaga wyliczenia podwójnej sumy dla każdego z n zadań. Następnie dla wybranego zadania t wskazywana jest maszyna, która umożliwia jego najszybsze przetworzenie, w czasie $\mathcal{O}(m)$. Nierówność (2.13) gwarantuje, że długość uszeregowania jest niezależna od kolejności uszeregowania zadań w zbiorze $\mathrm{J}\setminus\{t\}$, ponieważ realizacja dowolnego zadania zakończy się przed terminem $r_{i,t}$, $i = 1,2,\dots,m$. Ciekawą konsekwencją omawianej własności jest możliwość uproszczenia dowolnej instancji poprzez modyfikację terminów gotowości jednego zadania.

3) *Istnieją instancje problemu* $R2|r_{i,j}|C_{\max}$*, które można rozwiązać w czasie pseudowielomianowym.*

Niech $\mathrm{L} = \{t, l\}$ będzie zbiorem zawierającym dwa zadania, których parametry gwarantują najwcześniejsze możliwe zakończenie na różnych maszynach

$$\arg\min_{i=1,2}\left(r_{i,t} + p_{i,t}\right) \neq \arg\min_{i=1,2}\left(r_{i,l} + p_{i,l}\right) \tag{2.14}$$

oraz najpóźniejszą dostępność na pozostałych (różnych) maszynach

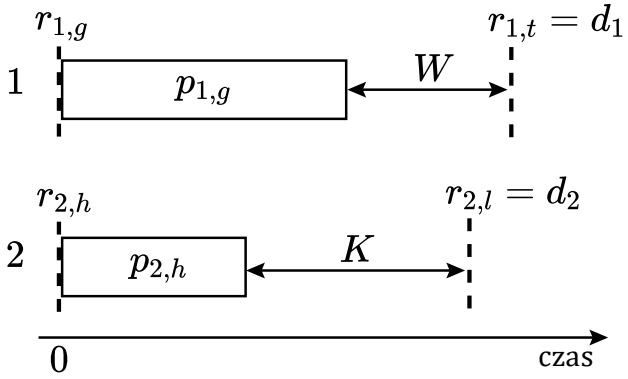
$$r_{i,t} \geq r_{i,l} > r_{i,j},\ i = 1,2,\ j = 1,2,\dots,m,\ j \neq t,\ j \neq l. \tag{2.15}$$

Niech $x^{(\mathrm{L})}$ oznacza harmonogram realizacji zadań zdefiniowanych w zbiorze J\{L} oraz niech zadania t i l najwcześniej kończą się na maszynach różnych maszynach $1 = \arg\min_{i=1,2}\left(r_{i,t} + p_{i,t}\right)$, $2 = \arg\min_{i=1,2}\left(r_{i,l} + p_{i,l}\right)$. Opierając się na (2.14) oraz (2.15) można zauważyć, że jeżeli zadania zdefiniowane w J\{L} są uszeregowane w taki sposób, że realizacja ostatniego zadania na obydwu maszynach zakończy się przed terminami $r_{1,t}$, $r_{2,l}$, to optymalna wartość kryterium długości uszeregowania zależy jedynie od parametrów zadań w zbiorze L. Przypisanie zadań t i l można zrealizować w stałym czasie $\mathcal{O}(1)$, ponieważ należy uszeregować dwa zadania na dwóch maszynach. Bazując na terminologii używanej w teorii szeregowania, terminy gotowości $r_{1,t}$, $r_{2,l}$ interpretujemy jako wymagane terminy zakończeń d_1, d_2 na maszynach dla zadań w zbiorze J\{L}. Weryfikacja możliwości zaprojektowania $x^{(\mathrm{L})}$ stanowi oryginalny decyzyjny problem szeregowania $R2|r_{i,j}, d_i|C_{\max}$.

Następnie wskażemy dwa szczególne przypadki problemu decyzyjnego $R2|r_{i,j}, d_i|C_{\max}$, które można rozwiązać w czasie pseudowielomianowym. Ograniczamy się do zbioru zadań J\{L} wraz z ich czasami przetwarzania oraz terminami gotowości oraz dwóch wymaganych terminów zakończeń d_1 i d_2.

W pierwszym przypadku zakładamy, że $\forall_{i \neq s} r_{i,j} = r_{s,t}$, $\forall_{i \neq s} p_{i,j} = p_{s,j}$ dla wszystkich zadań w zbiorze J\{L} oraz $\sum_{j \in \mathrm{J}\setminus\{\mathrm{L}\}} p_j = d_1 + d_2$, $p_j = p_{i,j}$. Taki problem można zapisać w notacji Grahama jako $R2|d_i|C_{\max}$. Jest to problem optymalizacyjny, w którym optymalnym rozwiązaniem jest uszeregowanie zadań w taki sposób, aby czasy ich realizacji mieściły się w oknach czasowych $[0, d_1]$ i $[0, d_2]$. Celem jest znalezienie podzbioru zadań, których czasy realizacji na jednej maszynie sumują się do d_1, co wymusi sumę czasów realizacji pozostałych zadań równą d_2 na drugiej maszynie. Wówczas rozwiązanie

rozważanego problemu szeregowania jest równoważne rozwiązaniu problemu sumy podzbioru, który jest NP-zupełny (Garey & Johnson, 1979). Należy podkreślić, że problem sumy podzbioru można rozwiązać optymalnie w czasie pseudowielomianowym za pomocą algorytmu implementującego programowanie dynamiczne (Koiliaris & Xu, 2019).



Rysunek 2.1. Instancja problemu $R2|r_{i,j}, d_i|C_{\max}$

W drugim przypadku również wskażemy instancję problemu, której rozwiązanie sprowadza się rozwiązania problemu sumy podzbioru. Instancję problemu przedstawiono na rysunku 2.1. Przyjmijmy, że w systemie zdefiniowano dwa zadania g i h, $\{g,h\} \subset \mathrm{J}\backslash\{\mathrm{L}\}$, dla których zachodzi (2.14), tzn. $1 = \arg\min_{i=1,2}(r_{i,g} + p_{i,g})$, $2 = \arg\min_{i=1,2}(r_{i,h} + p_{i,h})$ oraz g i h są dostępne najwcześniej na różnych maszynach (przyjęto, że są to maszyny gwarantujące najwcześniejsze zakończenie g i h). Wymuszenie przypisania zadań g i h na pierwszych pozycjach w sekwencjach ustanowionych na różnych maszynach jest konsekwencją założeń $p_{1,h} > d_1$ oraz $p_{2,g} > d_2$. Wówczas, gdy terminy zakończeń zadań g i h przekraczają terminy gotowości wszystkich pozostałych zadań ($\forall_{j\in \mathrm{J}\backslash\{g,h\} \wedge i\in\{1,2\}} r_{i,j} > r_{i,t} + p_{i,t}$, $t \in \{g,h\}$), $\forall_{i\neq s} p_{i,j} = p_{s,j}$ oraz $\sum_{j\in \mathrm{J}\backslash\{\mathrm{L},g,h\}} p_j = W + K$, to problem sprowadza się do $R2|d_i|C_{\max}$, ponieważ zadania $\mathrm{J}\backslash\{\mathrm{L}, g, h\}$ są gotowe do przetwarzania bez oczekiwania od razu po zakończeniu g oraz h. Optymalne rozwiązanie problemu wymaga uszeregowania podzbioru zadań w taki sposób, że suma czasów ich przetwarzania jest równa W na maszynie 1 (co wymusi sumę czasów przetwarzania pozostałych zadań na maszynie 2 równą K). Takie uszeregowanie można uzyskać stosując pseudowielomianowy algorytm rozwiązania problemu sumy podzbioru.

Dyskusja w niniejszym podrozdziale dotyczyła wpływu terminów gotowości i czasów przetwarzania zadań na złożoność obliczeniową problemu $Rm|r_{i,j}|C_{\max}$. Formułując warunki w (2.12) wskazaliśmy, że nie zawsze można zrównoważyć obciążenie maszyn w systemie, gdy ich wydajność jest skrajnie różna. Można zatem stwierdzić, że problem szeregowania na m maszynach jest w rzeczywistości problemem szeregowania na co najwyżej m maszynach, ponieważ wymuszanie użycia wszystkich maszyn może być nieuzasadnione z punktu widzenia kryterium długości uszeregowania. Formułując warunek (2.13) pokazano, że zbyt długie niezagospodarowane okna czasowe pomiędzy najpóźniejszymi terminami gotowości a terminami gotowości pozostałych zadań mogą upraszczać problem oraz mogą stanowić o istnieniu wielu uszeregowań optymalnych. Dodatkowo wskazano dwa szczególne przypadki problemu, które można rozwiązać w czasie pseudowielomianowym.

2.4. Szacowanie dolnego ograniczenia wartości funkcji celu

Rezultat oszacowania wartości kryterium problemu optymalizacyjnego jest wykorzystywany do oceny jakości rozwiązania problemu, w procesie projektowania algorytmów aproksymacyjnych lub jest podstawą ewaluacji bardziej złożonego kryterium. Opracowano pięć funkcji oszacowania wartości dolnego ograniczenia kryterium długości uszeregowania.

Docelowym obszarem zastosowania pierwszej funkcji są instancje, w których terminy gotowości zadań są rozmieszczone wewnątrz jednego (wąskiego) przedziału niepewności $\left[0, m^{-1}\sum_{j=1}^{n}\min_{i=1,2,\ldots,m} p_{i,j}\right]$ na wszystkich maszynach. Funkcja zwracająca wartość oszacowania (2.3) dla omówionego założenia ma postać

$$LB_1(\mathrm{S}) = \left(\min_{i=1,2,\ldots,m}\min_{j=1,2,\ldots,n} r_{i,j}\right) + m^{-1}\sum_{j=1}^{n}\min_{i=1,2,\ldots,m} p_{i,j}. \tag{2.16}$$

Obliczenie wartości (2.16) może prowadzić do naruszenia ograniczeń wynikających z parametrów problemu, gdy istnieje co najmniej jedno zadanie j spełniające nierówność $LB_1(\mathrm{S}) \leq r_{i,j}$. Wynika to z faktu, że operator uśredniania uwzględnia jedynie termin gotowości najwcześniej dostępnego zadania.

Drugie podejście jest oparte na założeniu, że $\delta = \lceil m^{-1}n \rceil$ podzbiorów zadań można przetworzyć równolegle na m maszynach. Stosujemy zaokrąglenie $\delta = \lceil m^{-1}n \rceil$, ponieważ

część maszyn może mieć przypisane jedno zadanie więcej w porównaniu do maszyn obsługujących najmniejszą liczbę zadań. W przeciwieństwie do (2.16) funkcja

$$LB_2(\mathrm{S}) = \min_{i=1,2,\dots,m} \min_{j=1,2,\dots,n} r_{i,j} + \lceil m^{-1}n \rceil \min_{i=1,2,\dots,m} \min_{j=1,2,\dots,n} p_{i,j}. \tag{2.17}$$

nie uśrednia czasów przetwarzania i na jej postać składa się zwielokrotnienie najkrótszego czasu przetwarzania. Oszacowanie (2.17) jest efektywne, gdy maszyny cechują się podobną wydajnością. Zarówno (2.16) jak i (2.17) zwracają niedoszacowaną wartość (2.3), gdy występują duże rozbieżności pomiędzy parametrami tego samego zadania na różnych maszynach. Taki przypadek można zdefiniować jako

$$\frac{\max_{i=1,2,\dots,m} r_{i,j} \triangleq \tilde{r}_j}{\min_{i=1,2,\dots,m} r_{i,j} \triangleq \bar{r}_j} > \lambda \ \vee \ \frac{\max_{i=1,2,\dots,m} p_{i,j} \triangleq \tilde{p}_j}{\min_{i=1,2,\dots,m} p_{i,j} \triangleq \bar{p}_j} > \tilde{\lambda}, \quad j = 1,2,\dots,n, \tag{2.18}$$

gdzie wartości λ, $\tilde{\lambda}$ są określane przez decydenta (arbitralnie przyjęto, że $\lambda > 2$, $\tilde{\lambda} > 2$).

W definicji trzeciej funkcji założono wykorzystanie terminu gotowości oraz czasu przetwarzania jednego (tego samego) zadania, których suma jest zdeterminowana poprzez podejście „maxmin”

$$LB_3(\mathrm{S}) = \max_{i=1,2,\dots,m} \min_{j=1,2,\dots,n} \left(r_{i,j} + p_{i,j}\right). \tag{2.19}$$

Funkcja (2.19) pozwala odpowiednio skompensować znaczne rozbieżności pomiędzy $r_{i,j}$ i $p_{i,j}$, co czyni funkcję odporną dla przypadku $\lambda > 2$, $\tilde{\lambda} > 2$. Dodatkowo (2.19) zapewnia dobre rezultaty, gdy terminy gotowości są rzadko dystrybuowane na osi czasu. Termin „rzadko” odnosi się do przypadku, w którym istnieją okna czasowe pomiędzy kolejnymi terminami gotowości, wewnątrz których można ulokować co najmniej jedno zadanie.

Czwarta funkcja jest najbardziej złożona i uwzględnia $\bar{r}_{j_k}$ i $\bar{p}_{j_k}$ posortowane rosnąco. Zadania są wówczas obsługiwane na „idealnej” maszynie, oferującej najlepsze parametry dostępności i przetwarzania zadań. Procedurę można przedstawić w trzech krokach:

1. posortuj terminy gotowości zadań rosnąco $\bar{r}_{j_1} \leq \bar{r}_{j_2} \leq \cdots \leq \bar{r}_{j_k} \leq \cdots \leq \bar{r}_{j_n}$,
2. posortuj czasy przetwarzania zadań rosnąco $\bar{p}_{j_1} \leq \bar{p}_{j_2} \leq \cdots \leq \bar{p}_{j_k} \leq \cdots \leq \bar{p}_{j_n}$,
3. zastosuj metodę ERD dla każdej pary zgodnie z danym porządkiem $(\bar{r}_{j_1}, \bar{p}_{j_1}), (\bar{r}_{j_2}, \bar{p}_{j_2}), \dots, (\bar{r}_{j_k}, \bar{p}_{j_k}), \dots, (\bar{r}_{j_n}, \bar{p}_{j_n})$, jako rezultat zwróć uśrednioną długość uszeregowania dla m maszyn.

Metodę ERD dla posortowanych par zdefiniowano jako

$$\bar{C}(k) = \bar{p}_{j_k} + \max\{\bar{C}(k-1), \bar{r}_{j_k}\}, \;\; k = 1,2,\dots,n, \;\; \bar{C}(0) = 0, \tag{2.20}$$

oraz wartość oszacowania jest równa

$$LB_4(\mathrm{S}) = m^{-1}\bar{C}(n). \tag{2.21}$$

Piąta metoda oszacowania wartości (2.3) opiera się na własności 2.1. Funkcja oszacowania ma postać

$$LB_5(\mathrm{S}) = \max_{j=1,2,\dots,n}\left\{\bar{r}_j + m^{-1}\sum_{j\in \mathrm{D}_j}\bar{p}_j\right\}, \tag{2.22}$$

gdzie $\mathrm{D}_j = \{j \in \mathrm{J} \mid \bar{r}_j \le \bar{r}_t, t = 1,2,\dots,n\}$ jest zbiorem zadań dostępnych od momentu $\bar{r}_j$.

Z omówionych funkcji szacowania wartości kryterium w problemie $Rm|r_{i,j}|C_{\max}$ jedynie (2.19) szacuje dolne ograniczenie nie naruszając ograniczeń wynikających ze sformułowania problemu dla dowolnej instancji. Naruszanie ograniczeń zachodzi na przykład, gdy stosowany jest operator uśredniania podany w (2.16), który ignoruje terminy gotowości $n-1$ zadań (uwzględniany jest jedynie najwcześniejszy możliwy termin gotowości). W rezultacie może dochodzić do niedoszacowania wartości kryterium. Funkcja $LB_3(\mathrm{S})$ zwraca najmniejsze możliwe oszacowanie długości uszeregowania, ponieważ obliczany jest termin zakończenia zadania, który jest najpóźniejszy z najwcześniejszych możliwych terminów zakończeń wszystkich zadań. Opracowane metody cechuje elastyczność, ponieważ obejmują przypadki terminów gotowości gęsto i rzadko rozmieszczonych na osi czasu. Największa wartość $LB(\mathrm{S}) = \max_{t=1,2,\dots,5} LB_t(\mathrm{S})$ stanowi oszacowanie wartości kryterium długości uszeregowania (2.3).

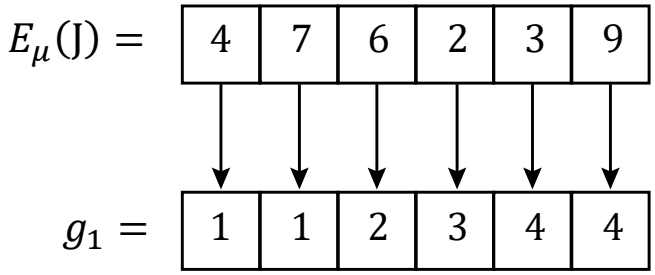
2.5. Algorytm bazujący na przeglądzie wyczerpującym

Pierwszy algorytm szeregowania **D_Brute-Force (BF)** implementuje technikę przeglądu wyczerpującego (ang. brute-force search lub exhaustive search) dla wybranych arbitralnie podzbiorów zadań. Stosowanie przeglądu wyczerpującego nawet dla instancji małej wielkości ($n \le 20,\ m \le 3$) jest niepraktyczne z powodu wykładniczej złożoności czasowej. Konsekwencją wykładniczej złożoności czasowej jest zastosowane dekompozycji na niezależne podpdoblemy. Dlatego kluczowym parametrem, decydującym o jakości rozwiązania i czasie obliczeń, jest określenie liczby zadań, które należy ustalić dla każdego podproblemu. Taki parametr, określony jako współczynnik podziału, oznaczono jako

$b \in \mathbb{N}_+$. Opracowany algorytm rozwiązuje optymalnie $\omega = \lceil nb^{-1} \rceil$ niezależnych podproblemów szeregowania zadań. W podproblemie o indeksie μ, $\mu = 1,2,\dots,\omega$, należy uszeregować zadania zdefiniowane w sekwencji $E_\mu(\mathrm{J}) = \left(j_{\mu+1}, j_{\mu+2}, \dots, j_{\mu+w}, \dots, j_{\mu+b}\right)$, $\left\{j_{\mu+1}, j_{\mu+2}, \dots, j_{\mu+w}, \dots, j_{\mu+b}\right\} \subseteq \mathrm{J}$. Dobór zadań w $E_1(\mathrm{J}), E_2(\mathrm{J}), \dots, E_\mu(\mathrm{J}), \dots, E_\omega(\mathrm{J})$ jest realizowany w oparciu o uśrednioną wartość terminów gotowości zadań. W pierwszej kolejności wyliczana jest średnia wartość terminu gotowości każdego zadania $r_{j_w}^{(avg)} = m^{-1} \sum_{i=1}^{m} r_{i,j_w}$, $j_w = 1,2,\dots,n$. Następnie z posortowanych rosnąco uśrednionych wartości

$$r_{j_1}^{(avg)} \le r_{j_2}^{(avg)} \le \cdots \le r_{j_{\mu+1}}^{(avg)} \le r_{j_{\mu+2}}^{(avg)} \le \cdots \le r_{j_{\mu+b}}^{(avg)} \le \cdots \le r_{j_n}^{(avg)}, \tag{2.23}$$

w podproblemie identyfikowanym indeksem μ rozważane są tylko zadania w sekwencji $E_\mu(\mathrm{J}) = \left(j_{\mu+1}, j_{\mu+2}, \dots, j_{\mu+w}, \dots, j_{\mu+b}\right)$. Sekwencja $E_\omega(\mathrm{J})$ nie zawiera b elementów, jeżeli n nie jest podzielne przez b, $b \le n$.



Rysunek 2.2. Przypisanie zadań zdefiniowanych w sekwencji $E_\mu(\mathrm{J})$ wskazuje pierwszy wiersz g_1 macierzy $G(\mathrm{M}, b, \mu)$. Dwie jedynki oznaczają, że zadania 4 i 7 muszą zostać przypisane do sekwencji na maszynie pierwszej w kolejności danej w wektorze $E_\mu(\mathrm{J})$

Proces optymalizacji jest oparty na wariacjach z powtórzeniami wygenerowanych na podstawie sekwencji $E_\mu(\mathrm{J})$, $\mu = 1,2,\dots,\omega$. Wszystkie możliwe wariacje reprezentuje macierz

$$G(\mathrm{M}, b, \mu) = \begin{bmatrix} i_{1,1}(\mu) & i_{1,2}(\mu) & \dots & i_{1,w}(\mu) & \dots & i_{1,b}(\mu) \\ i_{2,1}(\mu) & i_{2,2}(\mu) & \dots & i_{2,w}(\mu) & \dots & i_{2,b}(\mu) \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ i_{b^b,1}(\mu) & i_{b^b,2}(\mu) & \dots & i_{b^b,w}(\mu) & \dots & i_{b^b,b}(\mu) \end{bmatrix}, \tag{2.24}$$

gdzie wartość $i_{k,w}(\mu) \in \mathrm{M}$ wskazuje pozycję zadania $j_{\mu+w}$ w harmonogramie (przykład pokazano na rysunku 2.2). W związku z faktem, że liczba wariacji m elementowych o długości m z powtórzeniami jest równa m^m, to macierz (2.24) składa się z b^b wierszy.

Algorytm **BF** oblicza długość uszeregowania dla każdej wariacji wygenerowanej na podstawie sekwencji $E_\mu(\mathrm{J})$, uwzględniając przypisania zrealizowane w rozwiązaniach podproblemów identyfikowanych indeksami $\mu-1, \mu-2, \ldots, 1$. Indeks μ w harmonogramie oznaczonym jako $x(\mu)$ wskazuje, że uszeregowano zadania znajdujące się w sekwencjach $E_1(\mathrm{J}), E_2(\mathrm{J}), \ldots, E_\mu(\mathrm{J})$.

Algorytm **D_Brute-Force (BF)**

Require: $\mathrm{J}, \mathrm{M}, r, p, b$
Ensure: x_{BF}

1. **if** $|\mathrm{J}| \leq b$ **then** użyj algorytmu brute-force search dla pełnego zbioru J **return** x_{BF} **end if**
2. Przypisz $\omega := \lceil n^{-1}b \rceil$, $\mu := 1$ i $x(\mu := 0) = \left[x_{i,k,j}(\mu) := 0\right]_{\substack{i=1,2,\ldots,m \\ k,j=1,2,\ldots,n}}$.
3. Podziel zbiór J na ω sekwencji $E_1(\mathrm{J}), E_2(\mathrm{J}), \ldots, E_\mu(\mathrm{J}), \ldots, E_\omega(\mathrm{J})$.
4. **while** $\mu \leq \omega$
5. Wygeneruj macierz $G(\mathrm{M}, b, \mu)$.
6. Oblicz długość uszeregowania dla każdego wiersza w $G(\mathrm{M}, b, \mu)$.
7. Zaktualizuj harmonogram $x(\mu-1)$ umieszczając zadania wskazane w $E_\mu(\mathrm{J})$.
8. Zwiększ wartość indeksu $\mu := \mu + 1$.
9. **end while (4)**
10. $x_{\mathrm{BF}} := x(\omega)$

Złożoność operacji generowania rozwiązania początkowego w Linii 1 (pseudokod algorytmu **BF**), w którym każda decyzja jest początkowo ustawiona na zero, wynosi $\mathcal{O}(n^2m)$. Złożoność obliczeniowa wymagana do obliczenia długości uszeregowania (2.2) wynosi $\mathcal{O}(n^2m)$, ponieważ algorytm musi wykonać iteracje po trójwymiarowej macierzy binarnej. Obliczenie wartości funkcji celu w podproblemie μ wymaga czasu $\mathcal{O}(b^2m)$ przy zapamiętanych terminach zakończeń ostatnich przypisanych zadań na każdej maszynie w podproblemie $\mu-1$. W konsekwencji złożoność obliczeniowa algorytmu **BF** jest równa $\mathcal{O}(b^b\omega b^2m)$, ponieważ długość uszeregowania jest obliczana b^b razy dla ω macierzy (Linie 4-9). Złożoność pamięciowa wynosi $\mathcal{O}(b^b b + m + n^2m)$ i jest zdeterminowana rozmiarem macierzy (2.24), rozmiarem wektora zawierającego terminy zakończeń ostatnich zadań w dowolnym podproblemie oraz rozmiarem reprezentacji rozwiązania.

Przedstawiony algorytm **BF** wykorzystuje dekompozycję. Wykluczenie dekompozycji, poprzez wybór parametru $b = n$, skutkuje optymalnym rozwiązaniem problemu $Rm|r_{i,j}|C_{\max}$ przy złożoności obliczeniowej $\mathcal{O}(n^n\omega n^2m)$ oraz pamięciowej $\mathcal{O}(n^n n + n^2m)$.

2.6. Algorytm oparty na metaheurystyce Simulated Annealing

Algorytm **S_Annealing**, bazujący na metaheurystyce Simulated Annealing (**SA**), został zaprojektowany zgodnie z wytycznymi podanymi przez Kirkpatrick, Gelatt & Vecchi (1983). Realizacja procedury optymalizacji jest poprzedzona wygenerowaniem losowego rozwiązania początkowego $x(\varphi = 1)$, gdzie φ jest indeksem iteracji a $x(\varphi)$ oznacza rozwiązanie uzyskane w iteracji o numerze φ. Jedno rozwiązanie sąsiednie $x_{\mathrm{N}}(\varphi) \in \Omega_{\mathrm{SA}}\big(x(\varphi)\big)$, w stosunku do rozwiązania $x(\varphi)$, jest generowane poprzez zmianę pozycji zadania

$$\begin{aligned} t &= \arg \max_{j=1,2,\dots,n} \left\{\left(C_{i,k}(x(\varphi);\mathrm{S}) - p_{i,j}\right) - \bar{r}_j\right\}, \\ i &= 1,2,\dots,m, \quad k = 1,2,\dots,n, \text{ dla } x_{i,k,j} = 1, \end{aligned} \tag{2.25}$$

którego przetwarzanie rozpoczyna się najpóźniej względem najwcześniejszego możliwego terminu gotowości.

Zadanie t zostaje przeniesione na maszynę i pozycję w sekwencji umożliwiające jego najwcześniejsze zakończenie. Ustalenie najwcześniejszego możliwego terminu gotowości zadania t nie gwarantuje natychmiastowego wykonania zadania (w terminie jego gotowości), ponieważ nie realizujemy zmian harmonogramu zadań, których rozpoczęcie poprzedza termin gotowości zadania t. Zbiór m rozwiązań sąsiednich ma postać $\Omega_{\mathrm{SA}}\big(x(\varphi)\big) = \left\{x_{\mathrm{N}}^{(1)}(\varphi), x_{\mathrm{N}}^{(2)}(\varphi), \dots, x_{\mathrm{N}}^{(m)}(\varphi)\right\}$ a najlepsze rozwiązanie jest równe $x_{\mathrm{N}}(\varphi) = \arg \min_{x'_{\mathrm{N}} \in \Omega_{\mathrm{SA}}(x(\varphi))} C_{\max}(x'_{\mathrm{N}};\mathrm{S})$. W każdej iteracji algorytmu generujemy m rozwiązań sąsiednich. Unikanie lokalnych optimów jest realizowane poprzez przyjęcie rozwiązania o gorszej jakości, gdy funkcja $f(x(\varphi), x_N(\varphi);\mathrm{S}) = e^{v(\varphi)^{-1}\left(C_{\max}(x(\varphi);\mathrm{S}) - C_{\max}(x_{\mathrm{N}}(\varphi);\mathrm{S})\right)}$ (Michalewicz & Fogel, 2013) zwraca wartość większą niż losowo wygenerowana liczba z zakresu [0,1). Liczba iteracji zależy od wybranej strategii zmiany temperatury $v \in \{v_1(\varphi), v_2(\varphi), v_3(\varphi)\}$. W algorytmie **SA** wykorzystano $v_1(\varphi) = V_0(r,p)\alpha^{\varphi}$, $v_2(\varphi) = V_0(r,p) - \alpha\varphi$ lub $v_3(\varphi) = V_0(r,p)(1 + \log(\varphi + 1))^{-1}$ (Kirkpatrick, Gelatt & Vecchi, 1983; Hajek, 1988), gdzie $\alpha \in \{0.85, 0.9, 0.95\}$ i $V_0(r,p)$ oznacza temperaturę początkową. Prawdopodobieństwo akceptacji nowego rozwiązania musi być bliskie jedności, aby uczynić proces podejmowania decyzji losowym w początkowej fazie działania algorytmu. Dlatego przyjęto, że wartość temperatury początkowej

$$V_0(r,p) = \max_{i=1,2,\dots,m} \max_{j=1,2,\dots,n} r_{i,j} + m^{-1} \sum_{j=1}^{n} \max_{i=1,2,\dots,m} p_{i,j}, \tag{2.26}$$

odpowiada górnemu ograniczeniu wartości kryterium, przy założeniu wykorzystania wszystkich maszyn. Algorytm **SA** wykonuje obliczenia, dopóki temperatura jest większa od zera, czas obliczeń nie przekroczy Γ_{max} lub liczba iteracji bez poprawy jakości rozwiązania przekroczy ϱ.

Algorytm **S_Annealing (SA)**

Require: J, M, $r, p, \Gamma_{max}, \varrho, v$
Ensure: x_{SA}

1. Przypisz $\varphi \coloneqq 1$, oblicz $V_0(r,p)$, wygeneruj losowy harmonogram $x(\varphi)$.
2. **while** $\Gamma_{max} > t$ **or** $v(\varphi) > 0$ **or** $\varrho > 0$
3. Wyznacz $t \coloneqq \arg \max_{j=1,2,\dots,n} \{(C_{i,k}(x(\varphi); \mathrm{S}) - p_{i,j}) - \bar{r}_j\}$ dla $x_{i,k,j} = 1$.
4. Jeżeli wynikiem w Linii 3 jest zbiór rozwiązań, to wygeneruj zbiór rozwiązań sąsiednich dla każdego rozwiązania.
5. Wygeneruj zbiór rozwiązań sąsiednich $\Omega_{\mathrm{SA}}(x(\varphi))$ i $x_{\mathrm{N}}(\varphi) = \arg \min_{x'_{\mathrm{N}} \in \Omega_{\mathrm{SA}}(x(\varphi))} C_{\max}(x'_{\mathrm{N}}; \mathrm{S})$.
6. **if** $C_{\max}(x(\varphi); \mathrm{S}) > C_{\max}(x_N(\varphi); \mathrm{S})$ **or** $\mathrm{random}[0,1) < f(x(\varphi), x_{\mathrm{N}}(\varphi); \mathrm{S})$ **then** $x(\varphi) \coloneqq x_{\mathrm{N}}(\varphi)$ i zapamiętaj rezultat $C_{\max}(x(\varphi); \mathrm{S})$ **end if**
7. Zaktualizuj wskaźnik czasu t, wskaźnik poprawy ϱ oraz indeks iteracji $\varphi \coloneqq \varphi + 1$.
8. **end while (2)**
9. $x_{\mathrm{SA}} \coloneqq x(\varphi)$

Złożoność generowania rozwiązania początkowego zależy od wybranego generatora losowego GEN i wynosi $\mathcal{O}(\mathrm{GEN}n^2m)$, ponieważ generator jest uruchamiany dla różnych decyzji do momentu wypełnienia całego harmonogramu. Rozwiązanie problemu w Linii 3 wymaga przeglądu całego harmonogramu w czasie $\mathcal{O}(n^2m)$ pod warunkiem agregacji terminów zakończeń zadań w harmonogramie. Obliczenie wartości funkcji celu dla m rozwiązań sąsiednich wymaga $\mathcal{O}(n^2m^2)$. Operacje porównania, przypisania oraz modyfikacji indeksu poprawy jakości rozwiązania i czasu są stałe czasowo (Linie 6-7). Liczba iteracji pętli głównej (Linie 2-8) zależy od wybranej strategii zmiany temperatury. Ostatecznie złożoność obliczeniowa algorytmu **SA** wynosi $\mathcal{O}(\max\{\mathrm{GEN}n^2m, n^2m^2v_{\max}\})$, gdzie $v_{\max}$ jest największą możliwą liczbą iteracji pętli głównej przy rozważanych strategiach zmiany temperatury. Złożoność pamięciowa wynosząca $\mathcal{O}(m + n + n^2m)$ jest zdeterminowana rozmiarem zbioru rozwiązań sąsiednich, agregacją terminów zakończeń oraz rozmiarem reprezentacji rozwiązania.

2.7. Algorytm zachłanny

Wadą podejścia wyczerpującego (algorytm **BF**) oraz przeszukiwania losowego (algorytm **SA**) jest brak możliwości zagwarantowania ograniczonej straty jakości rozwiązania względem oszacowanego rozwiązania optymalnego. Z tego powodu opracowano algorytm **Greedy** (**GD**) implementujący strategię zachłanną z planowaniem długoterminowym dla decyzji o takiej samej jakości (decyzje dowolne z punktu widzenia kryterium długości uszeregowania). Procedura dzieli proces podejmowania decyzji na n zależnych od siebie etapów. Podczas etapu o indeksie η można uszeregować jedno zadanie $j \in \mathrm{J}(\eta)$, gdzie $\mathrm{J}(\eta)$ jest zbiorem zadań nieuszeregowanych w etapach $1,2,\dots,\eta-1$, uwzględniając poprzednie decyzje. Podproblem w η-tym etapie optymalizacji wyrażono wzorem

$$\Lambda = \arg \min_{\substack{i=1,2,\dots,m \\ k=1,2,\dots,n \\ j\in \mathrm{J}(\eta)}} \left\{C_{i,k}\left(x(\eta) = \left[x_{i,k,j}(\eta) = 1\right]; \mathrm{S}\right)\right\}, \tag{2.27}$$

gdzie Λ jest zbiorem krotek reprezentujących rozwiązanie podproblemu o liczności $|\Lambda|$. Krotka $(s,d,t) \in \Lambda$ stanowi podstawę do podjęcia decyzji $x_{s,d,t}(\eta) = 1$.

Wyróżnikiem opracowanego podejścia zachłannego jest sposób podejmowania decyzji w przypadku, gdy mamy do czynienia z decyzjami o tej samej jakości. Wówczas, jeżeli co najmniej dwa rozwiązania w (2.27) ($|\Lambda| > 1$) gwarantują taką samą (najmniejszą) wartość kryterium, to w procesie optymalizacji pod uwagę brane są najwcześniejsze możliwe terminy zakończeń zadań w Λ. Dla każdego zadania apriorycznie wskazywane są wszystkie maszyny pozwalające na jego najwcześniejsze zakończenie (niezależnie od uszeregowania), wyliczając

$$\arg \min_{i=1,2,\dots,m} \left\{r_{i,j} + p_{i,j}\right\}, \qquad j = 1,2,\dots,n. \tag{2.28}$$

Informacje o wydajności przetwarzania na poszczególnych maszynach są agregowane w zbiorze $\mathrm{J}_F(\eta) = \left\{\mathrm{J}_{F_1}(\eta), \mathrm{J}_{F_2}(\eta), \dots, \mathrm{J}_{F_i}(\eta), \dots, \mathrm{J}_{F_m}(\eta)\right\}$, gdzie $\mathrm{J}_{F_i}(\eta) \subseteq \mathrm{J}$ zawiera zadania mogące być przetwarzane najszybciej na maszynie i. Jeżeli (2.27) w η-tym etapie zwraca co najmniej dwa równorzędne rozwiązania, to wybieramy zadanie $j \in \mathrm{J}_{F_i}(\eta)$, które może być przetwarzane najefektywniej (najszybciej) na najmniejszej liczbie maszyn oraz $i = \arg \min_{s=1,2,\dots,m} \left|\mathrm{J}_{F_s}(\eta)\right|$. Taka strategia ma na celu równoważenie obciążenia szybko przetwarzających maszyn oraz zapobiega decyzjom, które w kolejnych iteracjach mogą

pogarszać jakość kryterium z powodu konieczności wyboru maszyn długo przetwarzających zadania. Po podjęciu każdej decyzji uszeregowane zadanie jest usuwane z podzbiorów w $\mathrm{J}_F(\eta)$.

Algorytm **Greedy (GD)**

Require: $\mathrm{J}, \mathrm{M}, r, p$
Ensure: x_{GD}

1. Przypisz $\eta \coloneqq 1$, $\mathrm{J}(\eta) \coloneqq \mathrm{J}$, $x(\eta) \coloneqq \left[x_{i,k,j}(\eta) \coloneqq 0\right]_{\substack{i=1,2,\dots,m \\ k,j=1,2,\dots,n}}$ i wygeneruj zbiór $\mathrm{J}_F(\eta)$.
2. **while** $\mathrm{J}(\eta) \neq \emptyset$
3. Wyznacz krotki $\Lambda \coloneqq \arg \min_{\substack{i=1,2,\dots,m \\ k=1,2,\dots,n \\ j\in \mathrm{J}(\eta)}} \left\{C_{i,k}\left(x(\eta) \coloneqq \left[x_{i,k,j}(\eta) \coloneqq 1\right]; \mathrm{S}\right)\right\}$.
4. **if** $|\Lambda| > 1$ **then** Wybierz zadanie $j \in \mathrm{J}_{F_i}(\eta)$ takie, że $i = \arg \min_{s=1,2,\dots,m} \left|\mathrm{J}_{F_s}(\eta)\right|$ **end if**
5. Przypisz $x_{s,d,t}(\eta) \coloneqq 1$, $(s,d,t) \in \Lambda$ \\ lub zadanie j, gdy spełniony jest warunek w Linii 4.
6. Zaktualizuj zawartość zbiorów $\mathrm{J}(\eta+1) \coloneqq \mathrm{J}(\eta)\backslash\{t\}$, $\forall_{i=1,2,\dots,m}\, \mathrm{J}_{F_i}(\eta) \coloneqq \mathrm{J}_{F_i}(\eta)\backslash\{t\}$ oraz indeks $\eta \coloneqq \eta + 1$.
7. Zwiększ liczbę zadań na maszynie $n_s(\eta) \coloneqq n_s(\eta-1) + 1$ i zapamiętaj $C_{s,n_s}(x(\eta); \mathrm{S})$.
8. **end while (2)**
9. $x_{\mathrm{GD}} \coloneqq x(\eta)$

Wygenerowanie rozwiązania początkowego i zbiorów zawierających najefektywniej przetwarzane zadania w Linii 1 wymagają $\mathcal{O}(n^2m)$. Rozwiązanie problemu w Linii 3 wymaga jedynie przeglądu indeksów maszyn i nieuszeregowanych zadań w czasie $\mathcal{O}(nm)$, ponieważ zapamiętujemy długość sekwencji na każdej maszynie w Linii 6. W przypadku spełnienia warunku w Linii 4, przegląd zawartości zbiorów wymaga $\mathcal{O}(m)$. Operacje porównania, przypisania oraz modyfikacji indeksu iteracji są stałe czasowo (Linie 5-7). Złożoność obliczeniowa algorytmu **GD** wynosi $\mathcal{O}(n^2m)$, ponieważ problem w Linii 3 jest rozwiązywany n razy. Złożoność pamięciowa wynosząca $O(n + n^2 + n^2m)$ jest zdeterminowana wektorem długości sekwencji, rozmiarem zbioru $\mathrm{J}_F(\eta)$ oraz rozmiarem reprezentacji rozwiązania.

Górną i dolną granicę długości uszeregowania, które są rezultatem uszeregowania zwróconego przez algorytm **GD**, można wyrazić oszacowaniem (2.19). W skrajnym przypadku wszystkie zadania należy uszeregować na jednej maszynie, co uzasadnia, że największa wartość kryterium jaką algorytm **GD** może uzyskać jest równa

$$\overline{C}_{\max}(x_{\mathrm{GD}}; \mathrm{S}) = n \max_{i=1,2,\dots,m} \min_{j=1,2,\dots,n} \left(r_{i,j} + p_{i,j}\right). \tag{2.29}$$

Z drugiej strony, jeżeli liczba zadań jest równa liczbie maszyn i każde z nich jest przetwarzane najszybciej na innej maszynie, to najmniejsza wartość kryterium jaką algorytm **GD** może uzyskać jest równa

$$\underline{C}_{\max}(x_{\mathrm{GD}};\mathrm{S}) = \max_{i=1,2,\dots,m} \min_{j=1,2,\dots,n} \left(r_{i,j} + p_{i,j}\right). \tag{2.30}$$

Wówczas długość uszeregowania jest ograniczona przez

$$C_{\max}(x_{\mathrm{GD}};\mathrm{S}) \leq \max_{i=1,2,\dots,m} \min_{j=1,2,\dots,n} \left(r_{i,j} + p_{i,j}\right) \max_{i=1,2,\dots,m} \left|\mathrm{J}_{F_i}(\eta)\right|. \tag{2.31}$$

Przykład obliczeniowy: Działanie algorytmu **GD** zilustrujemy na przykładzie obliczeniowym. Przyjmujemy następujące dane: $\mathrm{J} = \{1,2,3\}$, $\mathrm{M} = \{1,2,3\}$, $m = 3$, $n = 3$ oraz

$$r = \begin{bmatrix} 1 & 2 & 4 \\ 1 & 1 & 2 \\ 1 & 3 & 6 \end{bmatrix}, p = \begin{bmatrix} 3 & 6 & 11 \\ 3 & 5 & 8 \\ 3 & 7 & 4 \end{bmatrix}.$$

Pierwszy etap optymalizacji $\boldsymbol{\eta = 1}$: algorytm **GD** wskazuje zadanie i maszynę minimalizujące

$$\min \begin{Bmatrix} \boldsymbol{r_{1,1} + p_{1,1} = 1 + 3}, \boldsymbol{r_{2,1} + p_{2,1} = 1 + 3}, \boldsymbol{r_{3,1} + p_{3,1} = 1 + 3}, \\ r_{1,2} + p_{1,2} = 2 + 6, r_{2,2} + p_{2,2} = 1 + 5, r_{3,2} + p_{3,2} = 3 + 7, \\ r_{1,3} + p_{1,3} = 4 + 11, r_{2,3} + p_{2,3} = 2 + 8, r_{3,3} + p_{3,3} = 6 + 4 \end{Bmatrix} = \boldsymbol{4}.$$

Przypisanie zadania $j = 1$ do dowolnej maszyny jest optymalną decyzją z punktu widzenia strategii zachłannej. Wówczas decyzja o uszeregowaniu jest podejmowana na podstawie zawartości zbiorów $\mathrm{J}_{F_1}(\eta = 1) = \{1\}$, $\mathrm{J}_{F_2}(\eta = 1) = \{1,2\}$, $\mathrm{J}_{F_3}(\eta = 1) = \{1,3\}$. Dla zadania $j = 1$ wybrano maszynę $i = 1$, ponieważ $\left|\mathrm{J}_{F_1}(1)\right| < \left|\mathrm{J}_{F_2}(1)\right| \leq \left|\mathrm{J}_{F_3}(1)\right|$.

Zadanie $j = 1$ zostało przypisane do maszyny $i = 1$ na pierwszej pozycji w sekwencji i $\boldsymbol{C_{1,1}(x(1);\mathrm{S}) = 4}$, $\boldsymbol{x_{1,1,1}(1) = 1}$.

Drugi etap optymalizacji $\boldsymbol{\eta = 2}$: algorytm **GD** wskazuje zadanie i maszynę minimalizujące

$$\min \begin{Bmatrix} C_{1,1}(x(1);\mathrm{S}) + p_{1,2} = 4 + 6, \boldsymbol{r_{2,2} + p_{2,2} = 1 + 5}, r_{3,2} + p_{3,2} = 3 + 7, \\ C_{1,1}(x(1);\mathrm{S}) + p_{1,3} = 4 + 11, r_{2,3} + p_{2,1} = 2 + 8, r_{3,3} + p_{3,3} = 6 + 4 \end{Bmatrix} = \boldsymbol{6}.$$

Zadanie $\mathrm{j} = 2$ zostało przypisane do maszyny do maszyny $\mathrm{i} = 2$ na pierwszej pozycji w sekwencji i $\boldsymbol{C_{2,1}(x(2);\mathrm{S}) = 6}$, $\boldsymbol{x_{2,1,2}(2) = 1}$.

Trzeci etap optymalizacji $\boldsymbol{\eta = 3}$: algorytm **GD** wskazuje zadanie i maszynę minimalizujące

$$\min\begin{Bmatrix} C_{1,1}(x(1);\mathrm{S}) + p_{1,3} = 4 + 11, \\ C_{2,1}(x(2);\mathrm{S}) + p_{2,3} = 6 + 8, \boldsymbol{r_{3,3} + p_{3,3} = 6 + 4} \end{Bmatrix} = \mathbf{10}.$$

Zadanie j = 2 zostało przypisane do maszyny do maszyny i = 2 na pierwszej pozycji w sekwencji i $\boldsymbol{C_{3,1}(x(3);\mathrm{S}) = 10}$, $\boldsymbol{x_{3,1,3}(3) = 1}$.

Długość uszeregowania wynosi $\boldsymbol{C_{\max}(x;\mathrm{S}) = \max\{C_{1,1}(x;\mathrm{S}) = 4, C_{2,1}(x(2);\mathrm{S}) = 6, C_{3,1}(x(3);\mathrm{S}) = 10\} = 10}$ dla decyzji $\boldsymbol{\{x_{1,1,1} = 1, x_{2,1,2} = 1, x_{3,1,3} = 1\}}$.

2.8. Ocena eksperymentalna i statystyczna algorytmów

W niniejszym podrozdziale przeprowadzono następujące badania:

1. **Badanie 2.1**: Eksperymentalne porównanie jakości uszeregowań generowanych przez algorytm **GD** względem oszacowanej wartości optymalnej.
2. **Badanie 2.2**: Eksperymentalne porównanie jakości uszeregowań generowanych przez algorytmy **BF**, **GD** i **SA**.
3. **Badanie 2.3**: Eksperymentalne porównanie jakości uszeregowań generowanych przez algorytm **GD** z rozwiązaniami optymalnymi.
4. **Badanie 2.4**: Statystyczne porównanie jakości uszeregowań generowanych przez algorytmy **GD** i **BF**.

Wszystkie algorytmy w tej pracy zostały zaimplementowane w języku Python w wersji 3.7 oraz badania przeprowadzono wykorzystując procesor Intel Core i7 (4700MQ) z 8 GB pamięci RAM.

Badania przeprowadzono wykorzystując dodatnie liczby całkowite zdefiniowane w zbiorach $D_p = \{\underline{p}, \underline{p} + 1, \dots, \bar{p}\}$ (dopuszczalne zakresy czasów przetwarzania zadań) $D_r = \{\underline{r}, \underline{r} + 1, \dots, \bar{r}\}$ (dopuszczalne zakresy terminów gotowości zadań). Parametry wygenerowanego zbioru danych są następujące: $\underline{p} = 5$, $\bar{p} = 50$, $\underline{r} = 0$, $\bar{r} = 500$. Wartości ze zbiorów D_p i D_r są wybierane losowo (zgodnie z rozkładem równomiernym). Dla każdego zadania losowano m razy wartość ze zbioru D_p oraz zbioru D_r. Generowanie danych zrealizowano w dwóch etapach. Podczas pierwszego etapu generujemy największą rozważaną instancję problemu $n = 400$ i $m = 15$. Następnie pozostałe instancje są tworzone poprzez losowe (z równym prawdopodobieństwem) usuwanie wymaganej liczby parametrów zadań.

Wartość współczynnika podziału b w algorytmie **BF** jest kluczowa w kontekście czasu obliczeń. Zaproponowano możliwie najszerszy przegląd wartości b, zachowując ograniczenie czasowe. Algorytm **BF** uruchamiano dla każdej instancji i dla kolejnych wartości $b = 1,2,\dots,n$ do momentu, w którym łączny czas obliczeń przekroczy $\Gamma_{max} = 100$ [s] oraz wybierano najlepsze rozwiązanie. Jeżeli algorytm **BF** zwrócił najlepsze rozwiązanie dla kilku różnych wartości b, to wybierano długość uszeregowania uzyskaną w najkrótszym czasie.

W algorytmie **SA** strojeniu poddawano dobór strategii zmiany temperatury $v \in \{v_1(\varphi,\alpha), \boldsymbol{v_2(\varphi,\alpha)}, v_3(\varphi,\alpha)\}$ oraz parametr szybkości zmiany temperatury $\alpha \in \{0{,}85; \boldsymbol{0{,}9}; 0.95\}$. Strojenie algorytmu **SA** przeprowadzono metodą Grid Search. Pogrubioną czcionką oznaczono parametry, które zostały wybrane. Wylosowano po 5 instancji z każdego zakresu $n^{(z)} \times m^{(z)}$, gdzie $n^{(z)} = \{[10,99],[101,200],[201,300]\}$, $m^{(z)} = \{[2,5],[6,10],[11,15]\}$, oraz czasy przetwarzania zadań wylosowano, z wykorzystaniem rozkładu równomiernego, z przedziału $[\underline{p},\bar{p}]$, $\underline{p} = 5$, $\bar{p} \in \{6,7,\dots,100\}$. Parametry dobierano uruchamiając algorytm 10 razy dla każdej wygenerowanej instancji i dla każdej możliwej pary strojonych parametrów. Wybrano parametry v oraz α, dla których algorytm **SA** zwracał najwięcej razy uszeregowania o najlepszej jakości. Algorytm **SA** uruchamiano 25 razy dla każdej instancji oraz wybierano najlepszy uzyskany rezultat. Ustalono, że liczba dopuszczalnych iteracji bez poprawy rozwiązania wynosi $\varrho = 20$ oraz ograniczono łączny czas obliczeń dla każdej instancji (25 uruchomień) do $\Gamma_{max} = 100$ [s]. Do generowania liczb pseudolosowych w algorytmach opracowanych na potrzeby rozprawy doktorskiej wykorzystano generator Mersenne Twister opracowany przez Matsumoto & Nishimura (1998).

Badanie 2.1

Badanie 2.1 ma na celu porównanie długości uszeregowań uzyskanych przez algorytm **GD** z oszacowanymi dolnym ograniczeniami wartości kryterium. Badania przeprowadzono dla instancji $n \times m = \{40,80,\dots,400\} \times \{2,5,10,15\}$. Rezultaty zaprezentowane w tabelach 2.1 oraz 2.2 opisują jakość rozwiązania x_{GD} względem dolnego oszacowania oraz uzasadniają wykorzystanie $LB_5(\mathrm{S})$ w oszacowaniu $LB(\mathrm{S})$. Do oceny jakości wykorzystano empiryczne współczynniki $\Omega = \frac{C_{\max}(x_{\mathrm{GD}};\mathrm{S})}{\max\limits_{t=1,2,\dots,4} LB_t(\mathrm{S})}$ oraz $\widetilde{\Omega} = \frac{C_{\max}(x_{\mathrm{GD}};\mathrm{S})}{\max\limits_{t=1,2,\dots,5} LB_t(\mathrm{S})}$.

Tabela 2.1. Wartości współczynnika Ω

n	$m = 2$	$m = 5$	$m = 10$	$m = 15$
40	**1,896**	**6,610**	**7,316**	**8,290**
80	1,250	**3,730**	**6,761**	**6,228**
120	1,143	2,374	4,402	**4,615**
160	1,088	1,812	3,427	3,520
200	1,070	1,573	2,742	3,305
240	1,058	1,441	2,326	2,863
280	1,038	1,337	2,115	2,722
320	1,033	1,275	2,051	2,599
360	1,034	1,237	1,882	2,344
400	1,027	1,204	1,809	2,189

Wysokie wartości Ω w tabeli 2.1 dla podzbiorów instancji wyróżnionych pogrubioną czcionką wynikają z niedoszacowania wartości dolnego ograniczenia $LB_3(S) = \max_{t=1,2,\ldots,4}\{LB_t(S)\}$, które jest obliczane na podstawie parametrów tylko jednego zadania. Dla pozostałych instancji najlepszym oszacowaniem wartości kryterium było uśrednianie $LB_1(\mathrm{S}) = \max_{t=1,2,\ldots,5}\{LB_t(\mathrm{S})\}$ i wzrost wartości $LB_1(\mathrm{S})$ dla każdego dodatkowego podzbioru zadań jest znacznie większy niż wartości pozostałych oszacowań. Niedoszacowanie wyników podanych w tabeli 2.1 jest widoczne, gdy porównujemy je z wynikami podanymi w tabeli 2.2.

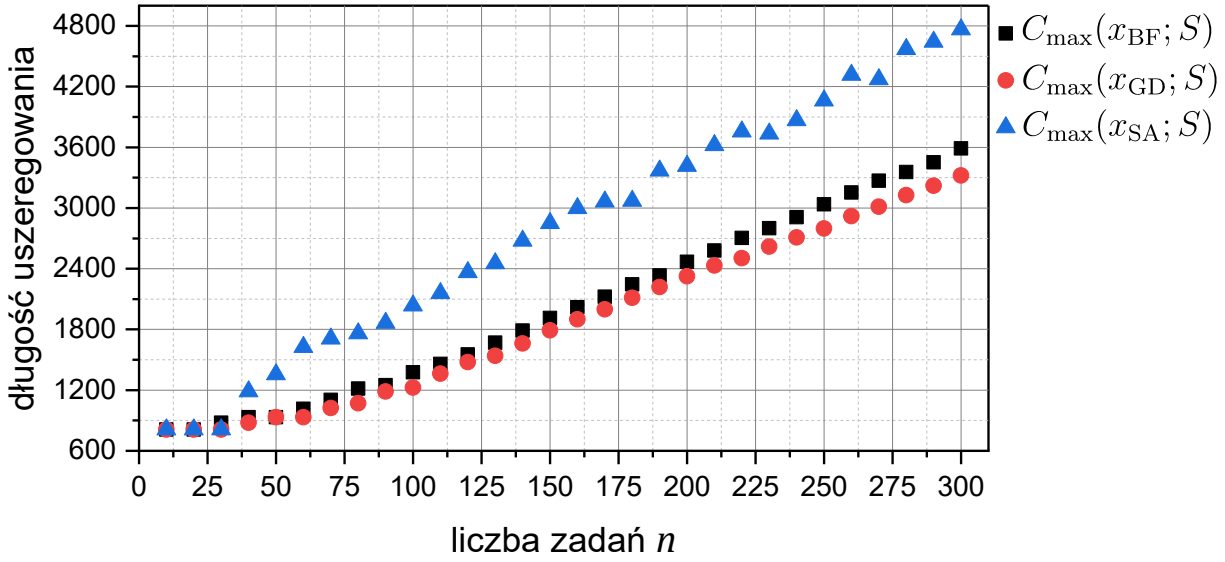
Tabela 2.2. Wartości współczynnika $\widetilde{\Omega}$ dla instancji o takich samych parametrach jak podane w tabeli 2.1

n	$m = 2$	$m = 5$	$m = 10$	$m = 15$
40	1,022	1,037	1,174	1,353
80	1,166	1,037	1,084	1,353
120	1,132	1,037	1,181	1,377
160	1,088	1,061	1,211	1,166
200	1,069	1,128	1,114	1,364
240	1,058	1,230	1,322	1,399
280	1,038	1,327	1,171	1,316
320	1,033	1,275	1,297	1,444
360	1,034	1,237	1,316	1,455
400	1,027	1,204	1,400	1,505

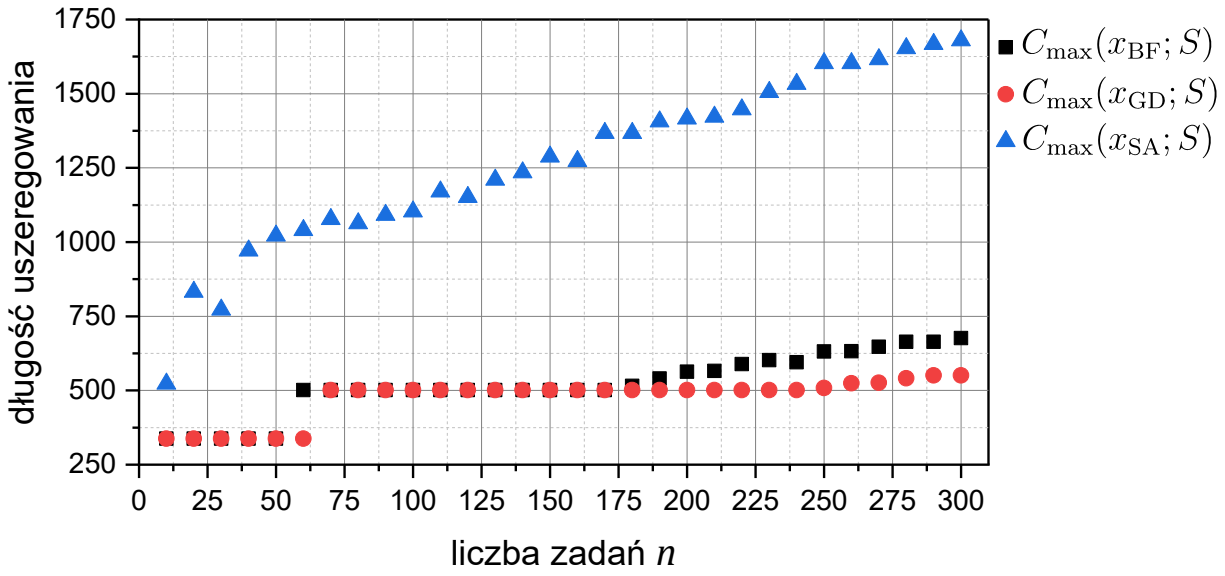
Specyfika problemu $Rm|r_{i,j}|C_{\max}$ wymusza wykorzystanie jak największej liczby terminów gotowości w efektywnym oszacowaniu dolnego ograniczenia. Efektywność $LB_5(\mathrm{S})$ wynika z pomijania podzbioru wcześniejszych terminów gotowości na rzecz późniejszych.

Badanie 2.2

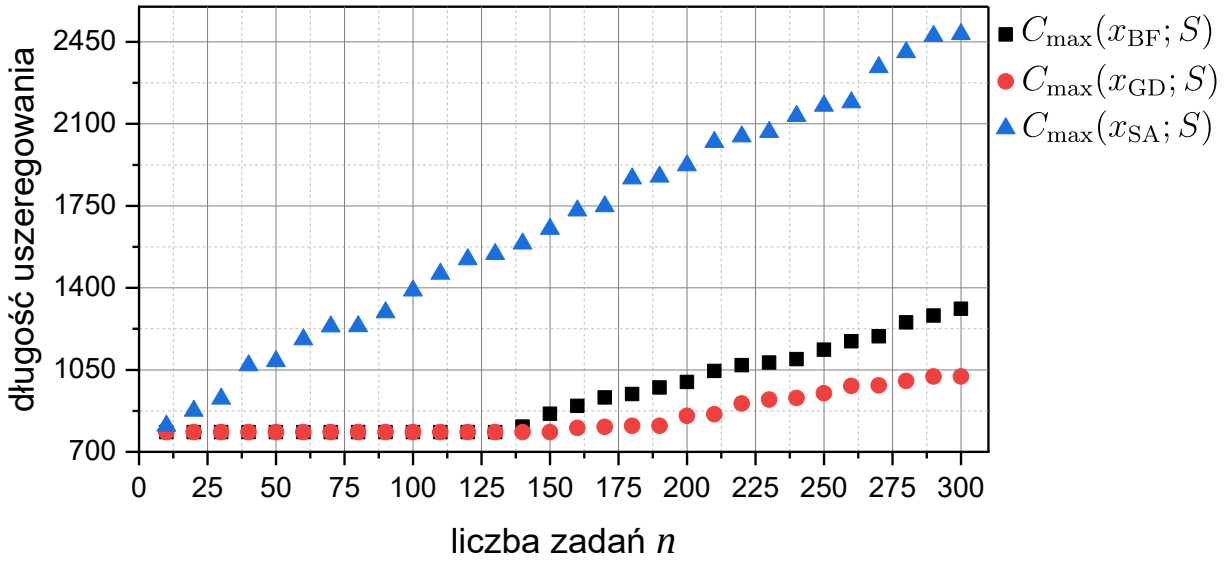
Drugie badanie ma na celu porównanie długości uszeregowań uzyskanych przez algorytmy **BF**, **GD** i **SA** oraz porównanie czasów zrealizowanych obliczeń. Wartości liczbowe rezultatów zaprezentowanych na rysunkach 2.3-2.8 umieszczono w tabelach A1-A5 (podrozdział A w dodatku).



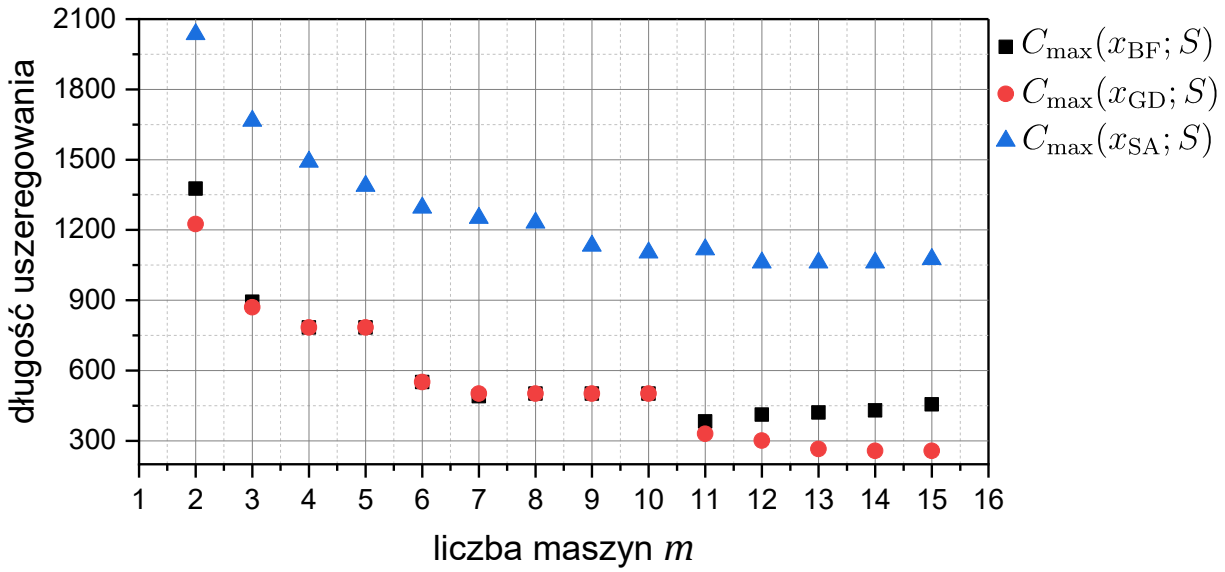
Rysunek 2.3. Wykres zależności długości uszeregowania od liczby zadań n dla $m = 2$



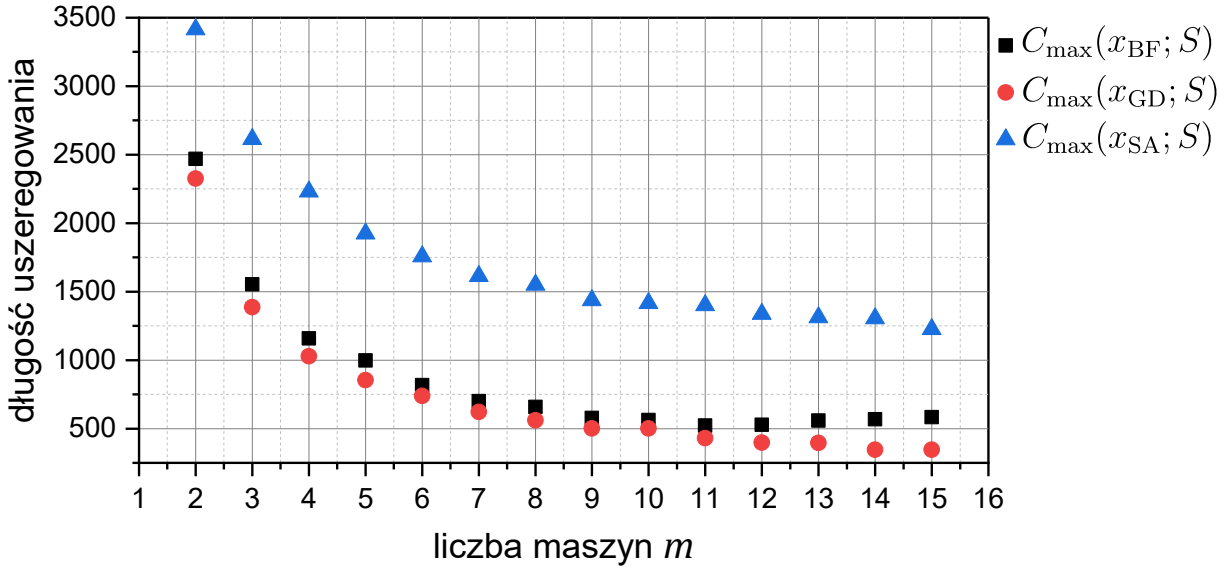
Rysunek 2.4. Wykres zależności długości uszeregowania od liczby zadań n dla $m = 5$



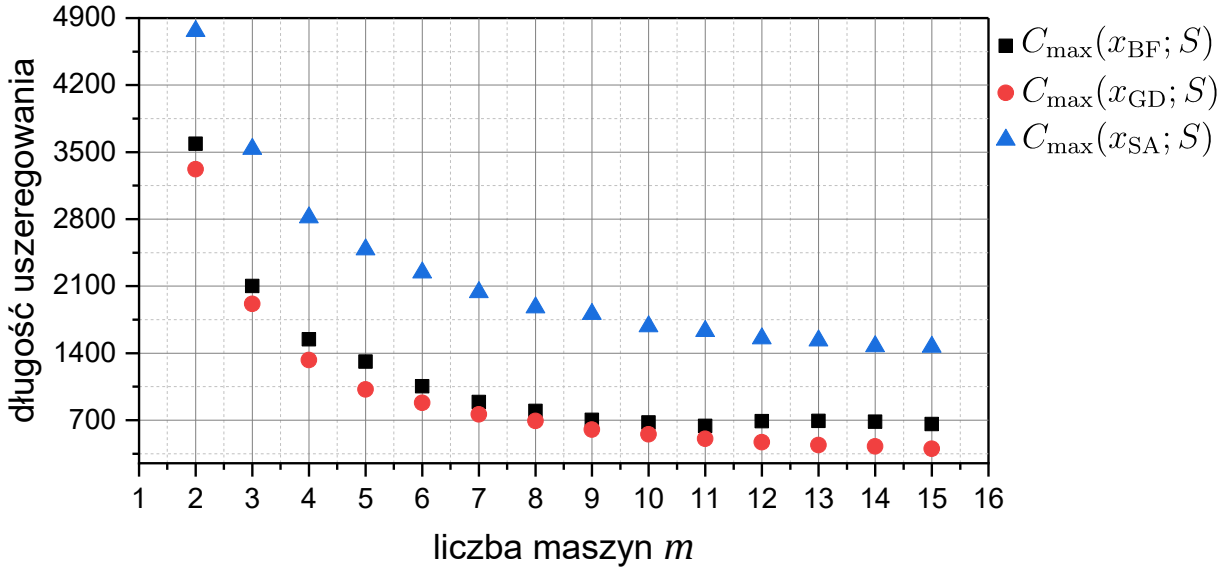
Rysunek 2.5. Wykres zależności długości uszeregowania od liczby zadań n dla $m = 10$



Rysunek 2.6. Wykres zależności długości uszeregowania od liczby maszyn m dla $n = 100$



Rysunek 2.7. Wykres zależności długości uszeregowania od liczby maszyn m dla $n = 200$

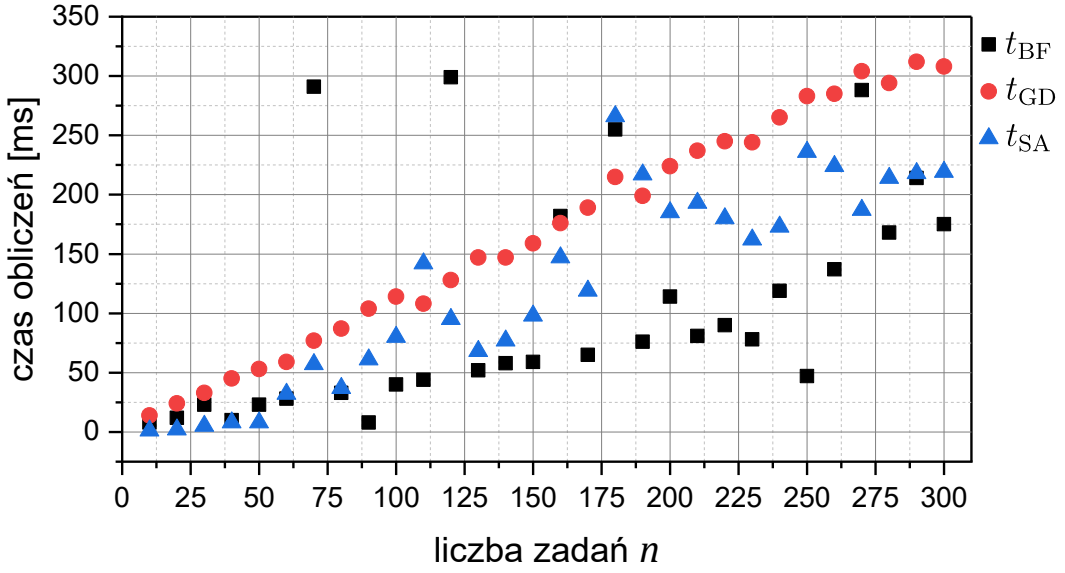


Rysunek 2.8. Wykres zależności długości uszeregowania od liczby maszyn m dla $n = 300$

Podstawową obserwacją, wynikającą z danych przedstawionych na rysunkach 2.3-2.8, jest fakt, że harmonogramy zadań wyznaczone przez algorytmy **BF** i **SA** nie zapewniły mniejszych wartości kryterium długości uszeregowania niż rezultaty uzyskane przez algorytm **GD**. Efektywność działania algorytmu **SA** nie ulegała poprawie nawet w przypadku, gdy rozwiązaniem początkowym był harmonogram utworzony przez algorytm **GD**. Łączny czas niewykorzystanych okien czasowych (możliwych do zagospodarowania przedziałów na osi czasu) był o wiele mniejszy w harmonogramach zaprojektowanych przez algorytmy **GD** i **BF** w porównaniu do rezultatów algorytmu **SA**. Skuteczność wypełniania

wolnych okien czasowych determinowała liniowy charakter wzrostu długości uszeregowania przy jednoczesnym zwiększaniu liczby zadań.

Efektywność algorytmu **GD** jest również widoczna również na rysunkach 2.6-2.8. Stabilizowanie się wyników wraz ze wzrostem liczby maszyn było konsekwencją wpływu terminów gotowości na sposób wyliczania wartości kryterium. Mianowicie, terminy gotowości zadań uniemożliwiają zmniejszenie długości uszeregowania, na przykład, gdy terminy gotowości podzbioru zadań są ustanowione tak późno na wszystkich maszynach, że większość zadań można uszeregować przed tymi terminami. Przy niedużej liczbie maszyn ($m < 7$) terminy gotowości były pokrywane czasami przetwarzania zadań i nie stanowiły dolnego ograniczenia wartości kryterium.



Rysunek 2.9. Wykres zależności czasu obliczeń od liczby zadań n dla $m = 2$

Cechą opracowanych algorytmów jest wyjątkowa szybkość realizacji obliczeń (rysunek 2.9). Czas obliczeń t_{SA} zrealizowanych przez algorytm **SA** dla pojedynczej instancji problemu jest równy medianie czasów z dwudziestu pięciu uruchomień algorytmu. Deterministyczny algorytm **GD** (najlepszy spośród opracowanych rozwiązań pod względem wartości kryterium) szeregował 300 zadań na dwóch maszynach w zaledwie $t_{GD} = 308$ [ms]. Zmienność czasów obliczeń realizowanych przez algorytm **SA** jest konsekwencją niedeterministycznego charakteru optymalizacji. Natomiast zmienność czasów obliczeń t_{BF} (czas znalezienia najlepszego rozwiązania) realizowanych przez algorytm **BF** jest wynikiem ustalenia odpowiedniej wartości parametru b.

Badanie 2.3

Sformułowanie problemu szeregowania $Rm|r_{i,j}|C_{\max}$, w którym uwzględniono jedną zmienną decyzyjną, ułatwia zrozumienie matematycznego modelu oraz podstawowych własności. Jednak rekurencyjna postać funkcji (2.1) utrudnia bezpośrednią implementację sformułowania problemu w oprogramowaniu dedykowanemu optymalizacji (np. solvery CPLEX, GUROBI itp.). Dlatego zaproponowano alternatywne sformułowanie problemu, które można zastosować w solverze CPLEX. Opracowane sformułowanie problemu stanowi rozszerzenie prac zrealizowanych przez Kooli & Serairi (2014) oraz Piasecki (2018) na przypadek uwzględniający wiele maszyn dowolnych oraz kryterium długości uszeregowania. Wprowadzono trzy zmienne optymalizacyjne:

1. Reprezentacja rozwiązania w postaci trójwymiarowej macierzy binarnej x (zmienna ma postać tożsamą z reprezentacją rozwiązania podaną w podrozdziale 2.3).
2. Termin zakończenia zadania $C_{i,k} \geq 0$ przetwarzanego na pozycji k w sekwencji zadań przypisanej do maszyny i. Jest to ciągła zmienna pomocnicza, która zastępuje rekurencyjną funkcję (2.1). Wszystkie terminy zakończeń są reprezentowane przez macierz C.
3. Długość uszeregowania $\tilde{C}_{\max} \geq 0$.

Problem szeregowania zadań $Rm|r_{i,j}|C_{\max}$ sprowadza się wówczas do

$$\min \tilde{C}_{\max}. \tag{2.32}$$

Na zmienne decyzyjne x, C oraz $\tilde{C}_{\max}$ nałożono następujące ograniczenia:

$$C_{i,k} \leq \tilde{C}_{\max}, \qquad i = 1,2,\dots,m,\ k = 1,2,\dots,n, \tag{2.33}$$

$$\sum_{j=1}^{n} x_{i,k,j}\left(r_{i,j} + p_{i,j}\right) \leq C_{i,k}, \qquad i = 1,2,\dots,m,\ k = 1,2,\dots,n, \tag{2.34}$$

$$C_{i,k-1} + x_{i,k,j}p_{i,j} \leq C_{i,k}, \qquad i = 1,2,\dots,m,\ k = 1,2,\dots,n,\ j = 1,2,\dots,n, \tag{2.35}$$

$$C_{i,0} = 0, \tag{2.36}$$

$$0 \leq C_{i,k},\ 0 \leq \tilde{C}_{\max}. \tag{2.37}$$

Ograniczenie długości uszeregowania podano w (2.33). Ograniczenie (2.34) powoduje, że przetwarzanie dowolnego zadania nie rozpocznie się przed terminem jego gotowości. Ograniczenie (2.35) wymusza kolejność realizacji zadań, która jest zgodna z wybranym

harmonogramem. Dopuszczalne wartości ciągłych zmiennych optymalizacyjnych podano w (2.36)-(2.37). Pozostałe ograniczenia podano w (2.4)-(2.7).

Aby porównać długość uszeregowania wyliczoną dla harmonogramu wyznaczonego przez algorytm **GD** z optymalną długością uszeregowania (uzyskaną z wykorzystaniem solvera CPLEX), zdefiniowano wskaźnik jakości

$$\hat{\theta} = \frac{C_{\max}(x_{\mathrm{GD}}; \mathrm{S}) - \tilde{C}^*_{\max}}{\tilde{C}^*_{\max}} 100\%, \tag{2.38}$$

gdzie $\tilde{C}^*_{\max}$ jest rozwiązaniem optymalnym. Dodatnia wartość $\hat{\theta}$ określa o ile procent wartość $C_{\max}(x_{\mathrm{GD}}; \mathrm{S})$ jest większa od optymalnej długości uszeregowania $\tilde{C}^*_{\max}$.

Dane do badań wygenerowano losowo dla każdej instancji z zakresów wartości podanych w niniejszym podrozdziale w badaniu 2.2. Rezultaty optymalizacji zaprezentowano w tabeli 2.3.

Tabela 2.3. Rezultaty optymalizacji dla $m = 2$

n	$C_{\max}(x_{\mathrm{GD}}; \mathrm{S})$	$\tilde{C}^*_{\max}$	t_{GD} [ms]	t_{CPLEX} [s]	$\hat{\theta}$ [%]
5	132	132	1	2	0,00
6	848	848	1	4	0,00
7	174	126	1	9	38,10
8	209	179	1	19	16,76
9	586	545	1	28	7,52
10	263	231	1	36	13,85
11	777	748	1	77	3,88
12	392	367	1	186	6,81
13	406	357	1	981	13,73
14	231	202	2	2538	14,36
15	551	526	2	6890	4,75

Uzyskane wyniki umożliwiają ocenę jakości rozwiązań generowanych przez algorytm **GD**. Dla przebadanych instancji problemu długość uszeregowania była gorsza nawet do 38,10%, ale były przypadki optymalne albo gorsze od optymalnego zaledwie o kilka procent. Natomiast czas działania solvera wynosił już około 114 minut dla $m = 2$ i $n = 15$.

Badanie 2.4

Omówione wyniki optymalizacji sugerują, że algorytm **GD** gwarantuje najlepszą jakość optymalizacji spośród opracowanych rozwiązań. Celem zweryfikowania tej tezy przeprowadzono badania statystyczne. Wygenerowano losowo 100 instancji, z których każda opisana jest krotką $(m, n, \bar{r}, \bar{p})$, gdzie $m \in \{1,2,\dots,15\}$, $n \in \{10,20,\dots,300\}$, $\underline{p} = 5$,

$\bar{p} \in \{6,7,\ldots,100\}$, $\underline{r} = 0$, $\bar{r} \in \{1,2,\ldots,100\}$. Parametry instancji oraz rezultaty optymalizacji zaprezentowano w tabeli A6 (podrozdział A w dodatku). Z badań wykluczono rezultaty zwracane przez algorytm **SA**, ponieważ dla żadnej wygenerowanej instancji algorytm nie uzyskał wyników lepszych od algorytmów **GD** i **BF**. Podczas pierwszego etapu badań statystycznych, do weryfikacji hipotezy zerowej mówiącej o tym, że rozkład rezultatów jest zbliżony do rozkładu normalnego wykorzystano test Shapiro-Wilka (Shapiro & Wilk, 1965). Wyniki testów przeprowadzonych dla poziomu istotności $\alpha = 0.05$:

- wartość statystyki $\mathrm{W}=0{,}6010$ oraz p-value $= 4{,}5\text{e-}15$ dla rezultatów algorytmu **BF**,
- wartość statystyki $\mathrm{W}=0{,}5822$ oraz p-value $= 2{,}0\text{e-}15$ dla rezultatów algorytmu **GD**,

pozwalają dla każdego zbioru odrzucić hipotezę zerową na rzecz hipotezy alternatywnej H_1, mówiącej o tym, że rozkład rezultatów nie jest zbliżony do rozkładu normalnego. Następnie przeprowadzono nieparametryczny test Wilcoxona dla par obserwacji (Wilcoxon, 1945) do weryfikacji hipotezy zerowej H_0 mówiącej o tym, że nie ma statystycznie istotnej różnicy pomiędzy rezultatami optymalizacji osiąganymi przez algorytmy **GD** i **BF**. Zgodnie z hipotezą alternatywna H_1, uszeregowanie zadań z wykorzystaniem algorytmu **BF** gwarantuje statystycznie większą długość uszeregowania niż algorytm **GD**. Sformułowane hipotezy mają postać:

$$\begin{aligned} H_0&: C_{\max}(x_{\mathrm{BF}}; \mathrm{S}) = C_{\max}(x_{\mathrm{GD}}; \mathrm{S}), \\ H_1&: C_{\max}(x_{\mathrm{BF}}; \mathrm{S}) > C_{\max}(x_{\mathrm{GD}}; \mathrm{S}). \end{aligned} \tag{2.39}$$

W 19% przypadków oba algorytmy zwracały harmonogramy gwarantujące równe długości uszeregowania. Z tego powodu zastosowano w teście Wilcoxona konserwatywne podejście, polegające na usunięciu takich samych par wartości. Wynikiem testu, przeprowadzonego na poziomie istotności $\alpha = 0.05$, jest $z_{val} = 7{,}6842$ oraz p-value $= 7{,}7\text{e-}15$, co pozwala odrzucić hipotezę zerową na rzecz hipotezy alternatywnej. Zastosowanie modyfikacji Pratta (Pratt, 1959) w teście Wilcoxona nie zmieniło rezultatu badania. Rezultatem badań statystycznych jest konkluzja, że harmonogramy będące rezultatem działania algorytm **GD** gwarantują statystycznie mniejszą długość uszeregowania niż harmonogramy uzyskane przez algorytm **BF**.

Rezultat badań statystycznych potwierdził prawdziwość tezy numer 1, którą sformułowano w podrozdziale 1.5.

3. Problemy szeregowania zadań z przedziałowymi (niepewnymi) terminami gotowości

3.1. Wprowadzenie

Rozdział jest poświęcony szczegółowej analizie niedeterministycznych problemów szeregowania zadań z przedziałowymi terminami gotowości na maszynach dowolnych z kryterium maksymalnego żalu dla długości uszeregowania. Rozważono dwa przypadki niedeterministycznej wersji problemu $Rm|r_{i,j}|C_{\max}$:

1. $Rm|r_{i,j}^{-} \le r_{i,j} \le r_{i,j}^{+}|Reg(C_{\max})$ (przedziałowe terminy gotowości zależą od maszyn),
2. $Rm|r_{j}^{-} \le r_{j} \le r_{j}^{+}|Reg(C_{\max})$ (przedziałowe terminy gotowości zależą jedynie od zadań).

Celem opracowania efektywnych algorytmów oraz precyzyjnego oszacowania wartości kryterium maksymalnego żalu wykazano, że liczbę scenariuszy dla terminów gotowości można ograniczyć do zbioru skończonego oraz udowodniono nieaproksymowalność omawianych problemów. Wskazano różnice w interpretacji kryterium, gdy przedziałowe wartości parametrów zależą od maszyn lub są cechami samych zadań. Dla problemu $Rm|r_{i,j}^{-} \le r_{i,j} \le r_{i,j}^{+}|Reg(C_{\max})$ opracowano dwa algorytmy konstrukcyjne, które bazują na zdekomponowanej wersji problemu optymalizacji odpornej, bazują na deterministycznej wersji problemu oraz wykorzystano metaheurystykę Tabu Search. Dla problemu $Rm|r_{j}^{-} \le r_{j} \le r_{j}^{+}|Reg(C_{\max})$ opracowano algorytm konstrukcyjny, który generuje harmonogram bazując na deterministycznym kryterium długości uszeregowania. Sformułowano warunki gwarantujące uzyskanie optymalnego rozwiązania w czasie wielomianowym. Interpretacja wyników obejmuje badania eksperymentalne oraz badania statystyczne. Fragmenty badań umieszczonych w niniejszym rozdziale zostały opublikowane w Józefczyk & Ławrynowicz M. (2021) oraz Ławrynowicz & Józefczyk (2021).

3.2. Problem $\boldsymbol{Rm|r_{i,j}^{-} \le r_{i,j} \le r_{i,j}^{+}|Reg(C_{\max})}$

Prezentowane sformułowanie zakłada niepewność przedziałową w odniesieniu do terminów gotowości zadań. Nie założono na przykład wstępnej wiedzy o rozkładzie prawdopodobieństwa niepewnego parametru (wszystkie wartości są równie prawdopodobne). Przyjmujemy, że termin gotowości zadania j na maszynie i jest niepewny i należy do znanego przedziału $u_{i,j} = \left[r_{i,j}^{-}, r_{i,j}^{+}\right]$, $r_{i,j}^{-} \le r_{i,j} \le r_{i,j}^{+}$. Niepewność terminów gotowości zadania j jest modelowana za pomocą iloczynu kartezjańskiego precyzyjnie określonych

przedziałów $\mathrm{U}_j = u_{1,j} \times u_{2,j} \times \ldots \times u_{i,j} \times \ldots \times u_{m,j}$. Zbiór wszystkich możliwych scenariuszy jest równy $\mathrm{U} = \mathrm{U}_1 \times \mathrm{U}_2 \times \ldots \times \mathrm{U}_j \times \ldots \times \mathrm{U}_n$, a pojedynczy scenariusz $u \in \mathrm{U}$ jest określony macierzą terminów gotowości $u = \left[r_{i,j}\right]_{\substack{i=1,2,\ldots,m \\ j=1,2,\ldots,n}}$. Harmonogram $\tilde{x}$, będący rozwiązaniem omawianego niedeterministycznego problemu szeregowania, jest reprezentowany przez trójwymiarową macierz binarną (analogicznie jak rozwiązanie problemu deterministycznego $Rm|r_{i,j}|C_{\max}$). Uszeregowanie jest reprezentowane przez macierz binarną $\tilde{x} = \left[\tilde{x}_{i,k,j}\right]_{\substack{i=1,2,\ldots,m \\ k,j=1,2,\ldots,n}}$, $\tilde{x}_{i,k,j} \in \{0,1\}$.

Funkcję żalu dla kryterium długości uszeregowania i przedziałowych terminów gotowości można przedstawić jako

$$Q(\tilde{x}, u; p) = C_{\max}(\tilde{x}, u; p) - C_{\max}(\tilde{x}_u^*; p), \tag{3.1}$$

gdzie $\tilde{x}_u^*$ jest optymalnym uszeregowaniem dla problemu deterministycznego $Rm|r_{i,j}|C_{\max}$ przy scenariuszu $u \in \mathrm{U}$. Wzór (3.1) określa odchylenie wartości kryterium dla harmonogramu $\tilde{x}$ wyznaczonego przez decydenta, przy scenariuszu u, od optymalnej wartości kryterium dla tego scenariusza. Zatem wartość maksymalnego żalu $Z(\tilde{x})$, względem wszystkich możliwych scenariuszy U, przyjmuje postać

$$Z(\tilde{x}) = \max_{u \in \mathrm{U}} Q(\tilde{x}, u; p) \tag{3.2}$$

oraz scenariusz $\hat{u} = \arg\max_{u \in \mathrm{U}} Q(\tilde{x}, u; p)$ nazywamy najgorszym scenariuszem.

Sformułowanie problemu można przedstawić w następujący sposób: Dla danych J, M, U, p, niedeterministyczny problem szeregowania wymaga znalezienia optymalnej macierzy $\tilde{x}^*$ minimalizującej maksymalny żal

$$Z(\tilde{x}^*) = \min_{\tilde{x}} Z(\tilde{x}), \tag{3.3}$$

przy ograniczeniach nałożonych na zmienną $\tilde{x}$, które są tożsame z ograniczeniami (2.4)-(2.7). Przyjęcie założenia $\underset{\substack{i=1,2,\ldots,m \\ j=1,2,\ldots,n}}{\forall} r_{i,j}^+ = r_{i,j}^-$ sprowadza niedeterministyczny problem szeregowania (3.3) do jego deterministycznego odpowiednika $Rm|r_{i,j}|C_{\max}$ z precyzyjnie określonymi terminami gotowości, który jest co najmniej NP-trudny. Z tego powodu niedeterministyczny problem (3.3) jest również co najmniej NP-trudny.

3.2.1. Podstawowe własności

W pierwszej kolejności wykażemy, że założona przedziałowa niepewność implikuje brak możliwości zaprojektowania algorytmu aproksymacyjnego.

Własność 3.1. *Nie można zaprojektować algorytmu* α*-aproksymacyjnego dla problemu* $Rm|r_{i,j}^- \le r_{i,j} \le r_{i,j}^+|Reg(C_{\max})$.

Dowód. Przypuśćmy, że istnieje algorytm α-aproksymacyjny gwarantujący rozwiązanie $\tilde{x}^{(s)}$ spełniające nierówność $Z(\tilde{x}^{(s)}) \le \alpha Z(\tilde{x}^*)$. Przyjęcie warunku $\underset{\substack{i=1,2,\dots,m \\ j=1,2,\dots,n}}{\forall} r_{i,j}^+ = r_{i,j}^-$ prowadzi do równości $Z(\tilde{x}^{(s)}) = Z(\tilde{x}^*) = 0$, ponieważ zbiór U zawiera tylko jeden scenariusz realizacji parametrów niepewnych, $|\mathrm{U}| = 1$. Algorytm aproksymacyjny musiałby generować uszeregowanie optymalne $\tilde{x}^{(s)} = \tilde{x}_u^{(s)*}$, gwarantujące optymalną długość uszeregowania $C_{\max}(\tilde{x}^{(s)}, u; p) = C_{\max}(\tilde{x}_u^{(s)*}; p)$, czego konsekwencją musiałoby być rozwiązanie problemu $Rm|r_{i,j}|C_{\max}$ w czasie wielomianowym. Q.E.D.

Fakt nieaproksymowalności problemów z niepewnością przedziałową, który jest rezultatem NP-trudności deterministycznej wersji problemu, jest częstą własnością problemów optymalizacji odpornej. Dowody nieaproksymowalności problemów szeregowania z przedziałowymi czasami przetwarzania zadań można znaleźć na przykład w Siepak (2013) lub Ćwik (2017).

Kluczową kwestią w obliczaniu oszacowanej wartości kryterium (3.3) jest wybór podzbioru scenariuszy, z których co najmniej jeden gwarantuje maksymalną wartość żalu (3.2). Scenariusze podzielono na dwa rodzaje:

- **scenariusz skrajny** $\tilde{u}^{i,j}$, w którym termin gotowości zadania j na maszynie i jest równy $r_{i,j} = r_{i,j}^+$, a pozostałe terminy gotowości przyjmujemy jako najwcześniejsze możliwe, czyli $\underset{\substack{s\in \mathrm{M}\setminus\{i\} \\ t\in \mathrm{J}\setminus\{j\}}}{\forall} r_{s,t} = r_{s,t}^-$,
- **scenariusz pośredni** $\tilde{u}_o^{i,j}$, w którym $r_{i,j} \in [r_{i,j}^-, r_{i,j}^+)$ oraz $\underset{\substack{s\in \mathrm{M}\setminus\{i\} \\ t\in \mathrm{J}\setminus\{j\}}}{\forall} r_{s,t} = r_{s,t}^-$.

Następujące twierdzenie uzasadnia fakt, że przegląd scenariuszy można ograniczyć do przeglądu skończonego zbioru oraz dwa wymienione wyżej typy scenariuszy (skrajny, pośredni) mogą jednocześnie maksymalizować wartość (3.2).

Twierdzenie 3.1. *Maksymalną wartość żalu można osiągnąć dla*

1. *scenariusza skrajnego* $\tilde{u}^{i,j}$, $i_m = \arg\max_{i=1,2,\dots,m} C_{i,n_i}(\tilde{x}, \tilde{u}^{i,j}; p)$, $\tilde{x}_{i_m,j,k} = 1$ *lub*
2. *scenariusza pośredniego* $\tilde{u}_o^{i,j}$.

Dowód. Bazując na własności 2.1, funkcję żalu można zapisać w następujący sposób:

$$
\begin{aligned}
Z(\tilde{x}) &= \max_{u\in \mathrm{U}} Q(\tilde{x}, u; p) = \max_{u\in \mathrm{U}} \{C_{\max}(\tilde{x}, u; p) - C_{\max}(\tilde{x}_u^*; p)\} \\
&= \max_{u\in \mathrm{U}} \left\{ \hat{r}_{i_m,j}^{(u)} + \sum_{e\in \mathrm{A}_{i_m,j}^{(u)}(\tilde{x})} p_{i_m,e} - \left(\hat{r}_{i_m^*,t^*}^{(u)} + \sum_{v\in \mathrm{A}_{i_m^*,t^*}^{(u)}(\tilde{x}_u^*)} p_{i_m,v} \right) \right\} \\
&= \max_{u\in \mathrm{U}} \left\{ \left(\hat{r}_{i_m,j}^{(u)} - \hat{r}_{i_m^*,t^*}^{(u)} \right) + \left(\sum_{e\in \mathrm{A}_{i_m,j}^{(u)}(\tilde{x})} p_{i_m,e} - \sum_{v\in \mathrm{A}_{i_m^*,t^*}^{(u)}(\tilde{x}_u^*)} p_{i_m,v} \right) \right\}.
\end{aligned}
\tag{3.4}
$$

Bieżący indeks j w (3.4) wskazuje najpóźniej rozpoczynające się zadanie, według sekwencji ustalonej na najdłużej pracującej maszynie $i_m = \arg\max_{i=1,2,\dots,m} C_{i,n_i}(\tilde{x}, u; p)$, które rozpoczyna się w terminie swojej gotowości $\hat{r}_{i_m,j}^{(u)}$ przy scenariuszu u. Termin gotowości $\hat{r}_{i_m,j}^{(u)} \in \left[r_{i_m,j}^-, r_{i_m,j}^+\right]$ jest odpowiednikiem terminu podanego w (2.9), ale jego wartość zależy od zrealizowanego scenariusza u. Natomiast zbiór $\mathrm{A}_{i_m,j}^{(u)}(\tilde{x})$ jest odpowiednikiem zbioru zdefiniowanego w (2.10). Optymalna wartość $\hat{r}_{i_m^*,t^*}^{(u)}$ oraz optymalna zawartość zbioru $\mathrm{A}_{i_m^*,t^*}^{(u)}(\tilde{x}_u^*)$ nie są znane, ponieważ ustalenie $\tilde{x}_u^*$ jest problemem co najmniej NP-trudnym. Wystarczy zauważyć, że największa wartość różnicy $\hat{r}_{i_m,j}^{(u)} - \hat{r}_{i_m^*,t^*}^{(u)}$ jest osiągana dla skrajnych wartości przedziałów niepewności $\hat{r}_{i_m,j}^{(u)} = r_{i_m,j}^+$, $\hat{r}_{i_m^*,t^*}^{(u)} = r_{i_m,t}^-$, a terminy gotowości zadań w zbiorze $\mathrm{J}\backslash\{t^*, j\}$ nie wpływają na wartość maksymalnego żalu. Zatem przegląd wszystkich scenariuszy skrajnych gwarantuje znalezienie najgorszego scenariusza $\hat{u}$ oraz maksymalnej wartości żalu $Z(\tilde{x})$.

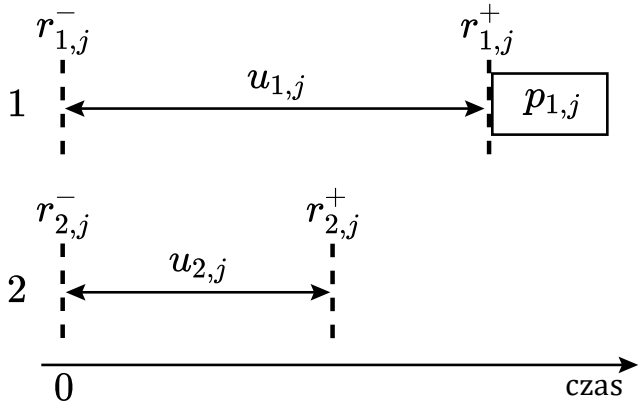
Sposób obliczania wartości kryterium determinuje również możliwość wyboru optymalnego scenariusza pośredniego. Niezerowa wartość żalu w (3.4) może wystąpić jedynie, gdy zadania występujące po terminie $\hat{r}_{i_m,j}^{(u)}$ w $\tilde{x}$ przesuną się na wcześniejsze pozycje i zakończoną się wcześniej według uszeregowania optymalnego $\tilde{x}_u^*$. Przyjmijmy, że termin $\hat{r}_{i_m,j}^{(u)}$ w (3.4) jest równy $\hat{r}_{i_m^*,t^*}^{(u)}$, czyli $\hat{r}_{i_m,j}^{(u)} = \hat{r}_{i_m^*,t^*}^{(u)}$. W konsekwencji wartość (3.14),

przy scenariuszu u, nie zależy od żadnego terminu gotowości i najgorszym scenariuszem może być zarówno scenariusz skrajny jak i scenariusz pośredni. Q.E.D.

Zbiór scenariuszy skrajnych, wystarczających do obliczenia wartości maksymalnego żalu dla ustalonego uszeregowania $\tilde{x}$, oznaczono w dalszej części rozprawy jako $\tilde{\mathrm{U}}$, $\left|\tilde{\mathrm{U}}\right| = nm$. Charakterystyczną cechą funkcji żalu (3.1) jest to, że do obliczenia maksymalnego żalu (3.2) wystarczy wykorzystać jedynie n scenariuszy skrajnych w $C_{\max}(\tilde{x}, \tilde{u}; p)$, $\tilde{u} \in \tilde{\mathrm{U}}$, ale nm scenariuszy w $C_{\max}(\tilde{x}^*_{\tilde{u}}; p)$. Jeżeli nie uszeregowano zadania j na maszynie i oraz pozycji k w sekwencji, czyli $\tilde{x}_{i,k,j} = 0$, to przyjęcie scenariusza $\tilde{u}^{i,j}$ nie wpływa na długość uszeregowania, ponieważ pesymistyczny termin $r^+_{i,j}$ nie jest uwzględniany w momencie wyliczania wartości $C_{\max}(\tilde{x}, \tilde{u}^{i,j}; p)$. W rezultacie, przy ustalonym harmonogramie $\tilde{x}$, wartość $C_{\max}(\tilde{x}, \tilde{u}^{i,j}; p)$ można obliczać jedynie, gdy $\tilde{x}_{i,k,j} = 1$, czyli dla n zadań. Natomiast optymalna długość uszeregowania $C_{\max}(\tilde{x}^*_{\tilde{u}}; p)$ zależy od wszystkich nm scenariuszy skrajnych.

Zwróćmy uwagę na jeszcze inną własność rozpatrywanego problemu. Okazuje się, że wartość kryterium maksymalnego żalu nie jest ograniczona sumą czasów realizacji zadań na najdłużej pracującej maszynie, które są przetwarzane od terminu $\hat{r}^{(u)}_{i_m,j}$. Interpretując (3.4) można zaobserwować, że możliwe jest spełnienie nierówności

$$\left(\hat{r}^{(u)}_{i_m,j} - \hat{r}^{(u)}_{i^*_m,t^*}\right) > \left(\sum_{e \in \mathrm{A}^{(u)}_{i_m,j}(\tilde{x})} p_{i_m,e} - \sum_{v \in \mathrm{A}^{(u)}_{i^*_m,t^*}(\tilde{x}^*_u)} p_{i_m,v}\right). \tag{3.5}$$



Rysunek 3.1. Instancja problemu: $\mathrm{M} = \{1,2\}$, $\mathrm{J} = \{1\}$, $|u_{1,j}| > p_{1,j}$, $|u_{2,j}| > p_{2,j}$, $|u_{1,j}| > |u_{2,j}|$, $r^-_{1,j} = r^-_{2,j} = 0$, $p_{1,j} > p_{2,j}$.
Długość przedziału niepewności oznaczono jako $|\ldots|$

W skrajnym przypadku wszystkie zadania gotowe do przetwarzania od terminu $\hat{r}_{i_m,j}^{(u)}$ według harmonogramu $\tilde{x}$ mogą być uszeregowane wcześniej lub na wydajniejszych maszynach według optymalnego harmonogramu $\tilde{x}_u^*$. Przypadek nieoptymalnej decyzji o uszeregowaniu jednego zadania, który spełnia nierówność (3.5), zaprezentowano na rysunku 3.1.

Z rysunku 3.1 można wywnioskować, że pomimo iż w systemie $n = 1$ to $Z(\tilde{x}^*) \neq 0$. W przypadku jednego zadania w systemie mogą wystąpić jedynie dwa scenariusze skrajne dla terminów gotowości $\tilde{u}^{1,j}$, $\tilde{u}^{2,j}$. Jeżeli podjęto decyzję $\tilde{x}_{1,1,j} = 1$, to $Z(\tilde{x}) = r_{1,j}^{+} + p_{1,j} - \left(0 + p_{2,j}\right) \neq 0$. Natomiast przy decyzji $\tilde{x}_{2,1,j} = 1$ również zachodzi $Z(\tilde{x}) = r_{2,j}^{+} + p_{2,j} - \left(0 + p_{1,j}\right) \neq 0$.

Z analitycznego i praktycznego punktu widzenia bardzo istotną cechą problemu jest możliwość wykluczenia podzbioru nieefektywnych scenariuszy skrajnych, które nie mogą maksymalizować wartości żalu. We własności 3.2 podano warunek konieczny, który musi być spełniony, aby wybrany scenariusz skrajny był najgorszym scenariuszem.

Własność 3.2. *Jeżeli $\tilde{u}^{i_m,j}$ jest najgorszym scenariuszem, to spełniona jest nierówność*

$$C_{i_m,k-1}\left(\tilde{x}, \tilde{u}^{i_m,j}; p\right) \leq r_{i_m,j}^{+}, \qquad \tilde{x}_{i_m,j,k} = 1, \qquad i_m = \arg \max_{i=1,2,\ldots,m} C_{i,n_i}\left(\tilde{x}, \tilde{u}^{i,j}; p\right). \tag{3.6}$$

Dowód. Jeżeli $C_{i_m,k-1}\left(\tilde{x}, \tilde{u}^{i_m,j}; p\right) > r_{i_m,j}^{+}$, to termin gotowości $r_{i_m,j}^{+}$ może jedynie zwiększać optymalną długość uszeregowania $C_{\max}\left(\tilde{x}_{\tilde{u}^{i_m,j}}^*; p\right)$ w porównaniu do $C_{\max}\left(\tilde{x}_{\tilde{u}_o^{i_m,j}}^*; p\right)$, gdzie $r_{i_m,j} \in \left[r_{i_m,j}^{-}, r_{i_m,j}^{+}\right)$, ponieważ $C_{\max}\left(\tilde{x}, \tilde{u}^{i_m,j}; p\right) = C_{\max}\left(\tilde{x}, \tilde{u}_o^{i_m,j}; p\right)$. Zatem nie może zachodzić

$$C_{\max}\left(\tilde{x}, \tilde{u}^{i_m,j}; p\right) - C_{\max}\left(\tilde{x}_{\tilde{u}^{i_m,j}}^*; p\right) > C_{\max}\left(\tilde{x}, \tilde{u}_o^{i_m,j}; p\right) - C_{\max}\left(\tilde{x}_{\tilde{u}_o^{i_m,j}}^*; p\right). \tag{3.7}$$

Q.E.D.

Konsekwencją własności 2.1 oraz 3.2 jest wzór $Z(\tilde{x}) = r_{i_m,j}^{+} + \sum_{e \in \mathrm{A}_{i_m,j}^{(\hat{u})}(\tilde{x})} p_{i_m,e} - C_{\max}(\tilde{x}_{\hat{u}}^*; p)$, gdzie $\hat{u} = \tilde{u}^{i_m,j}$ jest najgorszym scenariuszem. Bazując na twierdzeniu 3.1 oraz własności 3.2 zdefiniowano następujący zbiór **scenariuszy istotnych**

$$\mathrm{U}^{(\mathrm{I})}(\mathrm{J}, \mathrm{M}, \mathrm{U}, \tilde{x}) = \left\{ \begin{array}{c} \tilde{u}^{i_m,j} \in \mathrm{U} \mid C_{i_m,k-1}\left(\tilde{x}, \tilde{u}^{i_m,j}; p\right) \leq r_{i_m,j}^{+}, \qquad \tilde{x}_{i_m,j,k} = 1, \\ i_m = \arg \max_{i=1,2,\ldots,m} C_{i,n_i}\left(\tilde{x}, \tilde{u}^{i,j}; p\right) \ \wedge \\ C_{\max}\left(\tilde{x}, \tilde{u}^{i_m,j}; p\right) = r_{i_m,j}^{+} + \sum_{e \in \mathrm{A}_{i_m,j}^{\left(\tilde{u}^{i_m,j}\right)}(\tilde{x})} p_{i_m,e} \end{array} \right\}. \tag{3.8}$$

Zbiór $\mathrm{U}^{(\mathrm{I})}(\mathrm{J},\mathrm{M},\mathrm{U},\tilde{x}) \triangleq \tilde{\mathrm{U}}^{(\mathrm{I})}$ jest skończony oraz zawiera scenariusze skrajne spełniające warunek sformułowany w (3.6). Scenariusze w (3.8) pozwalają obliczyć przedział możliwych wartości kryterium (3.2). Aby wskazać dokładny zakres wartości, które może osiągnąć funkcja żalu (3.1), długość uszeregowania $C_{\max}\left(\tilde{x}^{*}_{\tilde{u}^{i,j}};p\right)$, $\tilde{u}^{i,j} \in \tilde{\mathrm{U}}^{(\mathrm{I})}$, zastępujemy funkcją (oszacowaniem) (2.19)

$$Q\left(\tilde{x},\tilde{u}^{i,j};p\right) \le Q^{(o)}\left(\tilde{x},\tilde{u}^{i,j};p\right) = C_{\max}\left(\tilde{x},\tilde{u}^{i,j};p\right) - \max_{s=1,2,\dots,m}\min_{t=1,2,\dots,n}\left(r_{s,t}+p_{s,t}\right)$$
$$= r^{+}_{i,j} + \sum_{e\in \mathrm{A}^{\left(\tilde{u}^{i_m,j}\right)}_{i_m,j}(\tilde{x})} p_e - \max_{s=1,2,\dots,m}\min_{t=1,2,\dots,n}\left(r_{s,t}+p_{s,t}\right). \tag{3.9}$$

Funkcja (2.19) zwraca najmniejszą możliwą wartość oszacowania $C_{\max}\left(\tilde{x},\tilde{u}^{i,j};p\right)$, ponieważ bazuje na parametrach jednego zadania i nie narusza ograniczeń wynikających z terminów gotowości zadań. W rezultacie wartość kryterium $Z(\tilde{x}^{*})$ zawiera się w przedziale

$$0 \le Z(\tilde{x}^{*}) \le \max_{\tilde{u}^{i,j}\in\tilde{\mathrm{U}}^{(\mathrm{I})}} Q^{(o)}\left(\tilde{x},\tilde{u}^{i,j};p\right). \tag{3.10}$$

Ograniczenia podane w (3.10) definiują cały (dokładny) zakres możliwych wartości kryterium maksymalnego żalu. Do ewaluacji kryterium (3.2) wykorzystano oszacowanie

$$\tilde{Z}(\tilde{x}) = \max_{\tilde{u}^{i,j}\in\tilde{\mathrm{U}}^{(\mathrm{I})}}\left\{C_{\max}\left(\tilde{x},\tilde{u}^{i,j};p\right) - LB\left(\tilde{u}^{i,j};p\right)\right\}, \tag{3.11}$$

w którym rozwiązanie problemu $C_{\max}\left(\tilde{x}^{*}_{\tilde{u}^{i,j}};p\right)$ zastąpiono funkcją oszacowania $LB\left(\tilde{u}^{i,j};p\right)$, opracowaną dla problemu szeregowania $Rm|r_{i,j}|C_{\max}$ i przedstawioną podrozdziale 2.4. Wartość (3.11) określa największą wartość odchylenia długości uszeregowania, która jest obliczona na podstawie harmonogramu $\tilde{x}$ zaproponowanego przez decydenta, od oszacowanej wartości kryterium długości uszeregowania przy uwzględnieniu wszystkich scenariuszy istotnych.

Przeanalizujmy wpływ wartości $LB\left(\tilde{u}^{i,j};p\right)$, $i = 1,2,\dots,m$, $j = 1,2,\dots,n$, na oszacowaną wartość maksymalnego żalu (3.11). Warto podkreślić fakt, że różnica pomiędzy wartościami $LB\left(\tilde{u}^{i,j};p\right)$ i $LB(\tilde{u}^{s,t};p)$, $i \neq s$, $j \neq t$, wynika z tego, że scenariusze $\tilde{u}^{i,j}$ i $\tilde{u}^{s,t}$ różnią się terminami gotowości jedynie dwóch zadań j i t. Zaprezentowane w podrozdziale 2.4 metody obliczania wartości $LB\left(\tilde{u}^{i,j};p\right)$ nie wykorzystują terminów gotowości innych niż najwcześniejsze, wyłączając (2.19) (jeżeli maszyna i, oferująca najpóźniejszy termin gotowości dla zadania j, umożliwia jego najwcześniejsze zakończenie). Zatem wybór scenariusza skrajnego $\tilde{u}^{i,j}$ może jedynie wykluczać termin gotowości $r^{+}_{i,j}$

z obliczania wartości $LB(\tilde{u}^{i,j};p)$, ponieważ następuje opóźnienie terminu gotowości zadania j względem najwcześniejszego terminu $r_{i,j}^{-}$. Wykluczenie przy scenariuszu $\tilde{u}^{i,j}$ nie następuje w przypadku, gdy termin $r_{i,j} \in u_{i,j}$ jest najwcześniejszy dla dowolnej wartości z przedziału $u_{i,j}$.

Nie zastosowano algorytmów heurystycznych do oszacowania wartości $C_{\max}(\tilde{x}^{*}_{\tilde{u}^{i,j}};p)$, aby uniknąć niedoszacowania wartości kryterium maksymalnego żalu (3.2), czyli aby uniknąć przypadku, w którym oszacowana maksymalna wartość żalu mogłaby być mniejsza niż optymalne rozwiązanie. Wykorzystanie dolnego ograniczenia $LB(\tilde{u}^{i,j};p)$ w (3.11) może jedynie gwarantować przeszacowanie wartości kryterium (3.2), ponieważ $C_{\max}(\tilde{x},\tilde{u}^{i,j};p) \geq LB(\tilde{u}^{i,j};p)$, co stanowi podejście pesymistyczne.

3.2.2. Algorytm konstrukcyjny D_Regret

Algorytm konstrukcyjny **D_Regret** (**DR**) szereguje zadania w oparciu o zdekomponowaną wersję problemu (3.3). Podczas każdego etapu podejmowania decyzji minimalizowane jest odchylenie długości uszeregowania od oszacowanej wartości optymalnej przy jednym scenariuszu skrajnym. Procedura szeregowania sprowadza się do rozwiązania n podproblemów

$$\Lambda = \arg \min_{\substack{i=1,2,\dots,m \\ k=1,2,\dots,n \\ j\in \mathrm{J}(\eta)}} \{C_{i,k}(\tilde{x}(\eta) = [\tilde{x}_{i,k,j}(\eta) = 1], \tilde{u}^{i,j};p) - LB(\tilde{u}^{i,j};p)\}, \tag{3.12}$$

gdzie $\mathrm{J}(\eta)$ jest zbiorem zadań nieuszeregowanych podczas rozwiązywania podproblemów o indeksach $1,2,\dots,\eta-1$ a krotka $(s,d,t) \in \Lambda$ stanowi podstawę do podjęcia decyzji $\tilde{x}_{s,d,t}(\eta) = 1$. Forma dekompozycji zaproponowana w (3.12) skutkuje optymalizacją z uwzględnieniem długiego horyzontu czasowego, ponieważ posługujemy się apriorycznie wyznaczoną wartością $LB(\tilde{u}^{i,j};p)$, obliczoną na podstawie parametrów wszystkich zadań i maszyn. Skutkiem wybranej strategii optymalizacji jest możliwość występowania ujemnych wartości różnicy $C_{\max}(\tilde{x}(\eta),\tilde{u}^{i,j};p) - LB(\tilde{u}^{i,j};p) < 0$ dla pewnego podzbioru podproblemów, jeżeli harmonogram $\tilde{x}(\eta)$ nie uwzględnia wszystkich zadań. Jeżeli rezultatem rozwiązania podproblemu o indeksie η w (3.12) są co najmniej dwie krotki ($|\Lambda| > 1$), to najmniejsza wartość wskaźnika

$$b_t(\tilde{x}(\eta),\tilde{u}^{s,t}) = \max\{C_{s,d-1}(\tilde{x}(\eta),\tilde{u}^{s,t};p) - r_{s,t}^{+}, 0\},\ \ \tilde{x}_{s,d,t} = 1, \tag{3.13}$$

determinuje decyzję o przypisaniu. Celem (3.13) jest wyliczenie najdłuższego czasu oczekiwania na gotowość zadania t przy scenariuszu skrajnym $\tilde{u}^{s,t}$. Konsekwencją decyzji $\tilde{x}_{s',d',t'}(\eta) = 1$, podjętej na podstawie $(s', d', t') = \arg\min_{(s,d,t)\in\Lambda} b_t(\tilde{x}(\eta), \tilde{u}^{s,t})$, jest zmniejszanie czasu trwania niezagospodarowanego okna czasowego w harmonogramie. Algorytm wskazuje dowolne zadanie, jeżeli $\left|\arg\min_{(s,d,t)\in\Lambda} b_t(\tilde{x}(\eta), \tilde{u}^{s,t})\right| > 1$.

Algorytm **D_Regret (DR)**

Require: J, M, U, p
Ensure: $\tilde{x}_{\mathrm{DR}}$

1. Przypisz $\eta \coloneqq 1$, $\mathrm{J}(\eta) \coloneqq \mathrm{J}$, $\tilde{x}(\eta) \coloneqq \left[\tilde{x}_{i,k,j}(\eta) \coloneqq 0\right]_{\substack{i=1,2,\dots,m \\ k,j=1,2,\dots,n}}$.
2. $\forall_{\substack{i=1,2,\dots,m \\ j=1,2,\dots,n}}$ oblicz dolne ograniczenie $LB\left(\tilde{u}^{i,j}; p\right)$.
3. **while** $\mathrm{J}(\eta) \neq \emptyset$
4. Wyznacz krotki $\Lambda \coloneqq \arg\min_{\substack{i=1,2,\dots,m \\ k=1,2,\dots,n \\ j\in\mathrm{J}(\eta)}} \left\{C_{i,k}\left(\tilde{x}(\eta) \coloneqq \left[\tilde{x}_{i,k,j}(\eta) \coloneqq 1\right], \tilde{u}^{i,j}; p\right) - LB\left(\tilde{u}^{i,j}; p\right)\right\}$.
5. **if** $|\Lambda| > 1$ **then** $(s', d', t') \coloneqq \arg\min_{(s,d,t)\in\Lambda} b_t(\tilde{x}(\eta), \tilde{u}^{s,t})$ **end if**
6. Przypisz $\tilde{x}_{s,d,t}(\eta) \coloneqq 1$ \\ krotka (s', d', t') jest wykorzystywana, gdy spełniony jest warunek w 5.
7. Usuń przypisane zadanie $\mathrm{J}(\eta + 1) \coloneqq \mathrm{J}(\eta)\backslash\{t\}$, $\eta \coloneqq \eta + 1$.
8. Zwiększ liczbę zadań na maszynie $n_{s'}(\eta) \coloneqq n_{s'}(\eta - 1) + 1$ i zapamiętaj $C_{s',n_{s'}}(\tilde{x}(\eta), \tilde{u}^{-}; p)$.
9. **end while (3)**
10. $\tilde{x}_{\mathrm{DR}} \coloneqq \tilde{x}(\eta)$

Początkowe przypisania oraz oszacowanie wartości kryterium dla problemu $Rm|r_j|C_{\max}$ wymagają czasu $\mathcal{O}(n^2m)$ (Linie 1-2). Zachłanny krok w Linii 4 można wykonać w czasie $\mathcal{O}(nm)$, ponieważ zapamiętujemy przy scenariuszu $\tilde{u}^{-}$, w którym przyjęto najwcześniejsze możliwe terminy gotowości wszystkich zadań, długość uszeregowania i liczbę zadań na każdej maszynie. Wybór rozwiązania w Linii 5 wymaga $\mathcal{O}(|\Lambda| = n)$, ponieważ ewaluacja funkcji, dla każdej krotki, jest stała czasowo $\mathcal{O}(1)$. Operacje przypisania, modyfikacji zbioru oraz indeksu iteracji są stałe czasowo, więc złożoność czasowa algorytmu wynosi $\mathcal{O}(n^2m)$. Na złożoność pamięciową równą $\mathcal{O}(n^2m + nm + m + n)$ składają się: rozmiar reprezentacji rozwiązania, rozmiar wektora wartości dolnych ograniczeń długości uszeregowania i zbiór nieprzypisanych zadań.

3.2.3. Algorytm konstrukcyjny G_Regret

Algorytm konstrukcyjny **G_Regret** (**GR**) wykorzystuje zachłanną strategię podejmowania decyzji w oparciu o ewaluację oszacowanej wartości kryterium maksymalnego żalu. Procedura szeregowania sprowadza się do rozwiązania n podproblemów

$$\Lambda = \arg \min_{\substack{i=1,2,\dots,m \\ k=1,2,\dots,n \\ j \in \mathrm{J}(\eta)}} \left\{ \max\left\{ C_{i,k}\left(\tilde{x}(\eta) = \left[\tilde{x}_{i,k,j}(\eta) = 1\right], \tilde{u}^{i,j}; p\right) - LB\left(\tilde{u}^{i,j}; p\right) \right\} \right\}, \tag{3.14}$$

gdzie krotka $(s, d, t) \in \Lambda$ stanowi podstawę do podjęcia decyzji $\tilde{x}_{s,d,t}(\eta) = 1$. Taka strategia umożliwia szerszą eksplorację przestrzeni możliwych rozwiązań, w porównaniu do (3.12), w krótkim horyzoncie czasu planowania. Zastosowanie operatora „max" we wzorze (3.14) umożliwia realizację przeglądu odchyleń długości uszeregowania od oszacowanej wartości optymalnej dla $|m\mathrm{J}(\eta)|$ scenariuszy skrajnych w podproblemie o indeksie η. Jeżeli w (3.14) istnieje wiele równoważnych rozwiązań, to o przypisaniu decyduje najmniejsza wartość wskaźnika

$$\tilde{b}_t\left(\tilde{x}(\eta), \tilde{\mathrm{U}}\right) = |\mathrm{J}(\eta)|^{-1} \sum_{i \in \mathrm{J} \setminus \mathrm{J}(\eta)} b_t\left(\tilde{x}(\eta), \tilde{u}^{i,t}\right), \tag{3.15}$$

wyrażającego średni czas oczekiwania na gotowość zadania t przy uwzględnieniu scenariuszy skrajnych zadań w harmonogramie $\tilde{x}(\eta)$. Złożoność czasowa **GR** wynosi $\mathcal{O}(n^4 m^2)$, ponieważ dla każdego z n nieuszeregowanych zadań weryfikowana jest pozycja w sekwencjach ustanowionych na m maszynach dowolnych poprzez obliczanie wartości (3.14), którą można uzyskać w czasie $\mathcal{O}(n^2 m)$. Złożoność pamięciowa algorytmu **GR** jest równa złożoności pamięciowej algorytmu **DR**. Harmonogram wyznaczony przez algorytm **GR** jest reprezentowany przez macierz $\tilde{x}_{\mathrm{GR}}$.

3.2.4. Algorytm oparty na metaheurystyce Tabu Search

Cechą charakterystyczną mataheursytyki Tabu Search jest możliwość wykluczania pewnego podzbioru rozwiązań lub pojedynczych decyzji w procesie optymalizacji. Lista tabu w opracowanym algorytmie **T_Search** (**TS)** zawiera zadania, których pozycja w uszeregowaniu została zmieniona. Wybór tego zadania zależy od jakości uszeregowania wygenerowanego jako rozwiązanie sąsiednie $\tilde{x}_{\mathrm{N}}(\varphi)$. W procesie generowania rozwiązania sąsiedniego rozwiązywane są dwa niezależne problemy, aby wskazać zadanie, którego pozycja może zostać zmodyfikowana. Pierwszy problem sprowadza się do wskazania zadania

$$j' = \arg \max_{j \in \mathrm{J}} \left\{ C_{\max}\left(\tilde{x}(\varphi) = \left[\tilde{x}_{i,k,j}(\varphi) = 1\right], \tilde{u}^{i,j}; p\right) - LB(\tilde{u}^{i,j}; p) \right\}, \quad i = 1,2, \dots, m, \tag{3.16}$$

którego pesymistyczny termin gotowości w scenariuszu skrajnym powoduje największe odchylenie od oszacowanej wartości kryterium. Rozwiązaniem drugiego problemu

$$j'' = \arg\max_{j\in \mathrm{J}} \left\{ C_{i,k}\left(\tilde{x}(\varphi), \tilde{u}^{i,j}; p\right) - \min\left(r_{i,j}^{-} + p_{i,j}\right)\right\}, \quad i = 1,2,\dots,m, \tag{3.17}$$

jest zadanie, dla którego różnica pomiędzy terminem jego zakończenia i najwcześniejszym możliwym terminem jego zakończenia jest największa. Następnie generowane są dwa rozwiązania sąsiednie (dla zadań wskazanych w (3.16) i (3.17)), w których zadania j' i j'' są przypisywane w taki sposób, aby terminy ich zakończeń były najwcześniejsze. Liczba rozwiązań sąsiednich ρ jest parametrem algorytmu, który podlega strojeniu. Do listy tabu $\mathrm{TL}(\varphi)$ dodajemy zadanie j, którego zmiana pozycji zapewnia $\tilde{x}_{\mathrm{N}}(\varphi) = \arg \min_{\tilde{x}'_{\mathrm{N}}\in\Omega_{\mathrm{TS}}(x(\varphi))} \tilde{Z}(\tilde{x}'_{\mathrm{N}})$. Wybierane jest również losowo jedno zadanie do zmiany pozycji. Kryterium aspiracji jest weryfikowane, gdy realizacja zadania znajdującego się na liście tabu nie rozpoczyna się w momencie jego dostępności na maszynie przy scenariuszu $\tilde{u}^{i,j}$, na której jest ono uszeregowane. Z listy tabu może zostać usunięte zadanie j, które spełnia warunek $C_{i,k}\left(\tilde{x}(\varphi), \tilde{u}^{i,j}; p\right) - r_{i,j}^{+} \neq p_{i,j}$, $j \in \mathrm{TL}(\varphi)$, $\tilde{x}_{i,k,j}(\varphi) = 1$, oraz jego przemieszczenie w harmonogramie na pozycje gwarantujące jego najwcześniejsze rozpoczęcie poprawia jakość rozwiązania (jakość przypisania jest weryfikowana na każdej maszynie). Liczba iteracji utrzymywania decyzji w $\mathrm{TL}(\varphi)$ jest równa τ i stanowi parametr algorytmu, który podlega strojeniu.

Algorytm **T_Search (TS)**

Require: J, M, U, p, ϱ, ρ, τ
Ensure: $\tilde{x}_{\mathrm{TS}}$
1. Przypisz $\varphi \coloneqq 1$ oraz wygeneruj losowy harmonogram $\tilde{x}(\varphi)$.
2. $\underset{\substack{i=1,2,\dots,m\\ j=1,2,\dots,n}}{\forall}$ oblicz dolne ograniczenie $LB\left(p, \tilde{u}^{i,j}\right)$.
3. **while** $\Gamma_{max} > t$ **or** $\varrho > 0$
4. $\underset{i=1,2,\dots,m}{\forall}$ wyznacz $j' \coloneqq \arg\max_{j\in \mathrm{J}} \left\{ C_{\max}\left(\tilde{x}(\varphi) \coloneqq \left[\tilde{x}_{i,k,j}(\varphi) \coloneqq 1\right], \tilde{u}^{i,j}; p\right) - LB\left(\tilde{u}^{i,j}; p\right)\right\}$.
5. $\underset{i=1,2,\dots,m}{\forall}$ wyznacz $j'' \coloneqq \arg\max_{j\in \mathrm{J}} \left\{ C_{i,k}\left(\tilde{x}(\varphi), \tilde{u}^{i,j}; p\right) - \min\left(r_{i,j}^{-} + p_{i,j}\right)\right\}$.
6. **if** $j' \notin \mathrm{TL}(\varphi)$ **or** $j'' \notin \mathrm{TL}(\varphi)$ **then**
7. Wybierz najlepsze rozwiązanie sąsiednie, $\tilde{x}_{\mathrm{N}}(\varphi) \coloneqq \arg \min_{\tilde{x}'_{\mathrm{N}}\in\Omega_{\mathrm{TS}}(x(\varphi))} \tilde{Z}(\tilde{x}'_{\mathrm{N}})$.
8. **if** $\tilde{Z}\left(\tilde{x}(\varphi)\right) > \tilde{Z}(\tilde{x}_{\mathrm{N}})$ **then** $\tilde{x}(\varphi) \coloneqq \tilde{x}_{\mathrm{N}}$ i zapamiętaj rezultat $\tilde{Z}\left(\tilde{x}(\varphi)\right)$ **end if**
9. Umieść zadanie, którego pozycja została zmieniona, na liście tabu $\mathrm{TL}(\varphi)$.
10. Zweryfikuj $C_{i,k}\left(\tilde{x}(\varphi), \tilde{u}^{i,j}; p\right) - r_{i,j}^{+} \neq p_{i,j}$, $j \in \mathrm{TL}(\varphi)$, $\tilde{x}_{i,k,j}(\varphi) = 1$ dla m maszyn.
11. Zaktualizuj wskaźnik czasu t, wskaźnik poprawy ϱ oraz indeks iteracji $\varphi \coloneqq \varphi + 1$.
12. Usuń elementy z $\mathrm{TL}(\varphi)$ znajdujące się na liście ρ iteracji.
13. **end if (6)**, **end while (3)** oraz przypisz $\tilde{x}_{\mathrm{TS}} \coloneqq \tilde{x}(\varphi)$

Złożoność czasowa generowania rozwiązania początkowego w Linii 1 zależy od wybranego generatora losowego GEN i wynosi $\mathcal{O}(\mathrm{GEN}n^2m)$. Złożoność czasowa obliczania dolnych ograniczeń w Linii 2 jest równa $\mathcal{O}(\mathrm{GEN}n^2m)$. Rozwiązania problemów w Linach 4-5 mają złożoność czasową równą $\mathcal{O}(n^3m)$, ponieważ dla każdego nieprzypisanego zadania realizowana jest ewaluacja kryterium. Weryfikacja warunku usunięcia zadań z listy tabu wymaga przeglądu całej listy o długości co najwyżej ρ. Zmiana pozycji zadania w harmonogramie, które zostało usunięte z listy tabu, wymusza ewaluację kryterium, co można zrealizować w czasie $\mathcal{O}(\rho n^2m)$ (dla ρ różnych zadań). Ostatecznie złożoność czasowa algorytmu wynosi $\mathcal{O}(n^3m)$. Na złożoność pamięciową równą $\mathcal{O}(n^2m + nm + 2m^2n^2)$ składają się: rozmiar reprezentacji rozwiązania, rozmiar wektora wartości dolnych ograniczeń długości uszeregowania oraz rozmiar zbioru rozwiązań sąsiednich.

3.2.5. Ocena eksperymentalna i statystyczna algorytmów

W niniejszym podrozdziale przeprowadzono następujące badania w celu eksperymentalnej i statystycznej oceny opracowanych algorytmów:

1. **Badanie 3.1**: Eksperymentalne porównanie jakości uszeregowań generowanych przez algorytmy **DR**, **GR** oraz **TS**.
2. **Badanie 3.2**: Statystyczne porównanie jakości uszeregowań generowanych przez algorytmy **DR** i **GR**.

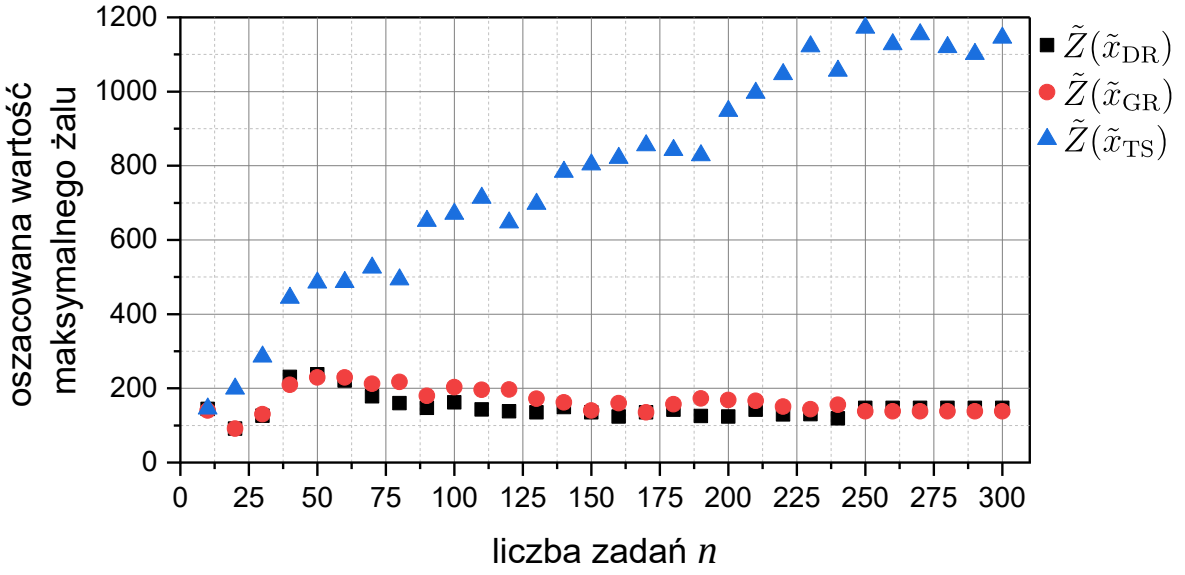
Metody generowania zbiorów danych do testowania jakości algorytmów rozwiązujących problemy optymalizacji odpornej zostały omówione dla znaczącej części problemów szeregowania zadań z przedziałowymi czasami ich przetwarzania (Allahverdi, Aydilek & Aydilek, 2014; Ćwik & Józefczyk, 2018; Sotskov, Egorova & Lai, 2009). Metodę generowania danych dla problemu szeregowania z przedziałowymi terminami gotowości na jednej maszynie opracowali Yue et al. (2018). Autorzy definiują wzorem $\left[avg_j - \mathrm{offset}_j, avg_j + \mathrm{offset}_j\right]$ przedział niepewności dla zadania j, gdzie avg_j jest uśrednionym terminem gotowości a offset_j jest losową wartością. Terminy gotowości wygenerowane na potrzeby badania 2.2 (podrozdział 2.8) wykorzystano do określenia dolnych wartości przedziałów niepewności $r_{i,j}^- = r_{i,j}, \quad i = 1,2,\dots,m, \quad j = 1,2,\dots,n.$ Górne granice przedziałów niepewności obliczono stosując wzór $u_{i,j} = \left[r_{i,j}^-, r_{i,j}^+\right] = \left[r_{i,j}^-, r_{i,j}^- + avg_j * \mathrm{offset}_{i,j}\right]$, $r_{i,j}^- < r_{i,j}^+$, gdzie avg_j jest uśrednionym czasem przetwarzania

zadania, a $\text{offset}_{i,j}$ jest liczbą całkowitą wylosowaną z zakresu $\{1,2,\ldots,10\}$ przy użyciu rozkładu jednostajnego.

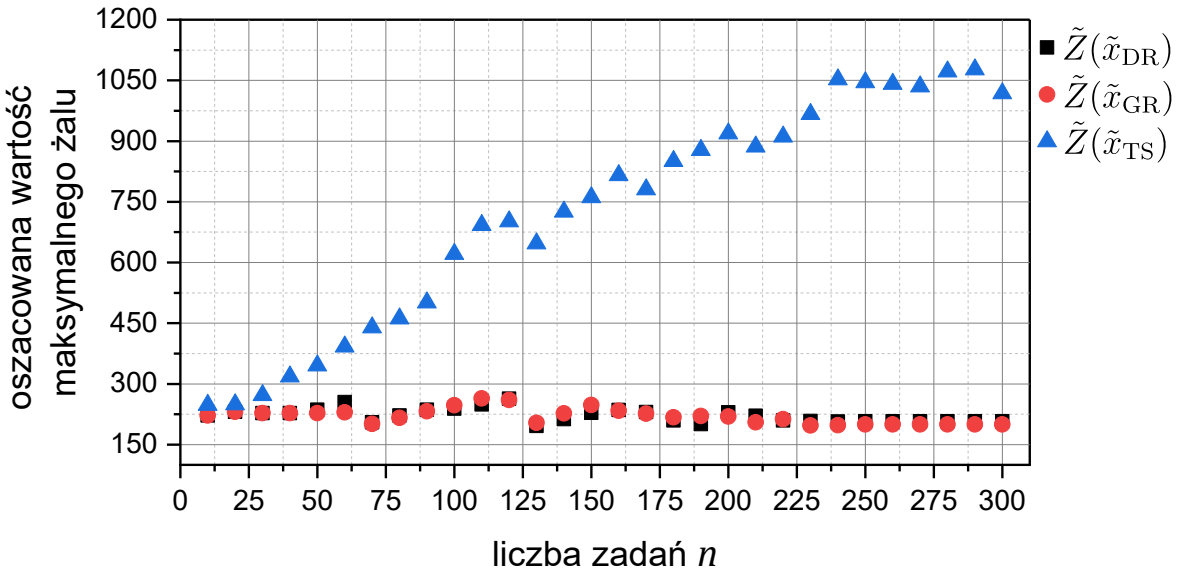
W algorytmie **TS** strojeniu poddawano dwa parametry: liczbę rozwiązań sąsiednich $\rho \in \{0.5m, \boldsymbol{m}, 1.5m, 2m\}$ oraz liczbę iteracji $\tau \in \left\{\left\lceil\frac{n}{m}\right\rceil, \left\lceil\frac{n}{2m}\right\rceil, \left\lceil\frac{n}{3m}\right\rceil, \left\lceil\frac{\boldsymbol{n}}{\boldsymbol{4m}}\right\rceil\right\}$ utrzymywania pojedynczej decyzji (wynikającej z przeglądu rozwiązań sąsiednich) na liście tabu. Wybrano parametry oznaczone pogrubioną czcionką. Metoda strojenia algorytmu **TS** jest tożsama z metodą zaproponowaną dla algorytmu **SA** w podrozdziale 2.8. Rezultatem strojenia były następujące wartości parametrów: $\rho = 1.5m$ oraz $\tau = \left\lceil\frac{n}{m}\right\rceil$. Algorytm **TS** uruchamiano 25 razy dla każdej instancji oraz wybierano najlepszy uzyskany rezultat. Ustalono, że liczba dopuszczalnych iteracji bez poprawy rozwiązania wynosi $\varrho = 20$ oraz ograniczono łączny czas obliczeń dla każdej instancji (25 uruchomień) ograniczono do $\Gamma_{max} = 10$ [min].

Badanie 3.1

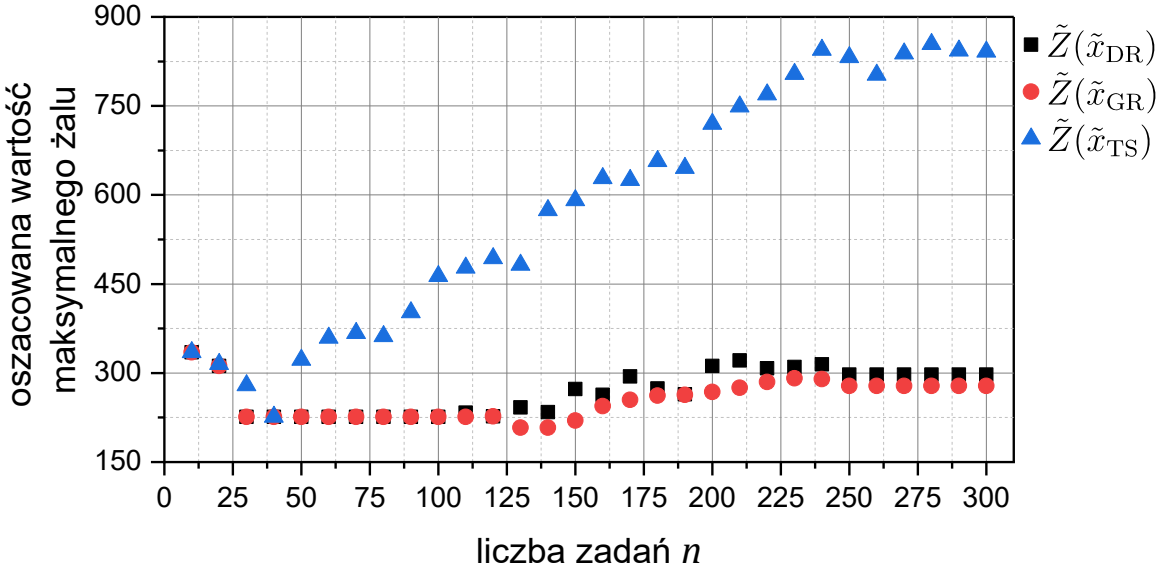
Pierwsze badanie ma na celu porównanie oszacowanych wartości maksymalnego żalu, które zostały obliczone na podstawie uszeregowań zwróconych przez algorytmy **DR**, **GR** oraz **TS**, oraz porównanie czasów zrealizowanych obliczeń. Wykorzystano czasy przetwarzania zadań wygenerowane na potrzeby badania 2.2. Wartości liczbowe rezultatów zaprezentowanych na rysunkach 3.2-3.8 umieszczono w tabelach B1-B5 (podrozdział B w dodatku).



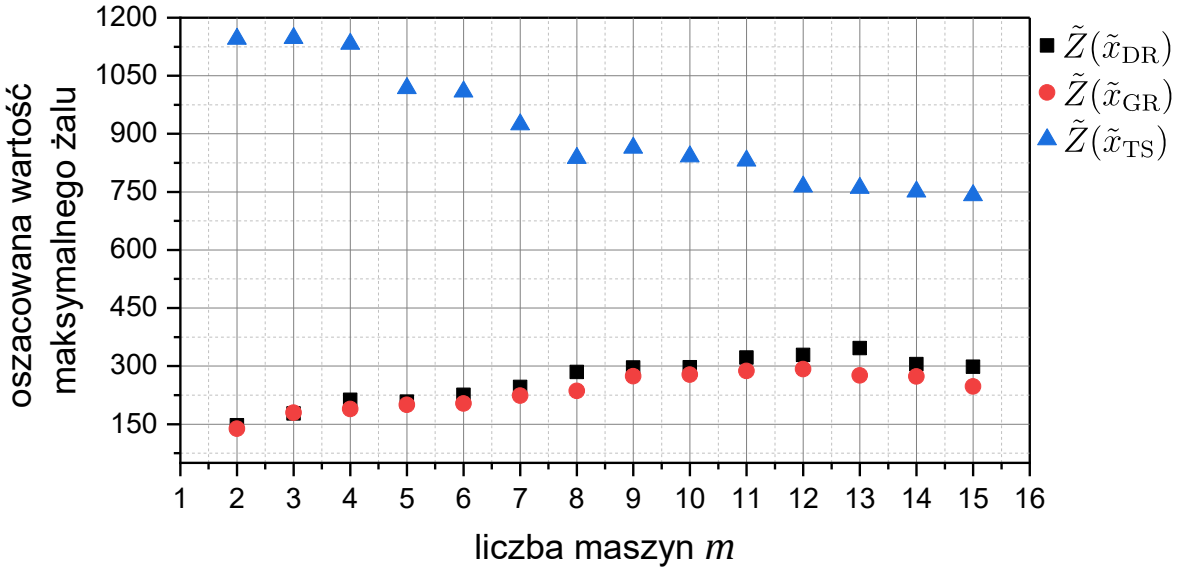
Rysunek 3.2. Wykres zależności oszacowanej wartości maksymalnego żalu od liczby zadań n dla $m = 2$



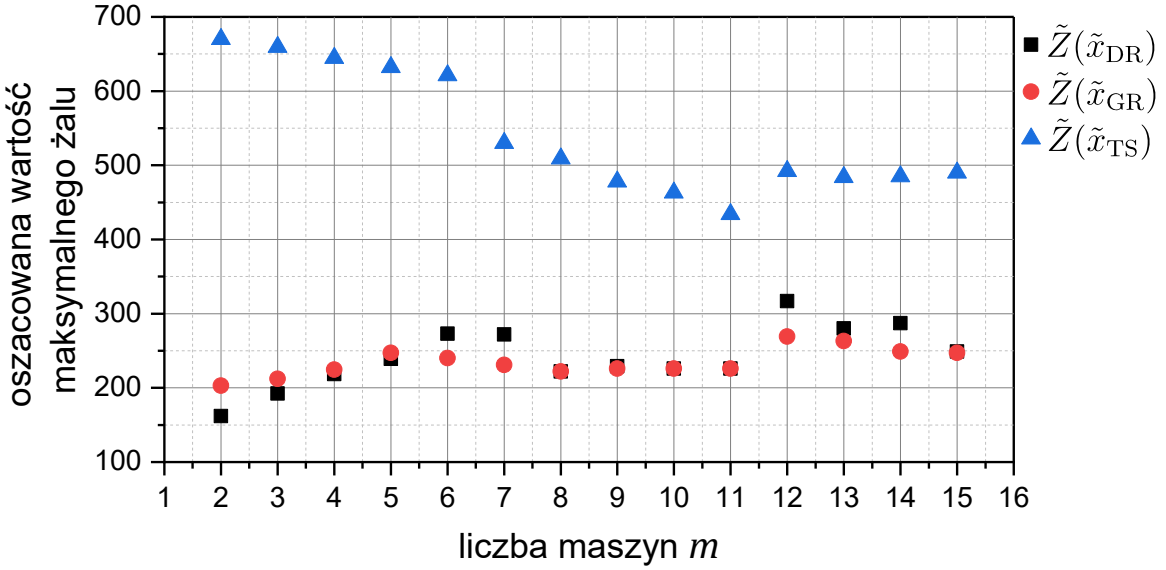
Rysunek 3.3. Wykres zależności oszacowanej wartości maksymalnego żalu od liczby zadań n dla $m = 5$



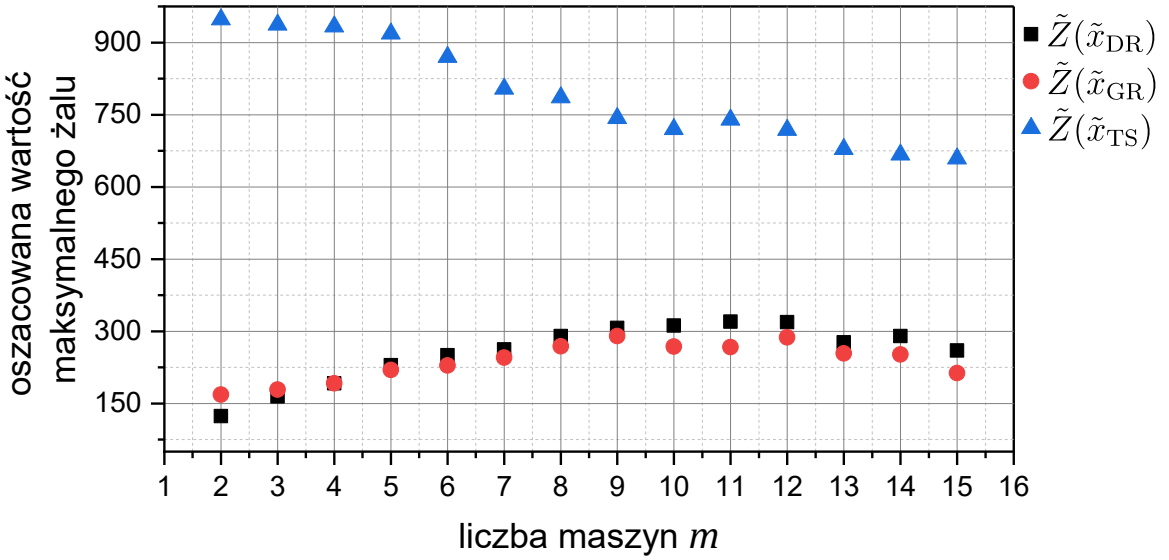
Rysunek 3.4. Wykres zależności oszacowanej wartości maksymalnego żalu od liczby zadań n dla $m = 10$



Rysunek 3.5. Wykres zależności oszacowanej wartości maksymalnego żalu od liczby maszyn m dla $n = 100$



Rysunek 3.6. Wykres zależności oszacowanej wartości maksymalnego żalu od liczby maszyn m dla $n = 200$

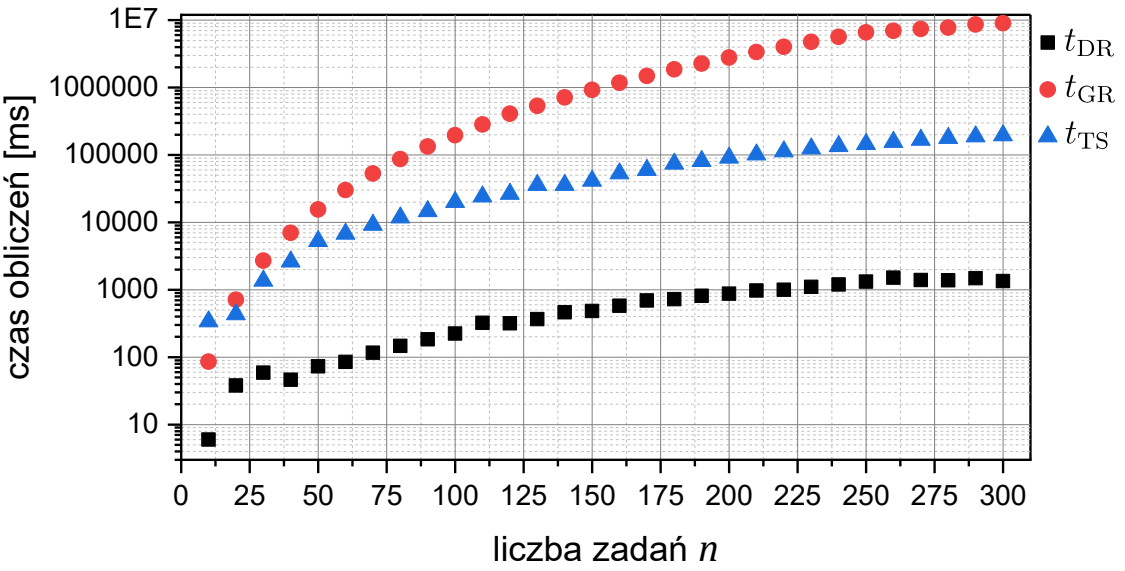


Rysunek 3.7. Wykres zależności oszacowanej wartości maksymalnego żalu od liczby maszyn m dla $n = 300$

Z rezultatów zaprezentowanych na rysunkach 3.2-3.7 wynika, że algorytmy **DR** i **GR** osiągały bardzo zbliżone rezultaty optymalizacji pomimo implementacji zupełnie innych strategii podejmowania decyzji. Przewaga jakości rozwiązań zwracanych przez algorytm **GR** wynikała z rozwiązywania wewnętrznego podproblemu maksymalizacji w (3.14), co odpowiada ewaluacji oszacowanej wartości kryterium żalu dla scenariuszy skrajnych odpowiadających zadaniom przypisanym w harmonogramie podczas η-tego etapu podejmowania decyzji. Warto podkreślić fakt, że minimalizacja odchylenia od długości uszeregowania, zaimplementowana w algorytmie **DR** dla jednego scenariusza skrajnego w każdym podproblemie $\eta = 1,2,\dots,n$ umożliwiała uzyskanie najlepszych rezultatów optymalizacji dla małych instancji problemu (np. $n = 80$ i $m = 2$) i dużych instancji problemu (np. $n = 220$ i $m = 5$). W obu przypadkach jedynie 1% do 2% wszystkich scenariuszy skrajnych stanowiły scenariusze istotne. Był to rezultat nakładania się większości przedziałów niepewności na każdej maszynie, czego konsekwencją było pełne pokrycie przedziałów czasami przetwarzania zadań oraz faktu, że suma czasów przetwarzania wszystkich zadań była większa niż łączna długość przedziałów niepewności.

Harmonogramy wyznaczone przez algorytm **TS** odbiegały znacząco jakością od uzyskanych przez algorytmy **DR** i **GR**, wyłączając małe instancje (np. $n = 20$ i $m = 10$). Są to przypadki, w których co najmniej połowa przedziałów niepewności na każdej maszynie była rozłączna, a rozwiązanie problemu wymagało równoważenia obciążenia maszyn. Tendencja do równoważenia obciążenia objawia się przy poprawie jakości rozwiązań wraz

ze wzrostem liczby maszyn (rysunki 3.5-3.7). Algorytm **TS** wykazywał również tendencje do utykania w lokalnych ekstremach, ponieważ podzbiór zadań (nieprzekraczający 15% wszystkich zadań) był wykorzystywany do generowania rozwiązań sąsiednich podczas wszystkich iteracji. Wprowadzenie losowych decyzji o wyborze zadania do przemieszczenia w harmonogramie nie poprawiało jakości rozwiązań. Niedostateczna eksploracja przestrzeni rozwiązań jest widoczna wyraźnie przy wzroście liczby zadań na rysunkach 3.2-3.4, gdy przedziały niepewności zostały pokryte czasami przetwarzania zadań we wczesnej fazie działania algorytmu (np. w wyniku wygenerowania rozwiązania losowego).



Rysunek 3.8. Wykres zależności czasu obliczeń od liczby zadań n dla $m = 2$. Wartości na osi y podano w skali logarytmicznej (log10)

Bardzo długi czas obliczeń t_{GR} realizowanych przez algorytm **GR** (rysunek 3.8) jest spowodowany koniecznością rozwiązywania wewnętrznego podproblemu maksymalizacji w (3.14). Efektywność czasowa algorytmu **DR** wynikała z przeglądu jednego scenariusza skrajnego dla każdej potencjalnej decyzji. Czas obliczeń t_{TS} zrealizowanych przez algorytm **TS** dla pojedynczej instancji problemu jest równy medianie czasów z dwudziestu pięciu uruchomień algorytmu. Na czas optymalizacji realizowanej przez algorytm **TS** rzutowało rozwiązywanie problemów (3.16)-(3.17).

Badanie 3.2

Dane do badań statystycznych wygenerowano w taki sam sposób jak w badaniu 2.4, a przedziały niepewności wyznaczono stosując formułę $u_{i,j} = [r_{i,j}^-, r_{i,j}^+] = [r_{i,j}^-, r_{i,j}^- + avg_j *$

$\text{offset}_{i,j}\rceil$. Parametry instancji oraz rezultaty optymalizacji zaprezentowano w tabeli B6 (podrozdział B w dodatku). Z badań wykluczono rezultaty zwracane przez algorytm **TS**, ponieważ dla żadnej wygenerowanej instancji algorytm nie uzyskał wyników lepszych od algorytmów **DR** i **GR**. Wygenerowano losowo 100 instancji problemu.

Podczas pierwszego etapu badań statystycznych, do weryfikacji hipotezy mówiącej o tym, że rozkład danych jest zbliżony do rozkładu normalnego wykorzystano test Shapiro-Wilka (Shapiro & Wilk, 1965). Wyniki testów przeprowadzonych dla poziomu istotności $\alpha = 0.05$:

- wartość statystyki $W = 0{,}9503$ oraz p-value $= 0{,}0009$ dla rezultatów algorytmu **DR**,
- wartość statystyki $W = 0{,}9489$ oraz p-value $= 0{,}0007$ dla rezultatów algorytmu **GR**,

pozwalają dla każdego zbioru odrzucić hipotezę zerowa zakładającą normalny rozkład wartości. Następnie przeprowadzono test Wilcoxona dla par (Wilcoxon, 1945), stosując konserwatywne podejście dla par o równej wartości (27% par), dla następujących hipotez statystycznych:

$$\begin{aligned} H_0&: \tilde{Z}(\tilde{x}_{\mathrm{DR}}) = \tilde{Z}(\tilde{x}_{\mathrm{GR}}), \\ H_1&: \tilde{Z}(\tilde{x}_{\mathrm{DR}}) > \tilde{Z}(\tilde{x}_{\mathrm{GR}}). \end{aligned} \tag{3.18}$$

Wynikiem testu, przeprowadzonego dla poziomu istotności $\alpha = 0.05$, jest $z_{val} = 1{,}9165$ oraz p-value $= 0{,}0276$, co pozwala odrzucić hipotezę zerową na rzecz hipotezy alternatywnej. Zastosowanie modyfikacji Pratta (Pratt, 1959) w teście Wilcoxona nie zmieniło rezultatu badania. Wynik badań statystycznych uzasadnia fakt, że algorytm **GR** wyznacza uszeregowania zapewniające statystycznie mniejszą wartość oszacowanego maksymalnego żalu niż algorytm **DR**.

W badaniach przeanalizowano również wpływ strategii zachłannej, w której wartość oszacowania dolnego ograniczenia $LB\left(\tilde{u}^{i,j}(\eta), p(\eta)\right)$ zależała od stanu rozwiązania, czyli tylko parametrów zadań przypisanych w harmonogramie. Apriorycznie wyliczaną wartość $LB\left(\tilde{u}^{i,j}; p\right)$ w algorytmie **GR** zastąpiono przez wartość $LB\left(\tilde{u}^{i,j}(\eta), p(\eta)\right)$. Wówczas rezultaty optymalizacji, dla wszystkich instancji rozważanych w badaniach 3.1 i 3.2, okazały się być co najwyżej tej samej jakości. Różnice na korzyść algorytmu **GR**, w porównaniu do opisanej wyżej zmodyfikowanej metody obliczania dolnego ograniczenia, przy danym scenariuszu skrajnym, sięgały 6,4%. Gorsze wyniki były rezultatem przedwczesnej zbieżności do lokalnego minimum. Na przykład, przypisanie dowolnego

zadania j podczas etapu $\eta = 1$ w oparciu o (3.14), gdzie $LB(\tilde{u}^{i,j}; p)$ zastąpiono przez oszacowanie $LB\left(\tilde{u}^{i,j}(\eta = 1), p(\eta = 1)\right)$, prowadzi do sytuacji, w której jest ono uszeregowane zawsze na maszynie oferującej jego najszybsze zakończenie, co jest konsekwencją zastosowania wzoru (2.19) do oszacowania $LB\left(\tilde{u}^{i,j}(\eta = 1), p(\eta = 1)\right)$ (różnica pomiędzy najwcześniejszym terminem zakończenia zadania j, przy scenariuszu skrajnym, a najwcześniejszym możliwym terminem jego zakończenia jest wówczas najmniejsza). Jeżeli m zadań może zakończyć się najwcześniej na m różnych maszynach, to decyzje podejmowane podczas m początkowych podproblemów odpowiadają wówczas minimalizacji długości uszeregowania, co rzutowało na końcowy rezultat optymalizacji.

3.3. Problem $Rm|r_j^- \le r_j \le r_j^+|Reg(C_{\max})$

Przedmiotem rozważań w tym podrozdziale jest analiza szczególnego przypadku problemu $Rm|r_{i,j}^- \le r_{i,j} \le r_{i,j}^+|Reg(C_{\max})$, w którym zakładamy niezależność terminów gotowości zadań od maszyn dowolnych. W nowym problemie, oznaczonym w zmodyfikowanej notacji Grahama jako $Rm|r_j^- \le r_j \le r_j^+|Reg(C_{\max})$, przedział niepewności terminu gotowości zadania j ma postać $\bar{u}_j = [r_j^-, r_j^+]$, $0 \le r_j^- < r_j^+$, i jego zakres jest cechą zadania. Opierając się na twierdzeniu 3.1, zbiór dopuszczalnych scenariuszy modelujemy jako $\bar{\mathrm{U}} = \bar{u}^1 \times \bar{u}^2 \times \ldots \times \bar{u}^j \times \ldots \times \bar{u}^n$, $\bar{\mathrm{U}} \subset \mathrm{U}$, realizacja pojedynczego scenariusza $\bar{u}_j$ jest opisana wektorem $\bar{u}^j = [r_1^-, \ldots, r_j^+, \ldots, r_n^-]^{\mathrm{T}}$, w którym $r_j = r_j^+$ oraz $\underset{t \in \mathrm{J}\setminus\{j\}}{\forall} r_t = r_t^-$. Bachler, Krumke & Le (2020) udowodnili, że przegląd scenariuszy skrajnych jest wystarczający do optymalnego rozwiązania problemu $1|r_j^- \le r_j \le r_j^+|Reg(C_{\max})$. Autorzy pracy nie wskazali scenariusza pośredniego jako maksymalizującego wartość funkcji celu. Dodatkowo deterministyczną wersję problemu $1|r_j^- \le r_j \le r_j^+|Reg(C_{\max})$ można rozwiązać optymalnie w czasie wielomianowym metodą ERD. Postać zmiennej decyzyjnej, ograniczeń oraz kryterium żalu w problemie $Rm|r_j^- \le r_j \le r_j^+|Reg(C_{\max})$ są tożsame z (2.4)-(2.7) i (3.1)-(3.3). Zmienną optymalizacyjną oraz wartość maksymalnego żalu dla rozważanego problemu oznaczono jako $\bar{x}$ oraz $\hat{Z}(\bar{x})$.

3.3.1. Podstawowe własności

Zależność terminów gotowości od maszyn w niedeterministycznym problemie $Rm|r_{i,j}^- \le r_{i,j} \le r_{i,j}^+|Reg(C_{\max})$ wymusza uwzględnianie czasów oczekiwania na gotowość zadań w wartości funkcji celu. Wartość kryterium maksymalnego żalu dla problemu

$Rm|r_j^- \le r_j \le r_j^+|Reg(C_{\max})$ szeregowania ma zupełnie inną interpretację, ponieważ nie uwzględnia czasów oczekiwania na gotowość zadań. Dowód tej własności przeprowadzono poniżej.

Własność 3.3. *Maksymalna wartość żalu nie może przekroczyć sumy czasów przetwarzania zadań uszeregowanych na najdłużej pracującej maszynie.*

Dowód. Bazując na własności 2.1, funkcję żalu można zapisać w następujący sposób:

$$\begin{aligned}\hat{Z}(\bar{x}) &= \max_{\bar{u}\in\bar{\mathrm{U}}} Q(\bar{x},\bar{u};p) = \max_{\bar{u}\in\bar{\mathrm{U}}}\{C_{\max}(\bar{x},\bar{u};p) - C_{\max}(\bar{x}_{\bar{u}}^*;p)\}\\ &= \max_{\bar{u}\in\bar{\mathrm{U}}}\left\{\hat{r}_j^{(\bar{u})} + \sum_{e\in \mathrm{A}_j^{(\bar{u})}(\bar{x})} p_e - \left(\hat{r}_j^{(\bar{u})} + \sum_{v\in \mathrm{A}_{j^*}^{(\bar{u})}(\bar{x}_{\bar{u}}^*)} p_{i_m,v} + T_{j^*}^{(\bar{u})}\right)\right\}\\ &= \max_{\bar{u}\in\bar{\mathrm{U}}}\left\{\left(\hat{r}_j^{(\bar{u})} - \hat{r}_j^{(\bar{u})}\right) + \left(\sum_{e\in \mathrm{A}_j^{(u)}(\bar{x})} p_{i_m,e} - \sum_{v\in \mathrm{A}_{j^*}^{(\bar{u})}(\bar{x}_{\bar{u}}^*)} p_{i_m,v} + T_{j^*}^{(\bar{u})}\right)\right\} \qquad (3.19)\\ &= \max_{\bar{u}\in\bar{\mathrm{U}}}\left\{\sum_{e\in \mathrm{A}_j^{(\bar{u})}(\bar{x})} p_{i_m,e} - \left(\sum_{v\in \mathrm{A}_{j^*}^{(\bar{u})}(\bar{x}_{\bar{u}}^*)} p_{i_m,v} + T_{j^*}^{(\bar{u})}\right)\right\} \ge 0.\end{aligned}$$

Bieżący indeks j, przy decyzji $\bar{x}_{i_m,j,k} = 1$, w (3.19) wskazuje najpóźniej rozpoczynające się zadanie, według sekwencji ustalonej na najdłużej pracującej maszynie $i_m = \arg\max_{i=1,2,\dots,m} C_{i,n_i}(\bar{x},\bar{u};p)$, które rozpoczyna się w terminie swojej gotowości $\hat{r}_j^{(\bar{u})}$ przy scenariuszu $\bar{u}$. Termin gotowości $\hat{r}_j^{(\bar{u})} \in \left[r_j^-, r_j^+\right]$ jest odpowiednikiem terminu podanego w (2.9), ale jego wartość zależy od zrealizowanego scenariusza $\bar{u}$. Natomiast zbiór $\mathrm{A}_j^{(\bar{u})}(\bar{x})$ jest odpowiednikiem zbioru zdefiniowanego w (2.10). Termin gotowości $\hat{r}_{j^*}^{(\bar{u})}$ nie jest znany, ponieważ uzyskanie optymalnego uszeregowania $\bar{x}_{\bar{u}}^*$ wymaga rozwiązania problemu, który jest co najmniej NP-trudny. Dlatego wprowadzono parametr $T_{j^*}^{(\bar{u})} \le 0$, którego celem jest wymuszenie zmniejszenia wartości $C_{\max}(\bar{x}_{\bar{u}}^*;p)$ względem $C_{\max}(\bar{x},\bar{u};p)$. Termin gotowości $\hat{r}_j^{(\bar{u})}$ może zostać uwzględniony do wyliczenia optymalnej wartości kryterium $C_{\max}(\bar{x}_{\bar{u}}^*;p)$, ponieważ jest on cechą zadania, przez co ogranicza optymalną długość uszeregowania $C_{\max}(\bar{x}_{\bar{u}}^*;p) > \hat{r}_j^{(\bar{u})}$. Aby dowieść sformułowanej własności wystarczy zauważyć, że jeżeli $\max_{\bar{u}\in\bar{\mathrm{U}}}\left\{\sum_{e\in \mathrm{A}_j^{(\bar{u})}(\bar{x})} p_{i_m,e} - \left(\sum_{v\in \mathrm{A}_{j^*}^{(\bar{u})}(\bar{x}_{\bar{u}}^*)} p_{i_m,v} + T_{j^*}^{(\bar{u})}\right)\right\} < 0$, to wartość

maksymalnego żalu mogłaby być ujemna, co prowadzi do sprzeczności, ponieważ $C_{\max}(\bar{x},\bar{u};p) \geq C_{\max}(\bar{x}^*_{\bar{u}};p)$ dla dowolnego scenariusza. Q.E.D.

Opierając się na własności 3.3 można wskazać dokładny zakres dopuszczalnych wartości kryterium $\hat{Z}(\bar{x}^*)$, wykorzystując jedynie czasy przetwarzania zadań:

$$0 \leq \hat{Z}(\bar{x}^*) \leq \max_{j=1,2,\ldots,n}\left\{\max_{i=1,2,\ldots,m} p_{i,j} - \min_{i=1,2,\ldots,m} p_{i,j} + \sum_{t\in \mathrm{J}\setminus\{j\}} \max_{i=1,2,\ldots,m} p_{i,t}\right\}. \tag{3.20}$$

Górne ograniczenie wartości $\hat{Z}(\bar{x}^*)$ jest efektem skrajnego przypadku, w którym zadanie j jest przetwarzane najdłużej zgodnie z $\bar{x}$ i zostaje ono przestawione według harmonogramu optymalnego $\bar{x}^*_{\hat{u}}$, $\hat{u} = \bar{u}^j$, na maszynę oferującą jego najszybszą realizację oraz wszystkie pozostałe zadania $\mathrm{J}\setminus\{j\}$, rozpoczynające się po zakończeniu zadania j zgodnie z $\bar{x}$, zostają uszeregowane według $\bar{x}^*_{\hat{u}}$ w taki sposób, że termin zakończenia każdego zadania w $\mathrm{J}\setminus\{j\}$ nie przekracza terminu gotowości $\hat{r}_j^{(\hat{u})}$. Wszystkie zadania uszeregowane po zadaniu j na maszynie i_m, zgodnie z ustalonym uszeregowaniem $\bar{x}$, mogą również zakończyć się przed $\hat{r}_j^{(\hat{u})}$ na tej samej maszynie, zgodnie z uszeregowaniem optymalnym $\bar{x}^*_{\hat{u}}$.

Ponadto można zauważyć, że w przypadku (3.20) scenariusz pośredni również może maksymalizować wartość kryterium żalu. Niech zadania w $\mathrm{J}\setminus\{j\}$ zostaną uszeregowane w taki sposób, że zgodnie z $\bar{x}^*_{\hat{u}}$ zakończą się przed terminem $\hat{r}_j^{(\hat{u})}$, $\hat{u} = \bar{u}^j$. Wówczas występuje okno czasowe $\hat{r}_j^{(\hat{u})} - C^{(\hat{u})}_{\mathrm{J}\setminus\{j\}} \neq 0$, $\bar{x}_{i_m,j,k} = 1$, gdzie $C^{(\hat{u})}_{\mathrm{J}\setminus\{j\}}$ jest optymalną długością uszeregowania wyznaczoną dla zadań $\mathrm{J}\setminus\{j\}$. W rezultacie termin $r_j^+ = \hat{r}_j^{(\hat{u})}$, będący składową scenariusza skrajnego $\hat{u} = \bar{u}^j$, można zastąpić dowolną wartością z przedziału $r_j \in \left[\hat{r}_j^{(\hat{u})} - C^{(\hat{u})}_{\mathrm{J}\setminus\{j\}}, \hat{r}_j^{(\hat{u})}\right]$, ponieważ w takim przypadku

$$\hat{Z}(\bar{x}) = \left(r_j + p_{i_m,j} + \sum_{t\in \mathrm{J}\setminus\{j\}} p_{i_m,t}\right) - \left(r_j + \min_{i=1,2,\ldots,m} p_{i,j}\right). \tag{3.21}$$

Własność 3.3 uzasadnia fakt, że szacowanie wartości kryterium maksymalnego żalu dla problemu, w którym terminy gotowości zależą od zadań, nie jest obarczone błędem wynikającym z niedoszacowania czasów oczekiwania na gotowość zadań. Podczas ewaluacji wartości kryterium dla problemu $Rm|r_j^- \leq r_j \leq r_j^+|Reg(C_{\max})$ posługujemy się następującym oszacowaniem:

$$\bar{Z}(\bar{x}) = \max_{\bar{u}\in\widehat{\mathrm{U}}}\{C_{\max}(\bar{x},\bar{u};p) - LB(\bar{u};p)\}, \tag{3.22}$$

gdzie $\widehat{\mathrm{U}}$ jest zbiorem scenariuszy istotnych, $\widehat{\mathrm{U}} \subset \widetilde{\mathrm{U}}$.

3.3.2. Algorytm konstrukcyjny S_Regret

Algorytm konstrukcyjny **S_Regret (SR)** wyznacza rozwiązanie niedeterministycznego problemu szeregowania $Rm|r_j^- \le r_j \le r_j^+|Reg(C_{\max})$ wykorzystując strategię zachłanną dla zastępczego deterministycznego problemu z kryterium długości uszeregowania $Rm|r_{i,j}|C_{\max}$. Działanie algorytmu opiera się na obliczaniu liczby potencjalnych zmian w harmonogramie optymalnym względem harmonogramu ustalonego przez decydenta. Zaproponowane podejście wynika z definicji kryterium maksymalnego żalu dla rozważanego problemu szeregowania, według której niezerową wartość kryterium uzyskujemy jedynie, gdy podzbiór zadań może zakończyć się wcześniej w $\bar{x}^*_{\widehat{u}}$ niż wynika to z decyzji macierzy decyzji $\bar{x}$. Bazując na tej obserwacji, wprowadzono zbiór

$$\mathrm{D}_j^+\left(\overline{\mathrm{U}},\mathrm{J}(\eta)\right) = \left\{t \in \mathrm{J}(\eta)\backslash\{j\} | r_t^- < r_j^+\right\} \tag{3.23}$$

nieprzypisanych zadań $\mathrm{J}(\eta)\backslash\{j\}$ w η-tym etapie szeregowania $(\eta = 1,2,\dots,n)$, które w harmonogramie $\bar{x}$ mogą być przypisane po zadaniu j, natomiast w harmonogramie optymalnym $\bar{x}^*_{\bar{u}^j}$ mogą się wykonać przed j, ponieważ cechują się wcześniejszymi terminami gotowości przy scenariuszu $\bar{u}^j$.

Algorytm **SR** w η-tym podproblemie realizuje dwie procedury:

1. Wskazanie zadania $\overline{\Lambda} = \arg\min_{j\in\mathrm{J}(\eta)}\left|\mathrm{D}_j^+\left(\overline{\mathrm{U}},\mathrm{J}(\eta)\right)\right|$, przed którym może wykonać się najmniejsza liczba nieprzypisanych zadań w harmonogramie optymalnym. Jeżeli $|\overline{\Lambda}| > 1$ to wartością rozstrzygającą jest najmniejsza średnia liczba jednostek czasu przetwarzania zadań $\arg\min_{j\in\overline{\Lambda}}\sum_{t\in\mathrm{J}(\eta)\backslash\{j\}}\sum_{i\in\mathrm{M}} m^{-1}p_{i,t}$, które mogą wykonać się przed j.
2. Uszeregowanie zadania $t \in \overline{\Lambda}$ w harmonogramie według strategii zachłannej

$$(s,d) = \arg\min_{\substack{i=1,2,\dots,m\\k=1,2,\dots,n}}\left\{C_{i,k}\left(\bar{x}(\eta) = \left[\bar{x}_{i,k,t}(\eta) = 1\right],\bar{u}^t;p\right)\right\}. \tag{3.24}$$

Ostatecznie krotka (s,d,t) stanowi podstawę do podjęcia decyzji $\bar{x}_{s,d,t}(\eta) = 1$. We wzorze (3.24) scenariusz skrajny również zastępujemy scenariuszem $\bar{u}^- = \left[r_1^-,\dots,r_j^-,\dots,r_n^-\right]^{\mathrm{T}}$, (algorytm **SR** jest zawsze uruchamiany dwukrotnie, dla scenariuszy skrajnych $\bar{u}^t$,

$t = 1,2,\dots,n$, oraz dla scenariusza optymistycznego $\bar{u}^-$, i wybierane jest najlepsze uszeregowanie). W podejściu opartym na scenariuszach skrajnych rozważamy pesymistyczne przypadki gotowości zadań dla deterministycznego kryterium (3.24). Dla scenariusza $\bar{u}^-$ przyjmujemy scenariusz optymistyczny dla wszystkich zadań. Wykorzystanie zbioru (3.23), zadań w $\bar{\Lambda}$ oraz strategii zachłannej, przy scenariuszu skrajnym $\bar{u}^t$ lub $\bar{u}^-$, umożliwiają podejmowanie decyzji bez wiedzy o optymalnej długości uszeregowania $C_{\max}(\bar{x}^*_{\bar{u}^t}; p)$. Takie podejście wyklucza konieczność obliczania wartości dolnego ograniczenia $LB(\bar{u}^t; p)$. Brak konieczności szacowania wartości $C_{\max}(\bar{x}^*_{\bar{u}^t}; p)$ pozwala w praktyce stosować arbitralnie przyjęte scenariusze, ponieważ rozważane jest jedynie kryterium deterministyczne.

Algorytm **S_Regret (SR)**

Require: J, M, $\bar{\mathrm{U}}, p$
Ensure: $\bar{x}_{\mathrm{SR}}$

1. Przypisz $\eta \coloneqq 1$, $\mathrm{J}(\eta) \coloneqq \mathrm{J}$, $\bar{x}(\eta) \coloneqq \left[\bar{x}_{i,k,j}(\eta) \coloneqq 0\right]_{\substack{i=1,2,\dots,m \\ k,j=1,2,\dots,n}}$.
2. $\forall_{j\in \mathrm{J}(\eta)}$ wygeneruj zbiór $\mathrm{D}_j^+\left(\bar{\mathrm{U}}, \mathrm{J}(\eta)\right)$.
3. **while** $\mathrm{J}(\eta) \neq \emptyset$
4. Wyznacz $\bar{\Lambda} \coloneqq \arg\min_{j\in \mathrm{J}(\eta)} \left|\mathrm{D}_j^+\left(\bar{\mathrm{U}}, \mathrm{J}(\eta)\right)\right|$.
5. **if** $|\bar{\Lambda}| > 1$ **then** $t \coloneqq \arg\min_{j\in\Lambda} \sum_{t'\in \mathrm{J}(\eta)\setminus\{j\}} \sum_{i\in \mathrm{M}} m^{-1} p_{i,t'}$ **end if**
6. Wyznacz $(s,d) \coloneqq \arg\min_{\substack{i=1,2,\dots,m \\ k=1,2,\dots,n}} \left\{C_{i,k}\left(\bar{x}(\eta) \coloneqq \left[\bar{x}_{i,k,t}(\eta) \coloneqq 1\right], \bar{u}^t; p\right)\right\}$.
7. Przypisz $\bar{x}_{s,d,t}(\eta) = 1$, $\mathrm{J}(\eta+1) \coloneqq \mathrm{J}(\eta)\setminus\{t\}$, $\mathrm{D}_j^+\left(\bar{\mathrm{U}}, \mathrm{J}(\eta)\right)\setminus\{t\}$, $j = 1,2,\dots,n$ i $\eta \coloneqq \eta + 1$.
8. Zwiększ liczbę zadań na maszynie $n_s(\eta) \coloneqq n_s(\eta-1) + 1$ i zapamiętaj $C_{s,n_s}(\bar{x}(\eta), \bar{u}^-; p)$.
9. **end while (3)**
10. $\bar{x}_{\mathrm{SR}} \coloneqq \bar{x}(\eta)$

Przypisania oraz wygenerowanie rozwiązania początkowego w Linii 1 można zrealizować w czasie $O(n^2 m)$. Generowanie podzbiorów w Linii 2 można wykonać w czasie $O(n^2)$, ponieważ najpóźniejszy termin gotowości r_j^+ jest porównywany z najwcześniejszymi terminami gotowości pozostałych zadań r_t^-, $\mathrm{J}\setminus\{t\}$ (procedura porównania jest wykonywana $n-1$ razy dla każdego zadania). Przegląd zadań w Linii 4 wymaga $O(n)$. W przypadku istnienia wielu równoważnych zadań wybór najlepszego może być wykonany w czasie $O(n^2 m)$ (Linia 5). Rozwiązanie problemu przy dowolnym scenariuszu w Linii 6 wymaga jedynie przeglądu indeksów maszyn i nieuszeregowanych zadań w czasie $O(nm)$, ponieważ zapamiętujemy długość sekwencji na każdej maszynie. Operacje przypisania, modyfikacji zbiorów i indeksu iteracji są stałe czasowo (Linia 7). Ostatecznie złożoność obliczeniowa algorytmu **SR** jest równa $O(n^3 m)$. Złożoność pamięciowa $O(n^2 + n^2 m)$ jest

zdeterminowana rozmiarem zbiorów agregujących zadania, które mogą się wykonać wcześniej w odniesieniu do wskazanego zadania oraz rozmiarem reprezentacji rozwiązania.

3.3.3. Analiza prostych instancji

W niniejszym podrozdziale wskazano instancje problemu $Rm|r_j^- \leq r_j \leq r_j^+|Reg(C_{\max})$, które można optymalnie rozwiązać w czasie wielomianowym przez algorytm **SR**. Efektywność algorytmu **SR** wynika z wykorzystania zadania $\overline{\Lambda}$. Sformułowano dwa warunki, których spełnienie gwarantuje optymalne rozwiązanie problemu.

W pierwszym przypadku założono rozłączność przedziałów niepewności terminów gotowości

$$\underset{\{j,t\}\subseteq \mathrm{J}, j\neq t}{\forall} \bar{u}_j \cap \bar{u}_t = \emptyset. \tag{3.25}$$

Własność 3.4. *Rozwiązanie* $\bar{x}_{\mathrm{SR}}$ *jest optymalne, jeżeli warunek* (3.25) *jest spełniony.*

Dowód. Posortowane rosnąco, dowolnie wybrane z przedziałów, terminy gotowości zadań gwarantują kolejność $r_{j_1} < \cdots < r_{j_\eta} < \cdots < r_{j_n}$, która nie zależy od zrealizowanego scenariusza. W takim przypadku zachodzi równość

$$\left|\mathrm{D}_{j_1}^+\left(\overline{\mathrm{U}}, \mathrm{J}(1)\right)\right| = \cdots = \left|\mathrm{D}_{j_\eta}^+\left(\overline{\mathrm{U}}, \mathrm{J}(\eta)\right)\right| = \cdots = \left|\mathrm{D}_{j_n}^+\left(\overline{\mathrm{U}}, \mathrm{J}(n)\right)\right| = 0. \tag{3.26}$$

Zatem przypisanie zadań według strategii zachłannej w kolejności $j_1, \ldots, j_\eta, \ldots, j_n$, powoduje, że pozycja każdego zadania w harmonogramie $\bar{x}_{\mathrm{SR}}$ i harmonogramie optymalnym $\bar{x}^*_{\bar{u}^{j_\eta}}$, $\eta = 1,2,\ldots,n$, jest taka sama. Dlatego niezależnie od scenariusza uzyskujemy najmniejszą możliwą długość uszeregowania optymalnego $C_{\max}(\bar{x}_{\mathrm{SR}}; p) = C_{\max}\left(\bar{x}^*_{\bar{u}^{j_\eta}}; p\right)$, co skutkuje zerową wartością maksymalnego żalu $\hat{Z}(\bar{x}^* = \bar{x}_{\mathrm{SR}}) = 0$. Q.E.D.

Zauważmy dodatkowo, że algorytmy **DR** i **GR**, w odróżnieniu do algorytmu **SR**, nie zwróciłyby optymalnych uszeregowań dla każdej instancji, przy założeniu (3.25), ponieważ w problemie $Rm|r_{i,j}^- \leq r_{i,j} \leq r_{i,j}^+|Reg(C_{\max})$ kolejność przedziałów niepewności nie musi być taka sama na wszystkich maszynach. W efekcie może dochodzić do zmiany harmonogramu optymalnego względem ustalonego przez algorytmy **DR** i **GR**, co skutkuje niezerową wartością kryterium maksymalnego żalu.

Drugi warunek

$$\underset{j\in \mathrm{J}}{\exists} \max_{t\in \mathrm{J}\setminus\{j\}} r_t^+ + \sum_{t\in \mathrm{J}\setminus\{j\}} \max_{i=1,2,\ldots,m} p_{i,t} \leq r_j^- \tag{3.27}$$

dotyczy przypadku, w którym może istnieć wiele równoważnych optymalnych harmonogramów oraz decyzja o przypisaniu jednego zadania determinuje optymalność rozwiązania.

Własność 3.5. *Rozwiązanie* $\bar{x}_{\mathrm{SR}}$ *jest optymalne, jeżeli warunek* (3.27) *jest spełniony.*

Dowód. Nierówność (3.27) determinuje $C_{\max}(\bar{x},\bar{u};p) \le r_j^-$ niezależnie od scenariusza $\bar{u}$ lub harmonogramu $\bar{x}$ wyznaczonego dla zadań w zbiorze J\{j}. Optymalną decyzją jest umieszczenie zadania j na ostatniej pozycji na najwydajniejszej maszynie, ponieważ przy takim przypisaniu $\left|\mathrm{D}_j^+\left(\overline{\mathrm{U}},\mathrm{J}(n)\right)\right| = 0$. Algorytm **SR** przypisze zadanie j jako ostatnie, ponieważ $\left|\mathrm{D}_t^+\left(\overline{\mathrm{U}},\mathrm{J}(\eta)\right)\right| < \left|\mathrm{D}_j^+\left(\overline{\mathrm{U}},\mathrm{J}(n)\right)\right|$, $\eta = 1,2,\dots,n-1$, $t \in$ J\{j}. Wówczas optymalną (zerową) wartość maksymalnego żalu można obliczyć uwzględniając parametry zadania j

$$\hat{Z}(\bar{x}^* = \bar{x}_{\mathrm{SR}}) = r_j + p_{i,j} - \left(r_j + p_{s,j}\right) = p_{i,j} - p_{s,j}, \quad r_j \in \bar{u}_j, \quad \{i,s\} \subseteq \mathrm{M}, \tag{3.28}$$

gdzie zachłanna decyzja gwarantuje równość $i = s$. Q.E.D.

Algorytmy **DR** i **GR** również wyznaczają optymalne uszeregowanie, gdy spełniony jest warunek (3.27) (dowód jest analogiczny).

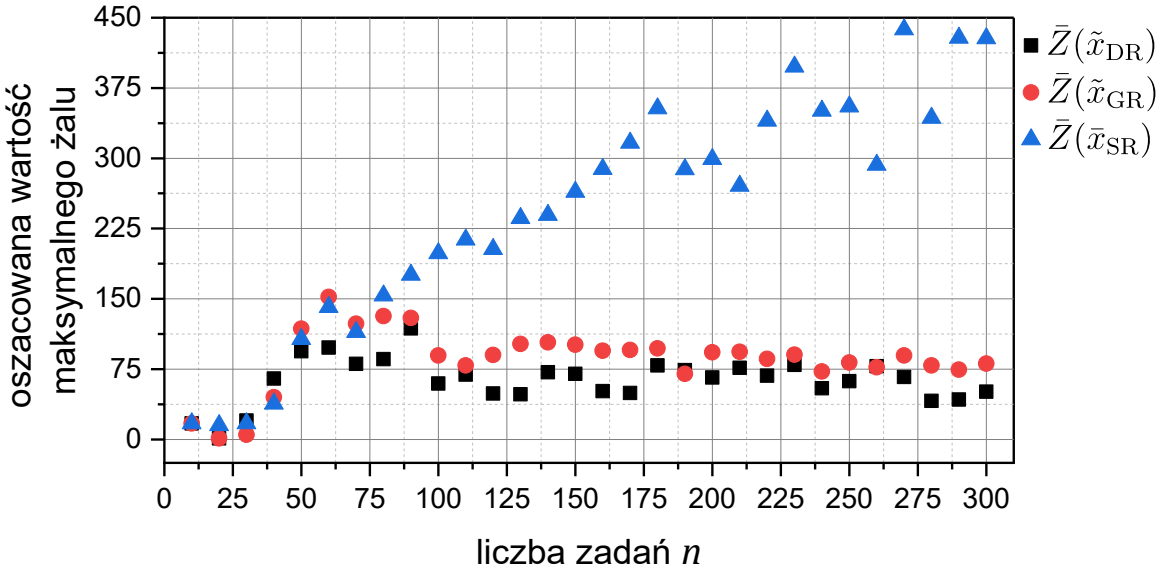
3.3.4. Ocena eksperymentalna i statystyczna algorytmów

W niniejszym podrozdziale przeprowadzono następujące badania w celu eksperymentalnej i statystycznej oceny opracowanych algorytmów:

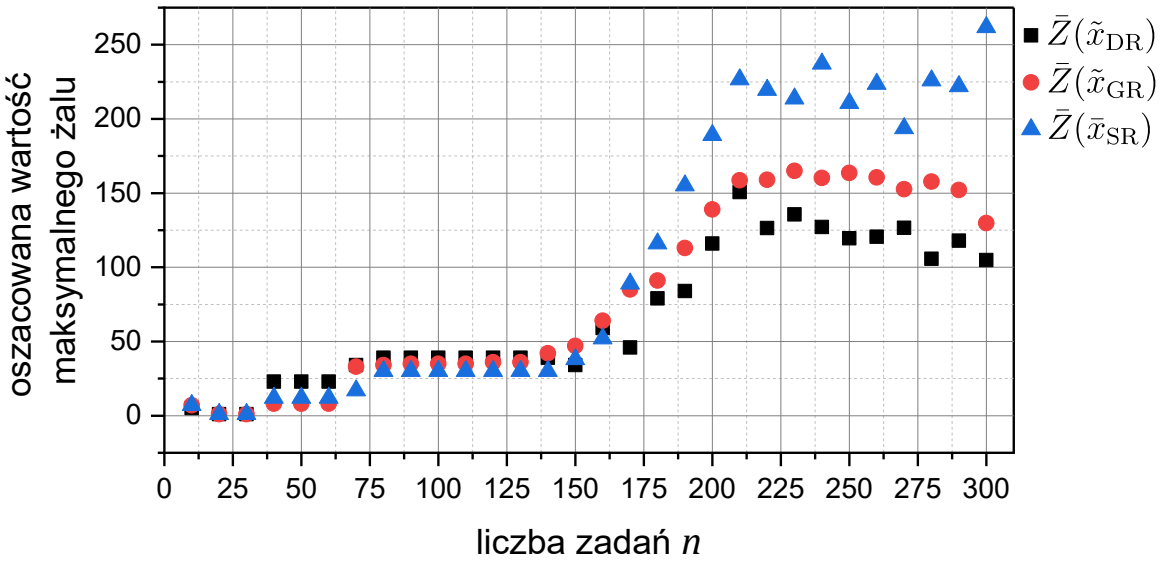
1. **Badanie 3.3**: Eksperymentalne porównanie jakości uszeregowań generowanych przez algorytmy **DR**, **GR** oraz **SR**.
2. **Badanie 3.4**: Statystyczne porównanie jakości uszeregowań generowanych przez algorytmy **DR**, **GR** i **SR**.

Badanie 3.3

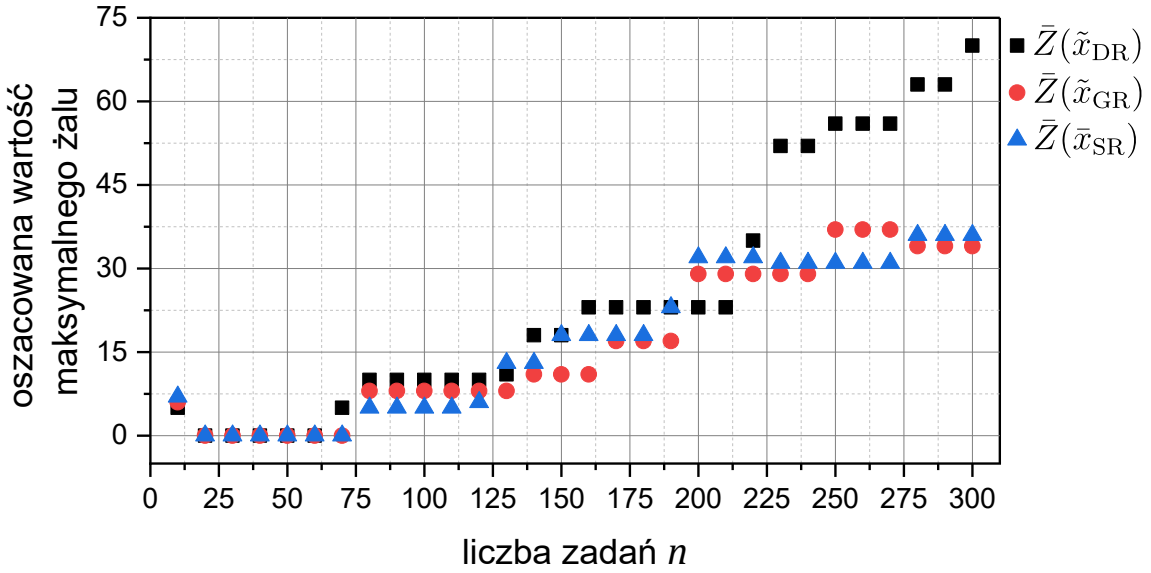
Pierwsze badanie ma na celu porównanie oszacowanych wartości maksymalnego żalu, które wyliczamy na podstawie uszeregowań zwróconych przez algorytmy **DR**, **GR** oraz **SR**, oraz porównanie czasów zrealizowanych obliczeń. Wartości liczbowe rezultatów zaprezentowanych na rysunkach 3.9-3.15 umieszczono w tabelach C1-C3 (podrozdział C w dodatku). Każdy przedział niepewności $\bar{u}_j$, $j = 1,2,\dots,n$, został wybrany losowo z przedziałów wygenerowanych w badaniu 3.1 oraz wykorzystano czasy przetwarzania wskazane w tym badaniu.



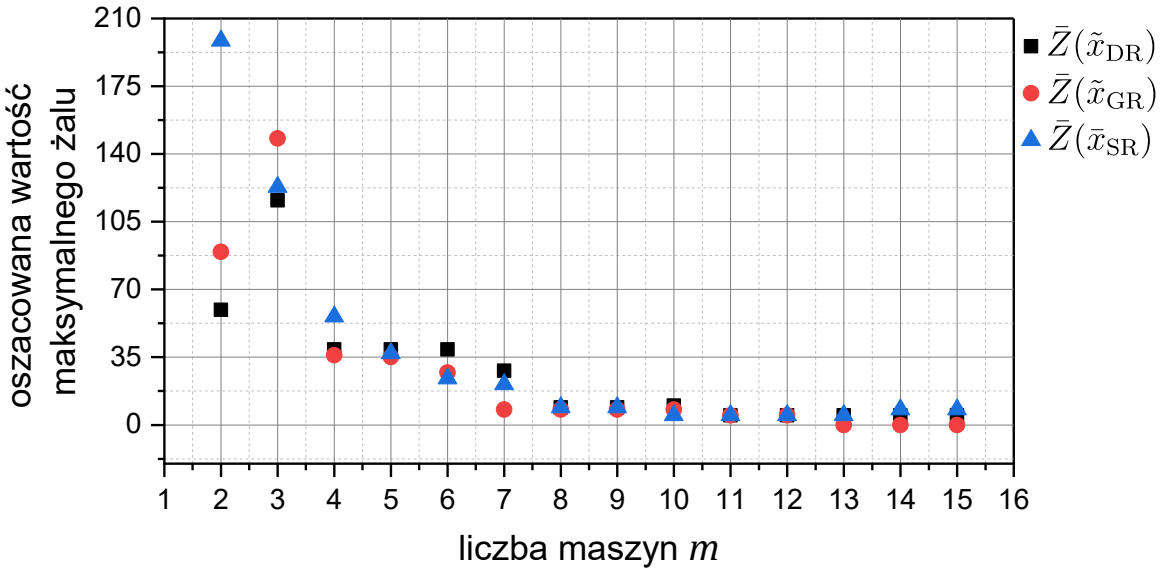
Rysunek 3.9. Wykres zależności oszacowanej wartości maksymalnego żalu od liczby zadań n dla $m = 2$



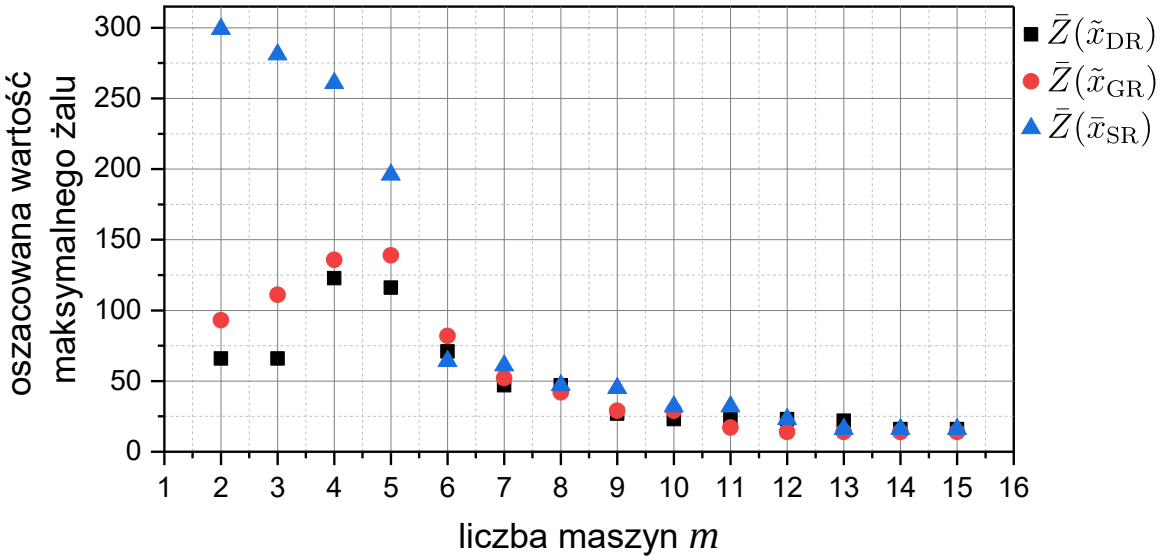
Rysunek 3.10. Wykres zależności oszacowanej wartości maksymalnego żalu od liczby zadań n dla $m = 5$



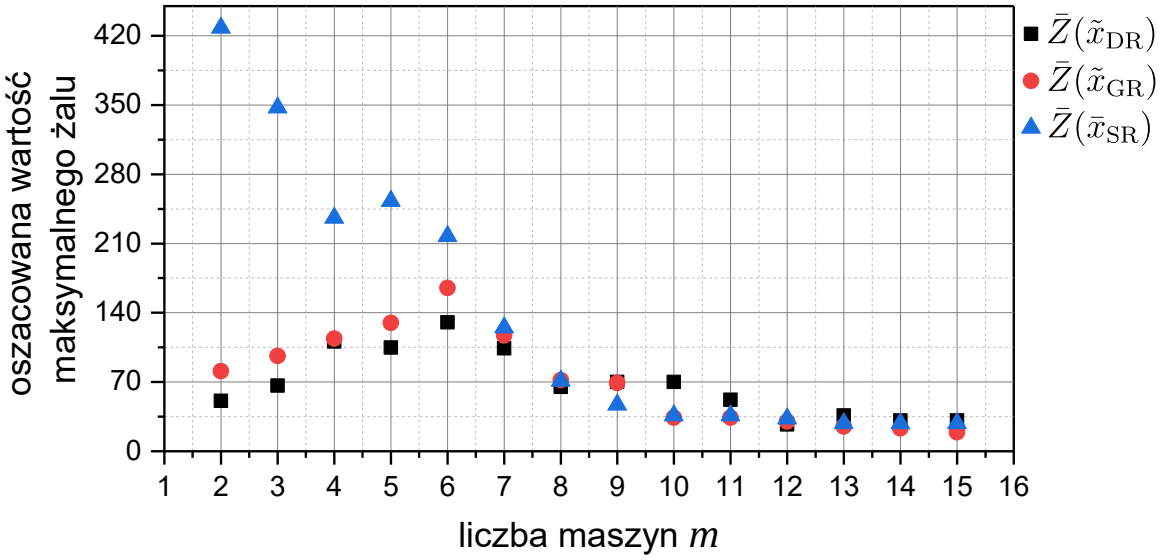
Rysunek 3.11. Wykres zależności oszacowanej wartości maksymalnego żalu od liczby zadań n dla $m = 10$



Rysunek 3.12. Wykres zależności oszacowanej wartości maksymalnego żalu od liczby maszyn m dla $n = 100$



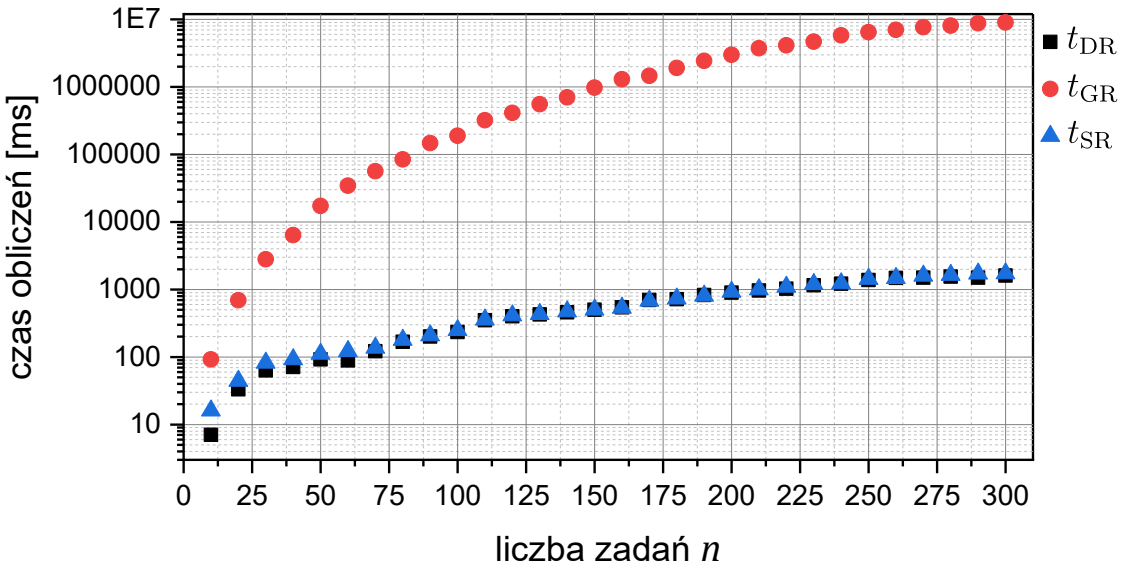
Rysunek 3.13. Wykres zależności oszacowanej wartości maksymalnego żalu od liczby maszyn m dla $n = 200$



Rysunek 3.14. Wykres zależności oszacowanej wartości maksymalnego żalu od liczby maszyn m dla $n = 300$

Analizując wyniki zaprezentowane na rysunkach 3.9-3.11 można zaobserwować charakterystyczne stabilizowanie się wartości oszacowanego maksymalnego żalu niezależnie od zwiększanej liczby zadań. Jest to efekt pokrycia przedziałów niepewności czasami przetwarzania zadań, w efekcie czego, na wartość kryterium miał wpływ jedynie niewielki podzbiór scenariuszy istotnych (co najwyżej 3%), który nie zmieniał się wraz z dodawaniem kolejnych zadań do uszeregowania. Wyniki zaprezentowane na rysunkach 3.9 (rezultaty obliczeń algorytmu **SR** dla $n > 80$ i $m = 2$) lub 3.13-3.14 (rezultaty obliczeń algorytmu **SR**

dla $m \leq 5$ i $n = 200$ lub $n = 300$) uwidoczniły fakt, że decyzje podejmowane w oparciu o zadania wybierane z podzbioru podanego w (3.23), są bardzo nieefektywne, gdy terminy gotowości zadań zostaną pokryte w pełni czasami przetwarzania podzbioru zadań $\mathrm{J}' \neq \mathrm{J}$ a pozostałe zadania muszą zostać przypisane w harmonogramie bez uwzględniania czasu oczekiwania na ich gotowość. Trzy algorytmy gwarantowały wyniki charakteryzujące się zbliżonymi trendami zmian wartości (rysunki 3.10-3.11). Okazało się, że algorytm **SR**, podejmujący decyzje w oparciu o kryterium długości uszeregowania, jest wstanie osiągać rezultaty porównywalne z algorytmami **DR** i **GR**, które opracowano dla problemu niedeterministycznego (np. $n < 70$ i $m = 10$ lub $n = 100$ i $m = 10$). Algorytm **SR**, dzięki zastosowaniu (3.23) i $\overline{\Lambda}$, zapewniał łączny czas niewykorzystanych okien czasowych w harmonogramie, przy uwzględnieniu wszystkich scenariuszy skrajnych, mniejszy niż w harmonogramach wyznaczonych przez algorytmy **DR** i **GR**. Stabilizowanie się wartości rezultatów zaprezentowanych na rysunkach 3.12-3.14 wraz ze wzrostem liczby maszyn było rezultatem równoważenia ich obciążenia.



Rysunek 3.15. Wykres zależności czasu obliczeń od liczby zadań n dla $m = 2$. Wartości na osi y podano w skali logarytmicznej (log10)

Zaprezentowana na rysunku 3.15 zbieżność czasów obliczeń t_{SR} (czas obliczeń dla każdej instancji jest sumą czasów dwóch podejść: pesymistycznego i optymistycznego) i t_{DR} wynikała z zastosowania zachłannych strategii podejmowania decyzji dla jednego scenariusza skrajnego w każdym etapie optymalizacji. Różnica w czasach obliczeń wynikała z konieczności obliczania liczby potencjalnych zmian harmonogramie optymalnym

względem ustalonego harmonogramu przy jednym scenariuszu skrajnym (algorytm **SR**) oraz obliczania oszacowań długości uszeregowań dla scenariuszy skrajnych (algorytm **DR**).

Badanie 3.4

Następnie zweryfikowano, czy algorytm **GR** gwarantuje statystycznie najlepsze rezultaty optymalizacji. Dane do badań statystycznych wygenerowano w taki sam sposób jak w badaniu 2.4, a przedziały niepewności wyznaczono stosując formułę $u_{i,j} = \left[r_j^-, r_j^+\right] = \left[r_j^-, r_j^- + avg_j * \text{offset}_j\right]$, gdzie offset_j jest liczbą całkowitą wylosowaną z zakresu $\{1,2,\dots,10\}$ przy użyciu rozkładu jednostajnego. Wyniki zaprezentowano w tabeli C4 (podrozdział C w dodatku). Podczas pierwszego etapu badań statystycznych, do weryfikacji hipotezy mówiącej o tym, że rozkład danych jest zbliżony do rozkładu normalnego wykorzystano test Shapiro-Wilka (Shapiro & Wilk, 1965). Wyniki testów przeprowadzonych dla poziomu istotności $\alpha = 0.05$:

- wartość statystyki $W=0{,}7534$ oraz p-value $= 1{,}2\text{e-}11$ dla rezultatów algorytmu **DR**,
- wartość statystyki $W=0{,}7304$ oraz p-value $= 2{,}9\text{e-}12$ dla rezultatów algorytmu **GR**,
- wartość statystyki $W=0{,}7007$ oraz p-value $= 5{,}6\text{e-}13$ dla rezultatów algorytmu **SR**,

pozwalają dla każdego zbioru odrzucić hipotezę zerowa. Następnie przeprowadzono dwa testy Wilcoxona dla par (Wilcoxon, 1945), stosując konserwatywne podejście dla par o równej wartości (15% par), dla następujących hipotez statystycznych:

$$\begin{aligned} H_0&: \bar{Z}(\tilde{x}_{\text{DR}}) = \bar{Z}(\tilde{x}_{\text{GR}}), \\ H_1&: \bar{Z}(\tilde{x}_{\text{DR}}) > \bar{Z}(\tilde{x}_{\text{GR}}), \end{aligned} \tag{3.29}$$

gdzie wynikami są $z_{val} = 2{,}2270$ oraz p-value $= 0{,}013$ (odrzucamy H_0 na rzecz H_1) oraz

$$\begin{aligned} H_0&: \bar{Z}(\bar{x}_{\text{SR}}) = \bar{Z}(\tilde{x}_{\text{GR}}), \\ H_1&: \bar{Z}(\bar{x}_{\text{SR}}) > \bar{Z}(\tilde{x}_{\text{GR}}), \end{aligned} \tag{3.30}$$

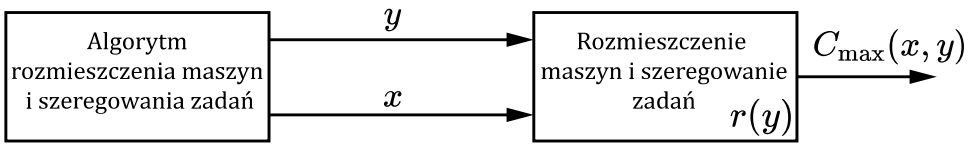
gdzie wynikami są $z_{val} = 7{,}1533$ oraz p-value $= 4{,}2\text{e-}13$ (odrzucamy H_0 na rzecz H_1; 11% par o tej samej wartości). Wynik badań statystycznych uzasadnia fakt, że algorytm **GR** generuje uszeregowania zapewniające statystycznie mniejszą długość uszeregowania niż algorytm **DR** i **SR**.

Rezultaty badań statystycznych potwierdziły prawdziwość tezy numer 2, którą sformułowano w podrozdziale 1.5.

4. Problem rozmieszczenia maszyn i szeregowania zadań

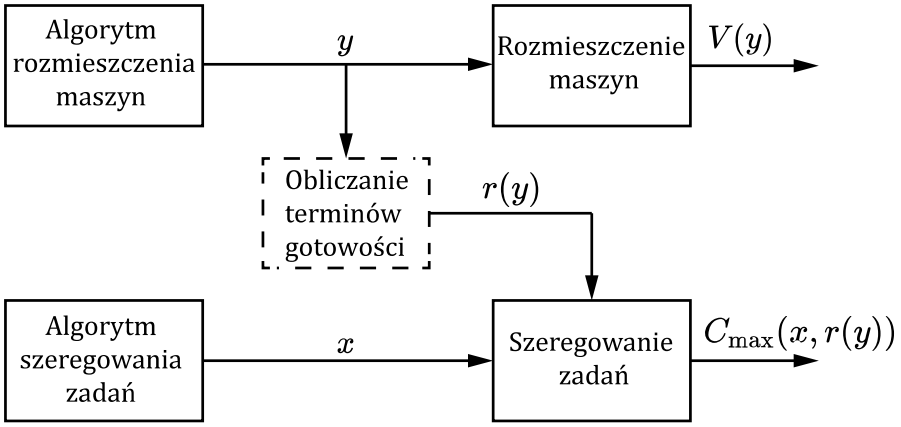
4.1. Wprowadzenie do optymalizacji w problemie ScheLoc

Treścią niniejszego rozdziału jest analiza efektywności różnych podejść do rozwiązania wybranego problemu rozmieszczenia maszyn i szeregowania zadań (ScheLoc): optymalizacji łącznej, optymalizacji sekwencyjnej z kryterium pośrednim oraz optymalizacji odpornej.



Rysunek 4.1. Podejście łączne (holistyczne) w problemie ScheLoc

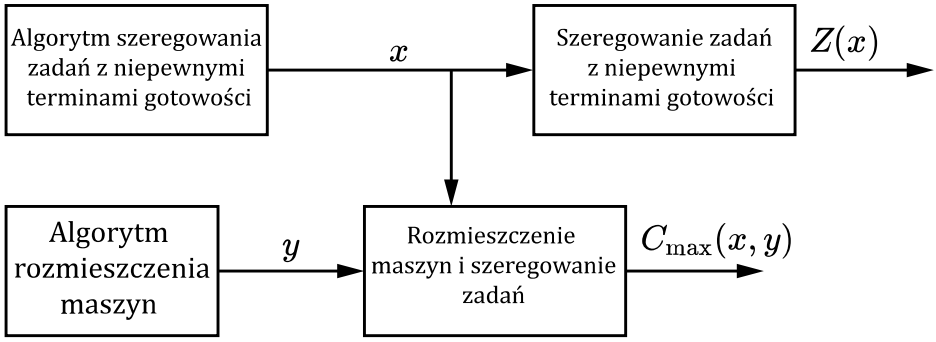
Schemat podejścia łącznego (nazywanego również holistycznym lub systemowym albo optymalizacją łączną) w problemie ScheLoc zaprezentowano na rysunku 4.1. Istotą tego podejścia jest jednoczesne podejmowanie decyzji o rozmieszczeniu maszyn oraz szeregowanie zadań ze względu na kryterium długości szeregowania. Zmienna decyzyjna y reprezentuje wybrane lokalizacje (współrzędne położenia w przestrzeni Euklidesowej) oraz zmienna decyzyjna x reprezentuje uszeregowanie. Terminy gotowości zadań $r(y)$ zależą od decyzji y, ponieważ są wyliczane na podstawie odległości pomiędzy danymi lokalizacjami zadań a wybranymi lokalizacjami maszyn.



Rysunek 4.2. Podejście sekwencyjne (dwuetapowe) w problemie ScheLoc

Podejście sekwencyjne w problemie ScheLoc (optymalizację sekwencyjną albo dwuetapową) zaprezentowano na rysunku 4.2. Jest to dwuetapowy problem optymalizacyjny, w którym wybór lokalizacji dla maszyn oraz szeregowanie zadań stanowią kolejne, odrębnie rozpatrywane podproblemy. Podczas pierwszego etapu optymalizacji wybierane są lokalizacje y dla maszyn (opierając się jedynie na danych współrzędnych lokalizacji zadań

oraz danych współrzędnych dopuszczalnych lokalizacji dla maszyn) w oparciu o arbitralnie przyjęte kryterium $V(y)$. Brak informacji o czasach realizacji zadań na poszczególnych maszynach wymusza wybór jedynie podzbioru lokalizacji bez wskazania rozmieszczenia maszyn (decyzje o przypisaniu maszyn dowolnych do lokalizacji nie są podejmowane). Podczas drugiego etapu optymalizacji, dla wybranego podzbioru lokalizacji, rozwiązujemy podproblem szeregowania zadań z terminami gotowości $r(y)$, przypisując jednocześnie maszyny do lokalizacji z podzbioru wskazanego podczas pierwszego etapu, aby minimalizować wartość kryterium $C_{\max}(x, r(y))$.



Rysunek 4.3. Podejście niepewne (odporne, niedeterministyczne) w problemie ScheLoc

Wykorzystanie optymalizacji odpornej (podejście niedeterministyczne albo niepewne) do rozwiązania problemu ScheLoc zaprezentowano na rysunku 4.3. Procedura składa się z dwóch części. Najpierw na podstawie danych wejściowych problemu ScheLoc definiujemy nowy niedeterministyczny problem szeregowania z przedziałowymi terminami gotowości zadań i kryterium maksymalnego żalu $Z(x)$ opartym na długości uszeregowania. Następnie pozycje maszyn w y są wybierane w taki sposób, aby harmonogram x, będący rezultatem rozwiązania niedeterministycznego problemu szeregowania, minimalizował długość uszeregowania $C_{\max}(x, y)$. Zadania przypisane do dowolnej maszyny wraz ze zdefiniowaną kolejnością ich realizacji nazywamy sekwencją. W podproblemie rozmieszczenia maszyny są przypisywane do wybranych lokalizacji razem z sekwencjami zadań, które zostały ustanowione podczas rozwiązywania problemu optymalizacji odpornej. Kolejność wykonywania zadań w sekwencjach nie jest modyfikowana. Fragmenty badań umieszczonych w niniejszym rozdziale zostały opublikowane w Ławrynowicz & Józefczyk (2019) oraz Józefczyk et al. (2022).

4.2. Sformułowanie łącznego problemu rozmieszczenia maszyn i szeregowania zadań

Niech $\mathrm{J} = \{1,2,\dots,j,\dots,n\}$ będzie zbiorem n niezależnych i niewywłaszczalnych zadań, które należy przetworzyć na co najwyżej m maszynach dowolnych, należących do zbioru $\mathrm{M} = \{1,2,\dots,i,\dots,m\}$. Założono dyskretny charakter rozmieszczenia maszyn. Zdefiniowano zbiór $\tilde{\mathrm{L}} = \{\tilde{l}_1, \tilde{l}_2, \dots, \tilde{l}_g, \dots, \tilde{l}_w\}$ zawierający w ustalonych współrzędnych lokalizacji dla maszyn, w którym lokalizacja $\tilde{l}_g = (\tilde{l}_{g,1}, \tilde{l}_{g,2}) \in \mathbb{R}^2$ jest opisana współrzędnymi w dwuwymiarowej przestrzeni Euklidesowej. Binarna decyzja o wyborze lokalizacji $\tilde{l}_g \in \tilde{\mathrm{L}}$ dla maszyny $i \in \mathrm{M}$ jest reprezentowana przez $y_{g,i} \in \{0,1\}$ i wszystkie decyzje są elementami składowymi binarnej macierzy decyzji $y = [y_{g,i}]_{\substack{g=1,2,\dots,w \\ j=1,2,\dots,m}}$. Współrzędne lokalizacji zadań są ustalone i zdefiniowane w zbiorze $\mathrm{L} = \{l_1, l_2, \dots, l_j, \dots, l_n\}$, w którym lokalizacja l_j jest opisana współrzędnymi położenia $l_j = (l_{j,1}, l_{j,2}) \in \mathbb{R}^2$ zadania $j \in \mathrm{J}$. Decyzje o rozmieszczeniu maszyn determinują terminy gotowości zadań. Bazując na elementach zbiorów L i $\tilde{\mathrm{L}}$ można obliczyć wszystkie możliwe terminy gotowości zadań niezależnie od rozmieszczenia maszyn za pomocą wzoru

$$r^{\mathrm{SL}} = [r^{\mathrm{SL}}_{g,j} = d(\tilde{l}_g, l_j)]_{\substack{g=1,2,\dots,w, \\ j=1,2,\dots,n}} \tag{4.1}$$

gdzie $d(\tilde{l}_g, l_j)$ jest funkcją odległości w dwuwymiarowej przestrzeni Euklidesowej, która zwraca wartość odległości pomiędzy współrzędnymi $\tilde{l}_g$ i l_j (lokalizacjami); $r^{\mathrm{SL}}_{g,j}$ jest terminem gotowości zadania j znajdującego się na pozycji l_j jeżeli na pozycji $\tilde{l}_g$ zostanie umieszona maszyna. Reprezentacja harmonogramu $x = [x_{i,k,j}]_{\substack{i=1,2,\dots,m \\ k,j=1,2,\dots,n}}$ będącego rozwiązaniem problemu ScheLoc jest tożsama z reprezentacją rozwiązania problemu $Rm|r_{i,j}|C_{\max}$, gdzie terminy gotowości $r_{i,j}$ są określone przez elementy macierzy r^{SL}, dla których $y_{g,i} = 1$. Dla zdefiniowanych parametrów problemu oraz zmiennych decyzyjnych długość uszeregowania jest równa

$$C_{\max}(x, y; \mathrm{S}_l) = \max_{\substack{g=1,2,\dots,w \\ i=1,2,\dots,m}} y_{g,i}\, C_{i,n_i}(x; \mathrm{S}_l), \tag{4.2}$$

gdzie $\mathrm{S}_l = \{\mathrm{L}, \tilde{\mathrm{L}}, p, r^{\mathrm{SL}}\}$ jest zbiorem parametrów problemu ScheLoc.

Sformułowanie łącznego (deterministycznego) problemu ScheLoc jest następujące: Dla danych J, M, L, $\tilde{\mathrm{L}}$, p, r^{SL} należy wyznaczyć macierze x^* i y^* minimalizujące długość uszeregowania

$$C_{\max}(x^*, y^*; \mathrm{S}_l) = \min_{x,y} C_{\max}(x, y; \mathrm{S}_l). \tag{4.3}$$

Na zmienną x nałożono ograniczenia (2.4)-(2.6) oraz przyjęto, że

$$\sum_{g=1}^{w} \sum_{i=1}^{m} y_{g,i} = m, \tag{4.4}$$

$$\sum_{i=1}^{m} y_{g,i} \leq 1, \qquad g = 1,2, \dots, w, \tag{4.5}$$

$$\sum_{i=1}^{m} \sum_{j=1}^{n} x_{i,k=1,j} \leq m, \tag{4.6}$$

$$x_{i,k,j} \in \{0,1\}, y_{g,i} \in \{0,1\}. \tag{4.7}$$

Ograniczenie (4.4) wymusza rozmieszczenie m maszyn (wybór m lokalizacji). W każdej lokalizacji może być umieszczona co najwyżej jedna maszyna (4.5). Przetwarzanie zadań można zrealizować na co najwyżej m maszynach (4.6). Ograniczenie (4.6) wynika z analizy złożoności obliczeniowej problemu $Rm|r_{i,j}|C_{\max}$ (przeprowadzonej w podrozdziale 2.3), uzasadniającej brak konieczności równoważenia obciążenia pracy maszyn. Wprowadzenie (4.6) ma na celu podkreślenie faktu, że harmonogram nie musi angażować całego systemu, którego architektura jest wymuszona w (4.4) i (4.5). Dziedziny wartości zmiennych decyzyjnych podano w (4.7).

4.3. Relacja pomiędzy problemem ScheLoc i problemem szeregowania $Rm|r_{i,j}|C_{\max}$

Złożoność obliczeniowa problemu ScheLoc wynika ze złożoności deterministycznego problemu szeregowania $Rm|r_{i,j}|C_{\max}$. Jeżeli liczba dopuszczalnych lokalizacji dla maszyn jest równa liczbie maszyn, tzn. $w = m$, lub jeżeli lokalizacje maszyn są apriorycznie ustalone macierzą y, to problem ScheLoc jest tożsamy problemowi szeregowania $Rm|r_{i,j}|C_{\max}$, który jest co najmniej NP-trudny.

Podproblem wyboru m spośród w dostępnych lokalizacji można rozwiązać w czasie wielomianowym poprzez realizację pełnego (wyczerpującego) przeglądu możliwych decyzji. Algorytm **Exhaustive** (**EX**) generuje $\frac{w!}{(w-m)!}$ sekwencji ($\frac{w!}{(w-m)!} \leq w^m$, $m \leq w$), z których

każda zawiera m wybranych lokalizacji. Liczba sekwencji $\frac{w!}{(w-m)!}$ jest równa liczbie unikalnych m-elementowych wariacji bez powtórzeń wybranych ze zbioru w elementowego.

Algorytm **Exhaustive (EX)**

Require: $\tilde{\mathrm{L}}$
Ensure: T // zbiór zawierający wszystkie możliwe rozmieszczenia maszyn

1. Zdefiniuj zbiór T $\coloneqq \emptyset$.
2. **for** $\tilde{l}^{(1)}$ **in** $\tilde{\mathrm{L}}$
3. **for** $\tilde{l}^{(2)}$ **in** $\tilde{\mathrm{L}}\backslash\{\tilde{l}^{(1)}\}$
…
4. **for** $\tilde{l}^{(m)}$ **in** $\tilde{\mathrm{L}}\backslash\{\tilde{l}^{(1)}, \tilde{l}^{(2)}, \dots, \tilde{l}^{(m-1)}\}$
5. Przekształć sekwencję $(\tilde{l}^{(1)}, \tilde{l}^{(2)}, \dots, \tilde{l}^{(i)}, \dots, \tilde{l}^{(m)})$, w której maszyna i jest umieszczona w lokalizacji $\tilde{l}^{(i)}$, do postaci macierzy y oraz umieść macierz y w zbiorze T.
6. **end for (2) – (5)**

Złożoność czasowa algorytmu **EX** jest równa $\mathcal{O}\left(\frac{w!}{(w-m)!}wm\right)$, ponieważ pętle for wykonają się $w(w-1)(w-2)\dots(w-m+1) = \frac{w!}{(w-m)!} \leq w^m$ razy oraz w Linii 5 następuje inicjalizacja macierzy wm elementowej. Liczba macierzy wm elementowych determinuje złożoność pamięciową równą $\mathcal{O}\left(\frac{w!}{(w-m)!}wm\right)$.

Możliwość realizacji wyczerpującego przeglądu rozmieszczeń maszyn w czasie wielomianowym gwarantuje, że optymalne rozwiązanie problemu (4.3) wymaga optymalnego rozwiązania $\frac{w!}{(w-m)!}$ niezależnych problemów szeregowania $Rm|r_{i,j}|C_{\max}$. Aby efektywnie rozwiązać problem ScheLoc w wersji łącznej, opracowano hybrydowy algorytm **ExGreedy** (**EG**), łączący algorytmy **EX** oraz **GD**.

Algorytm **ExGreedy (EG) – podejście łączne**

Require: J, M, L, $\tilde{\mathrm{L}}$, p
Ensure: $x_{\mathrm{EG}}, y_{\mathrm{EG}}$

1. Wygeneruj macierz r^{SL}, oblicz T $\coloneqq$ **EX(**$\tilde{\mathrm{L}}$**)** oraz wygeneruj losowo x_{B} i y_{B}.
2. **for** y **in** T
3. Wybierz elementy macierzy r^{SL}, które odpowiadają rozmieszczeniu w y, i umieść je w macierzy $r(y)$.
4. $x \coloneqq$ **GD(**J, M, $r(y), p$**)**
5. **if** $C_{\max}(x, y; \mathrm{S}_l) < C_{\max}(x_{\mathrm{B}}, y_{\mathrm{B}}; \mathrm{S}_l)$ **then** $x_{\mathrm{B}} \coloneqq x$ i $y_{\mathrm{B}} \coloneqq y$
6. **end for (2)**
7. $x_{\mathrm{EG}} \coloneqq x_{\mathrm{B}}$, $y_{\mathrm{EG}} \coloneqq y_{\mathrm{B}}$

Złożoność czasowa algorytmu **EG** wynosi $\mathcal{O}\left(\frac{w!}{(w-m)!}n^2m\right)$, ponieważ dla każdego z $\frac{w!}{(w-m)!}$ rozmieszczeń maszyn uruchamiany jest algorytm **GD**, którego złożoność czasowa jest równa $\mathcal{O}(n^2m)$. Złożoność pamięciowa $\mathcal{O}\left(mw\frac{w!}{(w-m)!}+n^2m\right)$ jest konsekwencją rozmiaru zbioru T oraz rozmiaru reprezentacji rozwiązania.

Uruchamianie algorytmu **EG** dla dużych instancji problemu ($m > 5$ i $w > 20$) jest niepraktyczne ze względu na nieakceptowalny czas obliczeń, który jest efektem dużej złożoności obliczeniowej wyczerpującego przeglądu dostępnych lokalizacji. Algorytmy dedykowane dla dużych instancji problemu podano w podrozdziałach 4.6.1 oraz 4.6.2.

4.4. Sformułowanie sekwencyjnego (dwuetapowego) problemu rozmieszczenia maszyn i szeregowania zadań

W dwuetapowym problemie optymalizacyjnym ScheLoc założono, że rozmieszczenie maszyn jest efektem rozwiązania podproblemu optymalizacyjnego (pośredniego), uwzględniającego jedynie współrzędne rozmieszczenia zadań dane zbiorem L i współrzędne dopuszczalnych lokalizacji dla maszyn dane zbiorem $\tilde{\mathrm{L}}$. Wskazywany jest podzbiór lokalizacji, w których należy umieścić maszyny, bez przypisywania konkretnych maszyn do lokalizacji. Takie podejście wynika z braku informacji o czasach przetwarzania zadań w pierwszym etapie optymalizacji. Wówczas wybór podzbioru lokalizacji jest równoważny rozmieszczeniu maszyn identycznych. W celu sformalizowania podproblemu pośredniego zdefiniowano funkcję obliczającą odległość między zadaniem znajdującym się w lokalizacji l_j i najbliżej zlokalizowaną pozycją dla maszyny, która została wybrana

$$d_{\min}(y, l_j) = \min_{\tilde{l}_g \in \tilde{\mathrm{L}}'(y)} d(\tilde{l}_g, l_j), \tag{4.8}$$

gdzie podzbiór $\tilde{\mathrm{L}}'(y) = \{\tilde{l}_g \in \tilde{\mathrm{L}} \mid y_{g,i} = 1, i = 1,2,\dots,m\}$, $\left|\tilde{\mathrm{L}}'(y)\right| = m$, zawiera współrzędne lokalizacji maszyn wskazanych przez algorytm optymalizacyjny dla podproblemu rozmieszczenia. Opisywany podproblem rozmieszczenia maszyn może być formułowany w różny sposób. W dalszym ciągu skoncentrowano się na dwóch wybranych wersjach, które wykorzystują znane z literatury zagadnienia rozmieszczenia (ang. location problems). W pierwszej wersji rozważamy problem optymalizacyjny, którego rozwiązanie wykorzystujemy do dyskretnego rozmieszczenia maszyn, polega on na minimalizacji największej odległości pomiędzy lokalizacją zadania i wybraną lokalizacją dla maszyny. Omówiony problem w literaturze jest określany jako m-center (Hakimi, 1964; Kariv &

Hakimi, 1979a) i dla oznaczeń przyjętych dla problemu ScheLoc może zostać sformułowany w następujący sposób:

$$V^{(\text{center})}(y^*) = \min_{y} V^{(\text{center})}(y) = \min_{y} \max_{l_j \in \mathrm{L}} d_{\min}(y, l_j). \tag{4.9}$$

Druga wersja podproblemu rozmieszczenia maszyn, nawiązująca do zagadnienia m-median (Hakimi, 1965; Kariv & Hakimi, 1979b), zakłada minimalizację sumy odległości pomiędzy wybranymi lokalizacjami dla maszyn oraz danymi lokalizacjami zadań:

$$V^{(\text{median})}(y^*) = \min_{y} V^{(\text{median})}(y) = \min_{y} \sum_{l_j \in \mathrm{L}} d_{\min}(y, l_j). \tag{4.10}$$

Rozmieszczenie maszyn w oparciu o arbitralnie przyjęte kryteria problemów m-center lub m-median ma na celu ograniczenie czasów przemieszczania się zadań do wybranych miejsc, co przekłada się bezpośrednio na skrócenie czasów potrzebnych do przygotowania zadań do realizacji. Rozwiązując problem (4.9) ograniczamy maksymalny czas przemieszczenia się zadań, natomiast w (4.10) minimalizujemy całkowity czas przemieszania się zadań do najbliższych lokalizacji w systemie. Podczas pierwszego etapu optymalizacji nie bierzemy pod uwagę elementów macierzy p. Wówczas przypisania maszyn do wybranych lokalizacji są dowolne, ponieważ nie wpływają na wartość kryteriów (4.9) i (4.10). Warto podkreślić, że wybrane wersje podproblemów są NP-trudne, gdy wartość m jest zmienną decyzyjną (Garey & Johnson, 1979). W rozważanym przypadku liczba maszyn nie jest zmienną decyzyjną, stąd podproblemy sformułowane w (4.9) i (4.10) można rozwiązać optymalnie algorytmem **EX**, który realizuje przegląd wszystkich możliwych rozmieszczeń maszyn, i wybiera macierz y minimalizującą wartość rozważanego kryterium.

Podczas drugiego etapu optymalizacji w problemie Scheloc, dla ustalonej macierzy y, terminy gotowości $r(y)$ są wyliczane w odniesieniu jedynie do wybranych m lokalizacji. W rezultacie, istnieje $m!$ optymalnych rozwiązań każdego z rozważnych problemów, ponieważ dla rozmieszczenia opisanego macierzą y można wygenerować $m!$ permutacji przypisań maszyn dowolnych. Aby optymalnie rozwiązać problem ScheLoc w podejściu sekwencyjnym, dla ustalonej macierzy y, należy optymalnie rozwiązać $m!$ problemów szeregowania $Rm|r_{i,j}|C_{\max}$. Pseudokod sekwencyjnego podejścia do rozwiązania problemu ScheLoc podano poniżej.

Algorytm **LOC+GD – podejście sekwencyjne**

Require: $\mathrm{J}, \mathrm{M}, \mathrm{L}, \tilde{\mathrm{L}}, p, c_r$ // $c_r \in \{\mathrm{center}, \mathrm{median}\}$
Ensure: x_{GD}, y

1. Wygeneruj macierz r^{SL}, oblicz $\mathrm{T} \coloneqq$ **LOC**$(\tilde{\mathrm{L}})$.
2. Wybierz ze zbioru T macierz y, która gwarantuje najmniejszą wartość $\tilde{y} = \arg\min_{y} V^{(c_r)}(y)$.
3. Uruchom algorytm **GD(**$\mathrm{J}, \mathrm{M}, r(\tilde{y}), p$**)** dla każdego z $m!$ możliwych rozmieszczeń maszyn w lokalizacjach wskazanych macierzą $\tilde{y}$.
4. Wybierz zmienne x_{GD} oraz y zapewniające najmniejszą długość uszeregowania $C_{\max}(x_{\mathrm{GD}}, y; \mathrm{S}_l)$ w problemie ScheLoc. Macierz y opisuje rozmieszczenie maszyn w lokalizacjach wskazanych macierzą $\tilde{y}$.

Złożoność czasowa podejścia sekwencyjnego wynosi $\mathcal{O}\left(\max\left\{m!\, n^2 m, \frac{w!}{(w-m)!} wm\right\}\right)$, ponieważ dla każdego z $m!$ rozmieszczeń maszyn uruchamiany jest algorytm **GD**, którego złożoność czasowa jest równa $\mathcal{O}(n^2 m)$. Złożoność pamięciowa $\mathcal{O}\left(\frac{w!}{(w-m)!} wm + n^2 m\right)$ jest konsekwencją rozmiaru zbioru T oraz rozmiaru reprezentacji rozwiązania. Akronim **LOC** określa wybrany algorytm optymalizacyjny dla podproblemu rozmieszczenia maszyn. W rozprawie rozważono algorytm **EX** oraz algorytm oparty na metaheurystyce (dedykowany dla dużych instancji, tzn. $m > 5$ i $w > 20$; podany w podrozdziale 4.6.3).

Różnica między podejściem łącznym i sekwencyjnym wynika z dekompozycji problemu ScheLoc. W podejściu sekwencyjnym rozmieszczenie maszyn nie może być modyfikowane w celu zmniejszenia długości uszeregowania, ponieważ lokalizacje dla maszyn są wybierane jeden raz (algorytmem **LOC**) przed procesem szeregowania zadań (realizowanym algorytmem **GD)**. Natomiast, w podejściu łącznym, algorytm modyfikuje rozmieszczenie maszyn oraz harmonogram, aby zminimalizować długość uszeregowania w problemie ScheLoc.

4.5. Sformułowanie problemu rozmieszczenia maszyn i szeregowania zadań z przedziałowymi terminami gotowości

Sformułowanie niedeterministycznego problemu ScheLoc zakłada przedziałowe (niepewne) terminy gotowości zadań wyznaczone na podstawie parametrów problemu deterministycznego. Zaproponowano dwa podejścia do formułowania a następnie do rozwiązywania niedeterministycznego problemu ScheLoc.

1) $\boldsymbol{Rm|r_j^- \le r_j \le r_j^+|Reg(C_{\max})}$ **+ ScheLoc**

Przedział niepewności $\bar{u}_j$, $j = 1,2,\dots,n$, jest określony przez najmniejszą oraz największą odległość pomiędzy $l_j \in$ L i elementami zbioru $\tilde{\text{L}}$ zgodnie ze wzorem

$$\bar{u}_j = \left[\min_{\tilde{l}_g \in \tilde{\text{L}}} d\left(\tilde{l}_g, l_j\right), \max_{\tilde{l}_g \in \tilde{\text{L}}} d\left(\tilde{l}_g, l_j\right) \right], \qquad j = 1,2,\dots,n. \tag{4.11}$$

W skrajnym przypadku wszystkie przedziały niepewności mogą być wyznaczone na podstawie jedynie dwóch elementów w $\tilde{\text{L}}$. Niech lokalizacje w $\tilde{\text{L}}$ będą rozmieszczone wzdłuż linii prostej oraz niech $m-2$ współrzędnych lokalizacji w $\tilde{\text{L}}$ jest rozmieszczone pomiędzy dwoma punktami $\tilde{l}_{g_k} = \left(\tilde{l}_{g_k,1}, \tilde{l}_{g_k,2}\right)$, $k = 1,2$, w taki sposób, że $\tilde{l}_{g_1,1} < \tilde{l}_{g_2,1}$, $\tilde{l}_{g_1,2} < \tilde{l}_{g_2,2}$. Jeżeli $l_{j,1} > \tilde{l}_{g_2,1}$ i $l_{j,2} > \tilde{l}_{g_2,2}$ lub $l_{j,1} < \tilde{l}_{g_1,1}$ i $l_{j,2} < \tilde{l}_{g_1,2}$, to przedział niepewności $\bar{u}_j$, $j = 1,2,\dots,n$, jest określony przez odległości do dwóch punktów $\tilde{l}_{g_k}$, $k = 1,2$. Rezultatem omówionych założeń jest powstanie odcinka, wzdłuż którego znajduje się nieokreślona liczba dopuszczalnych lokalizacji dla maszyn, do których zadanie j może się przemieścić w nieprecyzyjnie określonym czasie określonym przedziałem $\bar{u}_j$.

Rozwiązanie problemu szeregowania $Rm|r_j^- \le r_j \le r_j^+|Reg(C_{\max})$, przy założeniu (4.11), jest pierwszym etapem optymalizacji. Podczas kolejnego etapu podejmowania decyzji, opierając się na elementach zbioru $\tilde{\text{L}}$, wybieranych jest $\frac{w!}{(w-m)!}$ lokalizacji, w których można rozmieścić maszyny. Następnie, przy ustalonym harmonogramie, wybieramy rozmieszczenie maszyn, aby uzyskać jak najmniejszą długość uszeregowania w deterministycznym problemie ScheLoc. Pseudokod podejścia niepewnego, bazującego na algorytmach **W** ∈ {**DR**,**GR**,**SR**} i **EX**, dla problemu ScheLoc podano poniżej.

Algorytm **W** + **EX** – **podejście niepewne**

Require: J, M, L, $\tilde{\text{L}}$, p, **W**
Ensure: x_{W}, y

1. Oblicz przedziały niepewności $\bar{u}_j$, $j = 1,2,\dots,n$, zgodnie z (4.11), wykorzystując dane wejściowe problemu ScheLoc, oraz wygeneruj zbiór scenariuszy $\bar{\text{U}}$.
2. Uruchom algorytm $x_{\text{W}} :=$**W**$(\text{J}, \text{M}, \bar{\text{U}}, p)$**.**
3. Wygeneruj wszystkie macierze rozmieszczeń maszyn T := **EX**$(\tilde{\text{L}})$.
4. Przypisz maszyny (wraz z sekwencjami zadań) do lokalizacji wskazanych w macierzy $y \in$ T, które zapewniają najmniejszą długość uszeregowania $C_{\max}(x_{\text{W}}, y; \text{S}_l)$.

Czasowa złożoność obliczeniowa podejścia niepewnego opartego na (4.11) wynosi $\mathcal{O}\left(\frac{w!}{(w-m)!} n^2 m\right)$ (dla algorytmu **DR**), $\mathcal{O}\left(\max\left\{n^3 m, \frac{w!}{(w-m)!} n^2 m\right\}\right)$ (dla algorytmu **SR**),

$\mathcal{O}\left(\max\left\{n^4m^2, \frac{w!}{(w-m)!}n^2m\right\}\right)$ (dla algorytmu **GR**). Formuła $\mathcal{O}\left(\frac{w!}{(w-m)!}n^2m\right)$ w każdym przypadku określa złożoność czasową przypisania sekwencji zadań do wybranych lokalizacji, gdzie $\mathcal{O}(n^2m)$ jest złożonością czasową wyliczania oszacowanej wartości kryterium. Złożoność pamięciowa $\mathcal{O}\left(\frac{w!}{(w-m)!}wm + n^2m\right)$ jest konsekwencją rozmiaru zbioru T oraz rozmiaru reprezentacji rozwiązania.

2) $\boldsymbol{Rm|r_{i,j}^{-} \le r_{i,j} \le r_{i,j}^{+}|Reg(C_{max})}$ + ScheLoc

Obliczenie granic przedziałów $\tilde{u}_{i,j},\ i = 1,2,\dots,m$, dla zadania j wymaga zdefiniowania podzbioru $\tilde{\mathrm{L}}^j \subseteq \tilde{\mathrm{L}}$, który zawiera współrzędne m najdalej położonych, względem l_j, lokalizacji dostępnych dla maszyn. Przedziały niepewności dla zadania j na m maszynach wyliczamy na podstawie rekurencyjnego wzoru

$$\begin{gathered}\tilde{u}_{i,j} = \left[ref_{i,j}, ref_{i,j} + \max_{\tilde{l}_g \in \tilde{\mathrm{L}}^j \setminus \{\mathrm{I}(i-1)\}} d\left(\tilde{l}_g, l_j\right)\right], \qquad i = 1,2,\dots,m,\\ \mathrm{I}(0) = \emptyset, \qquad \mathrm{I}(i) = \mathrm{I}(i-1) \cup \arg \max_{\tilde{l}_g \in \tilde{\mathrm{L}}^j \setminus \{\mathrm{I}(i-1)\}} d\left(\tilde{l}_g, l_j\right),\end{gathered} \tag{4.12}$$

gdzie $ref_{i,j} = d\left(l_i, l_j\right)$ jest arbitralnie daną wartością (np. może stanowić odległość referencyjną pomiędzy l_j i arbitralnie wskazaną lokalizacją l_i), $\mathrm{I}(i-1)$ jest zbiorem lokalizacji wykorzystanych do wyznaczenia górnych granic przedziałów $\tilde{u}_{i,j-1}, \tilde{u}_{i-2,j}, \dots, \tilde{u}_{1,j}$.

W pracy przyjęto dwie metody wyliczania wartości $ref_{i,j}$. W pierwszym przypadku odległość referencyjna dla zadania j na m maszynach jest wyliczana na podstawie rekurencyjnego wzoru

$$\begin{gathered}ref_{i,j} = \min_{\tilde{l}_g \in \tilde{\mathrm{L}} \setminus \{\tilde{\mathrm{L}}^j \cup \mathrm{I}(i-1)\}} d\left(\tilde{l}_g, l_j\right), \qquad i = 1,2,\dots,m,\\ \mathrm{I}(0) = \emptyset, \qquad \mathrm{I}(i) = \mathrm{I}(i-1) \cup \arg \min_{\tilde{l}_g \in \tilde{\mathrm{L}} \setminus \{\tilde{\mathrm{L}}^j \cup \mathrm{I}(i-1)\}} d\left(\tilde{l}_g, l_j\right).\end{gathered} \tag{4.13}$$

W drugim przypadku odległość referencyjna dla zadania j jest wspólna dla m maszyn

$$ref_{i,j} = \min_{\tilde{l}_g \in \tilde{\mathrm{L}} \setminus \{\tilde{\mathrm{L}}^j\}} d\left(\tilde{l}_g, l_j\right). \tag{4.14}$$

Opracowane metody wyliczania $ref_{i,j}$ różnią się tym, że (4.13) równoważy wartości górnych granic przedziałów niepewności, wyliczonych za pomocą (4.12), natomiast (4.14) równoważy wartości dolnych granic przedziałów niepewności. Oba podejścia są komplementarne.

Różnice w wyliczaniu przedziałów niepewności zaprezentowano w tabeli 4.1 dla jednego zadania na przykładzie obliczeniowym, gdy $m = 2$, $w = 6$. Przyjęto wektor $D_j = [3,9,17,23,33,41]$, w którym każda wartość reprezentuje odległość w przestrzeni Euklidesowej między l_j a w dostępnymi lokalizacjami dla maszyn, na przykład wartość 3 reprezentuje odległości między l_j a najbliżej położoną lokalizacją $\tilde{l}_g$.

Tabela 4.1. Przykład obliczeniowy

$ref_{i,j}$ wyliczone zgodnie z (4.12)	$ref_{i,j}$ wyliczone zgodnie z (4.13)
$\tilde{u}_{1,j} = [3,3+41] = [3,44]$ $\tilde{u}_{2,j} = [9,9+33] = [9,42]$	$\tilde{u}_{1,j} = [3,3+41] = [3,44]$ $\tilde{u}_{2,j} = [3,3+33] = [3,36]$

Analizując wzory (4.12)-(4.14), można zauważyć, że przedział niepewności dla dowolnego zadania na maszynie $i = 1$ jest zawsze najdłuższy, ponieważ wraz ze wzrostem wartości indeksu i długość przedziału niepewności zmniejsza się. Z tego powodu, po obliczeniu przedziałów niepewności, rozważamy $m!$ reprezentacji niepewności na maszynach. Niech macierz

$$\tilde{\mathrm{U}}^{(i_1,i_n,\ldots,i_m)} = \begin{bmatrix} \tilde{u}_{i_1,1} & \tilde{u}_{i_2,1} & \ldots & \tilde{u}_{i_k,1} & \ldots & \tilde{u}_{i_m,1} \\ \tilde{u}_{i_1,2} & \tilde{u}_{i_2,2} & \ldots & \tilde{u}_{i_k,2} & \ldots & \tilde{u}_{i_m,2} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ \tilde{u}_{i_1,n} & \tilde{u}_{i_2,n} & \ldots & \tilde{u}_{i_k,n} & \ldots & \tilde{u}_{i_m,n} \end{bmatrix} \tag{4.15}$$

reprezentuje przedziały niepewności obliczone dla n zadań na m maszynach przy sekwencji $(i_1, i_n, \ldots, i_m)$, $\mathrm{M} = \{i_1, i_n, \ldots, i_m\}$. Algorytm $\mathbf{D} \in \{\mathbf{DR}, \mathbf{GR}\}$ jest uruchamiany $m!$ razy (dla każdej możliwej permutacji $i_1, i_n, \ldots, i_m$). Pseudokod podejścia niepewnego, bazującego na algorytmach $\mathbf{D} \in \{\mathbf{DR}, \mathbf{GR}\}$ i **EX**, wykorzystanego do rozwiązania problemu ScheLoc podano poniżej.

Algorytm **D + EX – podejście niepewne**

Require: J, M, L, $\tilde{\mathrm{L}}$, p, **D**
Ensure: $\tilde{x}_{\mathrm{D}}$, y

1. Oblicz przedziały niepewności $\tilde{u}_{i,j}$, $i = 1,2,\ldots,m$, $j = 1,2,\ldots,n$, zgodnie z (4.12)-(4.14), wykorzystując dane wejściowe problemu ScheLoc, oraz wygeneruj zbiór scenariuszy $\tilde{\mathrm{U}}$.
2. Uruchom $m!$ (dla każdej metody wyznaczania przedziałów niepewności) razy algorytm $\mathbf{D}(\mathrm{J}, \mathrm{M}, \tilde{\mathrm{U}}, p)$ oraz umieść uszeregowania w zbiorze $\tilde{\mathrm{X}}$.
3. Wygeneruj wszystkie macierze rozmieszczeń maszyn $\mathrm{T} \coloneqq \mathbf{EX}(\tilde{\mathrm{L}})$.
4. Przypisz maszyny (wraz z sekwencjami zadań) do lokalizacji wskazanych w macierzy $y \in \mathrm{T}$, które zapewniają najmniejszą długość uszeregowania $C_{\max}(\tilde{x}_{\mathrm{D}}, y; \mathrm{S}_l)$, $\tilde{x}_{\mathrm{D}} \in \tilde{\mathrm{X}}$.

Czasowa złożoność obliczeniowa podejścia niepewnego opartego na (4.12) jest równa $\mathcal{O}\left(\max\left\{m!\,n^2m, \frac{w!}{(w-m)!}n^2m\right\}\right)$ (dla algorytmu **DR**) lub $\mathcal{O}\left(\max\left\{m!\,n^4m^2, \frac{w!}{(w-m)!}n^2m\right\}\right)$ (dla algorytmu **GR**). Złożoność pamięciowa $\mathcal{O}\left(m!\,n^2m + \frac{w!}{(w-m)!}wm\right)$ jest konsekwencją rozmiarów zbiorów $\tilde{\mathrm{X}}$ oraz T. Przy zastosowaniu podejścia $Rm\left|r_j^- \le r_j \le r_j^+\right|Reg(C_{\max}) +$ ScheLoc nie ma konieczności uruchamiania $m!$ razy algorytmu szeregowania, ponieważ przedziały niepewności w (4.11) są niezależne od maszyn.

Wzory (4.11) i (4.12) definiują nieprecyzyjność wyrażoną przedziałowymi terminami gotowości zadań. Konsekwencją nieprecyzyjności jest również brak informacji o wszystkich elementach zbioru $\tilde{\mathrm{L}}$, ponieważ opierając się na przedziałach niepewności, nie można precyzyjnie wskazać wszystkich współrzędnych dopuszczalnych lokalizacji dla maszyn. W przypadku, gdy skrajne wartości przedziałów niepewności terminów gotowości zadań nie uwzględniają wszystkich elementów w $\tilde{\mathrm{L}}$, to również nie można określić rozmiaru tego zbioru. Zatem podczas pierwszego etapu optymalizacji, w podejściu niepewnym w problemie ScheLoc, decyzje są podejmowane dla danych niesprecyzowanych i niepełnych.

Omówione podejścia niepewne zakładają sformułowanie i wykorzystanie rozwiązań problemów optymalizacji odpornej do rozwiązania problemu ScheLoc. W praktyce sformułowane problemy optymalizacji odpornej wraz z metodami obliczania przedziałów niepewności stanowią problemy zastępcze dla sformułowanego problemu ScheLoc. Wówczas rozwiązanie problemu ScheLoc (sformułowanego dla ustalonych parametrów problemu) stanowi rozwiązanie referencyjne, rozumiane jako punkt odniesienia, do oceny jakości rezultatów podejść niepewnych. Z tego powodu do optymalizacji w podejściu niepewnym oraz w deterministycznym problemie Scheloc zastosowano algorytmy deterministyczne, i nie adaptowano algorytmów bazujących na metaheurystykach, aby wykluczyć błędy wynikające z podejmowania decyzji w warunkach pseudolosowości.

4.6. Algorytmy rozwiązania problemu rozmieszczenia maszyn i szeregowania zadań

W niniejszym podrozdziale zaprezentowano algorytmy opracowane dla dużych instancji problemu ScheLoc ($m > 5$ i $w > 20$).

4.6.1. Algorytm Hybrid (podejście łączne)

Algorytm **Hybrid** (**HS**) łączy metaheurystykę Simulated Annealing z algorytmem **GD**. Algorytm **HS** realizuje proces optymalizacji zaczynając od wygenerowania losowego

rozmieszczenia maszyn i wyznaczenia uszeregowania za pomocą algorytmu **GD**. Operator generowania rozwiązań sąsiednich realizuje sekwencję dwóch procedur:

1. zastępuje lokalizację $\tilde{l}_g$ wybraną dla maszyny i, $y_{g,i}(\varphi) = 1$, najbliższą wolną lokalizacją
$$l = \arg \min_{\tilde{l}_i \in \tilde{\mathrm{L}} \wedge y_{t,i}(\varphi) \neq 1, t=1,2,\ldots,w} d(\tilde{l}_g, \tilde{l}_i), \tag{4.16}$$
2. na podstawie nowego wektora rozmieszczenia $y_{\mathrm{N}}(\varphi)$ definiowana jest macierz $r(y_{\mathrm{N}}(\varphi))$ oraz uruchamiany jest algorytm **GD**.

W każdej iteracji generowane jest m sąsiednich rozwiązań, ponieważ w każdym rozwiązaniu sąsiednim modyfikujemy pozycje tylko jednej maszyny. Kolejne kroki algorytmu pokrywają się ze szkieletem metaheurystyki Simulated Annealing. Temperatura początkowa

$$\tilde{V}_0(p, \mathrm{L}, \tilde{\mathrm{L}}) = \max_{\substack{g=1,2,\ldots,w \\ j=1,2,\ldots,n}} d(\tilde{l}_g, l_j) + m^{-1} \sum_{j=1}^{n} \max_{i=1,2,\ldots,m} p_{i,j}, \tag{4.17}$$

jest ustalana poprzez wyliczenie wartości kryterium dla nieefektywnych decyzji. Funkcja energii: $\tilde{f}(x(\varphi), x_{\mathrm{N}}(\varphi), y(\varphi), y_{\mathrm{N}}(\varphi); \mathrm{S}_l) = e^{v(\varphi)^{-1}(C_{\max}(x(\varphi), y(\varphi); \mathrm{S}_l) - C_{\max}(x_{\mathrm{N}}(\varphi), y_{\mathrm{N}}(\varphi); \mathrm{S}_l))}$. Pseudokod algorytmu **HS** podano poniżej.

Algorytm **Hybrid** (**HS**)

Require: J, M, L, $\tilde{\mathrm{L}}$, p, Γ_{max}, ϱ
Ensure: x_{HS}, y_{HS}

1. Przypisz $\varphi \coloneqq 1$, oblicz $\tilde{V}_0(r^{\mathrm{SL}}, p, \mathrm{L}, \tilde{\mathrm{L}})$, wygeneruj losowo początkowe rozwiązanie $y(\varphi)$ oraz wyznacz macierz $r(y(\varphi))$
2. Wyznacz rozwiązanie początkowe $x(\varphi) \coloneqq$ **GD**$(\mathrm{J}, \mathrm{M}, r(y(\varphi)), p)$
3. **while** $\Gamma_{max} > t$ **or** $v(\varphi) > 0$ **or** $\varrho > 0$
4. $\forall_{\tilde{l}_g \in \tilde{\mathrm{L}} \wedge y_{g,i}(\varphi)=1}$ wyznacz $l \coloneqq \arg \min_{\tilde{l}_i \in \tilde{\mathrm{L}} \wedge y_{t,i}(\varphi) \neq 1, t=1,2,\ldots,w} d(\tilde{l}_g, \tilde{l}_i)$, wygeneruj rozwiązanie sąsiednie i umieść je w zbiorze $\mathrm{Y_N}(\varphi)$.
5. $\forall_{y_{\mathrm{N}}(\varphi) \in \mathrm{Y_N}(\varphi)}$ uruchom $x_{\mathrm{N}}(\varphi) \coloneqq$ **GD**$(\mathrm{J}, \mathrm{M}, r(y_{\mathrm{N}}(\varphi)), p)$ i wybierz parę $(x_{\mathrm{N}}(\varphi), y_{\mathrm{N}}(\varphi)) = \arg \min_{x'_{\mathrm{N}}, y'_{\mathrm{N}} \in \Omega_{\mathrm{SA}}(x(\varphi))} C_{\max}(x_{\mathrm{N}}(\varphi), y_{\mathrm{N}}(\varphi); \mathrm{S}_l)$
6. **if** $C_{\max}(x(\varphi), y(\varphi); \mathrm{S}_l) > C_{\max}(x_{\mathrm{N}}(\varphi), y_{\mathrm{N}}(\varphi); \mathrm{S}_l)$ **or** $\mathrm{random}[0,1) < \tilde{f}(x(\varphi), x_{\mathrm{N}}(\varphi), y(\varphi), y_{\mathrm{N}}(\varphi); \mathrm{S}_l)$ **then** $x(\varphi) \coloneqq x_{\mathrm{N}}(\varphi)$ i $y(\varphi) \coloneqq y_{\mathrm{N}}(\varphi)$ **end if**
7. Zaktualizuj wskaźnik czasu t, wskaźnik poprawy ϱ oraz indeks iteracji $\varphi \coloneqq \varphi + 1$.
8. **end while (3)**
9. $x_{\mathrm{HS}} \coloneqq x(\varphi)$, $y_{\mathrm{HS}} \coloneqq y(\varphi)$

Złożoność generowania rozwiązania początkowego w Liniach 1-2 zależy od czasowej złożoności obliczeniowej wybranego generatora losowego GEN i wynosi $\mathcal{O}(\max\{\mathrm{GEN}wm, n^2m\})$, ponieważ losowo wskazujemy decyzje w co najwyżej wm elementach macierzy $y(\varphi = 1)$ oraz dla wygenerowanej macierzy $y(\varphi = 1)$ uruchamiany jest algorytm **GD**. Wyliczenie odległości pomiędzy współrzędnymi lokalizacji wybranymi dla m maszyn a pozostałymi $w - m$ współrzędnymi niewybranych lokalizacji wymaga $\mathcal{O}\big(m(w - m)\big)$ (Linia 4, obliczenie odległości jest stałe czasowo). Uruchomienie algorytmu **GD** dla m nowych rozmieszczeń maszyn wymaga $\mathcal{O}(n^2m^2)$ (Linia 5). Operacje porównania, przypisania oraz modyfikacji wskaźników t, φ i ϱ są stałe czasowo (Linie 6-7). Ostatecznie czasowa złożoność obliczeniowa algorytmu **HS** wynosi $\mathcal{O}(v_{\max}n^2m^2)$, gdzie $v_{\max}$ jest największą możliwą liczbą iteracji pętli głównej przy rozważanych strategiach zmiany temperatury. Złożoność pamięciowa $\mathcal{O}\big((m + 1)(wm + n^2m)\big)$ jest zdeterminowana rozmiarem zbioru m rozwiązań sąsiednich oraz rozmiarem reprezentacji rozwiązania.

4.6.2. Algorytm Genetic (podejście łączne)

Zaprojektowanie algorytmu genetycznego (ewolucyjnego) wymaga zakodowania rozwiązania w postaci wektora lub macierzy, które podlegają modyfikacjom realizowanym przez odpowiednie operatory. Zbiór

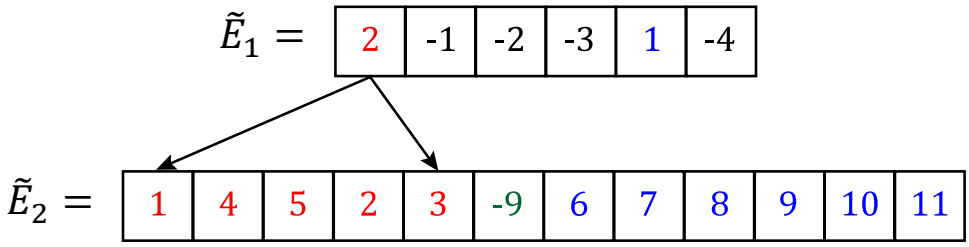
$$\tilde{E} = \{\tilde{E}_1, \tilde{E}_2\} = \{[e_1, e_2, \dots, e_w], [e_{w+1}, \dots, e_{n+m-1}]\}, \tag{4.18}$$

reprezentujący decyzje w x i y jest podzielony na dwie części (rysunek 4.4):

1. Składowe wektora $\tilde{E}_1$ opisują rozmieszczenie maszyn. W wektorze $\tilde{E}_1$ umieszczone są liczby dodatnie odpowiadające indeksom maszyn w zbiorze M oraz arbitralnie przyjęte liczby ujemne. Jeżeli $e_g > 0$, $g = 1,2,\dots,w$, to maszyna o indeksie $e_g \in \mathrm{M}$ zostaje umieszczona w lokalizacji $\tilde{l}_g$. Ujemna wartość oznacza brak wyboru tej lokalizacji.
2. Składowe wektora $\tilde{E}_2$ umożliwiają wskazanie sekwencji zadań realizowanych na każdej z m maszyn. Wektor $\tilde{E}_2$ zawiera $n + m - 1$ elementów, z których n liczb określa indeksy zadań a $m - 1$ arbitralnie przyjętych różnych ujemnych liczb (nazywanych separatorami) wskazuje sekwencje zadań na maszynach. Sekwencja zadań przypisana do maszyny i-tej znajduje się w wektorze $\tilde{E}_2$ pomiędzy separatorami o indeksach $i - 1$ i i. Do wskazania sekwencji zadań przypisanych

do maszyny ulokowanej na pozycji o indeksie $g = 1$ lub $g = m$ wystarczy jeden separator jak pokazano na rysunku 4.4.

Na rysunku 4.4 przedstawiono kodowanie rozwiązania w $\tilde{E}$. Numery indeksów wektora $\tilde{E}_1$, w których umieszczono liczby dodatnie wskazują indeks lokalizacji (wartość umieszczona pod konkretnym indeksem oznacza indeks maszyny). Do maszyny o indeksie 2 umieszczonej na pozycji o indeksie 1 w $\tilde{E}_1$ przypisujemy sekwencję (1,4,5,2,3) umieszczoną w $\tilde{E}_2$. Wartość -9 jest separatorem.



Rysunek 4.4. Kodowanie rozwiązania w algorytmie genetycznym dla M = {1,2}, $m = 2$, $w = 6$ oraz J = {1,2, …,11}, $n = 11$

Zaletą opracowanej metody kodowania rozwiązania jest jego elastyczność. Użycie ujemnych liczba dla niewybranych pozycji w $\tilde{E}_1$ oraz separatorów w $\tilde{E}_2$ umożliwia stosowanie dowolnych operatorów krzyżowania/mutacji dedykowanych wektorom zawierającym różne (nie powtarzające się) wartości liczbowe. Elastyczność uzyskano kosztem wydajności pamięciowej, ponieważ ujemne wartości zwiększając rozmiary wektorów w $\tilde{E}$. Operatory krzyżowania oraz mutacji są używane niezależnie dla $\tilde{E}_1$, $\tilde{E}_2$. Proces selekcji jest realizowany metodą Stochastic Universal Sampling opracowaną przez Baker'a (1987), a do rekombinacji zastosowano operator Ordered Crossover (OX1) opracowany przez Davis'a (1985). Parametrami algorytmu są: liczba osobników ω_1 w populacji, prawdopodobieństwo krzyżowania chromosomów ω_2, liczba osobników w pamięci długoterminowej ω_3 oraz prawdopodobieństwo mutacji chromosomu ω_4.

Algorytm **Genetic (GN)**

Require: J, M, L, $\tilde{\mathrm{L}}$, p, e, $\omega_1, \omega_2, \omega_3, \omega_4, \varrho, \Gamma_{max}$
Ensure: $x_{\mathrm{GN}}, y_{\mathrm{GN}}$

1. Przypisz $\varphi := 1$, wygeneruj początkową populację losowych rozwiązań i umieść je w zbiorze $\Upsilon(\varphi)$, $|\Upsilon(\varphi)| = \omega_1$ (najlepsze rozwiązanie w zbiorze to $y(\varphi)$, $x(\varphi)$), $\mathrm{B}(\varphi) = \emptyset$.
2. **while** $\Gamma_{max} > t$ **or** $\varrho > 0$
3. Wyselekcjonuj najlepsze rozwiązania wykorzystując metodę Stochastic Universal Sampling.

4. Wygeneruj nowe rozwiązania wykorzystując operator Ordered Crossover (OX1) do rekombinacji par w $\mathrm{B}(\varphi) \cup \Upsilon(\varphi)$ z prawdopodobieństwem ω_3.
5. Zastosuj operator mutacji z prawdopodobieństwem ω_4 dla każdego elementu w $\Upsilon(\varphi)$. Uaktualnij zbiór $\Upsilon(\varphi)$ o nowe rozwiązania.
6. Wyznacz $\left(x_{\mathrm{N}}(\varphi), y_{\mathrm{N}}(\varphi)\right) \coloneqq \arg \min_{(x'(\varphi),y'(\varphi))\in \Upsilon(\varphi)\cup \mathrm{B}(\varphi)} C_{\max}(x'(\varphi), y'(\varphi); \mathrm{S}_l)$ oraz umieść w $\mathrm{B}(\varphi)$ co najwyżej ω_3 najlepszych wyników, $|\mathrm{B}(\varphi)| = \omega_3$.
7. **if** $C_{\max}(x(\varphi), y(\varphi); \mathrm{S}_l) > C_{\max}(x_{\mathrm{N}}(\varphi), y_{\mathrm{N}}(\varphi); \mathrm{S}_l)$ **then** $x(\varphi) \coloneqq x_{\mathrm{N}}(\varphi)$ i $y(\varphi) \coloneqq y_{\mathrm{N}}(\varphi)$ **end if**
8. Zaktualizuj wskaźnik czasu t, wskaźnik poprawy ϱ oraz indeks iteracji $\varphi \coloneqq \varphi + 1$.
9. **end while (2)**
10. $x_{\mathrm{GN}} \coloneqq x_{\mathrm{N}}(\varphi)$, $y_{\mathrm{GN}} \coloneqq y_{\mathrm{N}}(\varphi)$

Złożoność generowania zbioru ω losowych rozwiązań początkowych zależy od wybranego generatora losowego GEN i wynosi $\mathcal{O}(\omega_1(n^2m + wm)\mathrm{GEN})$. Selekcję rozwiązań metodą Stochastic Universal Sampling można zrealizować w czasie $\mathcal{O}\big(\omega_1(w + n + m - 1)\big)$. Proces krzyżowania dwóch chromosomów operatorem OX1 wymaga liniowego czasu, więc krzyżowanie każdej pary wymaga $\mathcal{O}\big(\mathrm{GEN}\,(\omega_1 + \omega_3)(w + n + m - 1)\big)^2$. Ewaluacja jakości rozwiązań w Linii 6 wymaga sprawdzenia każdej pary rozwiązań w czasie $\mathcal{O}\big((\omega_1 + \omega_3)^2(wm + n^2m)\big)$. Zastosowanie operatora mutacji w Linii 5 oraz modyfikacja wskaźników t, φ i ϱ w Linii 7 są stałe czasowo. Ostatecznie czasowa złożoność obliczeniowa algorytmu **GN** wynosi $\mathcal{O}\big((\omega_1 + \omega_3)^2(wm + n^2m)\big)$. Złożoność pamięciowa $\mathcal{O}\big((\omega_1 + \omega_3)^2(wm + n^2m)\big)$ jest konsekwencją rozmiaru zbioru $\Upsilon(\varphi) \cup \mathrm{B}(\varphi)$ oraz rozmiaru reprezentacji rozwiązania.

4.6.3. Algorytm S_Annealing_Loc (podejście sekwencyjne)

Algorytm **S_Annealing_Loc** (**SA_L**), opracowany dla podproblemu wyboru lokalizacji dla maszyn, jest zmodyfikowaną wersją algorytmu **HS**. Współrzędne lokalizacji wyznaczone przez algorytm **SA_L** są wykorzystywane w drugim etapie podejścia sekwencyjnego. Rozwiązania sąsiednie są generowane zgodnie z (4.16) a jakość rozmieszczenia jest oceniana przez pryzmat (4.9) lub (4.10). Temperatura początkowa jest równa sumie odległości $\sum_{\tilde{l}_g\in\tilde{\mathrm{L}}}\sum_{l_j\in\mathrm{L}} d\big(\tilde{l}_g, l_j\big)$, co stanowi górne ograniczenie dla wartości obu rozważanych kryteriów rozmieszczenia maszyn (4.9) i (4.10). Funkcja energii ma postać $\hat{f}(y(\varphi), \tilde{y}, c_r) = e^{v(\varphi)^{-1}\left(V^{(c_r)}(y(\varphi)) - V^{(c_r)}(\tilde{y})\right)}$. Rozwiązania początkowe wyznaczono algorytmami opracowanymi przez Dyer & Frieze (1985) (dla podproblemu m-center) oraz Resende & Werneck (2004) (dla podproblemu m-median).

Algorytm **Simulated Annealing_Loc** (**SA_L**)

Require: $\tilde{\mathrm{L}}, \Gamma_{max}, \varrho, c_r \in \{\mathrm{center}, \mathrm{median}\}$
Ensure: $\tilde{y}_{c_r}$

1. Przypisz $\varphi \coloneqq 1$, oblicz temperaturę początkową ze wzoru $\sum_{\tilde{l}_g \in \tilde{\mathrm{L}}} \sum_{l_j \in \mathrm{L}} d(\tilde{l}_g, l_j)$ (nieefektywne rozwiązanie) oraz wyznacz rozwiązanie początkowe $y(\varphi)$.
2. **while** $\Gamma_{max} > t$ **or** $v(\varphi) > 0$ **or** $\varrho > 0$
3. $\underset{\tilde{l}_g \in \tilde{\mathrm{L}} \wedge y_{g,i}(\varphi)=1}{\forall}$ wyznacz $l = \arg \min_{\tilde{l}_i \in \tilde{\mathrm{L}} \wedge y_{t,i}(\varphi) \neq 1, t=1,2,\ldots,w} d(\tilde{l}_g, \tilde{l}_i)$, wygeneruj rozwiązanie sąsiednie i umieść je w zbiorze $\mathrm{Y_N}(\varphi)$.
4. Wybierz macierz $\tilde{y} = \arg \min_{y_{\mathrm{N}}(\varphi) \in \mathrm{Y_N}(\varphi)} V^{(c_r)}(y_{\mathrm{N}}(\varphi))$.
5. **if** $V^{(c_r)}(y(\varphi)) > V^{(c_r)}(\tilde{y})$ **or** random[0,1) $< f(y(\varphi), \tilde{y}, c_r)$ **then** $\mathbf{y}(\varphi) \coloneqq \tilde{y}$ **end if**
6. Zaktualizuj wskaźnik czasu t, wskaźnik poprawy ϱ oraz indeks iteracji $\varphi \coloneqq \varphi + 1$.
7. **end while (2)**
8. $y_{c_r} \coloneqq y(\varphi)$

Czasowa złożoność obliczeniowa algorytmu **SA_L** wynosi $\mathcal{O}(v_{\max} w m^2)$, ponieważ w każdej iteracji pętli while wartości kryterium, którego wyliczenie wymaga czasu $\mathcal{O}(wm)$, jest wyliczana dla m rozwiązań sąsiednich. Złożoność pamięciowa $\mathcal{O}((m+1)wm)$ jest konsekwencją rozmiaru zbioru m rozwiązań sąsiednich oraz rozmiarem uszeregowania.

4.7. Ocena eksperymentalna i statystyczna opracowanych podejść

W niniejszym podrozdziale przeprowadzono następujące badania w celu eksperymentalnej oceny opracowanych algorytmów, a w szczególności porównania rozpatrywanych podejść do rozwiązywania badanego zagadnienia ScheLoc:

1. **Badanie 4.1**: Eksperymentalne porównanie jakości uszeregowań zwróconych przez algorytm hybrydowy **HS** i algorytm genetyczny **GN** (podejście łączne).
2. **Badanie 4.2**: Porównanie podejścia łącznego z podejściem sekwencyjnym dla różnych kryteriów rozmieszczenia maszyn.
3. **Badanie 4.3**: Porównanie podejść łącznego i niepewnego.

Badanie 4.1

Współrzędne lokalizacji w $\tilde{\mathrm{L}}$ i L wylosowano wewnątrz obszaru w kształcie kwadratu o boku $\gamma = 1000$, wykorzystując rozkład jednostajny, oraz przyjęto, że dolny lewy róg kwadratu jest umieszony w punkcie (0,0). Termin gotowości $r_{g,j}$ jest równy odległości w dwuwymiarowej przestrzeni Euklidesowej pomiędzy współrzędnymi $\tilde{l}_g$ i l_j. Badania przeprowadzono dla czasów przetwarzania zadań wygenerowanych na potrzeby badania 2.2.

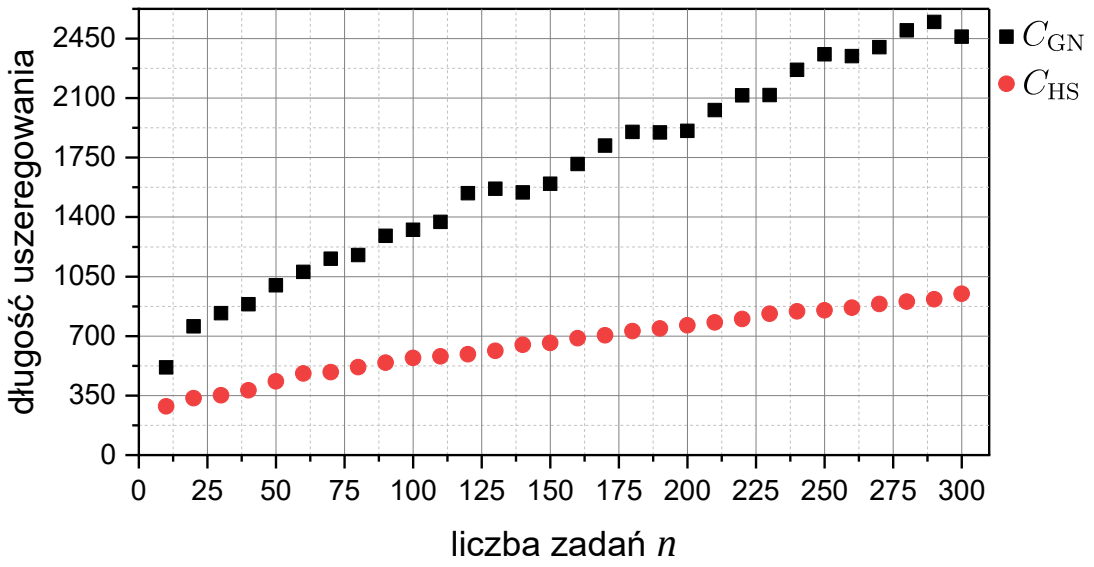
Algorytmy **HS** i **GN** uruchamiano 25 razy dla każdej instancji problemu oraz przyjęto, że liczba iteracji bez poprawy rozwiązania jest równa $\varrho = 20$ i czas obliczeń ograniczono do $\Gamma_{max} = 100$ [s]. Nie strojono parametrów algorytmu **HS**, ponieważ w każdej iteracji algorytm **SA** generuje m rozwiązań sąsiednich, z których każde różni się tylko jedną decyzją o rozmieszczeniu maszyny. Przyjęto funkcję zmiany temperatury $v(\varphi) = \tilde{V}_0\left(r^{\mathrm{SL}}, p, \mathrm{L}, \tilde{\mathrm{L}}\right)\alpha^{\varphi}$, gdzie $\alpha = 0{,}9$. Przyjęto oznaczenia $C_{\max}(x_{\mathrm{GN}}, y_{\mathrm{GN}}; \mathrm{S}_l) \triangleq C_{\mathrm{GN}}$, $C_{\max}(x_{\mathrm{HS}}, y_{\mathrm{HS}}; \mathrm{S}_l) \triangleq C_{\mathrm{HS}}$.

Strojenie algorytmu **GN** przeprowadzono metodą Random Search (losowego przeglądu parametrów algorytmu) dla arbitralnie przyjętych instancji. Strojeniu poddano:

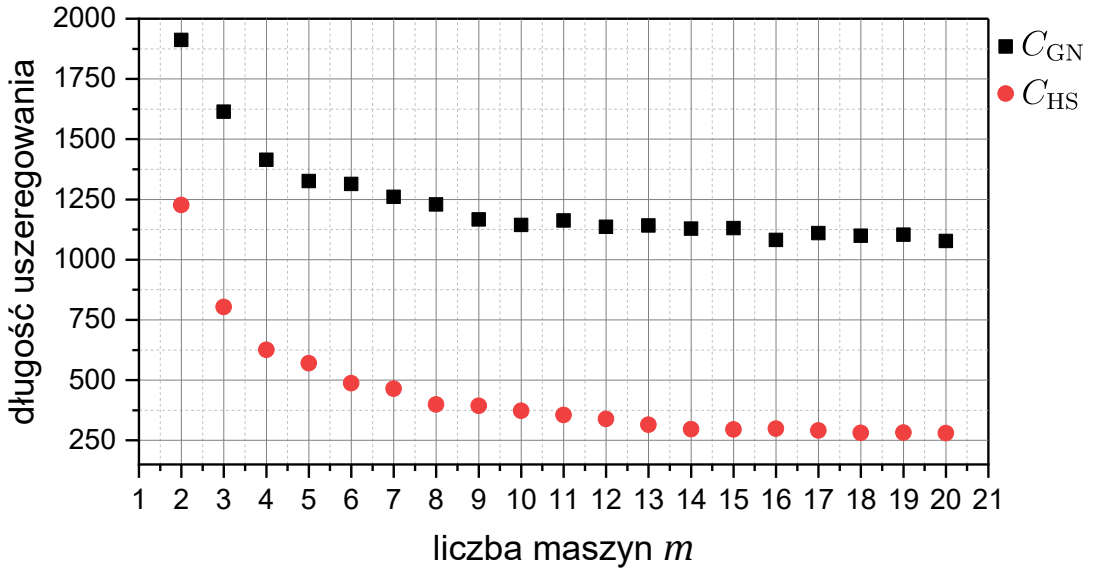
- rozmiar populacji początkowej $\omega_1 \in \left\{\left\lceil\frac{n}{0{,}5m}\right\rceil, \left\lceil\frac{\boldsymbol{n}}{\boldsymbol{0{,}4m}}\right\rceil, \left\lceil\frac{n}{0{,}3m}\right\rceil, \left\lceil\frac{n}{0{,}2m}\right\rceil, \left\lceil\frac{n}{0{,}1m}\right\rceil\right\}$,
- prawdopodobieństwo krzyżowania chromosomów $\omega_2 \in \{0{,}65;\ \boldsymbol{0{,}7};\ 0{,}75; 0{,}8; 0{,}85\}$,
- liczba osobników w pamięci długoterminowej $\omega_3 \in \{\lceil\boldsymbol{0{,}02\omega_1}\rceil, \lceil 0{,}05\omega_1\rceil, \lceil 0{,}08\omega_1\rceil\}$.

Prawdopodobieństwo mutacji nie podlegało strojeniu i jest równe $\omega_4 = 0{,}05$. Parametry algorytmu genetycznego dobierano uruchamiając algorytm 10 razy dla każdej instancji w $n^{(z)} \times m^{(z)} \times w^{(z)}$, gdzie $n^{(z)} = \{[0{,}50], [51{,}100], [101{,}150], [151{,}200]\}$, $m^{(z)} \in \{[2{,}8], [9{,}15], [16{,}20]\}$, $w^{(z)} = \{[m+1, m+5], [m+6, m+10], [m+11, m+15]\}$ oraz czasy przetwarzania zadań wylosowano, z wykorzystaniem rozkładu równomiernego, z przedziału $\left[\underline{p}, \bar{p}\right]$, $\underline{p} = 5$, $\bar{p} \in \{6{,}7, \ldots, 100\}$. Wybrano parametry ω_1, ω_2, ω_3, dla których algorytm **GN** najczęściej zwracał najlepsze rozwiązanie. Wybrane parametry oznaczono pogrubioną czcionką.

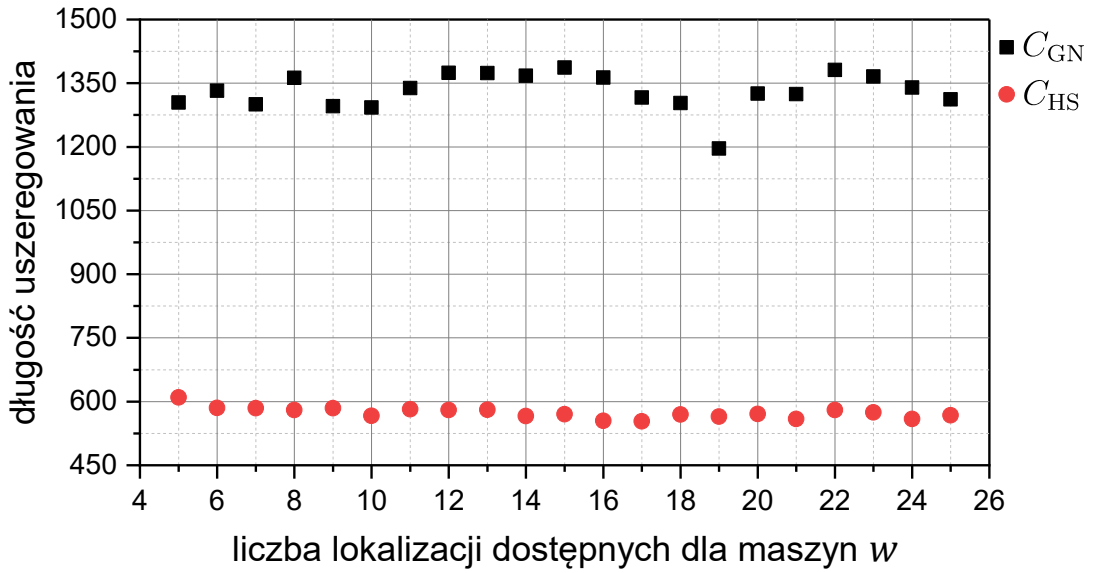
Do badań, których rezultaty zaprezentowano na rysunkach 4.5-4.8, instancje wygenerowano w taki sposób, że z największej wygenerowanej instancji problemu $m = 20$, $n = 200$, $w = 25$ oraz $\left[\underline{p} = 5, \bar{p} = 100\right]$ usuwano losowo wymaganą ilość danych, aby utworzyć mniejsze instancje. Rozbieżność rezultatów uzyskanych przez algorytmy stanowi uzasadnienie formy reprezentacji wyników na wykresach. Na rysunkach 4.5-4.7 zaprezentowano najmniejsze długości uszeregowania jakie zostały uzyskane przez opracowane algorytmy, ponieważ przedziały wartości osiąganych przez obie heurystyki były rozłączne w przypadku każdej rozważanej instancji problemu. Wartości liczbowe podano w dodatku w tabelach D1-D3.



Rysunek 4.5. Wykres zależności długości uszeregowania od liczby zadań n dla $m = 5, w = 20$

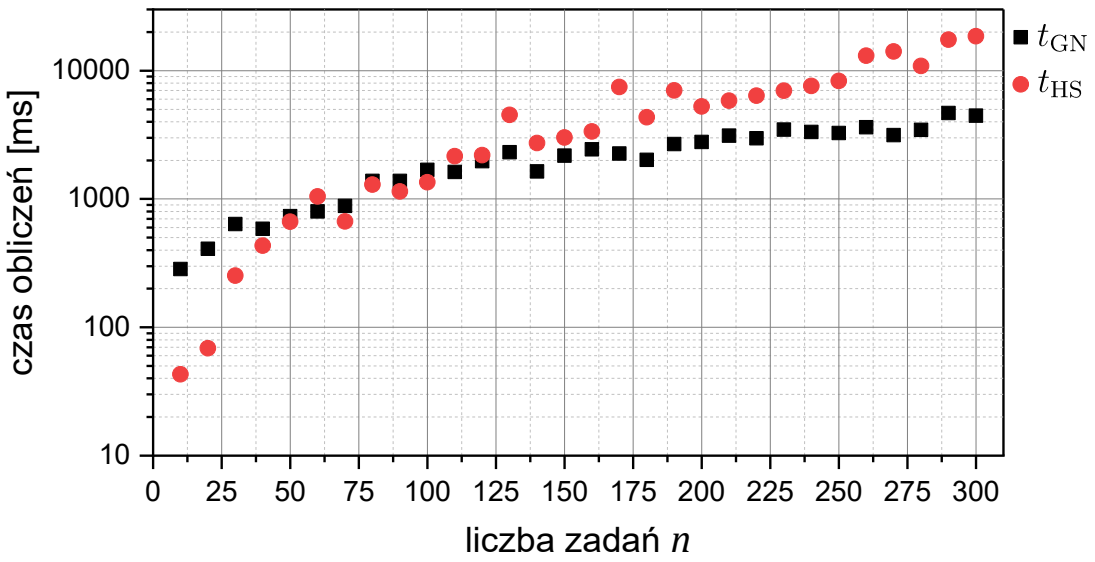


Rysunek 4.6. Wykres zależności długości uszeregowania od liczby maszyn m dla $m = 5, w = 20$



Rysunek 4.7. Wykres zależności długości uszeregowania od liczby dostępnych lokalizacji dla maszyn w dla $n = 100, m = 5$

Algorytm **HS** gwarantował mniejszą wartość funkcji celu niż algorytm **GN** dla wszystkich instancji ujętych na rysunkach 4.5-4.7. Przewaga podejścia hybrydowego wynikała efektywności algorytmu **GD**, którego uszeregowanie zapewniało mniejszą wartość kryterium również, gdy decyzje o rozmieszczeniu w y były tożsame w **HS** i **GN**. Algorytm **GN** pomimo prób intensyfikacji (krzyżowania jedynie fragmentów chromosomu) i dywersyfikacji (zwiększania rozmiaru populacji), w procesie eksploracji przestrzeni rozwiązań, nie był wstanie uzyskać rezultatów lepszych niż podejście hybrydowe (nawet, gdy elementem populacji początkowej było rozwiązanie uzyskane przez algorytm **HS**).



Rysunek 4.8. Wykres zależności median czasów obliczeń od liczby zadań n dla $m = 5$, $w = 20$. Wartości na osi y podano w skali logarytmicznej (log10)

Analizując dane zaprezentowane na rysunku 4.8 (wartości liczbowe podano tabeli D4) można zaobserwować zwiększanie się różnicy w czasach obliczeń pomiędzy **HS** i **GN** wraz ze wzrostem liczby zadań. Charakterystyczny wzrost czasu obliczeń realizowanych przez algorytm **HS** był spowodowany koniecznością weryfikacji każdego z m sąsiednich rozwiązań poprzez uruchamianie algorytmu **GD** dla coraz większej liczby zadań. Wzrost liczby zadań powodował duży rozrzut czasów obliczeń realizowanych przez algorytm **HS**. Jest to efekt modyfikacji, w każdej iteracji algorytmu, tylko jednej decyzji o wyborze lokalizacji maszyny w oparciu o kryterium odległości (zastępowanie wybranej lokalizacji najbliżej położoną niewybraną lokalizacją). Z tego powodu algorytm wykazywał tendencje do wyboru ograniczonego podzbioru rozmieszczeń, czego efektem było przedwczesne kończenie obliczeń. Rozwiązaniem tego problemu był losowy wybór wszystkich lokalizacji w przypadku ponownego uruchomienia algorytmu **HS**.

W kolejnej części badań wygenerowano losowo 100 instancji problemu. Każdą instancję opisano krotką $(m, n, \gamma, \bar{p})$, gdzie $m \in \{1,2,\dots,10\}$, $w \in \{m+1, m+2, \dots, 30\}$, $n \in \{10,11,\dots,100\}$, $\underline{p} = 5$, $\bar{p} \in \{6,7,\dots,100\}$ oraz długość boku kwadratu $\gamma \in \{100,101,\dots,1000\}$, wewnątrz którego rozmieszczane są zadania i dopuszczalne lokalizacje dla maszyn (lewy dolny róg kwadratu znajduje się w punkcie (0,0)). Testy statystyczne pominięto, ponieważ algorytm **HS** zwracał uszeregowania zapewniające mniejszą długość uszeregowania dla każdej instancji problemu.

Badanie 4.2

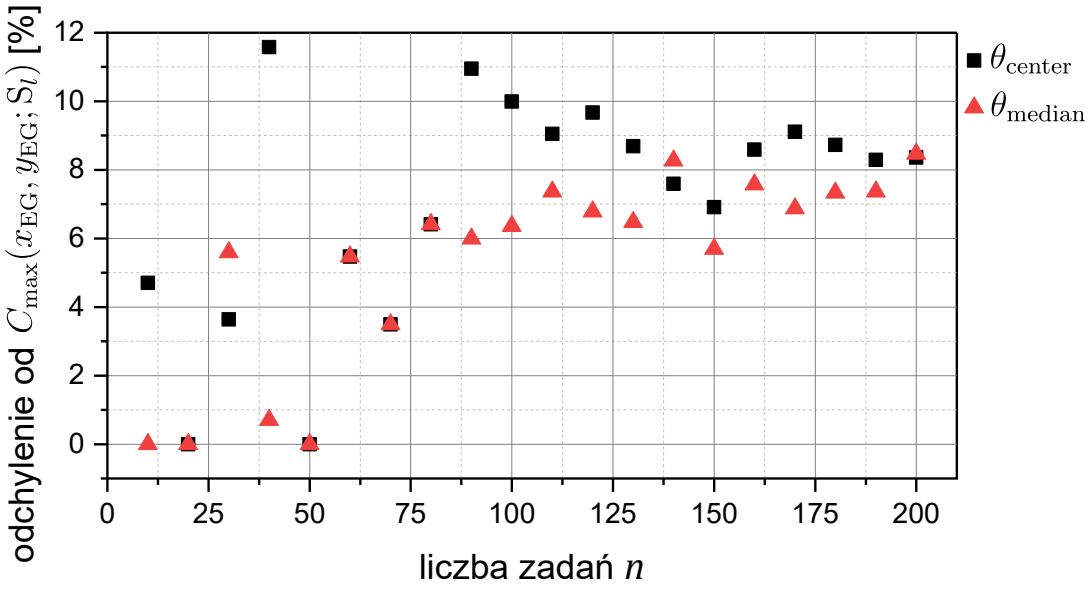
Do weryfikacji jakości dwuetapowej optymalizacji w problemie ScheLoc wykorzystano algorytmy **LOC** $\in$ {**EX**,**SA_L**}, **HS**, **GD** oraz **EG**. Algorytm **EX** zastosowano do znalezienia optymalnych macierzy rozwiązania problemów (4.9) i (4.10) dla małych instancji problemu, gdy których $w \leq 20$. Natomiast algorytm **SA_L** zastosowano dla przypadków $w \leq 100$.

Pierwszą część ewaluacji jakości rozwiązań przeprowadzono wykorzystując zbiór danych wskazany w badaniu 4.1 oraz algorytmy **LOC** $=$ **EX**, **GD** i **EG**. Aby porównać rezultaty optymalizacji sekwencyjnej i łącznej, zdefiniowano następujący wskaźnik jakości

$$\theta_t = \frac{C_{\max}(x_{\mathrm{GD}}, y_t; \mathrm{S}_l) - C_{\max}(x_{\mathrm{EG}}, y_{\mathrm{EG}}; \mathrm{S}_l)}{C_{\max}(x_{\mathrm{EG}}, y_{\mathrm{EG}}; \mathrm{S}_l)} 100\%, \quad t \in \{\mathrm{center}, \mathrm{median}\}, \tag{4.19}$$

gdzie y_{center} jest optymalnym rozwiązaniem problemu (4.9), które uzyskano realizując przegląd zupełny algorytmem **EX** (analogicznie uzyskano macierz y_{median} dla problemu

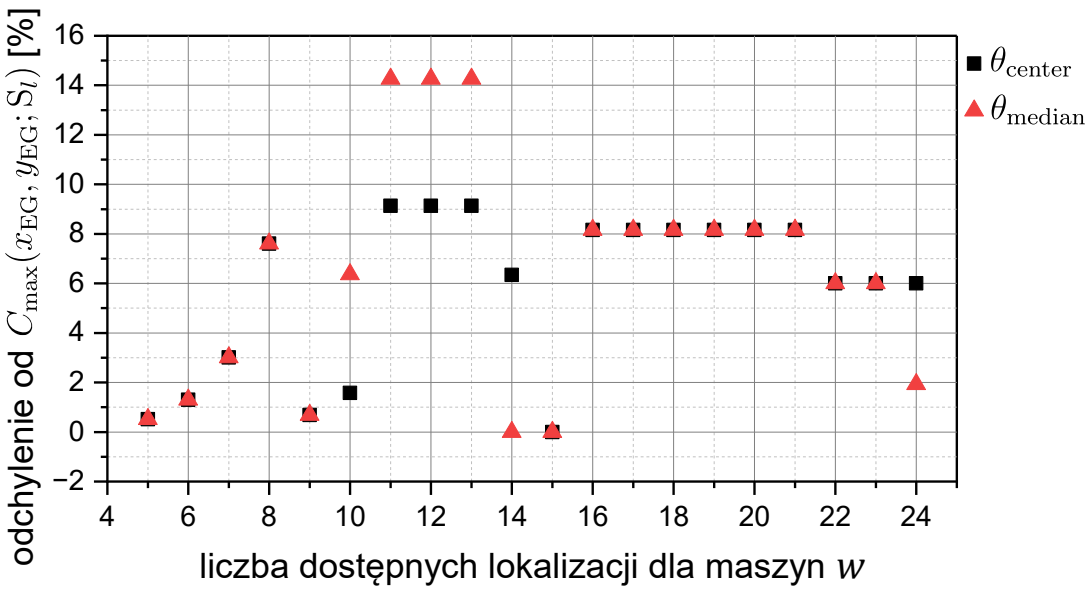
(4.10)). Rozmiary instancji problemów zostały ograniczone ($m \leq 8,\ n \leq 200,\ w \leq 20$) celem wykonania obliczeń algorytmem **EX** w akceptowalnym czasie. Wartości liczbowe podano w tabelach D5-D6 (podrozdział D w dodatku).



Rysunek 4.9. Wykres zależności θ_{center} i θ_{median} od liczby zadań n dla $m = 4$ i $w = 20$

Na rysunku 4.9 widać stabilizowanie się odchyleń θ_{center} i θ_{median} wraz ze wzrostem liczby zadań. Jest to efekt kompensacji nieoptymalnego, z punktu widzenia problemu ScheLoc, wyboru lokalizacji dla maszyn przez zachłanną strategię szeregowania zadań. Wpływ kryteriów rozmieszczenia maszyn na jakość rozwiązań problemu ScheLoc jest szczególnie widoczny dla małych instancji $n \leq 50$. Zgodnie z kryterium dla podproblemu m-center maszyny są rozmieszczane w taki sposób, aby zminimalizować najpóźniejszy termin gotowości jednego zadania wybrany spośród najwcześniejszych terminów gotowości, co skutkowało ignorowaniem pozostałych współrzędnych. Przewaga podejścia z zastosowanym podproblemem m-median wynikała z uwzględniania najbliżej położonych współrzędnych lokalizacji m zadań zamiast jednej najbardziej oddalonej lokalizacji. Ostatecznie takie podejście umożliwiało zrównoważenie obciążenia pracy maszyn.

Niezerowe wartości odchyleń θ_{center} i θ_{median} zaprezentowane na rysunku 4.10 są w większości przypadków zbieżne. Dodawanie, w sposób losowy (zgodnie z rozkładem równomiernym), kolejnych współrzędnych lokalizacji potwierdzało ten trend. Rozbieżność w wynikach jest konsekwencją innego rozmieszczenia maszyn w przypadku, gdy zbiór $\tilde{L}$ zawierał współrzędne odstające. Rozmieszczenia, dla których wyniki optymalizacji nie były tożsame różniły się jedną decyzją.



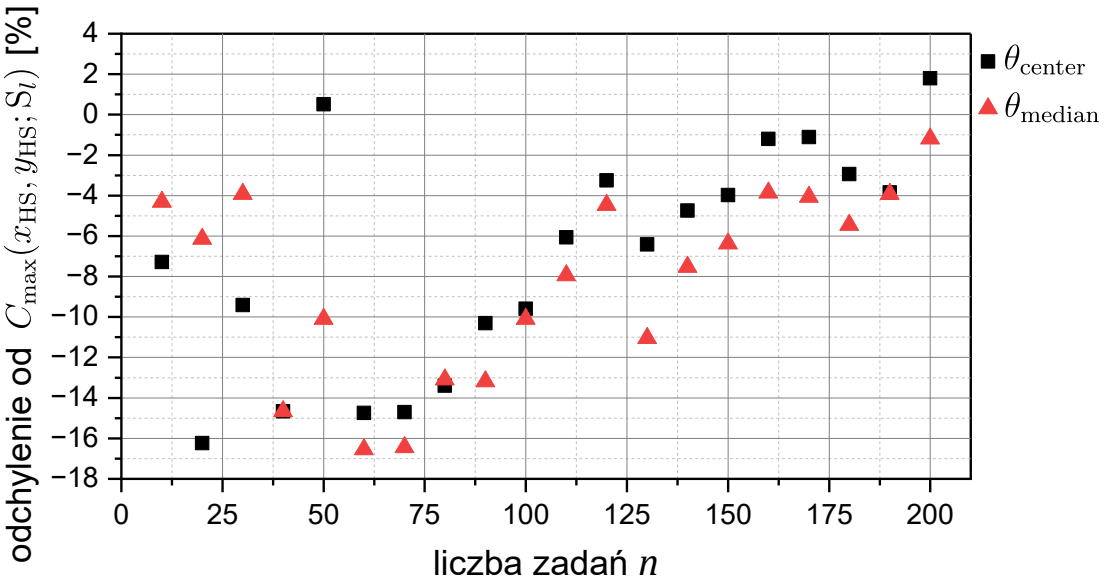
Rysunek 4.10. Wykres zależności θ_{center} i θ_{median} od liczby dostępnych lokalizacji dla maszyn w dla $n = 50$ i $m = 4$

Tabela 4.2. Rezultaty optymalizacji dla instancji $n = 50, w = 20$

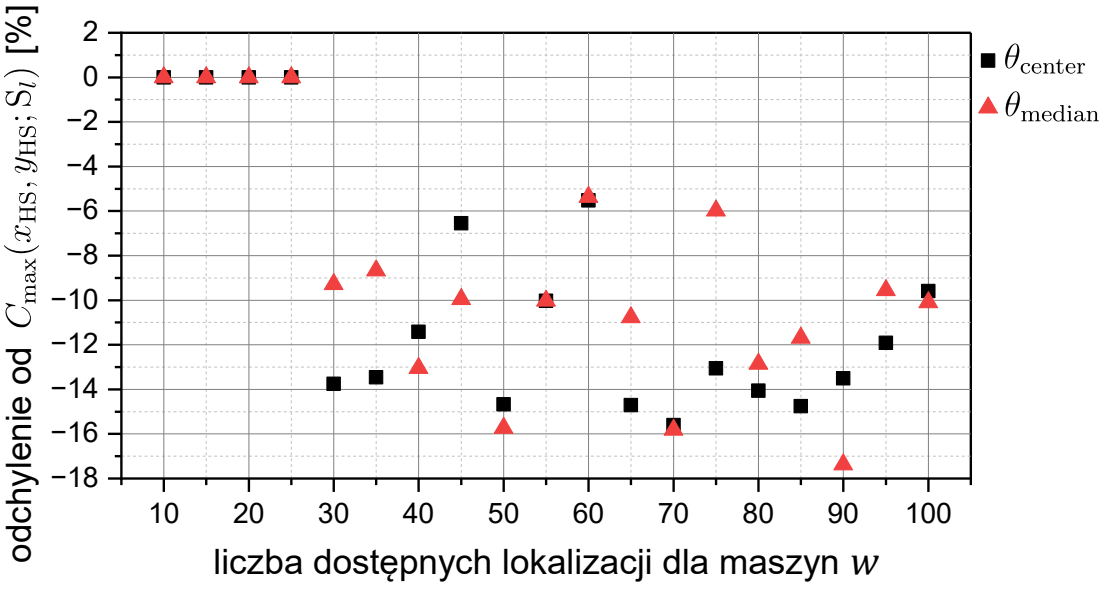
m	$C_{\max}(x_{\mathrm{GD}}, y_{\mathrm{center}})$	$C_{\max}(x_{\mathrm{GD}}, y_{\mathrm{median}})$	$C_{\max}(x_{\mathrm{EG}}, y_{\mathrm{EG}})$	θ_{center} [%]	θ_{median} [%]
2	800,34	839,29	790,48	1,25	6,18
3	655,34	686,64	573,01	14,37	19,83
4	501,80	501,80	501,80	0,00	0,00
5	430,32	455,54	407,34	5,64	11,83
6	377,83	365,30	365,30	3,43	0,00
7	345,81	345,81	345,81	0,00	0,00
8	331,23	331,23	331,23	0,00	0,00

Zwiększanie liczby maszyn, przy ustalonej liczbie w, naturalnie prowadziło do zrównania jakości optymalizacji sekwencyjnej i łącznej, ponieważ zmniejsza się liczba potencjalnych decyzji, które mogą powodować odchylenie od najbardziej efektywnego rozmieszczenia przy założeniu kryterium długości uszeregowania w problemie ScheLoc.

W drugiej części badań zwiększono liczbę dostępnych lokalizacji $w \leq 100$, aby wymusić stosowanie heurystyk. Badania przeprowadzono wykorzystując zbiór danych wskazany w punkcie badanie 4.1 (dodatkowo wygenerowano nowe współrzędne dostępnych lokalizacji dla maszyn) oraz wykorzystano algorytmy **LOC = SA_L**, **GD** i **HS**. Zdefiniowano wskaźnik oceny $\bar{\theta}_t = \frac{C_{\max}(x_{\mathrm{GD}}, \tilde{y}_t; \mathrm{S}_l) - C_{\max}(x_{\mathrm{HS}}, y_{\mathrm{HS}}; \mathrm{S}_l)}{C_{\max}(x_{\mathrm{HS}}, y_{\mathrm{HS}}; \mathrm{S}_l)} 100\%$, $t \in \{\mathrm{center}, \mathrm{median}\}$.



Rysunek 4.11. Wykres zależności $\bar{\theta}_{\mathrm{center}}$ i $\bar{\theta}_{\mathrm{median}}$ od liczby zadań n dla $m = 6$ i $w = 100$



Rysunek 4.12. Wykres zależności $\bar{\theta}_{\mathrm{center}}$ i $\bar{\theta}_{\mathrm{median}}$ od liczby dostępnych lokalizacji dla maszyn w dla $n = 50$ i $m = 6$

Tabela 4.3. Rezultaty optymalizacji dla instancji $n = 50, w = 100$

m	$C_{\max}(x_{\mathrm{GD}}, \tilde{y}_{\mathrm{center}})$	$C_{\max}(x_{\mathrm{GD}}, \tilde{y}_{\mathrm{median}})$	$C_{\max}(x_{\mathrm{HS}}, y_{\mathrm{HS}})$	$\bar{\theta}_{\mathrm{center}}$ [%]	$\bar{\theta}_{\mathrm{median}}$ [%]
2	747,02	820,03	740,99	0,81	10,67
3	541,33	577,47	568,75	-4,82	1,53
4	411,97	411,84	437,69	-5,88	-5,91
5	373,10	360,02	407,22	-8,38	-11,59
6	319,20	336,20	376,40	-15,20	-10,68
7	296,45	294,76	342,40	-13,42	-13,92
8	268,42	268,42	288,42	-6,93	-6,93
9	235,53	232,43	235,53	0,00	-1,32
10	213,45	209,88	219,98	-2,97	-4,59

Na rysunkach 4.11-4.12 (wartości liczbowe podano w dodatku w tabelach D7 i D9) oraz w tabeli 4.3 zaprezentowano wyniki, które wskazują, że podejście sekwencyjne może zapewniać lepsze rezultaty optymalizacji niż podejście łączne dla dużych instancji. Przewaga podejścia sekwencyjnego wynikała z przeglądu zupełnego możliwych przypisań sekwencji do wybranych, podczas pierwszego etapu podejmowania decyzji, lokalizacji rozmieszczenia maszyn. Efektywne rezultaty optymalizacji sekwencyjnej były obciążone znacznym wzrostem czasu obliczeń w porównaniu do podejścia łącznego. Przykładowe czasy obliczeń zaprezentowano w dodatku w tabeli D8.

W następującej obserwacji wskazano podzbiór instancji problemu ScheLoc, dla których podejścia łączne i sekwencyjne (dwuetapowe) w optymalizacji gwarantują optymalne rozwiązanie problemu ScheLoc.

Obserwacja 4.1. *Jeżeli elementy dwóch rozłącznych podzbiorów* $\tilde{\mathrm{L}}_1 \cap \tilde{\mathrm{L}}_2 = \emptyset$, $\tilde{\mathrm{L}} = \tilde{\mathrm{L}}_1 \cup \tilde{\mathrm{L}}_2$, $\left|\tilde{\mathrm{L}}_2\right| = m$, *spełniają nierówność*

$$\underset{l \in \mathrm{L} \wedge \tilde{l}_1 \in \tilde{\mathrm{L}}_1 \wedge \tilde{l}_2 \in \tilde{\mathrm{L}}_2}{\forall} d(\tilde{l}_1, l) > d(\tilde{l}_2, l), \tag{4.20}$$

to rezultaty zarówno optymalizacji łącznej jak i sekwencyjnej, uwzględniając (4.9) *lub* (4.10) *jako podproblem rozmieszczenia maszyn, zapewniają optymalne rozwiązanie problemu ScheLoc.*

Dowód. Warunek (4.20) oznacza, że m najwcześniejszych możliwych terminów gotowości dla dowolnego zadania jest zagwarantowane, gdy współrzędne rozmieszczenia m maszyn są dane zbiorem $\tilde{\mathrm{L}}_2$, co stanowi optymalne rozmieszczenie maszyn w problemie ScheLoc. Wówczas, przy wyborze współrzędnych w $\tilde{\mathrm{L}}_2$, suma najwcześniejszych terminów gotowości wszystkich zadań (kryterium dla podproblemu m-median) jest najmniejsza oraz najpóźniejszy termin gotowości jednego zadania wybrany spośród najwcześniejszych

terminów gotowości wszystkich zadań (kryterium dla podproblemu m-center) jest najwcześniejszy. Q.E.D.

Następnie przeprowadzono badania statystyczne, aby rozstrzygnąć, który podproblem rozmieszczenia maszyn gwarantuje statystycznie mniejszą długość uszeregowania dla wygenerowanych instancji w sekwencyjnym podejściu do rozwiązania problemu ScheLoc.

Do badań statystycznych wygenerowano 100 instancji problemu. Obliczenia realizowano algorytmami **EX+GD** lub **SA_L+GD**. W obu przypadkach stosowano podejście sekwencyjne. Jeżeli czas obliczeń realizowanych algorytmem **EX** przekraczał 10 minut, to dla prodproblemu rozmieszczenia uruchamiano algorytm **SA_L**. Wykorzystano takie same parametry metaheurystyki jak w przypadku algorytmu **HS**. Wybierano najlepszy uzyskany rezultat z dwudziestu pięciu uruchomień algorytmu dla każdej instancji. Wyniki zaprezentowano w tabeli D10 (podrozdział D w dodatku).

Podczas pierwszego etapu badań statystycznych, do weryfikacji hipotezy mówiącej o tym, że rozkład danych jest zbliżony do rozkładu normalnego wykorzystano test Shapiro-Wilka (Shapiro & Wilk, 1965). Wyniki testów przeprowadzonych dla poziomu istotności $\alpha = 0.05$:

- wartość statystyki W=0,8602 oraz p-value = 2,9e-08 dla $C_{\max}(x, y_{\text{center}}; \mathrm{S}_l)$,
- wartość statystyki W=0,8620 oraz p-value = 3,4e-11 dla $C_{\max}(x, y_{\text{median}}; \mathrm{S}_l)$,

pozwalają dla każdego zbioru odrzucić hipotezę zerowa. Następnie przeprowadzono test Wilcoxona dla par (Wilcoxon, 1945), stosując konserwatywne podejście dla par o równej wartości (19% par), dla następujących hipotez statystycznych:

$$\begin{aligned} H_0&: C_{\max}(x, y_{\text{center}}; \mathrm{S}_l) = C_{\max}(x, y_{\text{median}}; \mathrm{S}_l), \\ H_1&: C_{\max}(x, y_{\text{center}}; \mathrm{S}_l) > C_{\max}(x, y_{\text{median}}; \mathrm{S}_l), \end{aligned} \tag{4.21}$$

gdzie wynikami są $z_{val} = 2{,}406$ oraz p-value = 0,0081 (odrzucamy H_0 na rzecz H_1). Na podstawie badań statystycznych można stwierdzić, że rozmieszczenie maszyn w oparciu o podproblem m-median gwarantuje statystycznie mniejszą długość uszeregowania w problemie ScheLoc.

Rezultat badań statystycznych potwierdził prawdziwość tezy numer 3, którą sformułowano w podrozdziale 1.5.

Badanie 4.3

W pierwszej części badań wskazano, że stosując podejście niepewne w problemie ScheLoc można uzyskać optymalne rozwiązanie łącznego problemu ScheLoc.

Obserwacja 4.2. *Jeżeli przedziały niepewności zostały obliczone za pomocą* (4.11) *i jest spełniony warunek* (3.27), *to optymalne uszeregowanie* $\bar{x}^*_{\mathrm{SR}}$ *pozwala uzyskać optymalną długość uszeregowania w łącznym problemie rozmieszczenia maszyn i szeregowania zadań, czyli*

$$C_{\max}(x^*, y^*; \mathrm{S}_l) = C_{\max}(\bar{x}^*_{\mathrm{SR}}, y^*; \mathrm{S}_l). \tag{4.22}$$

Dowód. Niech dla dowolnego rozmieszczenia maszyn zachodzi

$$\sum_{\tilde{l}_g \in \tilde{\mathrm{L}}} \sum_{j \in \mathrm{J} \setminus \{t\}} \left(d(\tilde{l}_g, l_j) + \max_{i=1,2,\dots,m} p_{i,j} \right) \leq r_{i,t}, \qquad i = 1,2,\dots,m. \tag{4.23}$$

Wówczas algorytm **SR** (dla ustalonych przedziałów $\bar{u}_j$, $j = 1,2,\dots,m$) przypisze, w rozwiązaniu optymalnym $\bar{x}^*_{\mathrm{SR}}$, zadanie t na ostatniej pozycji w sekwencji ustanowionej na maszynie umożliwiającej jego najwcześniejsze zakończenie. Jest to konsekwencja spełnienia warunku (3.27). Algorytm **GD** również przypisze zadanie t na ostatniej pozycji w rozwiązaniu optymalnym (dla dowolnej macierzy y), ponieważ spełnienie nierówności (4.23) jest równoznaczne ze spełnieniem (2.13). W rezultacie optymalne uszeregowanie $\bar{x}^*_{\mathrm{SR}}$ jest również optymalnym uszeregowaniem w problemie ScheLoc. Q.E.D. Powyższa obserwacja jest również spełniona dla nieoptymalnych rozwiązań $\tilde{x}_{\mathrm{DR}}$ i $\tilde{x}_{\mathrm{GR}}$.

Należy podkreślić fakt, że w przypadku jednej maszyny $m = 1$ uszeregowanie będące rezultatem zastosowania podejścia niedeterministycznego z przedziałowymi terminami gotowości zadań i kryterium maksymalnego żalu dla długości uszeregowania nie może zapewnić mniejszej długości uszeregowania niż rozwiązanie uzyskane dla podejścia łącznego (deterministycznego) w problemie ScheLoc. Optymalne rozwiązanie łącznego problemu ScheLoc z jedną maszyną i kryterium długości uszeregowania sprowadza się do wyboru lokalizacji zapewniającej najmniejszą długość uszeregowania po przypisaniu sekwencji zadań zgodnie z metodą ERD. Podejście niedeterministyczne wymusza podejmowanie decyzji w oparciu o scenariusze skrajne, co skutkuje wymuszeniem opóźnienia terminu gotowości, równym długości przedziału niepewności (przyjmowana jest górna granica przedziału niepewności) co najmniej jednego zadania w procesie podejmowania decyzji. Opóźnienie może powodować zmianę kolejności gotowości zadań w porównaniu

do najwcześniejszych możliwych terminów gotowości, co może jedynie pogarszać jakość rozwiązania w porównaniu do metody ERD.

W drugiej części badań przeprowadzono ewaluację jakości uszeregowań uzyskanych przez algorytmy opracowane dla problemów niedeterministycznych, przy założeniu (4.11) lub (4.12), w deterministycznym (łącznym) problemie ScheLoc. Aby porównać rezultaty optymalizacji, zdefiniowano następujący wskaźnik jakości

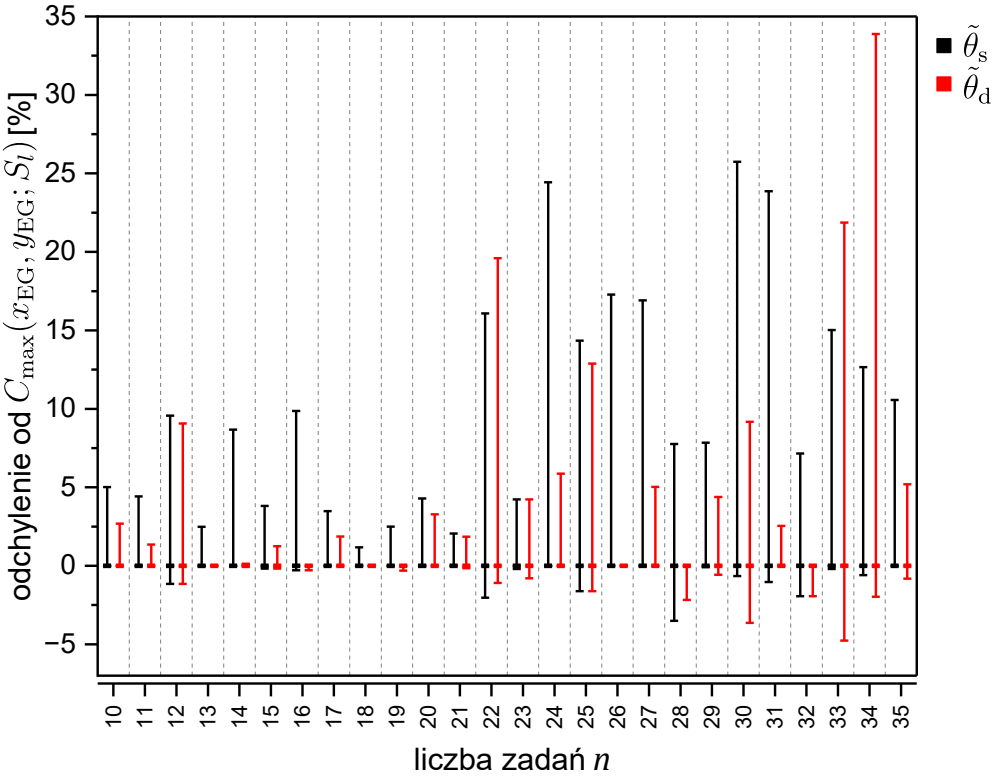
$$\tilde{\theta}_t = \frac{C_{\max}(x_t, y_{\mathrm{EX}}; \mathrm{S}_l) - C_{\max}(x_{\mathrm{EG}}, y_{\mathrm{EG}}; \mathrm{S}_l)}{C_{\max}(x_{\mathrm{EG}}, y_{\mathrm{EG}}; \mathrm{S}_l)} 100\%, \quad t \in \{\mathrm{d}, \mathrm{s}\}, \tag{4.24}$$

gdzie d ∈ {**DR**,**GR**}, s ∈ {**DR**,**GR**,**SR**} są akronimami algorytmów, d jest zbiorem akronimów algorytmów rozwiązania problemu $Rm|r_{i,j}^{-} \le r_{i,j} \le r_{i,j}^{+}|Reg(C_{\max})$ oraz s jest zbiorem akronimów algorytmów rozwiązania problemu $Rm|r_j^{-} \le r_j \le r_j^{+}|Reg(C_{\max})$. Ujemna wartość (4.24) oznacza, że algorytm **DR**, **GR** lub **SR** zwrócił harmonogram, który zapewnił długość uszeregowania mniejszą niż $C_{\max}(x_{\mathrm{EG}}, y_{\mathrm{EG}}; \mathrm{S}_l)$.

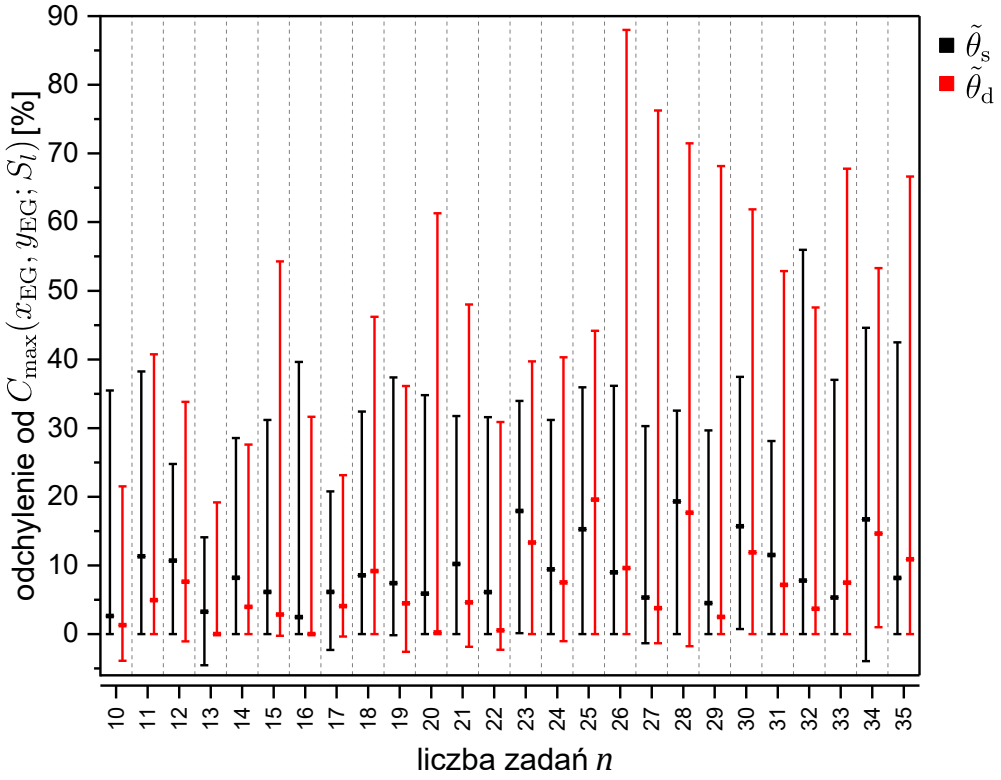
Podczas generowania współrzędnych w L i $\tilde{\mathrm{L}}$ nałożono geometryczne ograniczenia na obszary, w których mogą być rozmieszczone dostępne lokalizacje dla maszyn, aby zweryfikować wpływ długości przedziałów niepewności na rezultaty optymalizacji w problemie ScheLoc. Współrzędne dane zbiorami $\tilde{\mathrm{L}}$ i L wylosowano wewnątrz dwóch obszarów w kształcie kwadratu. Lewy dolny róg obu kwadratów był umieszczony w punkcie (0,0). Współrzędne rozmieszczenia zadań generowano wewnątrz obszaru o boku $\gamma_k^{(\mathrm{L})}$ oraz współrzędne rozmieszczenia dostępnych lokalizacji dla maszyn generowano wewnątrz obszaru o boku $\gamma_k^{(\tilde{\mathrm{L}})}, k = 1,2,3$. Rozważono następujące przypadki: $I_1: \gamma_1^{(\mathrm{L})} = 1000$, $\gamma_1^{(\tilde{\mathrm{L}})} = 100$; $I_2: \gamma_2^{(\mathrm{L})} = 1000$, $\gamma_2^{(\tilde{\mathrm{L}})} = 500$; $I_3: \gamma_3^{(\mathrm{L})} = 1000$, $\gamma_3^{(\tilde{\mathrm{L}})} = 1000$. W badaniach nie wykorzystywano algorytmów opartych na metaheurystykach, aby ograniczyć błąd związany z wyborem losowych i nieefektywnych decyzji. Dla każdej pary $\gamma_k^{(\mathrm{L})}$, $\gamma_k^{(\tilde{\mathrm{L}})}$ i ustalonych wartości n, m i w wygenerowano 50 instancji (dla każdej generowano losowo elementy zbiorów $\tilde{\mathrm{L}}$ i L oraz wartości w macierzy p z zakresów podanych w badaniu 2.2. Rezultaty badań zaprezentowano na rysunkach 4.13-4.15 (wartości liczbowe podano w tabeli D11 w dodatku). Każdy wynik na wykresie (dla ustalonego n, m i w) reprezentuje maksymalne odchylenie od $C_{\max}(x_{\mathrm{EG}}, y_{\mathrm{EG}}; \mathrm{S}_l)$ (górna pozioma linia), minimalne odchylenie od $C_{\max}(x_{\mathrm{EG}}, y_{\mathrm{EG}}; \mathrm{S}_l)$ (dolna pozioma linia) i medianę odchyleń (pogrubiona pozioma linia wewnątrz słupka) dla rozwiązań 50 losowo wygenerowanych instancji. W każdym

przypadku wybierano najlepszą uzyskaną wartość odchylenia, tzn. $\tilde{\theta}_{\mathrm{s}} = \min\{\tilde{\theta}_{\mathrm{DR}}, \tilde{\theta}_{\mathrm{GR}}, \tilde{\theta}_{\mathrm{SR}}\}$, $\tilde{\theta}_{\mathrm{d}} = \min\{\tilde{\theta}_{\mathrm{DR}}, \tilde{\theta}_{\mathrm{GR}}\}$.

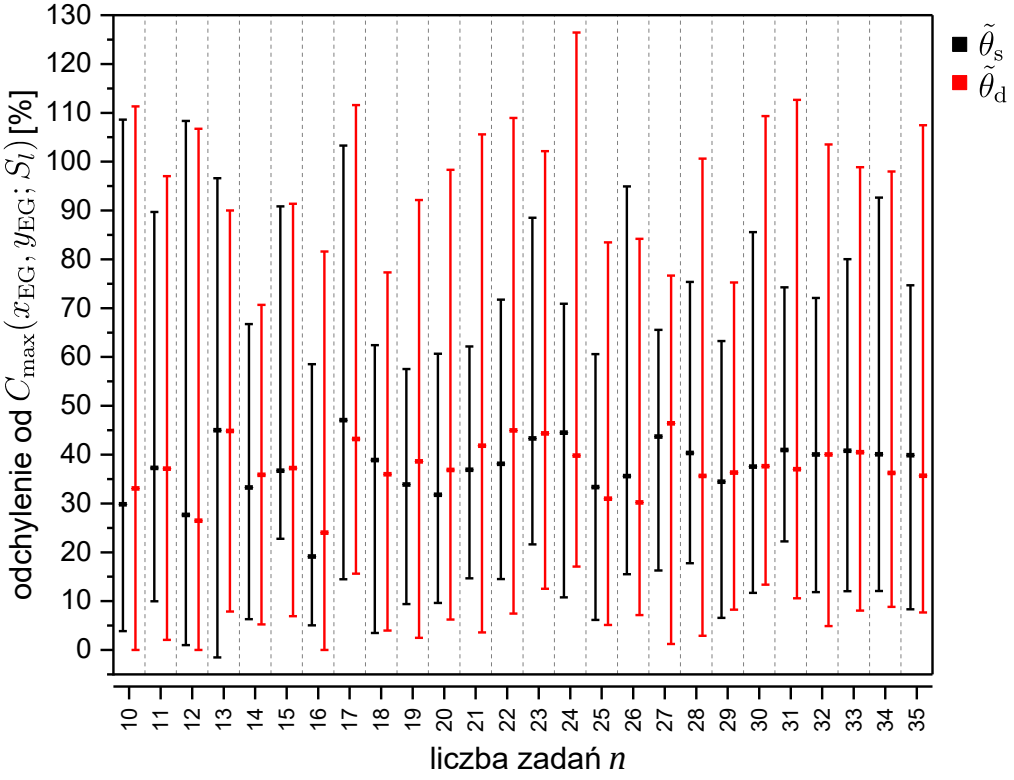
W badaniach rozważono $m = 2$ oraz $w = 30$. Znaczna dysproporcja między liczbą maszyn i liczbą dostępnych lokalizacji do ich rozmieszczenia jest intencjonalna, ponieważ wewnątrz każdego przedziału niepewności pożądana jest jak największa liczba terminów gotowości, które są reprezentowane (niejawnie) przez dostępne lokalizacje dla maszyn. Celem nałożenia ograniczeń na rozmieszczanie współrzędnych lokalizacji dostępnych dla maszyn w I_k, $k = 1,2,3$, jest zmiana długości przedziałów niepewności, tzn. przedziały niepewności obliczone według (4.11) lub (4.12) przy ograniczeniu $\gamma_1^{(\tilde{\mathrm{L}})} = 100$ były krótsze niż przy $\gamma_3^{(\tilde{\mathrm{L}})} = 1000$. W podejściu opartym na (4.12) stosowano dwie metody obliczania punktu referencyjnego (4.13) i (4.14). Wybierano uszeregowania gwarantujące mniejszą długość uszeregowania.



Rysunek 4.13. Wykres zależności $\tilde{\theta}_{\mathrm{s}}$ i $\tilde{\theta}_{\mathrm{d}}$ od liczby zadań n dla $w = 30$, $m = 2$, $\gamma_1^{(\mathrm{L})} = 1000$, $\gamma_1^{(\tilde{\mathrm{L}})} = 100$



Rysunek 4.14. Wykres zależności $\tilde{\theta}_{\mathrm{s}}$ i $\tilde{\theta}_{\mathrm{d}}$ od liczby zadań n dla $w = 30$, $m = 2$, $\gamma_2^{(\mathrm{L})} = 1000$, $\gamma_2^{(\tilde{\mathrm{L}})} = 500$



Rysunek 4.15. Wykres zależności $\tilde{\theta}_{\mathrm{s}}$ i $\tilde{\theta}_{\mathrm{d}}$ od liczby zadań n dla $w = 30$, $m = 2$, $\gamma_3^{(\mathrm{L})} = 1000$, $\gamma_3^{(\tilde{\mathrm{L}})} = 1000$

Rezultaty optymalizacji zaprezentowane na rysunkach 4.13-4.15 wskazują na zależność wartości $\tilde{\theta}_{\mathrm{s}}$ i $\tilde{\theta}_{\mathrm{d}}$ od długości przedziałów niepewności. W przypadku krótkich przedziałów niepewności (rysunek 4.13) mediany odchyleń $\tilde{\theta}_{\mathrm{s}}$ i $\tilde{\theta}_{\mathrm{d}}$ były równe zero dla każdej rozważanej instancji problemu i obydwu metod definiowania niepewności. Był to rezultat pokrycia przedziałów niepewności czasami przetwarzania zadań w początkowym etapie optymalizacji. W konsekwencji, po początkowych przypisaniach zadań, scenariusze skrajne powiązane z kolejnymi zadaniami nie wpływały na wartość kryterium i algorytmy **DR**, **GR** i **SR** szeregowały te zadania, stosując strategię zachłanną, jak w problemie deterministycznym.

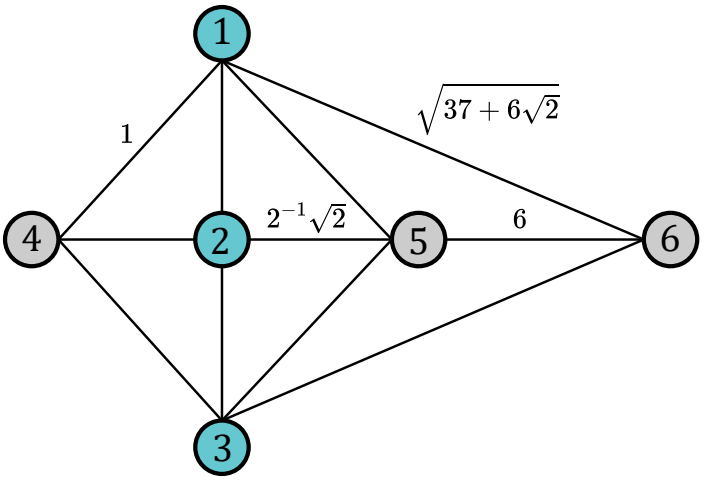
Jakość uszeregowań uzyskanych przez algorytmy **DR**, **GR** i **SR** pogarszała się, z punktu widzenia kryterium problemu ScheLoc, wraz z wydłużaniem się przedziałów niepewności (rysunki 4.14-4.15). Jednak istnieją instancje, dla których rozwiązania zwrócone przez algorytm **EG** są gorszej jakości niż rozwiązania zwrócone przez **DR**, **GR** i **SR** (np. rysunek 4.14, $n = 27$, gdzie algorytmy zwróciły różne harmonogramy zapewniające taką samą długość uszeregowania). Wówczas wymuszanie opóźnień terminów gotowości podzbiorów zadań, poprzez podejmowanie decyzji w oparciu o scenariusze skrajne w obydwu niedeterministycznych problemach szeregowania, poprawiało wartość kryterium długości szeregowania, ponieważ nieefektywne decyzje zachłanne, podejmowane przez algorytm **EG** w deterministycznym problemie ScheLoc, były odkładane w czasie.

Algorytmy **DR** nie szeregował zadań efektywnie, gdy zachodziła znacząca dysproporcja (powyżej 40%) między długościami przedziałów niepewności na różnych maszynach dla tych samych zadań. Algorytm zwracał uszeregowania, zgodnie z którymi maszyny były nierównomiernie obciążone, co zwiększało długość uszeregowania. Dopuszczenie modyfikacji harmonogramów w taki sposób, że możliwa była zamiana sekwencji zadań między maszynami (bez modyfikacji kolejności w sekwencji) nie poprawiało rezultatów optymalizacji.

Sumaryczną statystykę rezultatów optymalizacji w podejściu niepewnym opisano w postaci „**ALGORYTM**(procent instancji, dla których algorytm zwrócił najlepsze rozwiązanie)”. Najlepsze rozwiązanie rozumiemy jako najmniejszą znalezioną długość uszeregowania w problemie ScheLoc. Rezultaty są następujące: $\tilde{\theta}_{\mathrm{s}}$: **DR** (86%), **GR** (91%), **SR** (93%) oraz $\tilde{\theta}_{\mathrm{d}}$: **DR** (84%), **GR** (81%). Różnica pomiędzy jakością rezultatów uzyskanych dla (4.13) i (4.14) nie była istotna statystycznie.

Przeprowadzone badania, porównujące podejścia łączne i niepewne, uzupełniono przykładem obliczeniowym, w którym podejście niepewne zapewniło lepsze wyniki. Rozważono oba przypadki określania przedziałów niepewności, tzn. (4.11) i (4.12), dla których odpowiednio wykorzystano algorytmy **SR** i **DR**. Zaproponowane instancje są nietrywialne w tym sensie, że przedziały niepewności nie są rozłączne. Przykłady obliczeniowe wskazują jak odkładanie zachłannych decyzji w czasie, wynikające z zastosowania podejścia niepewnego, poprawia jakość uszeregowania.

Na rysunku 4.16 przedstawiono graf, w którym długości krawędzi spełniają nierówność trójkąta. Lokalizacje zadań są dane wierzchołkami w kolorze niebieskim oraz dopuszczalne lokalizacje dla maszyn są dane wierzchołkami w kolorze szarym. Wierzchołki grafu o numerach 1,5,3,4 pokrywają się z wierzchołkami kwadratu o boku 1. Wierzchołek 2 leży na przecięciu przekątnych kwadratu. Wierzchołki 1,2,5,6 oraz 3,2,5,6 konstruują trójkąty prostokątne o takich samych wymiarach, gdzie przeciwprostokątne 1,6 oraz 3,6 mają długość $\sqrt{\left(2^{-1}\sqrt{2}\right)^2 + \left(2^{-1}\sqrt{2}+6\right)^2} = \sqrt{37+6\sqrt{2}}$. W macierzy r^{SL} przyjęto, że pierwszy wiersz opisuje odległości pomiędzy wierzchołkiem grafu numer 4 a wierzchołkami 1 (pierwsza kolumna), 2 (druga kolumna), 3 (trzecia kolumna). W drugim i trzecim wierszu odległości są wyliczane analogicznie względem wierzchołka numer 6 i 5.



Rysunek 4.16. Parametry instancji problemu: $\tilde{\mathrm{L}} = \{4,5,6\}$, $w = 3$, $\mathrm{M} = \{1,2\}$, $m = 2$, $\mathrm{L} = \mathrm{J} = \{1,2,3\}$, $n = 3$, $p = \begin{bmatrix} 3 & 5 & 4 \\ 4 & 6 & 3 \end{bmatrix}$, $r^{\mathrm{SL}} = \begin{bmatrix} 1 & 2^{-1}\sqrt{2} & 1 \\ 1 & 2^{-1}\sqrt{2} & 1 \\ \sqrt{37+6\sqrt{2}} & 2^{-1}\sqrt{2}+6 & \sqrt{37+6\sqrt{2}} \end{bmatrix}$

Dla tak określonych danych rozważmy trzy przykłady obliczeniowe.

1) Łączny problem rozmieszczenia maszyn i szeregowania zadań – algorytm EG

Wszystkie możliwe rozmieszczenia maszyn $\mathrm{M} = \{1,2\}$ w obrębie wierzchołków grafu $\tilde{\mathrm{L}} = \{4,5,6\}$ przedstawiono w postaci dwuelementowych zbiorów decyzji: $\{(4,1),(5,2)\}$, $\{(4,2),(5,1)\}$, $\{(4,1),(6,2)\}$, $\{(4,2),(6,1)\}$, $\{(5,1),(6,2)\}$, $\{(5,2),(6,1)\}$. Zbiór $\{(4,1),(5,2)\}$ jest równoważny decyzjom $y_{4,1} = 1$ (maszyna 1 jest przypisana do wierzchołka 4) oraz $y_{5,2} = 1$. Uszeregowania zadań są wyznaczane za pomocą algorytmu **GD** dla każdej pary decyzji.

Optymalizacja w łącznym problemie ScheLoc rozpoczyna się od obliczeń realizowanych dla instancji problemu zaprezentowanej w tabeli 4.4. Liczba etapów optymalizacji jest równa liczbie zadań.

Tabela 4.4. Instancja problemu $Rm|r_{i,j}|C_{\max}$. Terminy gotowości zadań obliczono na podstawie zbioru $\{(4,1),(5,2)\}$ oraz czasy przetwarzania zadań

j	$y_{4,1} = 1$		$y_{5,2} = 1$	
1	$r_{1,1} = 1$	$p_{1,1} = 3$	$r_{2,1} = 1$	$p_{2,1} = 4$
2	$r_{1,2} = 2^{-1}\sqrt{2}$	$p_{1,2} = 5$	$r_{2,2} = 2^{-1}\sqrt{2}$	$p_{2,2} = 6$
3	$r_{1,3} = 1$	$p_{1,3} = 4$	$r_{2,3} = 1$	$p_{2,3} = 3$

Pierwszy etap optymalizacji $\boldsymbol{\eta = 1}$: algorytm **GD** wybiera zadanie i maszynę minimalizujące

$$\min\begin{Bmatrix}\boldsymbol{r_{1,1} + p_{1,1} = 1 + 3}, r_{2,1} + p_{2,1} = 1 + 4, r_{1,2} + p_{1,2} = 2^{-1}\sqrt{2} + 5,\\ r_{2,2} + p_{2,2} = 2^{-1}\sqrt{2} + 6, r_{1,3} + p_{1,3} = 1 + 4, \boldsymbol{r_{2,3} + p_{2,3} = 1 + 3}\end{Bmatrix} = \mathbf{4}.$$

Przypisanie zadania $j = 1$ do maszyny $i = 1$ oraz przypisanie zadania $j = 3$ do maszyny $i = 2$ są optymalnymi decyzjami z punktu widzenia strategii zachłannej. Decyzja o uszeregowaniu jest podejmowana na podstawie zawartości zbiorów $\mathrm{J}_{F_1}(\eta = 1) = \{1,2\}$ oraz $\mathrm{J}_{F_2}(\eta = 1) = \{3\}$. Wybrano zadanie $j = 3$, ponieważ $\left|\mathrm{J}_{F_1}(1)\right| > \left|\mathrm{J}_{F_2}(1)\right|$.

Zadanie $j = 3$ zostało przypisane do maszyny $i = 2$ (umieszczonej na wierzchołku numer 5) na pierwszej pozycji w sekwencji i $\boldsymbol{C_{2,1}(x(1); \mathrm{S}) = 4}$, $\boldsymbol{x_{2,1,3}(1) = 1}$.

Drugi etap optymalizacji $\boldsymbol{\eta = 2}$: algorytm **GD** wybiera zadanie i maszynę minimalizujące

$$\min\begin{Bmatrix}\boldsymbol{r_{1,1} + p_{1,1} = 1 + 3}, C_{2,1}(x(1); \mathrm{S}) + p_{2,1} = 4 + 4,\\ r_{1,2} + p_{1,3} = 2^{-1}\sqrt{2} + 5, C_{2,1}(x(1); \mathrm{S}) + p_{2,3} = 4 + 6\end{Bmatrix} = \mathbf{4}.$$

Zadanie $j = 1$ zostało przypisane do maszyny do maszyny $i = 1$ (umieszczonej na wierzchołku numer 4) na pierwszej pozycji w sekwencji i $\boldsymbol{C_{1,1}(x(2); S) = 4}$, $\boldsymbol{x_{1,1,1}(2) = 1}$.

Trzeci etap optymalizacji $\boldsymbol{\eta = 3}$: algorytm **GD** wybiera zadanie i maszynę minimalizujące

$$\min\{\boldsymbol{C_{1,1}(x(2); S) + p_{1,2} = 4 + 5}, C_{2,1}(x(2); S) + p_{2,2} = 4 + 6\} = \boldsymbol{9}.$$

Zadanie $j = 2$ zostało przypisane do maszyny do maszyny $i = 1$ (umieszczonej na wierzchołku numer 4) na drugiej pozycji w sekwencji i $\boldsymbol{C_{1,2}(x(3); S_l) = 9}$, $\boldsymbol{x_{1,2,2}(3) = 1}$.

Długość uszeregowania $\boldsymbol{C_{\max}(x_{GD}, y; S_l) = \max\{C_{1,2}(x_{GD}, y; S_l) = 9, C_{2,1}(x_{GD}, y; S_l) = 4\} = 9}$ dla decyzji $\boldsymbol{\{y_{4,1} = 1, y_{5,2} = 1\}, \{x_{1,1,1} = 1, x_{1,2,2} = 1, x_{2,1,3} = 1\}}$.

Szeregowanie zadań przy rozmieszczeniu $\{(4,2),(5,1)\}$ nie zmniejsza długości uszeregowania, ponieważ terminy gotowości zadań na maszynach pozostają takie same jak w przypadku $\{(4,1),(5,2)\}$. Z tego powodu pominięto szeregowanie zadań dla $\{(4,2),(5,1)\}$.

Wybór lokalizacji (wierzchołka grafu) numer 6 może jedynie pogarszać jakość rozwiązania, ponieważ przypisanie dowolnego zadania do maszyny umieszczonej w tej lokalizacji skutkuje długością uszeregowania przekraczającą 9. Przypisując dowolne zadanie do maszyny umieszczonej na wierzchołku o numerze 6 zachodzi

$$9 < \min\left\{\begin{array}{l} r_{1,1} + p_{1,1} = \sqrt{37 + 6\sqrt{2}} + 3, r_{1,2} + p_{1,2} = 2^{-1}\sqrt{2} + 6 + 5, \\ r_{1,3} + p_{1,3} = \sqrt{37 + 6\sqrt{2}} + 4, r_{2,1} + p_{2,1} = \sqrt{37 + 6\sqrt{2}} + 4, \\ r_{2,2} + p_{2,2} = 2^{-1}\sqrt{2} + 6 + 6, r_{2,3} + p_{2,3} = \sqrt{37 + 6\sqrt{2}} + 3 \end{array}\right\}.$$

Jeżeli (pomijając wierzchołek o numerze 6) wszystkie zadania zostaną przypisane do jednej maszyny to

$$9 < \min\begin{Bmatrix} r_{1,1} + p_{1,1} + p_{1,2} + p_{1,3} = 1 + 3 + 4 + 5 \\ r_{2,1} + p_{2,1} + p_{2,2} + p_{2,3} = 1 + 4 + 3 + 6 \end{Bmatrix}.$$

W rezultacie obliczenia dla rozmieszczeń maszyn $\{(4,1),(6,2)\}$, $\{(4,2),(6,1)\}$, $\{(5,1),(6,2)\}$, $\{(5,2),(6,1)\}$ można pominąć.

Rezultatem rozwiązania łącznego problemu rozmieszczenia maszyn i szeregowania zadań są decyzje $\boldsymbol{\{y_{4,1} = 1, y_{5,2} = 1\}}$, $\boldsymbol{\{x_{1,1,1} = 1, x_{1,2,2} = 1, x_{2,1,3} = 1\}}$ oraz długość uszeregowania $\boldsymbol{C_{\max}(x_{EG}, y_{EG}; S_l) = 9}$.

2) Problem rozmieszczenia maszyn i szeregowania zadań (podejście niepewne) – algorytm SR

Algorytm **SR** podejmuje decyzje w wariancie optymistycznym, tzn. w (3.24) wykorzystano scenariusz optymistyczny $\bar{u}^-$.

Tabela 4.5. Instancja problemu $Rm|r_j^- \leq r_j \leq r_j^+|Reg(C_{\max})$. Przedziałowe terminy gotowości zadań wyznaczone według wzoru (4.11) oraz czasy przetwarzania zadań

j	$i=1$			$i=2$		
1	$r_1^- = 1$	$r_1^+ = \sqrt{37+6\sqrt{2}}$	$p_{1,1} = 3$	$r_1^- = 1$	$r_1^+ = \sqrt{37+6\sqrt{2}}$	$p_{2,1} = 4$
2	$r_2^- = 2^{-1}\sqrt{2}$	$r_2^+ = 2^{-1}\sqrt{2}+6$	$p_{1,2} = 5$	$r_2^- = 2^{-1}\sqrt{2}$	$r_2^+ = 2^{-1}\sqrt{2}+6$	$p_{2,2} = 6$
3	$r_3^- = 1$	$r_3^+ = \sqrt{37+6\sqrt{2}}$	$p_{1,3} = 4$	$r_3^- = 1$	$r_3^+ = \sqrt{37+6\sqrt{2}}$	$p_{2,3} = 3$

Pierwszy etap optymalizacji $\boldsymbol{\eta = 1}$: Zachodzi równość $\left|\mathrm{D}_1^+(\overline{\mathrm{U}}, \mathrm{J}(1))\right| = \left|\mathrm{D}_2^+(\overline{\mathrm{U}}, \mathrm{J}(1))\right| = \left|\mathrm{D}_3^+(\overline{\mathrm{U}}, \mathrm{J}(1))\right| = 2$. O wyborze pierwszego zadania do uszeregowania decyduje najmniejsza oszacowana liczba jednostek czasu realizacji zadań, które mogą wykonać się wcześniej (po przypisaniu wybranego zadania). Dla zadania $j = 1$ oszacowanie jest równe $2^{-1}(3{,}5 + 5{,}5) = 4{,}5$, dla zadania $j = 2$ oszacowanie jest równe $2^{-1}(3{,}5 + 3{,}5) = 3{,}5$, dla zadania $j = 3$ oszacowanie jest równe $2^{-1}(3{,}5 + 5{,}5) = 4{,}5$.

Algorytm **SR** wybiera dla zadania $j = 2$ maszynę minimalizującą

$$\min\left\{\boldsymbol{r_2^- + p_{1,2} = 2^{-1}\sqrt{2} + 5}, r_2^- + p_{2,2} = 2^{-1}\sqrt{2} + 6\right\} = \boldsymbol{2^{-1}\sqrt{2} + 5}.$$

Zadanie $j = 2$ zostało przypisane do maszyny $i = 1$ na pierwszej pozycji w sekwencji i $\boldsymbol{C_{1,1}(\bar{x}(1), \bar{u}^-; p) = \sqrt{2}2^{-1} + 5}$, $\boldsymbol{\bar{x}_{1,1,2}(1) = 1}$.

Drugi etap optymalizacji $\boldsymbol{\eta = 2}$: zachodzi równość $\left|\mathrm{D}_1^+(\overline{\mathrm{U}}, \mathrm{J}(2))\right| = \left|\mathrm{D}_2^+(\overline{\mathrm{U}}, \mathrm{J}(2))\right| = 1$ oraz oszacowana liczba jednostek czasu realizacji zadań, które mogą wykonać się wcześniej jest równa $2^{-1}(3{,}5 + 4)$ dla $j = 1$ i $j = 3$. Decyzja o wyborze zadania do przypisania jest podejmowana arbitralnie.

Algorytm **SR** wybiera dla zadania $j = 1$ maszynę minimalizującą

$$\min\left\{C_{1,1}(\bar{x}(1), \bar{u}^-; p) + p_{1,1} = \sqrt{2}2^{-1} + 5 + 3, \boldsymbol{r_1^- + p_{2,1} = 1 + 4}\right\} = \boldsymbol{5}.$$

Zadanie $j = 1$ zostało przypisane do maszyny $i = 2$ na pierwszej pozycji w sekwencji i $\boldsymbol{C_{2,1}(\bar{x}(2), \bar{u}^-; p) = 5}$, $\boldsymbol{\bar{x}_{2,1,1}(2) = 1}$.

Trzeci etap optymalizacji $\boldsymbol{\eta = 3}$: zachodzi równość $\left|\mathrm{D}_2^+(\overline{\mathrm{U}}, \mathrm{J}(3))\right| = 0$.

Algorytm **SR** wybiera dla zadania $j = 3$ maszynę minimalizującą

$$\min \begin{Bmatrix} C_{1,1}(\bar{x}(2), \bar{u}^{-}; p) + p_{1,3} = \sqrt{2}2^{-1} + 5 + 4, \\ \boldsymbol{C_{2,1}(\bar{x}(2), \bar{u}^{-}; p) + p_{2,3} = 5 + 3} \end{Bmatrix} = \mathbf{8}.$$

Zadanie $j = 2$ zostało przypisane do maszyny $i = 2$ na drugiej pozycji w sekwencji i $\boldsymbol{C_{2,2}(\bar{x}(3), \bar{u}^{-}; p) = 8}$, $\boldsymbol{\bar{x}_{2,2,3}(3) = 1}$.

Uszeregowanie $\{\bar{x}_{1,1,2} = 1, \bar{x}_{2,1,1} = 1, \bar{x}_{2,2,3} = 1\}$ przy decyzjach $\{y_{4,2} = 1, y_{5,1} = 1\}$ (dobranych permutacyjnym generowaniem możliwych rozmieszczeń maszyn) gwarantuje wartość $C_{\max}(\bar{x}_{SR}, y_{EX}; S_l) = \max \begin{Bmatrix} C_{1,2}(\bar{x}_{SR}, y_{EX}; S_l) \\ = 2^{-1}\sqrt{2}5, C_{2,1}(\bar{x}_{SR}, y_{EX}; S_l) = 8 \end{Bmatrix} = 8$ w problemie ScheLoc, czyli długość uszeregowania mniejszą niż w podejściu łącznym, tzn. $C_{\max}(\bar{x}_{SR}, y_{EX}; S_l) < C_{\max}(x_{EG}, y_{EG}; S_l)$.

3) Problem rozmieszczenia maszyn i szeregowania zadań (podejście niepewne) – algorytm DR

W tabeli 4.6 przedziały niepewności obliczono przy założeniu, że punktem referencyjnym jest wierzchołek grafu o numerze 4.

Tabela 4.6. Instancja problemu $Rm|r_{i,j}^{-} \le r_{i,j} \le r_{i,j}^{+}|Reg(C_{\max})$. Przedziałowe terminy gotowości zadań wyznaczone według wzoru (4.12) i (4.14) oraz czasy przetwarzania zadań

j	$i = 1$			$i = 2$		
1	$r_{1,1}^{-} = 1$	$r_{1,1}^{+} = 1 + \sqrt{37 + 6\sqrt{2}}$	$p_{1,1} = 3$	$r_{2,1}^{-} = 1$	$r_{2,1}^{+} = 2$	$p_{2,1} = 4$
2	$r_{1,2}^{-} = 2^{-1}\sqrt{2}$	$r_{1,2}^{+} = \sqrt{2} + 6$	$p_{1,2} = 5$	$r_{2,2}^{-} = 2^{-1}\sqrt{2}$	$r_{2,2}^{+} = \sqrt{2}$	$p_{2,2} = 6$
3	$r_{1,3}^{-} = 1$	$r_{1,3}^{+} = 1 + \sqrt{37 + 6\sqrt{2}}$	$p_{1,3} = 4$	$r_{2,3}^{-} = 1$	$r_{2,3}^{+} = 2$	$p_{2,3} = 3$

Dolne ograniczenia długości uszeregowania dla wszystkich scenariuszach skrajnych są obliczane przed uruchomieniem głównej pętli algorytmu. Wybór punktu referencyjnego prowadzi do równości $LB\left(\tilde{u}^{i,j}(\eta), p(\eta)\right) = LB\left(\tilde{u}^{s,t}(\eta), p(\eta)\right)$, $\{j, t\} \subset \mathrm{J}$, $\{i, s\} \subset \mathrm{M}$.

Przykładowe obliczenia dolnych ograniczeń są następujące:

$$LB(\tilde{u}^{2,1}, p) = \max \begin{Bmatrix} LB_1 = 2^{-1}\sqrt{2} + 2^{-1}(5 + 3 + 3), \\ \boldsymbol{LB_2 = 2^{-1}\sqrt{2} + 6}, \\ LB_3 = 2^{-1}\sqrt{2} + 5, \\ LB_4 = 2^{-1}\left(2^{-1}\sqrt{2} + 5 + 3 + 3\right), \\ LB_5 = 2^{-1}\sqrt{2} + 2^{-1}(5 + 3 + 3) \end{Bmatrix} = \mathbf{2^{-1}\sqrt{2} + 6},$$

$$LB(\tilde{u}^{1,2},p)=\max\begin{Bmatrix}LB_1=2^{-1}\sqrt{2}+2^{-1}(5+3+3),\\ \boldsymbol{LB_2=2^{-1}\sqrt{2}+6},\\ \boldsymbol{LB_3=2^{-1}\sqrt{2}+6},\\ LB_4=2^{-1}\left(2^{-1}\sqrt{2}+5+3+3\right),\\ LB_5=2^{-1}\sqrt{2}+2^{-1}(5+3+3)\end{Bmatrix}=\boldsymbol{2^{-1}\sqrt{2}+6}.$$

Pierwszy etap optymalizacji $\boldsymbol{\eta=1}$: algorytm **DR** wybiera zadanie i maszynę minimalizujące

$$\left(r_{1,1}^{+}+p_{1,1}\right)-LB(\tilde{u}^{1,1},p)=\left(1+\sqrt{37+6\sqrt{2}}+3\right)-\left(2^{-1}\sqrt{2}+6\right),$$
$$\left(r_{2,1}^{+}+p_{2,1}\right)-LB(\tilde{u}^{2,1},p)=(2+4)-\left(2^{-1}\sqrt{2}+6\right),$$
$$\left(r_{1,2}^{+}+p_{1,2}\right)-LB(\tilde{u}^{1,2},p)=\left(\sqrt{2}+6+5\right)-\left(2^{-1}\sqrt{2}+6\right),$$
$$\left(r_{2,2}^{+}+p_{2,2}\right)-LB(\tilde{u}^{2,2},p)=\left(\sqrt{2}+6\right)-\left(2^{-1}\sqrt{2}+6\right),$$
$$\left(r_{1,3}^{+}+p_{1,3}\right)-LB(\tilde{u}^{1,3},p)=\left(1+\sqrt{37+6\sqrt{2}}+4\right)-\left(2^{-1}\sqrt{2}+6\right),$$
$$\boldsymbol{\left(r_{2,3}^{+}+p_{2,3}\right)-LB(\tilde{u}^{2,3},p)=(2+3)-\left(2^{-1}\sqrt{2}+6\right)}.$$

Zadanie $j=3$ zostało przypisane do maszyny $i=2$ na pierwszej pozycji w sekwencji i $\boldsymbol{C_{2,1}(\tilde{x}(1),\tilde{u}^{-};p)=4}$, $\boldsymbol{\tilde{x}_{2,1,3}(1)=1}$.

Drugi etap optymalizacji $\boldsymbol{\eta=2}$: algorytm **DR** wybiera zadanie i maszynę minimalizujące

$$\left(r_{1,1}^{+}+p_{1,1}\right)-LB(\tilde{u}^{1,1},p)=\left(1+\sqrt{37+6\sqrt{2}}+3\right)-\left(2^{-1}\sqrt{2}+6\right),$$
$$\boldsymbol{\left(4+p_{2,1}\right)-LB(\tilde{u}^{2,1},p)=(4+4)-\left(2^{-1}\sqrt{2}+6\right)},$$
$$\left(r_{1,2}^{+}+p_{1,2}\right)-LB(\tilde{u}^{1,2},p)=\left(\sqrt{2}+6+5\right)-\left(2^{-1}\sqrt{2}+6\right),$$
$$\left(4+p_{2,2}\right)-LB(\tilde{u}^{2,2},p)=(4+6)-\left(2^{-1}\sqrt{2}+6\right).$$

Zadanie $j=1$ zostało przypisane do maszyny $i=2$ na drugiej pozycji w sekwencji i $\boldsymbol{C_{2,2}(\tilde{x}(2),\tilde{u}^{-};p)=8}$, $\boldsymbol{\tilde{x}_{2,2,1}(2)=1}$.

Trzeci etap optymalizacji $\boldsymbol{\eta=3}$: algorytm **DR** wybiera zadanie i maszynę minimalizujące

$$\max\begin{Bmatrix}\boldsymbol{\left(r_{1,2}^{+}+p_{1,2}\right)-LB(\tilde{u}^{1,2},p)=\left(\sqrt{2}+6+5\right)-\left(2^{-1}\sqrt{2}+6\right)},\\ \left(8+p_{1,2}\right)-LB(\tilde{u}^{1,2},p)=(8+6)-\left(2^{-1}\sqrt{2}+6\right)\end{Bmatrix}$$

Zadanie $j=2$ zostało przypisane do maszyny $i=1$ na pierwszej pozycji w sekwencji i $\boldsymbol{C_{1,1}(\tilde{x}(3),\tilde{u}^{-};p)=2^{-1}\sqrt{2}+5}$, $\boldsymbol{\tilde{x}_{1,1,2}(3)=1}$.

Uszeregowanie $\{\tilde{x}_{1,1,2}=1, \tilde{x}_{2,1,3}=1, \tilde{x}_{2,2,1}=1\}$ przy decyzjach $\{y_{4,1}=1, y_{5,2}=1\}$ (dobranych permutacyjnym generowaniem możliwych rozmieszczeń maszyn) gwarantuje $C_{\max}(\tilde{x}_{\mathrm{DR}}, y_{\mathrm{EX}}; \mathrm{S}_l) = \max\begin{Bmatrix} C_{1,2}(\tilde{x}_{\mathrm{DR}}, y_{\mathrm{EX}}; \mathrm{S}_l) = 2^{-1}\sqrt{2}+5, \\ C_{2,1}(\tilde{x}_{\mathrm{DR}}, y_{\mathrm{EX}}; \mathrm{S}_l) = 8 \end{Bmatrix} = 8,$ czyli długość uszeregowania w problemie ScheLoc, która jest mniejsza niż w podejściu łącznym, tzn. $C_{\max}(\tilde{x}_{\mathrm{DR}}, y_{\mathrm{EX}}; \mathrm{S}_l) < C_{\max}(x_{\mathrm{EG}}, y_{\mathrm{EG}}; \mathrm{S}_l)$.

Wyróżnikiem podejścia niepewnego jest szeregowanie zadań w oparciu o przedziały niepewności obliczone na podstawie lokalizacji, które nie muszą być wykorzystane do rozmieszczenia maszyn. Algorytm **SR** uszeregował zadania w oparciu o skrajne scenariusze, w których dowolny najpóźniejszy termin gotowości odnosi się do lokalizacji, która nie została wybrana w podproblemie rozmieszczenia. W przypadku $Rm|r_j^- \leq r_j \leq r_j^+|Reg(C_{\max})$ (dane w tabeli 4.5) najgorszy scenariusz przy uszeregowaniu $\bar{x}_{\mathrm{SR}}$ uwzględnił wierzchołek numer 6 i stanowił podstawę wykluczenia tego wierzchołka w podproblemie rozmieszczenia maszyn, ponieważ terminy gotowości zadań byłyby nieakceptowalnie długie. Natomiast algorytm **DR** podejmował decyzje bazując na przedziałach niepewności, których wartości skrajne nie odzwierciedlają precyzyjnie współrzędnych lokalizacji dostępnych w podproblemie rozmieszczenia maszyn (górne granice przedziałów niepewności są wydłużane względem najdalej położonych lokalizacji dla maszyn). Należy podkreślić fakt, że rozwiązania uzyskane z wykorzystaniem zaproponowanych podejść niepewnych są optymalnymi rozwiązaniami instancji problemu ScheLoc, którą zaprezentowano na rysunku 4.16.

Rezultaty badań potwierdziły prawdziwość tezy numer 4, którą sformułowano w podrozdziale 1.5.

5. Ilustracyjne przykłady zastosowania

Przykłady zastosowania algorytmów opracowanych na potrzeby problemów rozważanych w pracy omówiono w trzech częściach. W pierwszej części omówiono wykorzystanie algorytmów opracowanych dla problemu $Rm|r_{i,j}^{-} \leq r_{i,j} \leq r_{i,j}^{+}|Reg(C_{\max})$ w procesie harmonogramowania produkcji odwodów drukowanych PCB (ang. Printed Circuit Board). W drugiej części omówiono łączny problem planowania infrastruktury przemysłowej (alokacji zasobów do produkcji) oraz harmonogramowania produkcji obwodów, który jest równoważny sformułowanemu w pracy problemowi ScheLoc. Rozważono podejście niepewne do rozwiązania problemu. W trzeciej części wskazano przykład potencjalnego wykorzystania algorytmów rozwiązania problemu $Rm|r_{i,j}|C_{\max}$ do szeregowania wykonywania kodów źródłowych w środowisku rozproszonym.

1) Harmonogramowanie produkcji obwodów drukowanych PCB

Bardzo ciekawym obszarem wykorzystania algorytmów rozwiązania problemu $Rm|r_{i,j}^{-} \leq r_{i,j} \leq r_{i,j}^{+}|Reg(C_{\max})$ jest harmonogramowanie procesu produkcji obwodów drukowanych PCB w wielu fabrykach. Popularną usługą jest oferowanie produkcji płytek PCB wraz z montażem elementów półprzewodnikowych za pośrednictwem Internetu. Projektanci układów elektronicznych umieszczają projekt płytki PCB (lub zbiór różnych projektów) na stronie internetowej producenta (w aplikacji webowej) oraz wskazują wymagane techniki i parametry montażu układów elektronicznych. Dodatkowo można opcjonalnie zlecać instalację oprogramowania (np. instalacja wybranej dystrybucji systemu Linux), sterowników oraz testowanie. W przypadku ograniczonych mocy produkcyjnych oraz konieczności zamawiania układów półprzewodnikowych u ich producentów (projekt odwodu drukowanego zakłada montaż układów scalonych, które nie są dostępne w miejscu produkcji obwodu) problemem jest harmonogramie terminów realizacji zamówień w wielu fabrykach.

Podstawowe pojęcia i dane problemu szeregowania $Rm|r_{i,j}^{-} \leq r_{i,j} \leq r_{i,j}^{+}|Reg(C_{\max})$ mają dla rozważanego przykładu następującą interpretację:

- zadanie j to zamówienie wykonania układu elektronicznego (serii płytek PCB z montażem układów półprzewodnikowych, instalacją oprogramowania, testowaniem),

- maszyna i odpowiada linii produkcyjnej i umieszczonej w fabryce (każda fabryka jest położona w innej lokalizacji geograficznej oraz każda linia produkcyjna jest uruchomiona w innej fabryce),
- termin gotowości $r_{i,j}$ jest momentem, w którym wszystkie komponenty elektroniczne (układy półprzewodnikowe) wymagane do realizacji zamówienia j zostaną skompletowane i dostarczone do linii produkcyjnej i,
- czas realizacji $p_{i,j}$ jest równy sumie czasów produkcji układów elektronicznych w zamówieniu j na linii produkcyjnej i, przetestowania oprogramowania, przetestowania układu oraz nadania przesyłki z układem do zamawiającego.

Termin „linia produkcyjna" stanowi uproszczenie, ponieważ w fabryce jest uruchomiona (zazwyczaj) więcej niż jedna linia produkcyjna. W deterministycznym problemie szeregowania taki przypadek można modelować poprzez wprowadzenie co najmniej dwóch maszyn (linii produkcyjnych) zapewniających równe terminy gotowości zadań, ponieważ są umieszczone w tej samej fabryce (w niedeterministycznym problemie przedziały niepewności dla terminów gotowości byłyby takie same na tych dwóch maszynach). W badaniach przyjęto, że realizacja produkcji j (jedno zadanie) na linii produkcyjnej i jest tożsama z zaangażowaniem wszystkich dostępnych (wolnych) mocy produkcyjnych na czas realizacji tego zadania w fabryce.

W praktyce terminy gotowości $r_{i,j}$ są nieprecyzyjne, ponieważ przygotowanie do realizacji zamówienia zależy od czasu dostarczenia brakujących komponentów półprzewodnikowych do miejsca produkcji układu elektronicznego lub poprawek w projekcie obwodu (np. błędy w postaci zbyt wąskich ścieżek, zbyt małych odległości pomiędzy ścieżkami i elementami układu są wykrywane i korygowane automatycznie oraz wymagają akceptacji projektanta). Czasy realizacji są zadeklarowane przez producenta obwodu PCB. Zróżnicowanie czasów realizacji jest konsekwencją mocy produkcyjnych w parku maszynowym oraz liczbą i specjalizacją pracowników.

W tabeli 5.1 przedstawiono przykładowe dane dla opisanego problemu z założeniem wykorzystania trzech linii produkcyjnych ($m = 3$) w różnych fabrykach (funkcjonujących w ramach jednego przedsiębiorstwa), których moce produkcyjne, w momencie harmonogramowania, mogą być zagospodarowane.

Tabela 5.1. Instancja problemu: $m = 3$, $n = 14$, $\mathrm{J} = \{1,2,\ldots,14\}$, $\mathrm{M} = \{1,2,3\}$. Terminy gotowości oraz czasy realizacji zadań są wyrażone w godzinach

j	$r_{1,j}^{-}$	$r_{1,j}^{+}$	$r_{2,j}^{-}$	$r_{2,j}^{+}$	$r_{3,j}^{-}$	$r_{3,j}^{+}$	$p_{1,j}$	$p_{2,j}$	$p_{3,j}$
1	6	12	8	24	18	48	1	1	1
2	12	48	24	48	6	12	2	4	2
3	24	72	8	48	12	24	3	6	3
4	6	24	24	72	48	72	6	6	7
5	6	18	24	48	48	95	4	4	5
6	2	10	1	6	8	18	2	2	3
7	1	2	1	4	1	6	5	4	5
8	4	12	8	12	10	24	4	4	5
9	10	20	12	24	18	24	1	4	3
10	24	48	10	18	10	48	7	8	6
11	10	14	8	20	24	48	5	4	4
12	24	72	48	95	12	36	2	2	2
13	12	36	8	48	4	24	3	5	2
14	48	96	24	96	12	96	2	1	2

Czasy produkcji podane w tabeli 5.1 odpowiadają konsumenckiemu charakterowi realizowanych zleceń. Czasy produkcji są znacznie mniejsze od terminów gotowości, ponieważ zlecenia obejmują produkcję niewielkich powierzchni obwodów drukowanych, wymagające zamówienia komponentów półprzewodnikowych u producentów. Do rozwiązania problemu $Rm|r_{i,j}^{-} \leq r_{i,j} \leq r_{i,j}^{+}|Reg(C_{\max})$ dla danych w tabeli 5.1 wykorzystano dedykowane algorytmy **DR** i **GR**. Wynikiem optymalizacji (oba algorytmy zwróciły takie same uszeregowania) jest harmonogram

$$\tilde{x}^{u_1} = \begin{bmatrix} 7 & 1 & 11 & 9 & 5 & 4 \\ 6 & 8 & 10 & 14 & 0 & 0 \\ 2 & 13 & 3 & 12 & 0 & 0 \end{bmatrix}.$$

W rozprawie posługiwano się harmonogramem w postaci trójwymiarowej macierzy binarnej. Na potrzeby przykładu, w celu poprawy czytelności harmonogramu, przyjęto uproszczoną reprezentację rozwiązania. Liczby w pierwszym wierszu macierzy $\tilde{x}^{u_1}$ definiują sekwencję zadań ustanowioną na maszynie $i = 1$ (wiersze $2, 3$ analogicznie wskazują sekwencje dla maszyn $i = 2, 3$). Zadania są realizowane od lewej do prawej strony w macierzy $\tilde{x}^{u_1}$. Wartość oszacowanego maksymalnego żalu dla harmonogramu $\tilde{x}^{u_1}$ jest równa 82 [h], tzn. produkcja obwodów drukowanych, przy harmonogramie $\tilde{x}^{u_1}$, może opóźnić się o co najwyżej 82 [h] (w najgorszym przypadku) w stosunku do najlepszego harmonogramu (długość uszeregowania dla najlepszego harmonogramu jest oszacowana). Pozornie długi

czas jest wynikiem długich (w porównaniu do czasów realizacji zamówień) przedziałów niepewności, ponieważ w wartości kryterium jest uwzględniany czas oczekiwania na komponenty do układów elektronicznych. Decydent może wykluczyć podzbiór zadań w planowaniu produkcji, aby zmniejszyć opóźnienie przy najgorszym scenariuszu, jeżeli jest ono nieakceptowalne. Na przykład wykluczenie zadania $j = 14$, którego termin gotowości $r_{i,j}^{+}$ jest najpóźniejszy na trzech maszynach $i = 1,2,3$, pozwala uzyskać harmonogram (oba algorytmy zwróciły takie same uszeregowania)

$$\tilde{x}^{u_2} = \begin{bmatrix} 7 & 1 & 11 & 9 & 5 & 4 \\ 6 & 8 & 10 & 0 & 0 & 0 \\ 2 & 13 & 3 & 12 & 0 & 0 \end{bmatrix},$$

dla którego wartość oszacowanego maksymalnego żalu jest równa $23{,}33$ [h]. Rozbieżność pomiędzy wartościami kryterium dla $\tilde{x}^{u_1}$i $\tilde{x}^{u_2}$ wynika z faktu, że zadanie $j = 14$ może, przy dowolnej decyzji o jego uszeregowaniu i najpóźniejszym terminie gotowości, rozpocząć się o wiele wcześniej na maszynie, na której nie jest ono realizowane według harmonogramu optymalnego.

Tabela 5.2. Instancja problemu: $m = 3$, $n = 14$, J $= \{1,2, \ldots ,14\}$, M $= \{1,2,3\}$ (terminy gotowości są równe podanym w tabeli 5.1). Czasy realizacji zadań są wyrażone w godzinach

j	$p_{1,j}$	$p_{2,j}$	$p_{3,j}$
1	12	10	11
2	23	27	28
3	17	16	13
4	25	20	27
5	24	26	25
6	12	12	13
7	15	15	15
8	22	21	28
9	18	19	21
10	17	20	16
11	15	14	14
12	18	22	20
13	13	15	12
14	12	11	14

Kolejne badanie przeprowadzono dla terminów gotowości podanych w tabeli 5.1 i czasów realizacji zadań w tabeli 5.2, które są dłuższe oraz, w pewnych przypadkach, są porównywalne z najwcześniejszymi możliwymi terminami gotowości zadań (np. $j = 11$). Wydłużenie czasów produkcji jest konsekwencją zwiększenia wolumenu produkcji (powierzchni płytek PCB) w ramach każdego zamówienia. Wówczas uszeregowania mają postać

$$\tilde{x}^{u_{\mathrm{DR}}} = \begin{bmatrix} 7 & 1 & 8 & 2 & 0 & 0 \\ 6 & 11 & 9 & 5 & 4 & 14 \\ 13 & 3 & 12 & 10 & 0 & 0 \end{bmatrix} \text{ oraz } \tilde{Z}(\tilde{x}_{\mathrm{DR}}) = 33{,}33 \text{ [h]},$$

$$\tilde{x}^{u_{\mathrm{GR}}} = \begin{bmatrix} 7 & 1 & 9 & 8 & 2 \\ 6 & 11 & 10 & 5 & 14 \\ 13 & 3 & 12 & 4 & 0 \end{bmatrix} \text{ oraz } \tilde{Z}(\tilde{x}_{\mathrm{GR}}) = 30{,}33 \text{ [h]}.$$

Wartości kryterium są mniejsze dla wszystkich zadań w harmonogramach, ponieważ dłuższe czasy realizacji zadań kompensują wpływ terminów gotowości na rezultaty optymalizacji.

2) Planowanie infrastruktury przemysłowej oraz harmonogramowanie produkcji obwodów drukowanych PCB

Precyzyjnie określony termin gotowości zadania j, odpowiadający precyzyjnie określonemu czasowi dostarczenia komponentów półprzewodnikowych (wymaganych do wyprodukowania obwodu drukowanego wraz z montażem komponentów) do wybranej lokalizacji, można zobrazować w dwuwymiarowej przestrzeni Euklidesowej w taki sposób, że odległość między współrzędnymi położenia zadania j a współrzędnymi położenia maszyny i odpowiada terminowi gotowości $r_{i,j}$. Jest to przypadek teoretyczny, ponieważ precyzyjne modelowanie czasów dostaw jest w praktyce niemożliwe. Zasadniczą kwestią jest odpowiednie dostosowanie mocy produkcyjnych, rozumiane jako jednoczesne planowanie infrastruktury przemysłowej oraz harmonogramowanie produkcji np. w przypadku zakontraktowania mocy produkcyjnych przez jednego klienta dla cyklicznych (lub częstych) zamówień produkcji obwodów, przy jednoczesnym uwzględnieniu niepewnych terminów dostaw komponentów półprzewodnikowych. W takim przypadku celem jest rozmieszczenie m linii produkcyjnych (maszyn) w obrębie w fabryk (lokalizacji dostępnych dla maszyn). Wybór $m < w$ maszyn jest podyktowany koniecznością zachowania wolnych mocy produkcyjnych. Przypadek teoretyczny (z precyzyjnie ustalonymi terminami gotowości) jest równoważny łącznemu problemowi rozmieszczenia maszyn oraz szeregowania zadań ScheLoc.

Problemy opisane w podpunktach **1)** i **2)** można traktować jako powiązane ze sobą, tzn. w **2)** dostosowywane są (zakontraktowane) moce produkcyjne (w postaci aranżacji linii produkcyjnych w fabrykach) w odniesieniu do pierwszego złożonego zamówienia, natomiast w **1)** (dla ustalonej infrastruktury przemysłowej) realizowane jest harmonogramowanie kolejnych zamówień.

Tabela 5.3. Współrzędne dostępnych lokalizacji dla maszyn, $(\tilde{l}_{g,1}, \tilde{l}_{g,2}) \in \mathbb{R}^2$, $\tilde{l}_g \in \tilde{\mathrm{L}}$, $w = 8$

g	$\tilde{l}_{g,1}$	$\tilde{l}_{g,2}$
1	164	197
2	116	150
3	198	127
4	111	145
5	152	187
6	144	169
7	131	111
8	171	183

Tabela 5.4. Instancja problemu: $m = 3$, $n = 14$, $w = 8$, $\mathrm{J} = \{1,2,\ldots,14\}$, $\mathrm{M} = \{1,2,3\}$, $l_j = (l_{j,1}, l_{j,1}) \in \mathbb{R}^2$, $l_j \in \mathrm{L}$.

Terminy gotowości oraz czasy realizacji zadań są wyrażone w godzinach

j	$l_{j,1}$	$l_{j,2}$	$r_{1,j}^{\mathrm{SL}}$	$r_{2,j}^{\mathrm{SL}}$	$r_{3,j}^{\mathrm{SL}}$	$r_{4,j}^{\mathrm{SL}}$	$r_{5,j}^{\mathrm{SL}}$	$r_{6,j}^{\mathrm{SL}}$	$r_{7,j}^{\mathrm{SL}}$	$r_{8,j}^{\mathrm{SL}}$	$p_{1,j}$	$p_{2,j}$	$p_{3,j}$
1	117	198	47	48	107	53	36	39	88	56	21	26	24
2	140	130	71	31	58	32	58	39	21	61	22	22	23
3	107	150	73	9	93	6	58	41	45	72	21	29	27
4	179	123	75	68	19	71	69	57	49	60	30	21	24
5	148	171	30	38	66	45	16	4	62	25	23	27	26
6	109	111	102	39	90	34	87	67	22	95	23	24	22
7	118	125	85	25	80	21	70	51	19	78	28	25	25
8	173	101	96	75	36	76	88	73	43	82	20	22	26
9	117	160	59	10	87	16	44	28	50	58	28	24	29
10	184	120	79	74	15	77	74	63	53	64	30	27	21
11	143	107	92	50	58	49	80	62	12	80	27	22	24
12	149	163	37	35	60	42	24	7	55	29	22	27	30
13	177	134	64	63	22	66	58	48	51	49	26	20	30
14	150	154	45	34	55	40	33	16	47	35	26	26	28

Termin gotowości $r_{g,j}^{\mathrm{SL}}$ jest równy odległości w przestrzeni Euklidesowej pomiędzy współrzędnymi $\tilde{l}_g$ oraz l_j (dla uproszczenia obliczeń pominięto wartości dziesiętne).

Rozwiązanie problemu, z wykorzystaniem algorytmu **EG**, dla precyzyjnie określonych danych w tabelach 5.3 oraz 5.4 (przypadek teoretyczny), podano macierzami

$$\tilde{x}^{u_{\mathrm{EG}}} = \begin{bmatrix} 5 & 12 & 1 & 3 & 8 \\ 11 & 2 & 13 & 4 & 6 \\ 14 & 10 & 7 & 9 & 0 \end{bmatrix}, y_{\mathrm{EG}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \text{ oraz } C_{\max}(x_{\mathrm{EG}}, y_{\mathrm{EG}}; \mathrm{S}_l) = 138 \text{ [h]}.$$

W macierzy y_{EG}: pierwsza maszyna (sekwencja zadań podana w pierwszym wierszu $\tilde{x}^{u_{\mathrm{EG}}}$ jest ustanowiona na pierwszej maszynie itd.) została przypisana do lokalizacji $g = 1$, druga maszyna została przypisana do lokalizacji $g = 7$, trzecia maszyna została przypisana do lokalizacji $g = 6$.

Następnie sprawdzono jak przyjęcie niepewności w procesie optymalizacji wpływa na jakość planowania infrastruktury przemysłowej i harmonogramowania produkcji.

Logistyka dostaw komponentów półprzewodnikowych do fabryk stanowi podstawę do przyjęcia niepewności w rozważanym problemie. W pierwszym przypadku rozważono podejście, w którym przedział niepewności zadania $j \in \{1,2,\ldots,14\}$ jest określony przez najbliższą i najbardziej odległą lokalizację w zbiorze $\tilde{\mathrm{L}}$, czyli zastosowano metodę podaną wzorem (4.11). Wówczas algorytm **SR** zwrócił najlepsze rozwiązanie

$$\bar{x}^{u_{\mathrm{SR}}} = \begin{bmatrix} 14 & 12 & 3 & 1 & 2 \\ 9 & 7 & 13 & 11 & 8 \\ 10 & 5 & 4 & 6 & 0 \end{bmatrix}, y = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \text{ oraz } C_{\max}(\bar{x}_{\mathrm{SR}}, y; \mathrm{S}_l) = 138 \text{ [h]}.$$

Przyjęcie takiej formy reprezentacji niepewności jest najbardziej pesymistycznym przypadkiem, bazującym jedynie na logistyce dostaw, ponieważ uwzględnia jedynie skrajne terminy dostarczenia komponentów do produkcji oraz nie rozróżnia wszystkich lokalizacji (przedział niepewności dla każdego zadania jest wyliczany jedynie w oparciu o dwie lokalizacje w $\tilde{\mathrm{L}}$).

W kolejnym przypadku przedziały niepewności obliczono według wzorów (4.12)-(4.14). Liczba przedziałów niepewności, obliczonych według (4.12)-(4.14), jest równa liczbie maszyn dla dowolnego zadania j. Dodawanie wartości referencyjnej do górnych granic przedziałów niepewności ma na celu „uodpornienie" sformułowania na przypadek nieprzewidzianych okoliczności (np. opóźnienia w dostawach komponentów do produkcji). Wówczas najlepsze rozwiązanie uzyskał algorytm **GR**, przy (4.12)-(4.13), które ma postać

$$\tilde{x}^{u_{\mathrm{GR}}} = \begin{bmatrix} 12 & 2 & 3 & 8 & 6 \\ 9 & 7 & 13 & 4 & 11 \\ 5 & 14 & 1 & 10 & 0 \end{bmatrix}, y = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \text{ oraz } C_{\max}(\tilde{x}_{\mathrm{GR}}, y; \mathrm{S}_l) = 126 \text{ [h]},$$

i zapewnia mniejszą długość uszeregowania niż rozwiązanie przypadku teoretycznego. Zatem, dla rozważanego zbioru danych, rezultat optymalizacji odpornej (podejście niepewne) zapewnił bardziej efektywne podejście w planowaniu infrastruktury przemysłowej i harmonogramowaniu produkcji niż rozwiązanie problemu deterministycznego.

3) Szeregowanie wykonywania kodów źródłowych

Nieoczywistym obszarem wykorzystania algorytmów rozwiązania problemu $Rm|r_{i,j}|C_{\max}$ jest harmonogramowanie wykonywania kodów źródłowych z założeniem możliwości ich tłumaczenia na różne języki programowania. Roziere et al. (2020) przedstawili pracę, w której omówili temat tłumaczenia kodów źródłowych. Autorzy zaproponowali metodę zmiany kodu źródłowego z języka A na język B. Skupiono się na kodzie wysokopoziomowym i wskazano, że funkcjonalność automatycznej zmiany kodu może być wykorzystywana w systemach informatycznych do zastępowania przestarzałego, niewydajnego lub niewspieranego kodu. Naturalnym rozszerzeniem badań nad tłumaczeniem kodów źródłowych jest uwzględnianie parametrów sprzętowych urządzeń, na których są one wykonywane. Jeżeli potrafimy w sposób zautomatyzowany przepisać kod napisany w języku A, który jest dedykowany specyficznej architekturze (np. assembler zaprojektowany dla procesora x86/x64), na kod źródłowy dedykowany innej architekturze (np. assembler dla procesora o architekturze ARM lub AVR), to uzyskujemy zdolność realizacji obliczeń wykorzystując różne urządzenia elektroniczne, na przykład będące elementami Internetu rzeczy (ang. Internet of Things). Mniejszą wydajność, wynikającą z mniejszej mocy obliczeniowej nowego układu scalonego, można częściowo kompensować

wykorzystywaniem jego specyficznej architektury (np. wielordzeniowości) lub wbudowanych specyficznych cech sprzętowych (np. wykorzystanie jednostki GPU).

Podstawowe pojęcia i dane problemu szeregowania $Rm|r_{i,j}|C_{\max}$ mają dla rozważanego przykładu następującą interpretację:

- zadanie j to kod źródłowy, który należy wykonać na wybranym układzie scalonym (procesorze, mikrokontrolerze),
- maszyna i to procesor wykonujący kod,
- termin gotowości $r_{i,j}$ jest momentem, w którym kod źródłowy, pierwotnie napisany w języku A, zostanie przepisany (przetłumaczony) na język B. Termin gotowości zadania zależy od trudności tłumaczenia kodu na konkretną platformę sprzętową (procesor),
- czas realizacji zadania $p_{i,j}$ jest równy czasowi wykonania kodu w języku B.

Wybór kryterium długości uszeregowania jest podyktowany koniecznością oczekiwania na rezultaty wykonania wszystkich kodów źródłowych. Można wówczas mówić o szeregowaniu wykonywania kodów źródłowych w środowisku rozproszonym (na wielu różnych procesorach).

Specjalnym przypadkiem zmiany kodu źródłowego z języka A na język B jest zmiana pseudokodu na kodu źródłowy. Informacje na ten temat są przedstawione między innymi w Kulal et al. (2016) i Zhong, Stern & Klein (2020). Automatyczne generowanie kodu źródłowego na podstawie pseudokodu oraz algorytm szeregowania dla problemu $Rm|r_{i,j}|C_{\max}$ umożliwiają harmonogramowanie badań naukowych, których celem jest porównanie efektywności algorytmów (np. opracowanych dla problemów optymalizacji lub przetwarzania danych) przedstawionych (w postaci pseudokodów) w wybranych publikacjach. Czasy realizacji obliczeń przez algorytmy (po przetłumaczeniu i uruchomieniu w nowym środowisku sprzętowym) można oszacować na podstawie wolumenów danych, czasów obliczeń oraz parametrów platform testowych podanych w publikacjach. W przypadku założenia niepewności czasów realizacji rozsądne jest przyjęcie najbardziej pesymistycznych czasów.

6. Podsumowanie pracy

6.1. Podsumowanie rezultatów rozprawy

W rozprawie doktorskiej rozpatrzono wybrane problemy szeregowania zadań na maszynach dowolnych z kryterium długości uszeregowania dla przypadku nieustalonych terminów gotowości. Przebadano oryginalny problem z terminami gotowości zadań zależnymi od maszyn. Rozważano również przypadek przedziałowych terminów gotowości zadań, co wymagało rozwiązania niedeterministycznej wersji odpowiedniego zagadnienia optymalizacji w postaci minimalizacji maksymalnego żalu opartego na długości uszeregowania. Wyniki uzyskane dla wymienionych już dwóch problemów, czyli odpowiednie algorytmy, wykorzystano w pracach nad rozwiązaniem złożonego problemu optymalizacyjnego ScheLoc, dla którego oprócz algorytmów dla wersji deterministycznej opracowano także algorytmy dla wersji niedeterministycznej. W trakcie badań uzyskano wyniki analityczne w postaci twierdzeń i własności, ale zasadniczymi rezultatami są algorytmy dla rozpatrywanych trzech głównych problemów oraz ich wersji szczegółowych, które poddano ocenie eksperymentalnej i statystycznej. Na szczególne podkreślenie zasługują następujące aspekty uzyskanych wyników.

a) Podczas analizy złożoności problemu $Rm|r_{i,j}|C_{\max}$ wykazano, że pozornie niewielka zmiana parametrów dowolnej instancji wpływa na trudność problemu. Poprawa szybkości przetwarzania zadań jednej maszyny lub wydłużenie czasu oczekiwania na gotowość jednego zadania umożliwiają optymalne rozwiązanie problemu w czasie wielomianowym. Wskazano przypadki, w których problem $Rm|r_{i,j}|C_{\max}$ może być rozwiązany w czasie pseudowielomianowym poprzez jego transformację do oryginalnych problemów $R2|r_{i,j}, d_i|C_{\max}$ oraz $R2|d_i|C_{\max}$ z wymaganymi terminami zakończeń zadań na maszynach d_i, które zależą od maszyn dowolnych. Opracowana metoda planowania długoterminowego w zachłannej strategii podejmowania decyzji, zaimplementowanej w algorytmie **GD**, pozwala zagwarantować (zależną od instancji) odległość od oszacowanego rozwiązania optymalnego. Ponadto na podstawie wykonanych badań statystycznych udowodniono, że harmonogramy wyznaczone przez algorytm **GD** gwarantują statystycznie mniejszą długość uszeregowania niż algorytmy **BF** i **SA**.

b) W analizie własności problemów $Rm|r_j^- \le r_j \le r_j^+|Reg(C_{\max})$ oraz $Rm|r_{i,j}^- \le r_{i,j} \le r_{i,j}^+|Reg(C_{\max})$ wykazano ich nieaproksymowalność oraz dowiedziono, że przegląd scenariuszy można ograniczyć do skończonego zbioru scenariuszy skrajnych. Dodatkowo

udowodniono, że najgorszy scenariusz może być scenariuszem skrajnym lub pośrednim. Opracowane algorytmy **DR** i **GR** zapewniały bardzo zbliżone rezultaty optymalizacji. Dla problemu $Rm|r_j^- \leq r_j \leq r_j^+|Reg(C_{\max})$ udowodniono, że instancje z rozłącznymi przedziałami niepewności dla terminów gotowości lub nieproporcjonalnie późnym jednym przedziałem niepewności można rozwiązać optymalnie w czasie wielomianowym. Wykazano również analitycznie, że maksymalna wartość żalu w problemie $Rm|r_j^- \leq r_j \leq r_j^+|Reg(C_{\max})$ nie może przekroczyć sumy czasów przetwarzania zadań uszeregowanych na najdłużej pracującej maszynie. Dowód jest podstawą do rozróżnienia interpretacji wartości kryterium maksymalnego żalu dla rozważanych problemów optymalizacji odpornej. Rezultaty badań empirycznych uzasadniły fakt, że algorytm **SR**, podejmujący decyzje w oparciu o deterministyczne kryterium, jest w stanie osiągać lepsze rezultaty optymalizacji (na przykład dla $m = 10$ i $n = 250$) niż algorytmy **DR** i **GR**, wykorzystujące kryterium maksymalnego żalu w procesie optymalizacji. Ewaluacja oszacowanej wartości funkcji celu w algorytmie **GR** (dla niepełnego harmonogramu w każdym stanie rozwiązania) okazała się być efektywniejszą techniką optymalizacji niż minimalizacja odchylenia przy jednym scenariuszu skrajnym (scenariuszu gwarantującym najmniejsze odchylenie od długości uszeregowania w każdym stanie rozwiązania) w algorytmie **DR**. Jednak wykorzystanie algorytmu **GR** wymaga zaangażowania wydajniejszych zasobów sprzętowych, w porównaniu do algorytmu **DR**, jeżeli kryterium wyboru jest również czas obliczeń. Wykazano, bazując na wynikach badań statystycznych, że harmonogramy wyznaczone przez algorytm **GR** gwarantują statystycznie mniejszą wartość oszacowanego maksymalnego żalu niż algorytmy **DR** i **SR**.

c) Warto podkreślić, że najlepsze rezultaty optymalizacji dla problemów szeregowania uzyskały algorytmy deterministyczne (adaptujące strategie zachłanne lub wykorzystujące własności problemów). Projektowanie algorytmów bazujących na metaheurystykach Simulated Annealing (dla problemu $Rm|r_{i,j}|C_{\max}$) oraz Tabu Search (dla problemu $Rm|r_{i,j}^- \leq r_{i,j} \leq r_{i,j}^+|Reg(C_{\max})$) okazało się trudne. Algorytmy **SA** oraz **TS** wykazywały tendencje do zmian pozycji ograniczonego podzbioru zadań (niezależnie od instancji), co skutkowało nierównomiernym obciążaniem maszyn, i w konsekwencji, zwiększało długość uszeregowania.

d) Sformułowanie problemu ScheLoc z ustaloną liczbą maszyn do rozmieszczenia umożliwiło zaprojektowanie algorytmu **EG**, łączącego algorytm dokładny dla podproblemu

rozmieszczenia i algorytm zachłanny **GD** dla podproblemu szeregowania zadań. Algorytm **EG** zwracał uszeregowania o najlepszej jakości, ale jest niepraktyczny (dla większych instancji problemu, np. $m > 5$ i $w > 20$) z powodu wysokiej (wielomianowej) czasowej złożoności obliczeniowej, która wynika z optymalnego rozwiązania podproblemu rozmieszczenia maszyn. Algorytm genetyczny **GN** i algorytm hybrydowy **HS** zaprojektowano dla większych instancji. Algorytm **HS** zwracał harmonogramy zapewniające mniejszą długość uszeregowania w porównaniu do algorytmu **GN** dla wszystkich wygenerowanych instancji problemu ScheLoc. Wykazano, bazując na wynikach badań statystycznych, że w podejściu sekwencyjnym statystycznie mniejszą wartość kryterium uzyskiwano dla lokalizacji będących rezultatem rozwiązania podproblemu m-median. Ponadto należy podkreślić fakt, że sformułowanie podproblemu rozmieszczenia m-median oraz m-center nie wykluczało uzyskania rozwiązania o takiej samej jakości jaką gwarantowało podejście łączne.

e) Bardzo ciekawym rezultatem badań, który potwierdził prawdziwość tezy numer 4 niniejszej rozprawy, jest wskazanie przykładowych instancji problemu ScheLoc, dla których zastosowanie podejścia niepewnego umożliwia wyznaczenie harmonogramu zapewniającego mniejszą długość uszeregowania niż zastosowanie podejścia łącznego z precyzyjnie określonymi parametrami problemu. Wskazano również podzbiór instancji, dla których uszeregowanie, będące optymalnym rozwiązaniem problemu $Rm|r_j^- \leq r_j \leq r_j^+|Reg(C_{\max})$, jest tożsame z optymalnym uszeregowaniem w problemie ScheLoc.

6.2. Kierunki dalszych prac

Kontynuacja prac nad łącznym problemem rozmieszczenia maszyn i szeregowania zadań lub jego składowymi podproblemami może obejmować badania analityczne, formułowanie nowych założeń, projektowanie nowych algorytmów rozwiązania oraz wyszukiwanie obszarów ich implementacji.

a) Przyszłe badania analityczne nad rozważanymi w rozprawie problemami będą skupiały się na definiowaniu nowych sformułowań (angażujących mniejszą liczbę zmiennych decyzyjnych lub bardziej efektywnych pamięciowo), ponieważ rozwój solverów, optymalizacji kwantowej oraz wzrost mocy obliczeniowej komputerów prawdopodobnie umożliwią w przyszłości rozwiązywanie złożonych problemów optymalizacyjnych dla coraz większych instancji. Analiza literatury dla problemu ScheLoc potwierdziła trend wykorzystywania solverów jako istotnych narzędzi rozwiązania.

b) Opracowanie dokładniejszych metod oszacowania długości uszeregowania dla problemu $Rm|r_{i,j}|C_{\max}$ może stanowić podstawę do opracowania algorytmów aproksymacyjnych i jednocześnie umożliwić dokładniejsze oszacowanie zakresu możliwych wartości kryterium maksymalnego żalu. Zaprojektowanie algorytmu aproksymacyjnego dla problemu $Rm|r_{i,j}|C_{\max}$ pozostaje zagadnieniem otwartym. Przyjęcie arbitralnych terminów zakończeń zadań d_i (terminy zakończeń zadań zależą od maszyn) w $R2|r_{i,j}, d_i|C_{\max}$ oraz $R2|d_i|C_{\max}$ prowadzi do szczególnie ciekawego założenia. Okazało się, że sam proces weryfikacji ich spełnialności jest problemem co najmniej NP-trudnym.

c) Naturalnym kierunkiem prowadzenia kolejnych badań nad problemami optymalizacji odpornej z niepewnymi terminami gotowości zadań jest zmiana sposobu reprezentacji niepewności lub wybór innego kryterium. Nowe (rozszerzone) sformułowania problemu ScheLoc mogą uwzględniać nowe kryteria (w tym podejście wielokryterialne), pewien stopień nieprecyzyjności parametrów problemu (np. przedziałowe czasy przetwarzania zadań) lub niepełną ilość informacji o instancji (np. zbiór zadań do uszeregowania jest niekompletny w momencie podejmowania decyzji). Bardzo ciekawym kierunkiem dalszych prac jest weryfikacja możliwości sformułowania problemu zastępczego dla problemu ScheLoc w taki sposób, aby algorytm rozwiązania problemu zastępczego cechował się mniejszą złożonością obliczeniową lub pamięciową i wyznaczał harmonogram zapewniający porównywalną wartość funkcji celu jak algorytmy dla problemu pierwotnego.

Literatura

Aissi H., Bazgan C. & Vanderpooten D. (2005). Complexity of the min-max and min-max regret assignment problem. *Operations Research Letters*, *33*(6), 634–640.

Aissi H., Bazgan C. & Vanderpooten D. (2009). Min–max and min–max regret versions of combinatorial optimization problems: A survey. *European Journal of Operational Research*, *197*(2), 427–438.

Aissi, H., & Roy, B. (2010). Robustness in multi-criteria decision aiding. *Trends in Multiple Criteria Decision Analysis*, 87-121.

Allahverdi, A., Aydilek, H., & Aydilek, A. (2014). Single machine scheduling problem with interval processing times to minimize mean weighted completion time. *Computers & Operations Research*, *51*, 200-207.

Avdeenko, T. V., & Mesentsev, Y. A. (2016). Efficient approaches to scheduling for unrelated parallel machines with release dates. *IFAC-PapersOnLine*, *49*(12), 1743-1748.

Averbakh, I. (2000). Minmax regret solutions for minimax optimization problems with uncertainty. *Operations Research Letters*, *27*(2), 57-65.

Averbakh, I. (2006). The minmax regret permutation flow-shop problem with two jobs. *European Journal of Operational Research*, *169*(3), 761-766.

Azar, Y., Champati, J. P., & Liang, B. (2018). 2-Approximation algorithm for a generalization of scheduling on unrelated parallel machines. *Information Processing Letters*, *139*, 39-43.

Azar, Y., Epstein, L., Richter, Y., & Woeginger, G. J. (2004). All-norm approximation algorithms. *Journal of Algorithms*, *52*(2), 120-133.

Bachtler, O., Krumke, S. O., & Le, H. M. (2020). Robust single machine makespan scheduling with release date uncertainty. *Operations Research Letters*, *48*(6), 816-819.

Bagheri, A., Wang, J., & Zhao, C. (2016). Data-driven stochastic transmission expansion planning. *IEEE Transactions on Power Systems*, *32*(5), 3461-3470.

Baker, J. E. (1987, July). Reducing bias and inefficiency in the selection algorithm. In *Proceedings of the Second International Conference on Genetic Algorithms* (Vol. 206, pp. 14-21).

Baker, K. R. (1984). The effects of input control in a simple scheduling model. *Journal of Operations Management*, *4*(2), 99-112.

Ben-Ameur, W. (2004). Computing the initial temperature of simulated annealing. *Computational Optimization and Applications*, *29*(3), 369-385.

Bertsimas, D., & Sim, M. (2003). Robust discrete optimization and network flows. *Mathematical Programming*, *98*(1), 49-71.

Bertsimas, D., & Sim, M. (2004). The price of robustness. *Operations Research*, *52*(1), 35-53.

Bierwirth, C., & Meisel, F. (2010). A survey of berth allocation and quay crane scheduling problems in container terminals. *European Journal of Operational Research*, *202*(3), 615-627.

Bożejko, W., Pempera, J., & Smutnicki, C. (2013). Parallel tabu search algorithm for the hybrid flow shop problem. *Computers & Industrial Engineering*, *65*(3), 466-474.

Buckhorst, A. F., do Canto, M. K. B., & Schmitt, R. H. (2022). The line-less mobile assembly system simultaneous scheduling and location problem. *Procedia CIRP*, *106*, 203-208.

Cappanera, P., & Scutellà, M. G. (2015). Joint assignment, scheduling, and routing models to home care optimization: A pattern-based approach. *Transportation Science*, *49*(4), 830-852.

Castillo-Salazar, J. A., Landa-Silva, D., & Qu, R. (2016). Workforce scheduling and routing problems: literature survey and computational study. *Annals of Operations Research*, *239*(1), 39-67.

Chen, H., Zhu, X., Liu, G., & Pedrycz, W. (2018). Uncertainty-aware online scheduling for real-time workflows in cloud service environment. *IEEE Transactions on Services Computing*, *14*(4), 1167-1178.

Chen, Z. L., & Vairaktarakis, G. L. (2005). Integrated scheduling of production and distribution operations. *Management Science*, *51*(4), 614-628.

Ćwik M. (2017). *Algorytmy rozwiązania odpornych problemów szeregowania zadań z niepewnością przedziałową* [Praca doktorska, Politechnika Wrocławska].

Ćwik, M., & Józefczyk, J. (2018). Heuristic algorithms for the minmax regret flow-shop problem with interval processing times. *Central European Journal of Operations Research*, *26*(1), 215-238.

Daniels, R. L., & Kouvelis, P. (1995). Robust scheduling to hedge against processing time uncertainty in single-stage production. *Management Science*, *41*(2), 363-376.

Dannerstedt, G. (1955). Production Scheduling for an Arbitrary Number of Periods Given the Sales Forecast in the Form of a Probability Distribution. *Journal of the Operations Research Society of America*, *3*(3), 300-318.

Dantzig, G. B. (1955). Linear programming under uncertainty. *Management Science*, *1*(3-4), 197-206.

Davis, E., & Jaffe, J. M. (1981). Algorithms for scheduling tasks on unrelated processors. *Journal of the ACM (JACM)*, *28*(4), 721-736.

Davis, L. (1985, August). Applying adaptive algorithms to epistatic domains. In *IJCAI* (Vol. 85, pp. 162-164).

Drwal, M., & Rischke, R. (2016). Complexity of interval minmax regret scheduling on parallel identical machines with total completion time criterion. *Operations Research Letters*, *44*(3), 354-358.

Dyer, M. E., & Frieze, A. M. (1985). A simple heuristic for the p-centre problem. *Operations Research Letters*, *3*(6), 285-288

El Fallahi, A., Prins, C., & Calvo, R. W. (2008). A memetic algorithm and a tabu search for the multi-compartment vehicle routing problem. *Computers & Operations Research*, *35*(5), 1725-1741.

Elvikis, D., Hamacher, H. W., & Kalsch, M. T. (2007). Scheduling and location (ScheLoc): makespan problem with variable release dates.

Elvikis, D., Hamacher, H. W., & Kalsch, M. T. (2009). Simultaneous scheduling and location (ScheLoc): the planar ScheLoc makespan problem. *Journal of Scheduling*, *12*(4), 361-374.

Ermoliev, Y. M., & Wets, R. B. (1988). *Numerical techniques for stochastic optimization*. Springer-Verlag.

Filcek, G., Jozefczyk, J., & Ławrynowicz, M. (2021). An evolutionary algorithm for joint bi-criteria location-scheduling problem. *International Journal of Industrial Engineering Computations*, *12*(2), 159-176.

Fu, Y., Ding, F., Mu, Z., Sun, C., & Gao, K. (2022). Integrating scheduling and routing decisions into home health care operation with skill requirements and uncertainties. *Journal of Simulation*, 1-24.

Gabrel, V., Murat, C., & Thiele, A. (2014). Recent advances in robust optimization: An overview. *European Journal of Operational Research*, *235*(3), 471-483.

Gantt, H. L. (1919). Efficiency and democracy. *Transactions of the American Society of Mechanical Engineers*, *40*, 799-808.

Garey, M. R., & Johnson, D. S. (1979). *Computers and intractability* (Vol. 174). San Francisco: freeman.

Gaspars-Wieloch, H. (2014). Modifications of the Hurwicz's decision rule. *Central European Journal of Operations Research*, *22*(4), 779-794.

Ghirardi, M., & Potts, C. N. (2005). Makespan minimization for scheduling unrelated parallel machines: A recovering beam search approach. *European Journal of Operational Research*, *165*(2), 457-467.

Glover, F. (1989). Tabu search—part I. *ORSA Journal on Computing*, *1*(3), 190-206.

Glover, F. (1990). Tabu search—part II. *ORSA Journal on Computing*, *2*(1), 4-32.

Glover, F. (1990). Tabu search: A tutorial. *Interfaces*, *20*(4), 74-94.

Graham, R. L., Lawler, E. L., Lenstra, J. K., & Kan, A. R. (1979). Optimization and approximation in deterministic sequencing and scheduling: a survey. In *Annals of Discrete Mathematics* (Vol. 5, pp. 287-326). Elsevier.

Gupta, S. K., & Rosenhead, J. (1968). Robustness in sequential investment decisions. *Management Science*, *15*(2), B-18.

Hakimi, S. L. (1964). Optimum locations of switching centers and the absolute centers and medians of a graph. *Operations Research*, *12*(3), 450-459.

Hakimi, S. L. (1965). Optimum distribution of switching centers in a communication network and some related graph theoretic problems. *Operations Research*, *13*(3), 462-475.

Harbaoui, H., & Khalfallah, S. (2020). Tabu-search optimization approach for no-wait hybrid flow-shop scheduling with dedicated machines. *Procedia Computer Science*, *176*, 706-712.

Hennes, H., & Hamacher, H. W. (2002). Integrated scheduling and location models: single machine makespan problems: Technical Report.

Heßler, C., & Deghdak, K. (2017). Discrete parallel machine makespan ScheLoc problem. *Journal of Combinatorial Optimization*, *34*(4), 1159-1186.

Hindi, K. S., & Toczyłowski, E. (1995). Detailed scheduling of batch production in a cell with parallel facilities and common renewable resources. *Computers & Industrial Engineering*, *28*(4), 839-850

Hochbaum, D. S. (1982). Heuristics for the fixed cost median problem. *Mathematical Programming*, *22*(1), 148-162.

Horowitz, E., & Sahni, S. (1976). Exact and approximate algorithms for scheduling nonidentical processors. Journal of the ACM (JACM), *23*(2), 317-327.

Hurwicz, L. (1951). Optimality criteria for decision making under ignorance (Vol. 370, pp. 1-11). Cowles Commission Discussion Paper, Statistics.

Ibarra, O. H., & Kim, C. E. (1977). Heuristic algorithms for scheduling independent tasks on nonidentical processors. *Journal of the ACM (JACM)*, *24*(2), 280-289.

Ingber, L. (1993). Simulated annealing: Practice versus theory. *Mathematical and Computer Modelling*, *18*(11), 29-57.

Jamili, A. (2017). A robust mathematical model and heuristic algorithms for integrated aircraft routing and scheduling, with consideration of fleet assignment problem. *Journal of Air Transport Management*, *58*, 21-30.

Józefczyk, J. (2001). Scheduling tasks on moving executors to minimise the maximum lateness. *European Journal of Operational Research*, *131*(1), 171-187.

Józefczyk, J., & Hojda, M. (2021). Systems Approach in Complex Problems of Decision-Making and Decision-Support. In *Automatic Control, Robotics, and Information Processing* (pp. 589-615). Springer, Cham.

Józefczyk, J., & Ławrynowicz, M. (2018). Heuristic algorithms for the Internet shopping optimization problem with price sensitivity discounts. *Kybernetes*, *47*(4), 831-852.

Józefczyk J., & Ławrynowicz M. (2021), On selected models and methods of robust decision-making and their applications, World Organisation of Systems and Cybernetics (WOSC Congress)

Józefczyk, J., Ławrynowicz, M., & Filcek, G. (2022). On problems and methods of coordinated scheduling and location. In Uncertainty and Imprecision in Decision Making and Decision Support: New Advances, Challenges, and Perspectives: Selected papers from BOS-2020, held on December 14-15, 2020, and IWIFSGN-2020, held on December 10-11, 2020 in Warsaw, Poland (pp. 145-163). Cham: Springer International Publishing.

Józefowska, J., Mika, M., Różycki, R., Waligóra, G., & Węglarz, J. (2001). Simulated annealing for multi-mode resource-constrained project scheduling. *Annals of Operations Research*, *102*, 137-155.

Kalaı, R., Lamboray, C., & Vanderpooten, D. (2012). Lexicographic α-robustness: An alternative to min–max criteria. *European Journal of Operational Research*, *220*(3), 722-728.

Kalsch, M. T., & Drezner, Z. (2010). Solving scheduling and location problems in the plane simultaneously. *Computers & Operations Research*, *37*(2), 256-264.

Kariv, O., & Hakimi, S. L. (1979a). An algorithmic approach to network location problems. I: The p-Centers. *SIAM Journal on Applied Mathematics*, *37*(3), 513-538.

Kariv, O., & Hakimi, S. L. (1979b). An algorithmic approach to network location problems. II: The p-Medians. *SIAM Journal on Applied Mathematics*, *37*(3), 539-560.

Kasperski, A. (2008). *Discrete optimization with interval data*. Berlin: Springer.

Kasperski, A., & Zieliński, P. (2008). A 2-approximation algorithm for interval data minmax regret sequencing problems with the total flow time criterion. *Operations Research Letters*, *36*(3), 343-344.

Kasperski, A., & Zieliński, P. (2016). Single machine scheduling problems with uncertain parameters and the OWA criterion. *Journal of Scheduling*, *19*(2), 177-190.

Kaufmann, C. (2014). A polynomial time algorithm for an integrated scheduling and location problem. In *14th International Conference on Project Management and Scheduling* (p. 124). Technische Universität München School of Management.

Keith, A. J., & Ahner, D. K. (2021). A survey of decision making and optimization under uncertainty. *Annals of Operations Research*, *300*(2), 319-353.

Kirkpatrick, S., Gelatt Jr, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. S*cience*, *220*(4598), 671-680.

Koiliaris, K., & Xu, C. (2019). Faster pseudopolynomial time algorithms for subset sum. *ACM Transactions on Algorithms (TALG)*, *15*(3), 1-20

Kooli, A., & Serairi, M. (2014). A mixed integer programming approach for the single machine problem with unequal release dates. *Computers & Operations Research*, *51*, 323-330.

Kouvelis, P., & Yu, G. (2013). *Robust discrete optimization and its applications* (Vol. 14). Springer Science & Business Media.

Kramer, R., & Kramer, A. (2021). An exact framework for the discrete parallel machine scheduling location problem. *Computers & Operations Research*, *132*, 105318.

Kramer, A., & Kramer, R. (2022). Integrating exact and heuristic methods to efficiently solve the ScheLoc problem. In *23ème congrès annuel de la Société Française de Recherche Opérationnelle et d'Aide à la Décision*.

Krumke, S. O., & Le, H. M. (2020). Robust absolute single machine makespan scheduling-location problem on trees. *Operations Research Letters*, *48*(1), 29-32.

Kulal, S., Pasupat, P., Chandra, K., Lee, M., Padon, O., Aiken, A., & Liang, P. S. (2019). Spoc: Search-based pseudocode to code. *Advances in Neural Information Processing Systems*, *32*.

Lenstra, J. K., Kan, A. R., & Brucker, P. (1977). Complexity of machine scheduling problems. In *Annals of discrete mathematics* (Vol. 1, pp. 343-362). Elsevier.

Lenstra, J. K., Shmoys, D. B., & Tardos, É. (1990). Approximation algorithms for scheduling unrelated parallel machines. *Mathematical Programming*, *46*(1), 259-271.

Li, Y., Côté, J. F., Callegari-Coelho, L., & Wu, P. (2021). Novel Formulations and Logic-Based Benders Decomposition for the Integrated Parallel Machine Scheduling and Location Problem. *INFORMS Journal on Computing*.

Li, Z., Wang, W., Yan, Y., & Li, Z. (2015). PS–ABC: A hybrid algorithm based on particle swarm and artificial bee colony for high-dimensional optimization problems. *Expert Systems with Applications*, *42*(22), 8881-8895.

Li, Y., Wen, X., Choi, T. M., & Chung, S. H. (2022). Optimal Establishments of Massive Testing Programs to Combat COVID-19: A Perspective of Parallel-Machine Scheduling-Location (ScheLoc) Problem. *IEEE Transactions on Engineering Management*.

Lin, Y. K. (2013). Particle swarm optimization algorithm for unrelated parallel machine scheduling with release dates. *Mathematical Problems in Engineering*, *2013*.

Liu, M., Lin, T., Chu, F., Zheng, F., & Chu, C. (2023). A new and general stochastic parallel machine ScheLoc problem with limited location capacity and customer credit risk. *RAIRO-Operations Research*, *57*(3), 1179-1193.

Liu, M., & Liu, R. (2019a). Risk-averse scheduling-location (ScheLoc) problem. In *2019 International Conference on Industrial Engineering and Systems Management (IESM)* (pp. 1-6). IEEE.

Liu, M., & Liu, X. (2019b). Distributionally robust parallel machine ScheLoc problem under service level constraints. *IFAC-PapersOnLine*, *52*(13), 875-880.

Liu, M., Liu, X., Zhang, E., Chu, F., & Chu, C. (2019). Scenario-based heuristic to two-stage stochastic program for the parallel machine ScheLoc problem. *International Journal of Production Research*, *57*(6), 1706-1723.

Liu, X., Chu, F., Zheng, F., Chu, C., & Liu, M. (2021). Parallel machine scheduling with stochastic release times and processing times. *International Journal of Production Research*, *59*(20), 6327-6346.

Lodwick, W. A., & Kacprzyk, J. (Eds.). (2010). *Fuzzy optimization: Recent advances and applications* (Vol. 254). Springer.

Lyu, X., Song, Y., He, C., Lei, Q., & Guo, W. (2019). Approach to integrated scheduling problems considering optimal number of automated guided vehicles and conflict-free routing in flexible manufacturing systems. *IEEE Access*, *7*, 74909-74924.

Ławrynowicz, M., & Filcek, G. (2020, September). A comparison of evolutionary and simulated annealing algorithms for bi-criteria location-scheduling problem. In *2020 15th Conference on Computer Science and Information Systems (FedCSIS)* (pp. 251-255). IEEE.

Ławrynowicz, M., & Józefczyk, J. (2019). A memetic algorithm for the discrete scheduling-location problem with unrelated machines. In *2019 24th International Conference on Methods and Models in Automation and Robotics (MMAR)* (pp. 158-163). IEEE.

Ławrynowicz, M., & Józefczyk, J. (2021). Robust unrelated parallel machine scheduling problem with interval release dates. *arXiv preprint arXiv:2107.09745*.

Ławrynowicz, M., & Józefczyk, J. (2022a). Scheduling jobs with machine-dependent release dates on unrelated machines. In *2022 26th International Conference on Methods and Models in Automation and Robotics (MMAR)* IEEE.

Ławrynowicz, M., & Józefczyk, J. (2022b). On the solvability of job scheduling problems with non-fixed release dates. In *2022 XXII Krajowa Konferencja Automatyzacji Procesów Dyskretnych (KKAPD)*.

Majdan, M., & Ogryczak, W. (2012, April). Determining OWA operator weights by mean absolute deviation minimization. In *International Conference on Artificial Intelligence and Soft Computing* (pp. 283-291). Springer, Berlin, Heidelberg.

Markowski M., & Józefczyk J.: Heuristic algorithms for solving uncertain routing-scheduling problem, *Artificial Intelligence and Soft Computing, ICAISC, Springer Lecture Notes in Computer Science*, 5097: 1052–1063, 2008.

Matsumoto, M., & Nishimura, T. (1998). Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, *8*(1), 3-30.

Mezentsev, Y. A., Estraykh, I. V., & Chubko, N. Y. (2019, October). Implementation of an efficient parametric algorithm for optimal scheduling on parallel machines with release dates. In *Journal of Physics: Conference Series* (Vol. 1333, No. 2, p. 022002). IOP Publishing.

Michalewicz, Z., & Fogel, D. B. (2013). *How to solve it: modern heuristics*. Springer Science & Business Media.

Michalewicz, Z., Dasgupta, D., Le Riche, R. G., & Schoenauer, M. (1996). Evolutionary algorithms for constrained engineering problems. *Computers & Industrial Engineering*, *30*(4), 851-870.

Mulvey, J. M., Vanderbei, R. J., & Zenios, S. A. (1995). Robust optimization of large-scale systems. *Operations Research*, *43*(2), 264-281.

Myszkowski, P. B., Olech, Ł. P., Laszczyk, M., & Skowroński, M. E. (2018). Hybrid differential evolution and greedy algorithm (DEGR) for solving multi-skill resource-constrained project scheduling problem. *Applied Soft Computing*, *62*, 1-14.

Nourani, Y., & Andresen, B. (1998). A comparison of simulated annealing cooling strategies. *Journal of Physics A: Mathematical and General*, *31*(41), 8373.

Özarık, S. S., Veelenturf, L. P., Van Woensel, T., & Laporte, G. (2021). Optimizing e-commerce last-mile vehicle routing and scheduling under uncertain customer presence. *Transportation Research Part E: Logistics and Transportation Review*, *148*, 102263.

Paraskevopoulos, D. C., Laporte, G., Repoussis, P. P., & Tarantilis, C. D. (2017). Resource constrained routing and scheduling: Review and research prospects. *European Journal of Operational Research*, *263*(3), 737-754.

Pereira, J. (2016). The robust (minmax regret) single machine scheduling with interval processing times and total weighted completion time objective. *Computers & Operations Research*, *66*, 141-152.

Piasecki B. (2018). *Wykorzystanie algorytmów sztucznej inteligencji do rozwiązania łącznego problemu szeregowania zadań i rozmieszczenia realizatorów* [Praca magisterska, Politechnika Wrocławska].

Piasecki B., & Józefczyk J. (2018) Evolutionary algorithm for joint task scheduling and deployment of executors. In: *Automation of Discrete Processes. Theory and Applications* (in Polish). Silesian University of Technology, *1*, 169-178.

Pinedo, M. L. (2012). *Scheduling* (Vol. 29). New York: Springer.

Potts, C. N. (1985). Analysis of a linear programming heuristic for scheduling unrelated parallel machines. *Discrete Applied Mathematics*, *10*(2), 155-164.

Pratt, J. W. (1959). Remarks on zeros and ties in the Wilcoxon signed rank procedures. *Journal of the American Statistical Association*, *54*(287), 655-667.

Rahbari, A., Nasiri, M. M., Werner, F., Musavi, M., & Jolai, F. (2019). The vehicle routing and scheduling problem with cross-docking for perishable products under uncertainty: Two robust bi-objective models. *Applied Mathematical Modelling*, *70*, 605-625.

Rajabzadeh, M., Ziaee, M., & Bozorgi-Amiri, A. (2016). Integrated approach in solving parallel machine scheduling and location (ScheLoc) problem. *International Journal of Industrial Engineering Computations*, *7*(4), pp. 573-584.

Raymond, F. E. (1929). Manufacturing Control through Economic Size of Production Lots. *SAE Transactions*, 524-533.

Resende, M. G., & Werneck, R. F. (2004). A hybrid heuristic for the p-median problem. *Journal of Heuristics*, *10*, 59-88.

Rockafellar, R. T. (2007). Coherent approaches to risk in optimization under uncertainty. In *OR Tools and Applications: Glimpses of Future Technologies* (pp. 38-61). Informs.

Rockafellar, R. T., & Wets, R. J. B. (1991). Scenarios and policy aggregation in optimization under uncertainty. *Mathematics of Operations Research*, *16*(1), 119-147.

Rodrigues, F., & Agra, A. (2021). An exact robust approach for the integrated berth allocation and quay crane scheduling problem under uncertain arrival times. *European Journal of Operational Research*, *295*(2), 499-516.

Rosenhead, J., Elton, M., & Gupta, S. K. (1972). Robustness and optimality as criteria for strategic decisions. *Journal of the Operational Research Society*, *23*(4), 413-431.

Rosing, K. E., Revelle, C. S., & Schilling, D. A. (1999). A gamma heuristic for the p-median problem. *European Journal of Operational Research*, *117*(3), 522-532.

Roy, B. (2010). Robustness in operational research and decision aiding: A multi-faceted issue. *European Journal of Operational Research*, *200*(3), 629-638.

Roziere, B., Lachaux, M. A., Chanussot, L., & Lample, G. (2020). Unsupervised translation of programming languages. *Advances in Neural Information Processing Systems*, *33*, pp. 20601-20611.

Savage, L. J. (1951). The theory of statistical decision. *Journal of the American Statistical association*, *46*(253), 55-67.

Schild, A. (1959). On inventory, production and employment scheduling. *Management Science*, *5*(2), 157-168.

Scholz, D. (2012). Integrated scheduling and location problems. In *Deterministic Global Optimization* (pp. 109-116). Springer, New York, NY.

Shahnejat-Bushehri, S., Tavakkoli-Moghaddam, R., Boronoos, M., & Ghasemkhani, A. (2021). A robust home health care routing-scheduling problem with temporal dependencies under uncertainty. *Expert Systems with Applications*, *182*, 115209.

Shapiro, S. S. & Wilk. MB 1965. An analysis of variance test for normality (complete samples). *Biometrika*, *52*(3/4), 591-611.

Shchepin, E. V., & Vakhania, N. (2005). An optimal rounding gives a better approximation for scheduling unrelated machines. *Operations Research Letters*, *33*(2), 127-133.

Shen, X., Sun, Y., & Zhang, M. (2016, July). An improved MOEA/D for multi-objective flexible job shop scheduling with release time uncertainties. In *2016 IEEE Congress on evolutionary computation (CEC)* (pp. 2950-2957). IEEE.

Siddiqui, A. W., & Verma, M. (2015). A bi-objective approach to routing and scheduling maritime transportation of crude oil. *Transportation Research Part D: Transport and Environment*, *37*, 65-78.

Siepak M. (2013). *Odporne algorytmy szeregowania zadań z niepewnością przedziałową* [Praca doktorska, Politechnika Wrocławska].

Snyder, L. V. (2006). Facility location under uncertainty: a review. *IIE transactions*, *38*(7), 547-564.

Sotskov, Y. N., Egorova, N. G., & Lai, T. C. (2009). Minimizing total weighted flow time of a set of jobs with interval processing times. *Mathematical and Computer Modelling*, *50*(3-4), 556-573.

Srivastava, B. (1998). An effective heuristic for minimising makespan on unrelated parallel machines. *Journal of the Operational Research Society*, *49*(8), 886-894.

Wald, A. (1945). Statistical decision functions which minimize the maximum risk. *Annals of Mathematics*, 265-280.

Wang, J., Li, X., Chu, J., & Tsui, K. L. (2020). A two-stage approach for resource allocation and surgery scheduling with assistant surgeons. *IEEE Access*, *8*, 49487-49496.

Wang, S., Wu, R., Chu, F., Yu, J., & Liu, X. (2020). An improved formulation and efficient heuristics for the discrete parallel-machine makespan ScheLoc problem. *Computers & Industrial Engineering*, *140*, 106238.

Wu, P., Wang, Y., Cheng, J., & Li, Y. (2022). An improved mixed-integer programming approach for bi-objective parallel machine scheduling and location. *Computers & Industrial Engineering*, *174*, 108813.

Wilcoxon F., Individual Comparisons by Ranking Methods, *Biometrics Bulletin*, *1*(6), 1945, 80-83.

Williamson, D. P., & Shmoys, D. B. (2011). *The design of approximation algorithms*. Cambridge university press.

Wu, C. C., Bai, D., Chen, J. H., Lin, W. C., Xing, L., Lin, J. C., & Cheng, S. R. (2021). Several variants of simulated annealing hyper-heuristic for a single-machine

scheduling with two-scenario-based dependent processing times. *Swarm and Evolutionary Computation*, *60*, 100765.

Wu, C. W., Brown, K. N., & Beck, J. C. (2005). Scheduling with uncertain release dates. *AICS'05*, 397.

Wu, P., Wang, Y., Cheng, J., & Li, Y. (2022). An improved mixed-integer programming approach for bi-objective parallel machine scheduling and location. *Computers & Industrial Engineering*, 108813.

Yager, R. R. (1988). On ordered weighted averaging aggregation operators in multicriteria decisionmaking. *IEEE Transactions on systems, Man, and Cybernetics*, *18*(1), 183-190.

Yang, J., & Yu, G. (2002). On the robust single machine scheduling problem. *Journal of Combinatorial Optimization*, *6*(1), 17-33.

Yang-Kuei, L., & Chi-Wei, L. (2013). Dispatching rules for unrelated parallel machine scheduling with release dates. *The International Journal of Advanced Manufacturing Technology*, *67*(1-4), 269-279.

Yu, G., & Jacobson, S. H. (2020). Approximation algorithms for scheduling C-benevolent jobs on weighted machines. *IISE Transactions*, *52*(4), 432-443.

Yue, F., Song, S., Zhang, Y., Gupta, J. N., & Chiong, R. (2018). Robust single machine scheduling with uncertain release times for minimising the maximum waiting time. *International Journal of Production Research*, *56*(16), 5576-5592.

Zhang, C., Li, Y., Cao, J., & Wen, X. (2022a). On the mass COVID-19 vaccination scheduling problem. *Computers & Operations Research*, 105704.

Zhang, C., Li, Y., Cao, J., Yang, Z., & Coelho, L. C. (2022b). Exact and matheuristic methods for the parallel machine scheduling and location problem with delivery time and due date. *Computers & Operations Research*, *147*, 105936.

Zhang, G., Hu, Y., Sun, J., & Zhang, W. (2020). An improved genetic algorithm for the flexible job shop scheduling problem with multiple time constraints. *Swarm and Evolutionary Computation*, *54*, 100664.

Zhang, Z., Ding, S., & Jia, W. (2019). A hybrid optimization algorithm based on cuckoo search and differential evolution for solving constrained engineering problems. *Engineering Applications of Artificial Intelligence*, *85*, 254-268.

Zhong, R., Stern, M., & Klein, D. (2020). Semantic scaffolds for pseudocode-to-code generation. *arXiv preprint arXiv:2005.05927*

김동욱. (2021). *Network Design and Route Planning for Integrated Logistics with Drones* [Doctoral dissertation, 서울대학교 대학원].

Dodatek

W dodatku umieszczono wartości liczbowe rezultatów optymalizacji deterministycznej i odpornej, które zaprezentowano na wykresach w rozprawie lub wykorzystywano podczas badań statystycznych. W opisach tabel wykorzystano następujące skrótowe oznaczenia:

A. $C_{\max}(x_{\mathrm{GD}};\mathrm{S}) \triangleq C_{\mathrm{GD}}$, $C_{\max}(x_{\mathrm{BF}};\mathrm{S}) \triangleq C_{\mathrm{BF}}$, $C_{\max}(x_{\mathrm{SA}};\mathrm{S}) \triangleq C_{\mathrm{SA}}^{(e)}$, $e \in \{\min, \mathrm{med}, \max\}$,

B. $\tilde{Z}(\tilde{x}_{\mathrm{DR}}) \triangleq \tilde{Z}_{\mathrm{DR}}$, $\tilde{Z}(\tilde{x}_{\mathrm{GR}}) \triangleq \tilde{Z}_{\mathrm{GR}}$, $\tilde{Z}(\tilde{x}_{\mathrm{TS}}) \triangleq \tilde{Z}_{\mathrm{TS}}^{(\mathrm{e})}$, $e \in \{\min, \mathrm{med}, \max\}$,

C. $\bar{Z}(\tilde{x}_{\mathrm{GR}}) \triangleq \bar{Z}_{\mathrm{GR}}$, $\bar{Z}(\tilde{x}_{\mathrm{DR}}) \triangleq \bar{Z}_{\mathrm{DR}}$, $\bar{Z}(\bar{x}_{\mathrm{SR}}) \triangleq \bar{Z}_{\mathrm{SR}}$,

D. $C_{\max}(x_{\mathrm{GN}}, y_{\mathrm{GN}};\mathrm{S}_l) \triangleq C_{\mathrm{GN}}^{(\mathrm{e})}$, $C_{\max}(x_{\mathrm{HS}}, y_{\mathrm{HS}};\mathrm{S}_l) \triangleq C_{\mathrm{HS}}^{(\mathrm{e})}$, $\in \{\min, \mathrm{med}, \max\}$,
$C_{\max}(x_{\mathrm{EG}}, y_{\mathrm{EG}};\mathrm{S}_l) \triangleq C_{\mathrm{EG}}$, $C_{\max}(x_{\mathrm{GD}}, y_{\mathrm{median}};\mathrm{S}_l) \triangleq C_{\mathrm{median}}$, $C_{\max}(x_{\mathrm{GD}}, y_{\mathrm{center}};\mathrm{S}_l) \triangleq C_{\mathrm{center}}$.

Wszystkie czasy podane w podpunktach A-C dla algorytmów niedeterministycznych są medianami czasów obliczeń z dwudziestu pięciu uruchomień algorytmów, $t_{\mathrm{SA}}^{(\mathrm{med})} \triangleq t_{\mathrm{SA}}$, $t_{\mathrm{TS}}^{(\mathrm{med})} \triangleq t_{\mathrm{TS}}$. Dla losowo wygenerowanych instancji, dla których rezultaty optymalizacji zaprezentowano w tabelach B6, C4, D10 oraz D11, ustalono $\underline{p} = 5$ oraz $\underline{r} = 0$.

A. Rezultaty optymalizacji dla problemu $Rm|r_{i,j}|C_{\max}$

Tabela A1. Wyniki zaprezentowane na rysunkach 2.3-2.5 (badanie 2.2)

n	$m = 2$		$m = 5$		$m = 10$	
	C_{BF}	C_{GD}	C_{BF}	C_{BF}	C_{BF}	C_{GD}
10	811	811	784	784	338	338
20	811	811	784	784	338	338
30	878	811	784	784	338	338
40	932	878	784	784	338	338
50	932	932	784	784	338	338
60	1016	932	784	784	501	338
70	1103	1023	784	784	501	501
80	1214	1069	784	784	501	501
90	1249	1186	784	784	501	501
100	1376	1225	784	784	501	501
110	1459	1362	784	784	501	501
120	1552	1476	784	784	501	501
130	1670	1538	784	784	501	501
140	1788	1662	807	784	501	501
150	1914	1792	862	784	501	501
160	2020	1901	896	802	501	501
170	2122	1998	932	806	501	501
180	2244	2113	946	811	515	501
190	2332	2220	975	811	540	501
200	2469	2326	998	853	563	501
210	2580	2431	1045	860	566	501
220	2705	2504	1070	906	589	501
230	2799	2619	1081	923	602	501

240	2909	2709	1096	930	595	501
250	3037	2798	1136	950	631	508
260	3152	2919	1172	981	632	524
270	3270	3014	1193	983	647	526
280	3355	3128	1252	1003	664	541
290	3452	3219	1281	1021	664	551
300	3588	3321	1310	1021	676	551

Tabela A2. Wyniki zaprezentowane na rysunkach 2.3-2.5 (badanie 2.2)

n	$m = 2$			$m = 5$			$m = 10$		
	$C_{\mathrm{SA}}^{(\min)}$	$C_{\mathrm{SA}}^{(\mathrm{med})}$	$C_{\mathrm{SA}}^{(\max)}$	$C_{\mathrm{SA}}^{(\min)}$	$C_{\mathrm{SA}}^{(\mathrm{med})}$	$C_{\mathrm{SA}}^{(\max)}$	$C_{\mathrm{SA}}^{(\min)}$	$C_{\mathrm{SA}}^{(\mathrm{med})}$	$C_{\mathrm{SA}}^{(\max)}$
10	811	875	1072	811	941,5	1039	522	892	994
20	811	1153,5	1215	875	1026	1106	832	957	1059
30	811	1296,5	1392	926	1093,5	1183	772	1017	1123
40	1187	1452,5	1575	1069	1160,5	1237	971	1035	1113
50	1356	1593	1707	1087	1213,5	1312	1021	1081,5	1180
60	1624	1738,5	1874	1180	1279	1362	1040	1108	1179
70	1707	1863,5	2024	1234	1323,5	1456	1077	1144,5	1215
80	1759	1929,5	2157	1235	1358,5	1485	1063	1140	1208
90	1863	2099,5	2335	1295	1392	1524	1091	1193,5	1232
100	2035	2228,5	2427	1388	1483	1588	1103	1216,5	1252
110	2157	2427,5	2519	1459	1538	1620	1170	1235	1306
120	2364	2570	2746	1521	1606,5	1707	1151	1278	1339
130	2452	2677	2778	1544	1672	1760	1210	1320	1417
140	2676	2804	2946	1588	1693,5	1783	1234	1323,5	1391
150	2850	2971,5	3150	1651	1764	1926	1287	1359	1419
160	2996	3120	3330	1730	1810	1936	1272	1412,5	1496
170	3061	3260,5	3361	1748	1890,5	2002	1367	1432	1508
180	3069	3327	3523	1866	1929,5	2000	1367	1459,5	1556
190	3369	3528,5	3681	1874	1974	2059	1406	1477	1545
200	3414	3616,5	3775	1922	2021,5	2129	1416	1513	1603
210	3619	3840	4024	2022	2110,5	2243	1422	1532	1624
220	3755	3907	4103	2045	2133	2330	1447	1550,5	1691
230	3733	4074,5	4251	2064	2218	2348	1505	1599	1677
240	3867	4134,5	4353	2133	2288,5	2371	1532	1631	1731
250	4061	4277	4450	2177	2326	2425	1602	1655,5	1723
260	4315	4482	4713	2191	2384	2509	1602	1678	1786
270	4272	4659	4803	2341	2450,5	2574	1616	1710,5	1827
280	4570	4731	4974	2405	2505	2601	1653	1758,5	1803
290	4642	4815,5	5061	2474	2591,5	2683	1667	1756,5	1842
300	4763	4982	5238	2482	2608	2719	1679	1790	1883

Tabela A3. Wyniki zaprezentowane na rysunkach 2.6-2.8 (badanie 2.2)

m	$n = 100$		$n = 200$		$n = 300$	
	C_{BF}	C_{GD}	C_{BF}	C_{GD}	C_{BF}	C_{GD}
2	1376	1225	2469	2326	3588	3321
3	893	870	1554	1387	2100	1914
4	784	784	1159	1028	1543	1328
5	784	784	998	853	1310	1021
6	551	551	817	739	1052	880
7	490	501	701	621	890	759
8	501	501	658	560	795	692
9	501	501	577	501	703	602
10	501	501	563	501	676	551

11	383	330	523	430	639	506
12	412	301	528	397	689	469
13	421	265	558	396	692	440
14	430	257	569	345	682	425
15	455	257	584	345	659	402

Tabela A4. Wyniki zaprezentowane na rysunkach 2.6-2.8 (badanie 2.2)

m	$n = 100$			$n = 200$			$n = 300$		
	$C_{\mathrm{SA}}^{(\min)}$	$C_{\mathrm{SA}}^{(\mathrm{med})}$	$C_{\mathrm{SA}}^{(\max)}$	$C_{\mathrm{SA}}^{(\min)}$	$C_{\mathrm{SA}}^{(\mathrm{med})}$	$C_{\mathrm{SA}}^{(\max)}$	$C_{\mathrm{SA}}^{(\min)}$	$C_{\mathrm{SA}}^{(\mathrm{med})}$	$C_{\mathrm{SA}}^{(\max)}$
2	2035	2228,5	2427	3414	3616,5	3775	4763	4982	5238
3	1665	1877	2002	2612	2788	2991	3531	3671	3981
4	1491	1551	1674	2230	2341,5	2499	2816	3001,5	3102
5	1388	1483	1588	1922	2021,5	2129	2482	2608	2719
6	1295	1376	1489	1757	1815,5	1991	2241	2331	2456
7	1251	1356	1399	1613	1702	1799	2036	2151	2299
8	1231	1301	1371	1549	1607	1784	1875	1936	2011
9	1132	1203	1273	1436	1502	1603	1806	1939	1999
10	1103	1216,5	1252	1416	1513	1603	1679	1790	1883
11	1116	1201,5	1250	1399	1499	1581	1628	1698	1789
12	1060	1141	1211	1337	1451	1590	1552	1603	1686
13	1060	1149,5	1221	1312	1387	1467	1531	1611,5	1689
14	1060	1149,5	1241	1304	1378	1441	1470	1571,5	1701
15	1075	1127	1241	1225	1283	1389	1464	1588	1721

Tabela A5. Wyniki zaprezentowane na rysunku 2.9 (badanie 2.2)

n	$m = 2$		
	t_{BF} [ms]	t_{GD} [ms]	t_{SA} [ms]
10	8	14	1
20	12	24	2
30	23	33	5
40	10	45	8
50	23	53	8
60	28	59	32
70	291	77	57
80	33	87	37
90	8	104	61
100	40	114	80
110	44	108	142
120	299	128	95
130	52	147	68
140	58	147	77
150	59	159	98
160	182	176	147
170	65	189	119
180	255	215	266
190	76	199	217
200	114	224	185
210	81	237	193
220	90	245	180
230	78	244	162
240	119	265	173
250	47	283	236
260	137	285	224
270	288	304	187

280	168	294	214
290	214	312	218
300	175	308	219

Tabela A6. Wyniki optymalizacji dla losowo wygenerowanych instancji (badanie 2.4)

$(m, n, \bar{r}, \bar{p})$	C_{BF}	C_{GD}	t_{BF} [ms]	t_{GD} [ms]
(2, 106, 409, 63)	1350	1209	3	31
(9, 184, 610, 13)	318	318	72	546
(12, 229, 976, 82)	698	509	215	1220
(5, 196, 199, 21)	392	310	19	369
(9, 159, 771, 28)	378	378	63	432
(12, 37, 580, 16)	170	170	40	36
(3, 111, 881, 72)	1319	1087	4	63
(10, 214, 84, 34)	221	180	79	622
(15, 10, 586, 26)	73	73	24	6
(11, 209, 265, 16)	210	152	168	1194
(10, 189, 167, 35)	237	175	85	742
(7, 155, 625, 33)	398	375	42	360
(15, 47, 690, 39)	310	146	207	68
(7, 230, 875, 44)	667	553	49	723
(4, 87, 324, 25)	310	267	7	62
(5, 150, 287, 83)	741	576	13	200
(15, 284, 746, 85)	735	435	545	2208
(12, 71, 513, 19)	141	138	70	146
(11, 65, 944, 69)	387	312	47	90
(13, 62, 897, 95)	377	285	77	96
(11, 275, 159, 20)	249	169	164	1649
(8, 90, 309, 47)	236	235	26	170
(14, 235, 543, 40)	421	263	380	1792
(13, 18, 993, 23)	210	206	32	14
(10, 240, 407, 68)	466	395	102	1132
(13, 249, 997, 29)	522	390	377	1822
(7, 42, 332, 35)	239	239	14	27
(12, 75, 429, 22)	159	153	83	170
(5, 248, 56, 51)	776	626	19	297
(14, 229, 706, 27)	412	280	416	1610
(5, 226, 466, 10)	342	339	32	514
(4, 28, 381, 27)	270	271	13	7
(15, 22, 408, 29)	136	136	57	23
(11, 211, 616, 38)	404	315	156	1123
(4, 71, 147, 93)	483	411	6	38
(15, 298, 398, 27)	371	205	491	3699
(5, 274, 318, 82)	1415	1097	29	566
(3, 38, 846, 87)	720	691	3	13
(12, 11, 647, 44)	63	70	15	8
(8, 235, 855, 63)	748	624	67	1145
(13, 116, 612, 78)	412	313	128	345
(12, 36, 955, 17)	337	337	40	35
(8, 63, 984, 22)	505	505	22	93
(5, 135, 210, 23)	302	250	14	162
(9, 153, 820, 89)	643	532	51	390
(7, 270, 705, 67)	845	681	50	884
(8, 88, 434, 38)	244	232	26	121
(10, 208, 884, 44)	514	514	98	828
(9, 159, 455, 90)	460	379	47	389
(15, 114, 721, 65)	383	273	220	372

(2, 294, 591, 64)	4236	3772	4	195
(10, 118, 223, 60)	241	189	48	247
(11, 141, 617, 21)	310	310	112	486
(9, 203, 633, 83)	608	526	86	854
(11, 166, 557, 68)	426	374	114	804
(7, 64, 551, 52)	311	311	15	65
(10, 200, 701, 72)	525	462	82	725
(12, 168, 787, 88)	561	405	140	613
(14, 260, 81, 24)	234	123	247	1314
(13, 89, 994, 50)	339	304	107	198
(7, 176, 827, 34)	483	486	38	399
(12, 77, 644, 12)	260	260	72	136
(2, 254, 796, 29)	1893	1748	4	187
(9, 120, 481, 38)	273	231	48	258
(15, 235, 192, 94)	479	246	266	1276
(3, 231, 173, 77)	2088	1759	7	154
(13, 192, 616, 27)	335	301	219	850
(12, 107, 546, 14)	162	148	95	275
(10, 57, 291, 34)	142	138	26	63
(12, 212, 666, 85)	661	420	148	920
(13, 81, 651, 85)	357	287	93	157
(4, 203, 663, 31)	796	651	12	287
(5, 224, 337, 80)	1209	940	19	344
(12, 168, 148, 51)	256	171	98	541
(3, 264, 955, 88)	3000	2486	8	268
(3, 50, 400, 43)	381	385	2	15
(12, 237, 382, 95)	534	365	144	1128
(5, 62, 571, 82)	511	462	6	38
(13, 162, 549, 48)	371	238	173	615
(8, 85, 674, 49)	323	324	26	107
(11, 277, 350, 38)	375	274	142	1427
(9, 292, 979, 42)	675	606	118	1512
(13, 146, 811, 85)	585	362	186	609
(2, 285, 851, 36)	2545	2267	4	292
(3, 177, 969, 46)	1230	1090	7	160
(4, 165, 858, 40)	789	733	14	225
(3, 46, 246, 34)	276	252	2	13
(13, 191, 52, 10)	126	82	161	683
(5, 144, 290, 56)	572	461	12	151
(9, 93, 397, 17)	200	200	37	155
(11, 83, 705, 39)	305	305	65	160
(4, 13, 489, 76)	343	343	2	2
(15, 185, 822, 83)	650	372	293	979
(13, 246, 199, 30)	285	169	274	1264
(3, 109, 53, 85)	1079	957	3	37
(5, 26, 811, 27)	549	549	5	8
(7, 77, 764, 45)	408	408	18	83
(10, 110, 675, 15)	359	359	62	225
(8, 74, 903, 31)	428	428	23	91
(3, 214, 306, 17)	665	581	6	177

B. Rezultaty optymalizacji dla problemu $\boldsymbol{Rm|r_{i,j}^{-} \leq r_{i,j} \leq r_{i,j}^{+}|Reg(C_{\max})}$

Tabela B1. Wyniki zaprezentowane na rysunkach 3.2-3.4 (badanie 3.1)

n	$m = 2$		$m = 5$		$m = 10$	
	$\tilde{Z}_{\mathrm{DR}}$	$\tilde{Z}_{\mathrm{GR}}$	$\tilde{Z}_{\mathrm{DR}}$	$\tilde{Z}_{\mathrm{GR}}$	$\tilde{Z}_{\mathrm{DR}}$	$\tilde{Z}_{\mathrm{GR}}$
10	144,0	140,0	222,0	222,0	335,0	335,0
20	91,0	91,0	231,0	231,0	312,0	312,0
30	126,0	130,0	228,0	228,0	226,0	226,0
40	230,5	209,5	228,0	228,0	226,0	226,0
50	238,0	229,0	236,0	228,0	226,0	226,0
60	221,0	229,0	255,0	230,0	226,0	226,0
70	178,0	212,0	205,0	201,0	226,0	226,0
80	160,0	217,0	222,0	216,0	226,0	226,0
90	147,0	180,0	236,0	232,0	226,0	226,0
100	162,0	203,0	239,0	247,0	226,0	226,0
110	143,0	196,0	249,0	264,0	233,0	226,0
120	138,0	197,0	263,4	260,4	227,0	227,0
130	135,0	172,0	196,0	204,0	242,0	208,0
140	149,0	162,0	213,0	227,0	234,0	208,0
150	135,0	140,0	228,6	247,6	273,0	220,0
160	124,0	160,0	235,4	233,4	263,0	244,0
170	135,0	135,0	230,2	226,2	294,0	255,0
180	142,0	157,0	209,6	217,6	274,0	262,0
190	125,5	172,5	200,4	220,4	264,0	263,0
200	123,5	168,5	229,4	219,4	312,0	268,0
210	142,0	166,0	221,0	205,0	321,0	275,0
220	129,5	150,5	209,6	212,6	308,0	285,0
230	130,5	143,5	208,6	197,6	310,0	291,0
240	119,0	156,0	206,8	197,8	314,7	289,7
250	147,0	138,0	207,8	199,8	297,1	278,1
260	147,0	138,0	207,8	199,8	297,1	278,1
270	147,0	138,0	207,8	199,8	297,1	278,1
280	147,0	138,0	207,8	199,8	297,1	278,1
290	147,0	138,0	207,8	199,8	297,1	278,1
300	147,0	138,0	207,8	199,8	297,1	278,1

Tabela B2. Wyniki zaprezentowane na rysunkach 3.2-3.4 (badanie 3.1)

n	$m = 2$			$m = 5$			$m = 10$		
	$\tilde{Z}_{\mathrm{TS}}^{(\min)}$	$\tilde{Z}_{\mathrm{TS}}^{(\mathrm{med})}$	$\tilde{Z}_{\mathrm{TS}}^{(\max)}$	$\tilde{Z}_{\mathrm{TS}}^{(\min)}$	$\tilde{Z}_{\mathrm{TS}}^{(\mathrm{med})}$	$\tilde{Z}_{\mathrm{TS}}^{(\max)}$	$\tilde{Z}_{\mathrm{TS}}^{(\min)}$	$\tilde{Z}_{\mathrm{TS}}^{(\mathrm{med})}$	$\tilde{Z}_{\mathrm{TS}}^{(\max)}$
10	145,0	190,5	320,0	248,0	271,0	343,0	335,0	338,0	430,0
20	199,0	226,5	291,0	249,0	300,0	348,0	315,0	393,0	437,0
30	285,0	372,5	453,0	272,0	327,5	397,0	279,0	318,0	385,0
40	443,5	493,5	550,0	318,0	370,5	405,0	226,0	330,0	434,0
50	485,0	523,0	555,0	345,0	434,0	471,0	322,0	360,3	423,0
60	486,0	531,0	618,0	392,0	496,5	524,0	359,0	409,0	501,0
70	525,0	601,0	691,0	439,0	481,5	526,0	367,0	425,0	490,0
80	493,0	607,0	718,0	461,0	560,5	630,0	362,0	463,0	531,0
90	651,0	700,5	758,0	501,0	620,0	674,0	402,2	481,0	526,0
100	670,0	709,5	892,0	621,0	653,0	724,0	463,0	508,0	554,0
110	713,0	814,0	862,0	692,0	753,5	782,0	477,1	562,0	595,0
120	647,0	827,0	922,0	701,4	792,9	842,4	493,0	582,0	615,0
130	697,0	802,0	891,0	647,0	718,0	819,0	482,0	557,4	662,0
140	783,0	820,5	953,0	725,0	793,0	904,0	574,2	609,0	714,0
150	803,0	922,5	1008,0	760,6	814,6	857,6	591,0	630,0	710,0

160	821,0	930,0	1069,0	815,4	862,4	992,4	628,0	660,0	736,0
170	855,0	972,0	1084,0	780,2	880,7	1032,2	625,0	716,3	780,0
180	842,0	983,0	1099,0	850,6	967,6	1050,6	657,0	753,0	873,0
190	827,5	1008,5	1163,5	877,4	948,9	1052,4	645,2	740,0	835,0
200	947,5	1108,5	1194,5	918,4	997,4	1087,4	719,5	789,2	818,0
210	996,0	1120,0	1257,0	886,0	1010,0	1133,0	748,7	822,7	821,7
220	1046,5	1211,5	1276,5	910,6	1026,6	1161,6	768,9	857,9	915,9
230	1121,5	1221,5	1464,5	966,6	1065,6	1194,6	803,7	910,7	982,7
240	1056,0	1218,0	1400,0	1051,8	1108,8	1229,8	844,1	895,1	1003,1
250	1172,0	1304,0	1352,0	1044,8	1153,8	1237,8	832,1	940,1	1001,1
260	1127,0	1264,0	1402,0	1040,8	1131,8	1282,8	802,1	900,1	982,1
270	1154,0	1273,0	1357,0	1034,8	1115,8	1279,8	838,1	936,1	984,1
280	1119,0	1304,0	1408,0	1071,8	1137,8	1223,8	854,1	907,1	970,1
290	1101,0	1288,0	1485,0	1076,8	1153,8	1217,8	843,1	908,1	965,1
300	1145,0	1280,0	1411,0	1017,8	1130,8	1308,8	841,2	911,9	969,1

Tabela B3. Wyniki zaprezentowane na rysunkach 3.5-3.7 (badanie 3.1)

m	$n = 100$		$n = 200$		$n = 300$	
	$\tilde{Z}_{\mathrm{DR}}$	$\tilde{Z}_{\mathrm{GR}}$	$\tilde{Z}_{\mathrm{DR}}$	$\tilde{Z}_{\mathrm{GR}}$	$\tilde{Z}_{\mathrm{DR}}$	$\tilde{Z}_{\mathrm{GR}}$
2	162,0	203,0	123,5	168,5	147,0	138,0
3	192,3	212,3	164,7	178,7	177,3	179,3
4	218,5	224,5	191,8	191,8	213,0	189,0
5	239,0	247,0	229,4	219,4	207,8	199,8
6	273,0	240,0	250,0	229,0	225,5	203,5
7	272,0	231,0	262,0	245,0	246,3	223,3
8	222,0	222,0	289,8	268,8	284,9	235,9
9	229,0	226,0	307,0	290,0	295,9	273,9
10	226,0	226,0	312,0	268,0	297,1	278,1
11	226,0	226,0	320,0	267,0	321,6	287,6
12	317,0	269,0	319,0	287,0	329,0	292,0
13	280,0	263,0	277,0	254,0	346,0	276,0
14	287,0	249,0	290,0	252,0	305,0	273,0
15	249,0	247,0	260,0	213,0	298,0	247,0

Tabela B4. Wyniki zaprezentowane na rysunkach 3.5-3.7 (badanie 3.1)

m	$n = 100$			$n = 200$			$n = 300$		
	$\tilde{Z}_{\mathrm{TS}}^{(\min)}$	$\tilde{Z}_{\mathrm{TS}}^{(\mathrm{med})}$	$\tilde{Z}_{\mathrm{TS}}^{(\max)}$	$\tilde{Z}_{\mathrm{TS}}^{(\min)}$	$\tilde{Z}_{\mathrm{TS}}^{(\mathrm{med})}$	$\tilde{Z}_{\mathrm{TS}}^{(\max)}$	$\tilde{Z}_{\mathrm{TS}}^{(\min)}$	$\tilde{Z}_{\mathrm{TS}}^{(\mathrm{med})}$	$\tilde{Z}_{\mathrm{TS}}^{(\max)}$
2	670,0	709,5	892,0	947,5	1108,5	1194,5	1145,0	1280,0	1411,0
3	659,3	726,8	829,3	936,7	1134,7	1217,7	1147,3	1323,3	1419,3
4	644,5	693,0	793,5	932,8	1037,8	1186,8	1132,0	1222,0	1307,0
5	632,0	661,0	729,0	918,4	997,4	1087,4	1017,8	1130,8	1308,8
6	621,0	653,0	724,0	870,0	970,0	1028,0	1008,5	1105,5	1195,5
7	530,0	546,5	658,0	804,0	935,0	952,0	924,3	998,3	1164,3
8	509,0	550,0	591,0	785,8	888,8	915,8	836,9	977,9	1033,9
9	478,0	536,0	611,0	742,7	840,7	900,7	863,9	933,9	1002,9
10	463,0	508,0	554,0	719,5	789,2	818,0	841,2	911,9	969,1
11	434,0	480,5	547,0	739,4	799,4	842,4	830,6	881,6	950,6
12	492,0	555,0	609,0	718,0	767,0	843,0	763,0	833,0	896,0
13	484,0	545,5	586,0	678,1	756,1	811,1	759,7	825,7	884,7
14	485,0	529,5	589,0	667,0	761,0	809,0	750,3	822,3	892,3
15	490,0	514,5	555,0	658,4	729,4	799,4	740,7	828,7	844,7

Tabela B5. Wyniki zaprezentowane na rysunku 3.8 (badanie 3.1)

n	$m = 2$		
	t_{DR} [ms]	t_{GR} [ms]	t_{TS} [ms]
10	6	86	337
20	38	711	431
30	59	2708	1356
40	46	6963	2602
50	73	15541	5239
60	85	30121	6765
70	116	52816	9176
80	147	86380	11784
90	184	133602	14561
100	224	196751	20022
110	324	284127	23973
120	317	407249	26280
130	366	536784	35713
140	461	710427	35598
150	484	923290	41068
160	576	1182777	52834
170	694	1493207	59502
180	721	1854126	73977
190	816	2270984	81057
200	877	2785245	90834
210	970	3364663	101239
220	997	4029366	112343
230	1100	4778449	124343
240	1199	5647110	134343
250	1317	6602931	144343
260	1505	6919584	155343
270	1386	7414928	166643
280	1377	7701143	177343
290	1476	8605922	187343
300	1337	8998274	194343

Tabela B6. Wyniki optymalizacji dla losowo wygenerowanych instancji (badanie 3.2)

$(m, n, \bar{r}, \bar{p})$	$\tilde{Z}_{\mathrm{DR}}$	$\tilde{Z}_{\mathrm{GR}}$	t_{DR} [ms]	t_{GR} [ms]
(7, 28, 729, 59)	324	324	10	1301
(2, 227, 849, 25)	257	277	172	832325
(6, 115, 473, 77)	263,7	264,7	140	186990
(3, 195, 282, 14)	55,3	57,3	191	701693
(14, 102, 92, 18)	52,3	42,3	259	297938
(4, 124, 419, 19)	120	109	125	165323
(15, 141, 706, 10)	346	346	504	1129423
(4, 116, 735, 65)	358	353	105	125820
(14, 237, 530, 12)	236	236	1356	7538224
(11, 196, 998, 69)	348	348	699	2772428
(7, 133, 895, 52)	299	309	208	385960
(15, 87, 812, 70)	410	410	225	177869
(2, 43, 293, 12)	91	88	6	1652
(5, 100, 612, 45)	193	174	87	90890
(14, 240, 146, 11)	41,4	39,4	1349	7907724
(14, 27, 958, 40)	296	296	18	2565
(10, 178, 213, 52)	109,8	102,8	577	1734834
(13, 213, 819, 14)	225	225	990	4623656
(11, 120, 170, 48)	106	95	263	427418
(9, 196, 830, 77)	376	369	623	2263376

(2, 105, 402, 66)	165,5	188,5	37	42410
(9, 160, 873, 11)	93	93	385	1030228
(5, 90, 287, 75)	161,2	161,2	80	62698
(11, 14, 982, 95)	553	553	6	226
(6, 106, 280, 79)	163,3	150,3	133	138348
(13, 44, 660, 51)	347	347	41	12821
(6, 201, 57, 23)	25,7	23,7	406	1614378
(7, 64, 147, 18)	73	55	49	24823
(14, 277, 206, 18)	95,8	75,8	1996	13856814
(11, 70, 928, 38)	385	385	117	55973
(14, 100, 175, 40)	123,9	108,9	315	281564
(2, 65, 694, 68)	288,5	281,5	17	7041
(10, 240, 398, 15)	145	124	990	5471609
(6, 207, 438, 32)	193,8	192,8	423	1816023
(12, 47, 350, 92)	187	153	45	14469
(8, 171, 525, 21)	182	182	434	1163073
(15, 131, 484, 28)	242	242	451	843433
(7, 96, 78, 35)	40,6	39,6	108	114830
(6, 183, 174, 91)	77	97	358	1130655
(13, 161, 520, 99)	296	296	569	1563683
(15, 60, 265, 77)	183	153	91	45209
(9, 113, 866, 58)	185	185	202	275709
(7, 297, 329, 75)	131,4	156,4	1088	8723826
(5, 201, 905, 35)	239	213	336	1337315
(7, 183, 997, 76)	352	346	424	1327015
(13, 287, 937, 19)	417	417	2024	14661355
(14, 164, 298, 81)	175	158	641	1810915
(3, 297, 487, 96)	141,7	182,7	452	3638431
(5, 287, 915, 95)	440,4	369,4	800	5374508
(7, 141, 355, 75)	185,3	174,3	231	484831
(8, 38, 874, 79)	493	493	20	4654
(3, 91, 418, 81)	123,7	163,7	41	37486
(2, 204, 935, 18)	337	349	136	549448
(4, 102, 156, 80)	85,3	85,3	81	77169
(8, 109, 940, 31)	442	442	171	209894
(2, 248, 536, 88)	99	136	202	1190385
(11, 102, 553, 81)	273	257	233	231580
(9, 297, 283, 67)	136,3	130,3	1410	11353664
(15, 67, 187, 62)	142	107	112	68736
(3, 235, 62, 16)	17,3	11,3	299	1443676
(14, 177, 316, 11)	112	112	731	2446705
(4, 229, 464, 21)	182,5	172,5	352	1759805
(6, 68, 513, 52)	213	200	46	26485
(10, 278, 817, 93)	367	368	1361	9808382
(9, 189, 146, 26)	60,2	53,2	548	1957642
(10, 206, 803, 44)	242	263	705	3033755
(4, 163, 208, 28)	73	78	200	471112
(6, 263, 356, 88)	164,7	150,7	745	4601273
(8, 177, 197, 43)	88,6	87,6	447	1337938
(7, 223, 207, 89)	109,7	112,7	579	2840082
(7, 65, 91, 43)	57,3	59,3	50	26210
(13, 19, 693, 81)	220	174	11	751
(7, 230, 377, 24)	126	108	647	3220964
(4, 296, 534, 83)	169,8	199,8	657	4826671
(7, 149, 395, 87)	241,1	220,1	292	601192
(3, 83, 389, 22)	76	64	42	26359
(3, 149, 720, 10)	98	98	136	245968
(12, 197, 435, 11)	161	161	841	3122204

(8, 296, 704, 82)	343,1	345,1	1248	9954948
(6, 70, 404, 90)	241	233	50	29026
(14, 240, 169, 89)	96,6	84,6	1498	7948822
(9, 245, 327, 59)	167,3	174,3	986	5353893
(11, 54, 134, 63)	108	91	53	21944
(14, 152, 259, 77)	168,2	155,2	578	1363859
(2, 97, 581, 49)	214	233	33	31724
(11, 206, 692, 69)	333	308	825	3363656
(14, 210, 91, 39)	42,7	40,7	1116	4721873
(12, 186, 731, 43)	225	235	715	2494373
(8, 90, 467, 33)	146	156	106	104354
(2, 80, 127, 60)	51,5	77,5	21	15770
(11, 202, 215, 11)	90,5	85,5	768	3119473
(8, 207, 261, 79)	134,8	136,8	573	2452837
(3, 149, 842, 87)	296,7	360,7	129	246702
(13, 182, 289, 15)	119	124	748	2505057
(11, 11, 328, 69)	145	145	3	121
(5, 89, 934, 14)	300	300	67	58812
(8, 146, 398, 88)	246,5	245,5	334	638843
(12, 146, 674, 11)	247	247	423	979252
(15, 32, 648, 55)	269	269	30	4862
(15, 109, 204, 27)	103	99	315	418030

C. Rezultaty optymalizacji dla problemu $\boldsymbol{Rm|r_j^- \le r_j \le r_j^+|Reg(C_{max})}$

Tabela C1. Wyniki zaprezentowane na rysunkach 3.9-3.11 (badanie 3.3)

n	$m = 2$			$m = 5$			$m = 10$		
	$\bar{Z}_{\mathrm{DR}}$	$\bar{Z}_{\mathrm{GR}}$	$\bar{Z}_{\mathrm{SR}}$	$\bar{Z}_{\mathrm{DR}}$	$\bar{Z}_{\mathrm{GR}}$	$\bar{Z}_{\mathrm{SR}}$	$\bar{Z}_{\mathrm{DR}}$	$\bar{Z}_{\mathrm{GR}}$	$\bar{Z}_{\mathrm{SR}}$
10	17,0	17,0	17,0	5,0	7,0	7,0	5,0	6,0	7,0
20	1,0	1,0	15,0	1,0	1,0	1,0	0,0	0,0	0,0
30	20,0	5,0	17,0	1,0	1,0	1,0	0,0	0,0	0,0
40	65,0	45,0	38,0	23,0	8,0	12,0	0,0	0,0	0,0
50	94,0	118,0	107,0	23,0	8,0	12,0	0,0	0,0	0,0
60	98,0	152,0	141,0	23,0	8,0	12,0	0,0	0,0	0,0
70	80,5	123,5	114,5	34,0	33,0	17,0	5,0	0,0	0,0
80	85,5	131,5	153,5	39,0	34,0	30,0	10,0	8,0	5,0
90	118,5	129,5	175,5	39,0	35,0	30,0	10,0	8,0	5,0
100	59,5	89,5	198,5	39,0	35,0	30,0	10,0	8,0	5,0
110	69,0	79,0	213,0	39,0	35,0	30,0	10,0	8,0	5,0
120	49,0	90,0	203,0	39,0	36,0	30,0	10,0	8,0	6,0
130	48,0	102,0	236,0	39,0	36,0	30,0	11,0	8,0	13,0
140	71,5	103,5	239,5	39,0	42,0	30,0	18,0	11,0	13,0
150	70,0	101,0	264,0	34,0	47,0	38,0	18,0	11,0	18,0
160	51,5	94,5	288,5	59,0	64,0	52,0	23,0	11,0	18,0
170	49,5	95,5	316,5	46,0	85,0	89,0	23,0	17,0	18,0
180	79,0	97,0	353,0	79,0	91,0	116,0	23,0	17,0	18,0
190	74,0	70,0	288,0	84,0	113,0	155,0	23,0	17,0	23,0
200	66,0	93,0	299,0	116,0	139,0	189,0	23,0	29,0	32,0
210	76,5	93,5	270,5	150,6	158,6	226,6	23,0	29,0	32,0
220	68,0	86,0	340,0	126,4	159,0	219,4	35,0	29,0	32,0
230	79,5	90,5	397,5	135,6	165,0	213,6	52,0	29,0	31,0
240	54,5	72,5	350,5	127,2	160,2	237,2	52,0	29,0	31,0
250	62,0	82,0	355,0	119,6	163,6	210,6	56,0	37,0	31,0
260	78,0	77,0	293,0	120,6	160,6	223,6	56,0	37,0	31,0
270	66,5	89,5	437,5	126,6	152,6	193,6	56,0	37,0	31,0

280	41,0	79,0	343,0	105,8	157,8	225,8	63,0	34,0	36,0
290	42,5	74,5	428,5	118,0	152,0	222,0	63,0	34,0	36,0
300	51,0	81,0	428,0	104,8	129,8	261,8	70,0	34,0	36,0

Tabela C2. Wyniki zaprezentowane na rysunkach 3.12-3.14 (badanie 3.3)

m	$n = 100$			$n = 200$			$n = 300$		
	$\bar{Z}_{\mathrm{DR}}$	$\bar{Z}_{\mathrm{GR}}$	$\bar{Z}_{\mathrm{SR}}$	$\bar{Z}_{\mathrm{DR}}$	$\bar{Z}_{\mathrm{GR}}$	$\bar{Z}_{\mathrm{SR}}$	$\bar{Z}_{\mathrm{DR}}$	$\bar{Z}_{\mathrm{GR}}$	$\bar{Z}_{\mathrm{SR}}$
2	59,5	89,5	198,5	66,0	93,0	299,0	51,0	81,0	428,0
3	116,0	148,0	123,0	66,0	111,0	281,0	66,3	96,3	347,3
4	39,0	36,0	56,0	122,8	135,8	260,8	110,8	113,8	235,8
5	39,0	35,0	37,0	116,0	139,0	196,0	104,8	129,8	252,8
6	39,0	27,0	24,0	71,0	82,0	64,0	130,2	165,0	217,2
7	28,0	8,0	21,0	47,0	52,0	61,0	104,0	117,0	125,0
8	9,0	8,0	9,0	47,0	42,0	47,0	65,0	72,0	71,0
9	9,0	8,0	9,0	27,0	29,0	45,0	70,0	69,0	47,0
10	10,0	8,0	5,0	23,0	29,0	32,0	70,0	34,0	36,0
11	5,0	5,0	5,0	23,0	17,0	32,0	52,0	34,0	36,0
12	5,0	5,0	5,0	23,0	14,0	23,0	27,0	30,0	33,0
13	5,0	0,0	5,0	22,0	14,0	16,0	36,0	25,0	28,0
14	5,0	0,0	8,0	16,0	14,0	16,0	31,0	23,0	28,0
15	5,0	0,0	8,0	16,0	14,0	16,0	31,0	19,0	28,0

Tabela C3. Wyniki zaprezentowane na rysunku 3.15 (badanie 3.3)

n	$m = 2$		
	t_{DR} [ms]	t_{GR} [ms]	t_{SR} [ms]
10	7	92	16
20	33	689	44
30	63	2789	82
40	71	6391	92
50	92	17314	111
60	89	34324	122
70	122	56542	137
80	168	84454	179
90	200	146553	209
100	234	190001	249
110	351	320433	358
120	402	414323	411
130	426	555643	431
140	457	701243	469
150	501	974932	500
160	541	1294321	532
170	700	1464324	677
180	714	1912344	729
190	829	2423432	803
200	900	2989321	921
210	963	3742376	1000
220	1029	4124344	1089
230	1152	4674299	1198
240	1219	5789421	1233
250	1374	6454035	1434
260	1488	7012033	1472
270	1499	7674242	1591
280	1554	8100324	1633
290	1499	8783892	1708
300	1601	8984324	1721

Tabela C4. Wyniki optymalizacji dla losowo wygenerowanych instancji (badanie 3.4)

$(m, n, \bar{r}, \bar{p})$	$\bar{Z}_{\mathrm{DR}}$	$\bar{Z}_{\mathrm{GR}}$	$\bar{Z}_{\mathrm{SR}}$	t_{DR} [ms]	t_{GR} [ms]	t_{SR} [ms]
(12, 185, 891, 87)	38,0	23,0	24,0	1302	2701120	106
(14, 31, 865, 49)	0,0	0,0	0,0	48	4963	3
(10, 129, 446, 21)	19,0	10,0	16,0	492	566226	62
(5, 248, 965, 24)	82,0	53,0	59,0	955	3457028	212
(9, 212, 305, 71)	117,0	121,0	144,0	1229	3425655	154
(5, 247, 599, 56)	307,4	290,4	382,4	982	3397947	214
(7, 255, 373, 23)	76,0	77,0	109,0	1401	5365826	231
(9, 19, 52, 64)	16,0	12,0	8,0	11	575	3
(12, 110, 107, 23)	27,0	23,0	30,0	456	373366	43
(8, 255, 800, 24)	35,0	21,0	34,0	1600	6109154	225
(15, 24, 804, 100)	11,0	5,0	5,0	32	2170	2
(11, 30, 782, 27)	0,0	0,0	0,0	32	3268	5
(10, 257, 382, 80)	123,0	133,0	151,0	2097	8040941	221
(6, 147, 968, 60)	43,0	55,0	47,0	386	544712	72
(6, 218, 398, 87)	246,8	242,8	360,8	885	2489912	157
(9, 269, 246, 81)	141,6	136,6	184,6	2039	8536952	260
(4, 145, 140, 29)	54,8	56,8	94,8	246	335515	62
(7, 33, 643, 40)	6,0	6,0	6,0	24	2705	4
(7, 287, 780, 14)	35,0	18,0	24,0	1722	8395486	283
(11, 249, 869, 52)	32,0	30,0	34,0	2070	7809758	218
(13, 30, 353, 56)	0,0	0,0	0,0	39	3942	3
(10, 43, 332, 99)	8,0	3,0	2,0	57	10152	6
(6, 166, 185, 71)	90,7	97,7	167,7	540	864991	99
(13, 36, 606, 84)	0,0	0,0	0,0	56	7368	4
(10, 32, 405, 86)	23,0	22,0	21,0	31	3624	3
(9, 296, 338, 55)	165,0	148,0	181,0	2428	12479050	321
(9, 142, 606, 31)	26,0	26,0	38,0	561	751652	71
(15, 102, 213, 40)	13,0	6,0	5,0	517	373947	31
(2, 164, 198, 53)	61,0	69,0	179,0	164	263727	80
(3, 244, 112, 15)	20,7	27,7	109,7	580	1862035	208
(14, 26, 654, 63)	19,0	6,0	11,0	29	2688	3
(5, 35, 365, 90)	50,0	40,0	47,0	22	2354	5
(10, 164, 614, 29)	17,0	11,0	31,0	812	1401322	82
(5, 241, 484, 94)	163,0	218,0	333,0	900	2997972	207
(6, 264, 121, 30)	52,2	48,2	106,2	1442	5208008	244
(15, 107, 492, 18)	6,0	3,0	7,0	604	431468	43
(15, 109, 295, 94)	23,0	23,0	27,0	553	466291	42
(9, 16, 625, 89)	2,0	2,0	2,0	9	328	2
(13, 193, 616, 90)	49,0	38,0	77,0	1488	3499062	119
(11, 229, 832, 10)	12,0	1,0	6,0	1782	5648590	190
(10, 256, 385, 53)	75,0	73,0	97,0	2029	7817861	236
(11, 298, 907, 86)	64,0	48,0	42,0	2948	15926077	326
(13, 221, 547, 34)	17,0	9,0	16,0	1994	5984805	161
(9, 79, 973, 70)	20,0	31,0	25,0	175	81327	24
(14, 211, 426, 65)	33,0	33,0	50,0	1918	5394896	144
(14, 271, 884, 74)	43,0	30,0	41,0	3099	14122955	257
(7, 276, 725, 97)	303,0	299,0	390,0	1662	7372926	267
(2, 266, 197, 89)	67,0	60,0	627,0	455	1753901	255
(4, 246, 327, 82)	131,0	142,0	331,0	748	2581700	211
(13, 73, 561, 50)	35,0	12,0	7,0	218	90019	17
(12, 247, 563, 58)	32,0	41,0	49,0	2246	8281677	209
(4, 92, 138, 40)	49,8	71,8	102,8	100	60652	25
(6, 292, 810, 17)	40,0	28,0	45,0	1574	7737370	301

(7, 23, 762, 44)	0,0	0,0	0,0	12	777	3
(3, 128, 319, 51)	128,3	135,3	243,3	157	155080	56
(9, 60, 928, 59)	19,0	18,0	13,0	110	29714	14
(5, 205, 535, 67)	234,4	283,4	378,4	667	1607738	132
(10, 221, 665, 64)	62,0	37,0	38,0	1517	4438653	178
(14, 299, 491, 47)	30,0	14,0	22,0	3871	20721155	309
(5, 185, 646, 27)	84,0	65,0	81,0	522	1079757	112
(4, 210, 736, 75)	308,0	375,0	459,0	548	1410186	139
(3, 32, 359, 90)	132,0	136,0	184,0	9	1019	3
(12, 22, 989, 40)	0,0	0,0	0,0	21	1305	2
(6, 166, 954, 35)	49,0	46,0	48,0	544	866975	96
(5, 49, 402, 96)	90,0	71,0	81,0	36	7540	8
(14, 58, 513, 16)	0,0	0,0	0,0	144	42237	11
(15, 193, 295, 21)	19,0	5,0	21,0	1698	4063353	115
(13, 131, 894, 73)	12,0	4,0	10,0	674	796230	51
(13, 54, 858, 11)	0,0	0,0	0,0	111	30219	12
(8, 80, 209, 86)	78,0	79,0	82,0	155	73582	20
(12, 211, 403, 62)	58,0	55,0	47,0	1625	4504608	147
(10, 232, 95, 56)	59,2	60,2	93,2	1674	5360599	182
(10, 250, 711, 76)	61,0	50,0	59,0	1921	7163734	221
(10, 268, 467, 76)	111,0	112,0	121,0	2174	9361593	242
(3, 296, 813, 63)	185,0	232,0	474,0	817	3971883	331
(4, 227, 847, 76)	381,8	400,8	572,8	636	1896421	163
(10, 144, 54, 62)	49,1	42,1	60,1	638	855160	61
(7, 279, 878, 67)	177,0	177,0	198,0	1668	7581286	273
(12, 255, 341, 75)	80,0	70,0	94,0	2498	9443177	230
(3, 166, 920, 16)	95,0	82,0	103,0	257	415707	82
(14, 152, 826, 63)	28,0	4,0	13,0	987	809390	69
(5, 127, 188, 58)	81,4	99,4	110,4	238	254599	48
(15, 184, 398, 55)	36,0	24,0	19,0	1511	3358670	104
(13, 99, 862, 79)	11,0	19,0	15,0	410	285729	29
(13, 164, 956, 31)	2,0	1,0	9,0	1063	1856881	94
(15, 87, 749, 80)	11,0	11,0	8,0	365	200384	23
(13, 282, 653, 81)	48,0	35,0	54,0	3215	8238899	269
(6, 83, 409, 32)	32,0	26,0	43,0	126	62521	22
(5, 191, 895, 50)	134,0	124,0	153,0	567	1218153	121
(14, 152, 590, 47)	10,0	2,0	10,0	985	1512773	69
(2, 58, 880, 25)	43,0	60,0	61,0	20	5381	11
(8, 124, 597, 32)	12,0	11,0	23,0	365	385597	45
(8, 119, 876, 41)	19,0	14,0	8,0	363	329759	44
(10, 79, 122, 16)	12,0	13,0	22,0	193	90640	20
(10, 142, 399, 26)	22,0	15,0	25,0	613	818391	73
(3, 265, 411, 68)	85,7	124,7	411,7	657	2608621	256
(5, 189, 670, 39)	118,0	132,0	157,0	560	1175909	122
(7, 134, 986, 82)	43,0	52,0	49,0	383	444458	66
(13, 47, 168, 38)	2,0	1,0	6,0	89	18410	8
(7, 73, 711, 38)	13,0	3,0	12,0	110	45586	16

D. Rezultaty optymalizacji dla problemu ScheLoc

Tabela D1. Wyniki zaprezentowane na rysunku 4.5 (badanie 4.1)

n	$C_{\mathrm{GN}}^{(\mathrm{min})}$	$C_{\mathrm{GN}}^{(\mathrm{med})}$	$C_{\mathrm{GN}}^{(\mathrm{max})}$	$C_{\mathrm{HS}}^{(\mathrm{min})}$	$C_{\mathrm{HS}}^{(\mathrm{med})}$	$C_{\mathrm{HS}}^{(\mathrm{max})}$
10	515,4	586,7	674,4	286,5	321,0	358,0
20	756,8	805,9	841,1	334,0	372,0	464,3
30	834,0	906,5	945,3	351,9	399,4	471,3
40	886,2	1003,0	1027,0	380,8	436,6	510,1
50	997,6	1077,5	1158,3	433,8	471,2	524,9
60	1077,6	1143,9	1234,1	479,7	496,4	540,5
70	1154,2	1233,2	1308,3	487,7	529,6	553,1
80	1176,5	1275,8	1333,1	516,9	550,9	589,2
90	1289,8	1340,3	1406,3	543,9	573,7	608,0
100	1325,4	1421,2	1467,1	570,5	602,6	639,5
110	1371,0	1471,1	1551,4	581,1	612,5	646,5
120	1539,4	1569,9	1640,1	593,1	625,3	681,5
130	1565,8	1607,7	1699,5	613,4	644,1	687,3
140	1544,8	1699,4	1797,5	649,1	678,5	704,4
150	1596,4	1763,0	1843,7	659,8	692,0	727,6
160	1711,9	1793,2	1922,2	686,6	714,1	751,2
170	1820,9	1893,8	1978,4	704,6	739,9	762,6
180	1900,0	1988,7	2073,8	729,7	759,7	780,2
190	1897,2	2028,1	2187,6	744,7	766,9	794,5
200	1906,8	2095,7	2254,4	762,7	787,8	818,0
210	2028,4	2180,3	2294,4	780,7	817,4	839,2
220	2116,4	2274,2	2367,5	800,6	831,4	853,0
230	2117,0	2307,5	2465,1	830,8	849,5	889,7
240	2266,4	2339,2	2497,6	845,3	866,0	892,9
250	2357,1	2448,8	2563,6	850,9	884,7	913,0
260	2347,0	2490,3	2633,7	867,5	895,0	928,6
270	2398,7	2588,6	2761,9	888,8	924,7	953,4
280	2498,4	2645,7	2884,2	902,9	945,3	971,1
290	2547,7	2683,0	2824,4	916,3	951,5	992,4
300	2460,2	2784,0	2884,7	949,1	977,7	1005,4

Tabela D2. Wyniki zaprezentowane na rysunku 4.6 (badanie 4.1)

m	$C_{\mathrm{GN}}^{(\mathrm{min})}$	$C_{\mathrm{GN}}^{(\mathrm{med})}$	$C_{\mathrm{GN}}^{(\mathrm{max})}$	$C_{\mathrm{HS}}^{(\mathrm{min})}$	$C_{\mathrm{HS}}^{(\mathrm{med})}$	$C_{\mathrm{HS}}^{(\mathrm{max})}$
2	1911,9	2024,9	2088,6	1226,8	1259,6	1277,8
3	1613,8	1651,8	1705,2	803,6	820,6	829,0
4	1414,6	1489,2	1552,5	625,4	683,3	716,3
5	1325,4	1421,2	1467,1	570,5	602,6	639,5
6	1314,0	1347,0	1401,4	486,7	523,7	564,3
7	1260,6	1311,8	1351,3	464,2	476,4	488,6
8	1229,1	1287,6	1319,7	398,5	427,9	479,9
9	1167,1	1276,2	1318,8	393,3	404,7	426,5
10	1144,1	1217,5	1275,8	373,0	381,8	396,5
11	1162,1	1186,3	1241,4	355,5	367,1	377,8
12	1136,1	1175,2	1238,0	338,4	349,8	363,2
13	1141,5	1166,1	1209,4	314,5	325,3	342,4
14	1128,9	1154,9	1205,7	296,7	324,2	338,9
15	1130,1	1171,3	1205,2	294,7	312,3	330,1
16	1081,1	1145,9	1176,8	298,1	307,6	316,2
17	1110,0	1137,9	1149,1	291,1	297,7	305,1

18	1098,9	1131,9	1167,9	280,8	290,4	291,5
19	1103,5	1134,5	1155,3	281,7	289,2	295,1
20	1077,4	1129,4	1158,4	279,3	287,4	293,3

Tabela D3. Wyniki zaprezentowane na rysunku 4.7 (badanie 4.1)

w	$C_{\mathrm{GN}}^{(\mathrm{min})}$	$C_{\mathrm{GN}}^{(\mathrm{med})}$	$C_{\mathrm{GN}}^{(\mathrm{max})}$	$C_{\mathrm{HS}}^{(\mathrm{min})}$	$C_{\mathrm{HS}}^{(\mathrm{med})}$	$C_{\mathrm{HS}}^{(\mathrm{max})}$
5	1304,4	1406,9	1459,3	610,0	613,1	624,2
6	1332,5	1399,9	1463,6	585,0	599,2	622,8
7	1300,1	1400,7	1436,2	584,3	597,8	621,8
8	1362,3	1421,2	1472,6	580,2	594,9	602,0
9	1295,9	1385,1	1420,4	584,3	595,6	640,8
10	1292,8	1394,9	1448,2	566,6	595,6	618,7
11	1338,7	1438,5	1476,4	581,6	598,9	639,5
12	1374,5	1444,8	1503,7	580,2	596,5	651,6
13	1373,6	1421,2	1475,7	580,7	607,2	632,8
14	1367,4	1448,6	1472,1	565,9	596,3	629,3
15	1386,7	1435,9	1466,8	570,0	602,8	644,4
16	1363,0	1425,1	1450,9	554,5	586,6	609,9
17	1316,1	1387,2	1424,7	553,3	581,5	598,3
18	1303,3	1415,0	1444,4	569,7	591,7	623,7
19	1196,0	1405,5	1541,2	564,8	614,3	639,1
20	1325,4	1421,2	1467,1	570,5	602,6	639,5
21	1324,3	1430,5	1492,6	559,0	594,3	638,6
22	1381,4	1427,9	1475,3	580,1	596,6	653,5
23	1365,5	1434,4	1497,5	574,4	598,6	639,5
24	1339,4	1422,3	1479,5	559,0	588,3	649,4
25	1311,9	1403,4	1501,4	567,4	594,3	645,2

Tabela D4. Wyniki zaprezentowane na rysunku 4.8 (badanie 4.1)

n	$t_{\mathrm{GN}}^{(\mathrm{min})}$ [ms]	$t_{\mathrm{GN}}^{(\mathrm{med})}$ [ms]	$t_{\mathrm{GN}}^{(\mathrm{max})}$ [ms]	$t_{\mathrm{HS}}^{(\mathrm{min})}$ [ms]	$t_{\mathrm{HS}}^{(\mathrm{med})}$ [ms]	$t_{\mathrm{HS}}^{(\mathrm{max})}$ [ms]
10	189	284	484	25	43	62
20	280	408,5	814	65	68,5	184
30	373	637	996	131	252,5	504
40	440	584	1144	221	433,5	870
50	509	732,5	1575	338	666	987
60	599	799	1909	485	1047,5	1909
70	669	881,5	1679	639	667	2518
80	788	1379	2922	845	1292,5	2579
90	886	1380	2074	1063	1145,5	3239
100	907	1687	2789	1298	1347,5	3904
110	1074	1623,5	3767	1683	2156,5	4776
120	1197	1966	4101	1945	2199,5	6065
130	1218	2310	4560	2216	4529	9215
140	1227	1639,5	2731	2553	2727	10025
150	1399	2178,5	5212	2880	3012	11641
160	1393	2432	5303	3248	3354	12947
170	1472	2262,5	3881	3691	7438	11419
180	1576	2019,5	4718	4203	4333,5	8684
190	1661	2676,5	5870	4608	7011	13732
200	1827	2772,5	4275	5126	5260,5	15387
210	1896	3109,5	5353	5694	5820	11631
220	1909	2974,5	4791	6238	6364,5	12722
230	2032	3476	5641	6841	6985	13790
240	2132	3323	9546	7363	7592	22288

250	2178	3265,5	5993	7926	8288	16310
260	2213	3624	6321	8700	13042	17591
270	2285	3141	8222	9363	14106,5	28257
280	2477	3446,5	7693	10148	10884,5	32099
290	2674	4663,5	16790	11256	17473	33983
300	2895	4450	8190	11975	18553	26307

Tabela D5. Wyniki zaprezentowane na rysunku 4.9 (badanie 4.2)

n	C_{center}	C_{median}	C_{EG}	θ_{center} [%]	θ_{median} [%]
10	334,79	319,77	319,77	4,70	0,00
20	354,41	354,41	354,41	0,00	0,00
30	379,63	386,77	366,28	3,64	5,59
40	493,63	445,50	442,41	11,58	0,70
50	501,80	501,80	501,80	0,00	0,00
60	552,39	552,39	523,77	5,47	5,47
70	582,66	582,66	562,99	3,49	3,49
80	634,49	634,49	596,26	6,41	6,41
90	674,04	643,87	607,50	10,95	5,99
100	697,90	674,85	634,50	9,99	6,36
110	723,43	712,24	663,39	9,05	7,36
120	755,99	736,03	689,31	9,67	6,78
130	784,43	768,43	721,72	8,69	6,47
140	810,98	816,08	753,76	7,59	8,27
150	836,15	826,64	782,10	6,91	5,69
160	874,04	865,79	804,89	8,59	7,57
170	905,50	886,91	829,92	9,11	6,87
180	926,16	914,21	851,81	8,73	7,33
190	956,31	948,05	883,09	8,29	7,36
200	969,64	970,64	894,81	8,36	8,47

Tabela D6. Wyniki zaprezentowane na rysunku 4.10 (badanie 4.2)

w	C_{center}	C_{median}	C_{EG}	θ_{center} [%]	θ_{median} [%]
5	589,22	589,22	586,16	0,52	0,52
6	578,02	578,02	570,59	1,30	1,30
7	587,79	587,79	570,59	3,01	3,01
8	587,79	587,79	546,25	7,60	7,60
9	550,03	550,03	546,25	0,69	0,69
10	550,03	575,88	541,47	1,58	6,36
11	550,03	575,88	504,01	9,13	14,26
12	550,03	575,88	504,01	9,13	14,26
13	550,03	575,88	504,01	9,13	14,26
14	533,63	501,80	501,80	6,34	0,00
15	501,80	501,80	501,80	0,00	0,00
16	501,80	501,80	463,96	8,16	8,16
17	501,80	501,80	463,96	8,16	8,16
18	501,80	501,80	463,96	8,16	8,16
19	501,80	501,80	463,96	8,16	8,16
20	501,80	501,80	463,96	8,16	8,16
21	501,80	501,80	463,96	8,16	8,16
22	491,80	491,80	463,96	6,00	6,00
23	491,80	491,80	463,96	6,00	6,00
24	491,80	472,92	463,96	6,00	1,93

Tabela D7. Wyniki zaprezentowane na rysunku 4.11 (badanie 4.2)

n	C_{center}	C_{median}	C_{HS}	θ_{center} [%]	θ_{median} [%]
10	174,1	179,7	187,8	-7,29	-4,31
20	223,4	250,3	266,7	-16,24	-6,15
30	272,3	288,8	300,6	-9,41	-3,93
40	296,2	296,2	347,1	-14,66	-14,66
50	326,5	324,6	361,1	0,51	-10,10
60	343,5	336,2	402,9	-14,74	-16,55
70	361,1	353,8	423,4	-14,71	-16,44
80	384,7	386,0	444,2	-13,39	-13,10
90	423,8	410,2	472,5	-10,31	-13,19
100	326,5	324,6	361,1	-9,59	-10,10
110	476,4	466,9	507,2	-6,07	-7,95
120	502,4	496,1	519,3	-3,25	-4,47
130	519	493,3	554,6	-6,42	-11,05
140	546,8	530,8	574,0	-4,74	-7,53
150	555,6	541,7	578,6	-3,98	-6,38
160	584	568,3	591,1	-1,20	-3,86
170	595,9	578,1	602,6	-1,11	-4,07
180	613,0	597,2	631,6	-2,94	-5,45
190	617,5	617,0	642,2	-3,85	-3,92
200	651,4	632,3	639,9	1,80	-1,19

Tabela D8. Czasy obliczeń dla rezultatów w tabeli D7 (badanie 4.2)

n	$t_{\text{center}}^{(\text{med})}$ [ms]	$t_{\text{median}}^{(\text{med})}$ [ms]	$t_{\text{HS}}^{(\text{med})}$ [ms]
10	4618	3702	266
20	3997	3945	346
30	5224	5584	629
40	6867	7738	1270
50	9037	9081	1161
60	10924	11257	1122
70	13567	14350	2164
80	18129	17357	2868
90	20765	21320	2764
100	25609	25943	4297
110	29245	29433	5401
120	34673	33993	4434
130	39459	39028	7150
140	45407	45494	5516
150	52019	53029	6516
160	58289	58287	7852
170	65804	64370	12587
180	72400	72384	9331
190	80416	80224	15782
200	89818	90496	12336

Tabela D9. Wyniki zaprezentowane na rysunku 4.12 (badanie 4.2)

w	C_{center}	C_{median}	C_{EG}	θ_{center} [%]	θ_{median} [%]
10	419,8	419,8	419,8	0,00	0,00
15	419,8	419,8	419,8	0,00	0,00
20	419,8	419,8	419,8	0,00	0,00
25	419,8	419,8	419,8	0,00	0,00

30	321,6	338,3	372,9	-13,75	-9,28
35	320,5	338,3	370,4	-13,46	-8,67
40	325,2	319,2	367,1	-11,42	-13,05
45	319,2	307,5	341,6	-6,55	-9,96
50	323,7	319,6	379,3	-14,67	-15,73
55	318,5	318,5	354,0	-10,03	-10,03
60	323,1	323,7	342,0	-5,52	-5,37
65	321,4	336,2	376,8	-14,71	-10,77
70	318,5	317,7	377,4	-15,61	-15,82
75	309,4	334,6	355,8	-13,06	-5,98
80	319,2	323,7	371,4	-14,06	-12,86
85	312,4	323,7	366,5	-14,76	-11,69
90	326,7	312,1	377,8	-13,51	-17,38
95	317,1	325,6	360,0	-11,91	-9,56
100	326,5	324,6	361,1	-9,59	-10,10

Tabela D10. Wyniki optymalizacji dla losowo wygenerowanych instancji (badanie 4.2)

$(m, n, w, \gamma, \vec{p})$	C_{center}	C_{median}	t_{center} [ms]	t_{median} [ms]
(2,59,5,77,76)	710,5	710,5	13	14
(7,87,13,419,89)	291,4	300,2	147207	132670
(5,88,14,864,98)	683,4	644,2	2647	2968
(3,54,9,179,73)	442,9	422,8	42	44
(3,45,8,691,88)	561,0	561,0	25	24
(8,72,11,148,93)	156,3	154,0	871818	841010
(6,53,13,548,62)	261,8	261,8	6514	6386
(5,30,9,596,91)	351,8	343,9	395	406
(4,83,9,284,98)	548,5	548,5	402	411
(6,44,11,651,34)	279,4	279,4	5066	4374
(4,16,11,432,45)	189,5	174,1	97	103
(3,77,9,701,42)	564,7	541,9	72	73
(2,75,4,619,34)	678,8	678,8	16	17
(6,17,13,164,31)	76,1	76,1	893	902
(8,79,12,533,43)	270,8	261,7	995444	977649
(6,57,11,340,31)	174,9	162,6	7212	7536
(6,42,12,582,86)	324,8	308,6	4056	4100
(6,46,13,349,73)	234,1	202,4	5429	5358
(3,64,6,698,76)	711,9	695,0	56	54
(3,99,7,411,45)	520,5	514,0	118	119
(8,88,11,660,88)	377,7	347,3	900176	998754
(7,23,13,685,99)	319,8	313,7	10750	10769
(5,64,10,968,55)	430,2	423,8	1448	1350
(8,96,10,497,75)	333,4	330,4	783310	790323
(7,26,9,159,54)	78,6	73,2	13854	13427
(2,91,21,875,37)	819,4	787,4	39	48
(4,38,14,482,80)	374,0	353,9	361	358
(6,30,29,714,44)	241,3	259,0	2776	2754
(5,99,19,370,49)	328,3	306,8	3347	3370
(5,78,19,681,28)	337,4	336,1	2187	2179
(5,52,7,564,82)	1422,3	1415,4	26	26
(8,76,17,591,11)	414,2	419,8	949	1025
(2,15,17,314,74)	144,3	170,7	953666	962294
(4,59,29,468,77)	311,1	319,0	5	5
(7,17,27,680,43)	403,7	406,3	8550	8226
(6,35,23,712,25)	169,4	167,4	6958	6882
(4,67,18,459,92)	204,5	208,6	3471	3292

(8,70,19,641,63)	518,5	518,5	1257	1238
(3,36,13,962,52)	281,7	281,5	823785	817542
(7,45,29,16,83)	534,1	520,3	36	37
(5,74,27,710,83)	96,1	95,9	40602	39661
(4,29,9,759,79)	504,5	476,6	2178	2055
(6,93, 10,673,21)	368,9	368,9	83	76
(2,51,8,104,28)	287,4	287,4	19450	20466
(3,74,13,332,96)	275,7	275,7	9	10
(6,84,30,778,40)	625,3	655,8	78	87
(3,14,7, 819,43)	344,8	329,8	16857	16519
(4,30,27,833,58)	160,3	155,9	136474	134305
(8,21,18,507,100)	360,0	337,6	5334	5921
(5,91,13,711,27)	171,9	180,9	80311	80723
(2,34,29,516,83)	331,6	341,3	2774	2697
(5,11,17,187,35)	610,3	593,5	15	22
(7,60,14,93,39)	82,6	69,2	237	253
(4,56,8,54,62)	81,7	81,3	67221	69218
(6,38,19,833,16)	186,8	181,4	197	201
(6,69,21,740,17)	658,7	658,7	3311	3062
(6,64,29,720,37)	219,1	278,6	10987	11275
(2,21,7,278,26)	288,3	273,1	10441	9385
(2,91,20,95,50)	200,8	200,8	5	7
(7,95,9,855,83)	715,2	711,8	40	32
(4,43,30,61,23)	472,2	472,2	159214	158463
(3,66,6,318,47)	86,2	79,6	8560	8800
(8,76,26,19,45)	383,2	373,3	48	46
(8,54,22,776,85)	67,3	68,5	959511	979147
(5,95,20,985,42)	304,1	308,5	492178	487464
(8,23,19,578,47)	465,8	468,0	3143	2966
(7,54,21,682,70)	210,4	210,4	94833	93162
(7,67,12,352,77)	324,0	282,4	53331	56960
(5,64,24,499,12)	254,1	246,1	83080	86299
(6,45,26,813,74)	184,7	178,0	1597	1538
(2,80,4,494,79)	363,2	327,7	5339	5422
(3,66,6,745,84)	1168,2	1157,3	18	15
(7,83,19,257,29)	700,0	680,1	135	133
(3,25,8,144,50)	143,8	141,7	133051	126901
(7,75,23,222,13)	173,8	158,6	13	12
(3,31,7,622,80)	86,6	96,0	100784	104543
(5,47,24,779,18)	454,7	450,6	15	13
(2,80,9,217,70)	240,9	250,3	1069	990
(7,59,18,446,19)	1065,3	1065,3	16	20
(4,82,15,283,13)	175,2	175,2	62936	65587
(7,90,29,201,73)	160,6	161,6	788	749
(10,99,21,377,70)	200,8	203,9	151842	155237
(6,49,22,72,31)	1330,7	1318,9	32	30
(5,100,8,326,69)	69,0	68,3	5922	6301
(6,49,30,298,13)	353,4	348,4	3274	3189
(7,39,11,249,75)	102,9	112,4	6053	6373
(4,32,20,528,59)	159,2	159,3	28584	29501
(4,20,24,932,71)	323,7	309,2	1491	1551
(2,81,17,245,30)	331,8	481,1	3018	3467
(4,61,24,745,53)	485,9	485,9	25	26
(2,33,13,898,31)	438,0	467,3	3797	3676
(7,33,26,665,45)	496,9	502,2	6	7
(7,79,22,188,24)	197,4	240,1	21021	21138
(9,42,16,789,86)	116,2	98,0	119553	116935
(5,18,20,979,13)	376,0	386,7	4691	4258

(3,89,23,367,86)	237,6	318,5	354	420
(4,82,6,980,65)	739,2	754,2	330	262
(6,29,20,978,32)	675,7	675,2	337	380
(2,57,5,8,90)	334,2	348,2	2379	2406
(4,72,26,204,65)	852,4	852,4	11	11

Tabela D11. Wyniki optymalizacji dla losowo wygenerowanych instancji (badanie 4.3). Dane zaprezentowane na rysunkach 4.13-4.15

j	$\gamma_1^{(L)} = 1000, \gamma_1^{(\tilde{L})} = 100$						$\gamma_2^{(L)} = 1000, \gamma_2^{(\tilde{L})} = 500$						$\gamma_3^{(L)} = 1000, \gamma_3^{(\tilde{L})} = 1000$					
	$\tilde{\theta}_s^{(a)}$	$\tilde{\theta}_s^{(b)}$	$\tilde{\theta}_s^{(c)}$	$\tilde{\theta}_d^{(a)}$	$\tilde{\theta}_d^{(b)}$	$\tilde{\theta}_d^{(c)}$	$\tilde{\theta}_s^{(a)}$	$\tilde{\theta}_s^{(b)}$	$\tilde{\theta}_s^{(c)}$	$\tilde{\theta}_d^{(a)}$	$\tilde{\theta}_d^{(b)}$	$\tilde{\theta}_d^{(c)}$	$\tilde{\theta}_s^{(a)}$	$\tilde{\theta}_s^{(b)}$	$\tilde{\theta}_s^{(c)}$	$\tilde{\theta}_d^{(a)}$	$\tilde{\theta}_d^{(b)}$	$\tilde{\theta}_d^{(c)}$
10	0,00	0,00	5,01	0,00	0,00	2,69	0,00	2,63	35,49	-3,88	1,31	21,52	3,85	29,83	108,62	0,00	33,09	111,34
11	0,00	0,00	4,42	0,00	0,00	1,36	0,00	11,34	38,26	0,00	4,94	40,75	9,99	37,30	89,71	2,08	37,15	97,05
12	-1,15	0,00	9,57	-1,15	0,00	9,07	0,00	10,70	24,78	-1,05	7,65	33,83	0,99	27,67	108,34	0,00	26,48	106,74
13	0,00	0,00	2,48	0,00	0,00	0,00	-4,54	3,27	14,11	0,00	0,00	19,17	-1,54	45,02	96,62	7,87	44,85	90,02
14	0,00	0,00	8,67	0,00	0,00	0,13	0,00	8,21	28,54	0,00	3,98	27,59	6,30	33,27	66,72	5,23	35,88	70,69
15	-0,18	0,00	3,82	-0,18	0,00	1,25	0,00	6,15	31,19	-0,23	2,86	54,29	22,77	36,72	90,84	6,94	37,26	91,39
16	-0,29	0,00	9,87	-0,29	0,00	0,00	0,00	2,47	39,65	0,00	0,00	31,65	5,04	19,15	58,52	0,00	24,02	81,59
17	0,00	0,00	3,48	0,00	0,00	1,87	-2,31	6,15	20,79	-0,36	4,07	23,16	14,49	47,06	103,29	15,64	43,21	111,58
18	0,00	0,00	1,18	0,00	0,00	0,00	0,00	8,55	32,40	0,00	9,19	46,20	3,48	38,90	62,41	3,97	35,99	77,31
19	0,00	0,00	2,50	-0,31	0,00	0,00	-0,17	7,41	37,39	-2,59	4,49	36,14	9,42	33,89	57,55	2,47	38,62	92,15
20	0,00	0,00	4,29	0,00	0,00	3,28	0,00	5,90	34,80	0,00	0,26	61,28	9,62	31,79	60,67	6,24	36,86	98,35
21	0,00	0,00	2,06	-0,16	0,00	1,85	0,00	10,21	31,75	-1,84	4,63	48,01	14,69	36,91	62,14	3,59	41,84	105,59
22	-2,03	0,00	16,08	-1,09	0,00	19,60	0,00	6,12	31,60	-2,28	0,56	30,90	14,51	38,11	71,73	7,46	44,95	108,93
23	-0,20	0,00	4,23	-0,80	0,00	4,23	0,13	17,93	33,97	0,00	13,35	39,71	21,63	43,31	88,51	12,53	44,35	102,13
24	0,00	0,00	24,43	0,00	0,00	5,87	0,00	9,42	31,19	-1,03	7,52	40,32	10,79	44,51	70,90	17,08	39,80	126,43
25	-1,62	0,00	14,35	-1,62	0,00	12,88	0,00	15,27	35,95	0,00	19,60	44,16	6,16	33,37	60,57	5,11	30,97	83,48
26	0,00	0,00	17,28	0,00	0,00	0,00	0,00	9,00	36,15	0,00	9,63	87,99	15,50	35,62	94,94	7,16	30,23	84,19
27	0,00	0,00	16,91	0,00	0,00	5,03	-1,32	5,34	30,29	-1,32	3,78	76,26	16,29	43,70	65,56	1,21	46,40	76,69
28	-3,50	0,00	7,76	-2,17	0,00	0,00	0,00	19,33	32,54	-1,76	17,69	71,48	17,76	40,33	75,38	2,90	35,66	100,62
29	-0,08	0,00	7,84	-0,57	0,00	4,39	0,00	4,52	29,68	0,00	2,50	68,15	6,57	34,45	63,26	8,26	36,33	75,26
30	-0,65	0,00	25,74	-3,64	0,00	9,18	0,74	15,71	37,47	0,00	11,89	61,86	11,71	37,57	85,56	13,39	37,61	109,35
31	-1,03	0,00	23,86	0,00	0,00	2,54	0,00	11,52	28,12	0,00	7,17	52,87	22,24	40,95	74,28	10,58	37,01	112,67
32	-1,94	0,00	7,16	-1,94	0,00	0,00	0,00	7,79	55,96	0,00	3,69	47,58	11,86	40,03	72,07	4,88	40,06	103,53
33	-0,20	0,00	15,02	-4,77	0,00	21,87	0,00	5,32	37,02	0,00	7,51	67,79	12,04	40,80	80,02	8,08	40,51	98,88
34	-0,60	0,00	12,66	-1,97	0,00	33,88	-3,93	16,72	44,60	1,00	14,64	53,29	12,08	40,08	92,64	8,84	36,26	98,00
35	0,00	0,00	10,57	-0,82	0,00	5,19	0,00	8,18	42,49	0,00	10,91	66,65	8,34	39,89	74,67	7,67	35,68	107,46

$a = \min, b = \text{med}, c = \max$