Na prawach rękopisu

INSTYTUT INFORMATYKI, AUTOMATYKI I ROBOTYKI
POLITECHNIKI WROCŁAWSKIEJ

Raport serii: PREPRINTY nr PRE-32/2006

Zagadnienia szeregowania zadań z uwzględnieniem transportu. Modele, własności i algorytmy

(rozprawa doktorska)

Adam Tyński

PROMOTOR:
Prof. dr hab. inż. Czesław Smutnicki

Słowa kluczowe:

- optymalizacja
- szeregowanie zadań
- transport
- AGV
- algorytmy przybliżone

WROCŁAW
czerwiec 2006

Spis treści

Streszczenie **v**

Abstract **vii**

1 Wprowadzenie **1**
1.1 Tezy pracy 3
1.2 Zakres pracy 4

2 Problemy szeregowania zadań **5**
2.1 Klasyfikacja problemów 6
2.1.1 Permutacyjny problem przepływowy 10
2.1.2 Problem gniazdowy 11
2.2 Algorytmy rozwiązywania problemów szeregowania zadań . 13
2.2.1 Algorytmy poszukiwań z zabronieniami 15
2.2.2 Algorytmy symulowanego wyżarzania 16
2.2.3 Algorytmy genetyczne 17
2.2.4 Algorytmy poszukiwania rozproszonego 19
2.3 Metody badania jakości algorytmów heurystycznych 20
2.4 Problematyka transportu w elastycznych systemach produkcyjnych 22

3 Własności problemu gniazdowego z ustalonym przydziałem wózków AGV **29**
3.1 Model matematyczny 30
3.1.1 Model permutacyjno-grafowy 33
3.1.2 Przykład ilustracyjny 34
3.2 Zbiory ruchów i sąsiedztwa 37
3.2.1 Efektywna metoda przeglądu sąsiedztwa 45
3.3 Własności grafu i permutacji częściowej 49
3.3.1 Efektywna metoda wyznaczania najlepszej pozycji . 56
3.4 Własności przestrzeni rozwiązań 60

3.4.1 Miary odległości 61
3.4.2 Wielka dolina 62
3.4.3 Inne własności przestrzeni rozwiązań dopuszczalnych 65
3.5 Wnioski i uwagi 67

4 Algorytmy rozwiązywania problemu gniazdowego z ustalonym przydziałem wózków AGV 69
4.1 Algorytmy konstrukcyjne 70
4.1.1 Algorytmy priorytetowe 70
4.1.2 Algorytm typu wstaw 72
4.2 Algorytmy lokalnych poszukiwań 72
4.2.1 Algorytmy poszukiwań z zabronieniami i poszukiwania rozproszonego 73
4.2.2 Algorytm symulowanego wyżarzania 77
4.2.3 Algorytmy genetyczne 81
4.3 Wyniki badań numerycznych 88
4.3.1 Instancje testowe 89
4.3.2 Wyniki badań algorytmów konstrukcyjnych 90
4.3.3 Wyniki badań algorytmów popraw – instancje TR . 93
4.3.4 Wyniki badań algorytmów popraw – instancje HK . 99
4.4 Wnioski i uwagi 104

5 Problem gniazdowy ze swobodnym przydziałem wózków AGV 107
5.1 Model matematyczny 108
5.1.1 Model permutacyjno-grafowy 110
5.2 Własności grafu i permutacji częściowej 111
5.2.1 Efektywna metoda wyznaczania najlepszej maszyny i pozycji 113
5.3 Zbiór ruchów i sąsiedztwo 117
5.3.1 Efektywna metoda przeglądu sąsiedztwa 118
5.4 Algorytmy konstrukcyjne 122
5.4.1 Algorytmy priorytetowe 123
5.4.2 Algorytm typu wstaw 125
5.5 Algorytmy lokalnych poszukiwań 126
5.5.1 Algorytm poszukiwań z zabronieniami 127
5.5.2 Algorytm genetyczny 130
5.6 Wyniki badań numerycznych 132
5.6.1 Instancje testowe 133
5.6.2 Wyniki badań algorytmów konstrukcyjnych 134
5.6.3 Wyniki badań algorytmów popraw – instancje TM . 137

5.6.4 Wyniki badań algorytmów popraw – instancje EX . 141
5.7 Wnioski i uwagi 147

6 Problem przepływowy z jednym wózkiem AGV 149
6.1 Model matematyczny 150
6.1.1 Model permutacyjno-grafowy 154
6.1.2 Przykład ilustracyjny 155
6.1.3 Polityka pracy wózka 158
6.2 Eliminacyjne własności bloków zadań 159
6.3 Algorytmy rozwiązywania problemu 163
6.4 Wyniki badań numerycznych 167
6.5 Wnioski i uwagi 172

7 Wnioski końcowe 175

Literatura 177

Spis tabel 189

Spis rysunków 191

Streszczenie

Praca dotyczy problemów optymalizacji równoczesnego planowania zadań i zarządzania transportem w elastycznych systemach produkcyjnych (ESP, ang. *flexible manufacturing system*). ESP to system wytwarzania złożony z w pełni zautomatyzowanych podsystemów wykonawczych, magazynowych i transportowych, których praca jest koordynowana przez komputerowy system sterowania. Ze względu na determinizm funkcjonowania ESP jako narzędzie do analizy i optymalizacji jego pracy wykorzystywana może być deterministyczna teoria szeregowania zadań. W rozprawie rozważano system automatycznego transportu realizowanego przez wózki AGV (z różną polityką obsługi) oraz przepływową i gniazdową strukturę stanowisk dla przepływu dowolnych zadań. Za kryterium przyjęto moment zakończenia wykonywania wszystkich zadań (maksymalny stopień wykorzystania parku maszynowego). W pracy proponuje się modele i algorytmy rozwiązywania postawionych zagadnień, które pozwalają na coraz doskonalsze modelowanie funkcjonowania coraz większej grupy rzeczywistych systemów produkcyjnych, co z kolei determinuje wzrost wydajności ich pracy. Szczególną uwagę zwrócono na efektywność proponowanych algorytmów w ich zastosowaniach w strukturach sterowania ESP.

Na pracę składa się 7 rozdziałów. W rozdziale pierwszym zawarte są tezy, cel i zakres pracy. Rozdział drugi zawiera wprowadzenie w problematykę, podstawowe definicje, problemy szeregowania stanowiące bazę do uogólnień w dalszej części pracy, klasyczne i najnowsze tendencje w optymalizacji dyskretnej oraz aktualny stan wiedzy na temat rozwiązywania problemów szeregowania zadań z uwzględnieniem ograniczeń transportowych. Rozdziały 3-6 stanowią zasadniczą część pracy, w której zaprezentowane są oryginalne rezultaty uzyskane przez autora.

W rozdziale trzecim rozważa się ESP o strukturze gniazdowej ze zbiorem dedykowanych wózków AGV w kontekście ogólnych i szczególnych własności zagadnienia. Dla tego problemu zaproponowano model matematyczny (kombinatoryczny) wraz z wygodną do analizy reprezentacją permutacyjno-grafową, uwzględniający szczególnie praktyczny tryb pracy wózków „zabierz i zostaw". Korzystając z wprowadzonej reprezentacji rozwiązania, zaproponowano lokalne sąsiedztwo generowane w oparciu o ruchy typu zamień sąsiednie operacje, dedykowane dla specjalizowanych algorytmów poszukiwań lokalnych. Dzięki wykazaniu pewnych własności teoretycznych sąsiedztwa, zaprezentowano wysoce efektywną metodę jego przeglądu. Wykazano dodatkowo nowe własności tzw. grafu częściowego, szczególnie przydatne dla niektórych algorytmów konstrukcyjnych i sąsiedztw generowanych przez ruchy typu "wstaw". Uzupełnieniem przedstawionych własności są ekspe-

rymentalne badania topologii przestrzeni rozwiązań (głównie Wielkiej Doliny) realizowane m.in. przy użyciu nowej, oryginalnie wprowadzonej miary odległości pomiędzy rozwiązaniami.

W rozdziale czwartym prezentuje się algorytmy przybliżonego rozwiązania problemu z rozdziału trzeciego. Algorytmy zaprojektowano wykorzystując odmienne techniki konstrukcji rozwiązań i różne techniki poszukiwań lokalnych, w tym m.in. metody symulowanego wyżarzania, poszukiwań z zabronieniami i rozproszonych. Nowatorskie wydaje się podejście genetyczne, w którym użyto nowego quasi-operatora krzyżowania, wykorzystującego wspomnianą wcześniej, oryginalną miarę odległości.

W rozdziale piątym poddano analizie ESP o strukturze gniazdowej ze swobodnym przydziałem wózków AGV do poszczególnych transportów. Analiza problemu prowadzona jest poprzez wychwycenia podobieństw i różnic oraz uogólnień względem wyników uzyskanych w rozdziałach 3-4. Zaproponowano rozszerzenie sąsiedztwa o ruchy typu "wstaw", pozwalające na zmianę kolejności obsługi zadań przez maszyny i środki transportowe, jak i na zmianę przydziału wózków do poszczególnych transportów. W dalszej części zaprojektowano efektywną metodę przeglądu sąsiedztwa. Sformułowano także algorytmy innych klas.

W rozdziale szóstym rozprawy rozważa się system produkcyjny, w którym wszystkie transporty wykonywane są przez jeden wózek pracujący w trybie jednokierunkowym i poruszający się cyklicznie pomiędzy maszynami zorganizowanymi w układ typu pętla. Ze względu na charakter procesu produkcyjnego i przyjęty arbitralny tryb pracy wózka, system produkcyjny można sprowadzić do specyficznej postaci tzw. permutacyjnego problemu przepływowego. Otrzymany model jest jednak formalnie bardziej skomplikowany od modeli problemów gniazdowych i stanowi dobrą bazę teoretyczną do przyszłej analizy problemów hybrydowych (m.in. systemów szeregowo-równoległych, wykorzystujących wiele wózków). Dzięki wyłonionym własnościom wprowadzonego modelu matematycznego i permutacyjno-grafowego przybliżone rozwiązanie problemu okazuje się możliwe przy zastosowaniu odpowiednich modyfikacji pewnych algorytmów rozwiązywania klasycznego permutacyjnego problemu przepływowego, znanych z literatury.

Wszystkie zaprojektowane algorytmy poddano badaniom numerycznym przy użyciu zestawów instancji testowych. Uzyskano empiryczne oceny jakości algorytmów, pozwalające wyłonić najlepszy z nich, a także poprawy wartości górnych ograniczeń dla znacznej liczby instancji znanych z literatury.

Abstract

The thesis concerns the optimization problems of simultaneous job planning and transport management in flexible manufacturing systems (FMS). FMS it is a manufacturing system composed of a full automated processing, storing and transporting subsystems coordinated by a computer control system. For the sake of the FMS deterministic behavior, the deterministic scheduling theory can be used as a tool for optimization of its work. In the thesis, the automated transportation system, employing a number of automated guided vehicles (with various control policies) with a flow-shop or a job-shop structure of service stages, are considered. The completion time of all jobs (the maximum level of a machine park utilization) was applied as the optimization criterion. There are presented, in the thesis, mathematical models and solution algorithms, allowing the major part of real manufacturing systems to be better modeled and, in turn, allowing the FMS performance to be improved. A special attention is also paid to the high effectiveness of algorithms in the context of control structures of the FMSs.

The thesis is consisted of 7 chapters. The first chapter consists of propositions, aim and scope of the thesis. The second chapter consists of an introduction of the scheduling issues, basic definitions, scheduling problems that are the basis for further generalizations, classic and recent trends in discrete optimization and the current state of a knowledge concerning job scheduling problems with the transport constraints. The chapters 3-6 constitute the main part of the thesis and they contains the original results obtained entirely by the author.

In chapter three, the FMS with a job-shop structure and the number of dedicated AGVs, in the context of general and problem-specific properties, is taken into account. There are presented for the considered problem the mathematical (combinatorial) model, particularly respecting a practical „pick and drop" vehicle working mode and a graph representation convenient for the analysis. Using the proposed solution representation and adjacent swap moves, a local neighborhood structure, dedicated to sophisticated local search algorithms, is proposed. According to certain theoretical neighborhood properties, a high effective neighborhood search method is developed. Additionally, new properties of, so called, partial solution graph are revealed, especially used for some construction algorithms and neighborhoods generated by insert moves. The supplementation of the properties provides experimental investigations on solution space topology (especially the Big Valley) that are realized, among others, using a new, originally introduced, distance measure between solutions.

In chapter four, there are presented approximate algorithms to solve the problem stated in chapter three. Algorithms are designed using different construction and local search techniques, for example, simulated annealing, tabu search and scatter search methods. However, the most interesting seems to be the genetic approach, where the new crossover quasi-operator utilizing the new distance measure is used.

In chapter five, the FMS with job-shop structure and unconstrained assignment of the AGVs to transport activities is considered. The analysis of the problem is carried out through picking out similarities, differences and generalizations of the results obtained in chapters 3-4. It is also proposed the neighborhood extension obtained by addition of insert move set (the neighborhood allows to change the job handling order by machines and AGVs as well the assignment of vehicles to transport activities) and the effective neighborhood searching method. Also, in this chapter algorithms of other kinds are formulated.

In chapter six of the thesis, the manufacturing system with the single AGV working in unidirectional mode, moving periodically among machines organized into loop layout and performing all transports is considered. According to production process and arbitrarily selected vehicle working mode, the system can be reduced to a specific form of, so called, permutation flow-shop problem. The obtained model is formally more complex than job-shop models and constitutes a good theoretical framework for further analysis of hybrid problems (among others, serial-parallel systems, with multiple vehicles). Thanks to emerged properties, of the introduced mathematical model and its permutation-graph representation, approximate solving of the problem in possible by using suitable modifications of some literature algorithms dedicated to classic permutation flow-shop scheduling problem.

Each designed algorithm was tested numerically with the use of test instance sets. Finally, the empirical evaluations of the algorithms' quality allow to select the best algorithm. Also, upper bounds for significant number of instances known from the literature were improved.

Rozdział 1

Wprowadzenie

Rozwój współczesnych systemów wytwarzania zmierza w kierunku automatycznych, zrobotyzowanych, bezludnych systemów elastycznych. Elastyczny system produkcyjny (ESP, ang. *flexible manufacturing system*) to system wytwarzania złożony ze zautomatyzowanych podsystemów wykonawczych, magazynowych i transportowych, których praca jest koordynowana przez komputerowy system sterowania. W ESP wszystkie czynności pomocnicze, wykonywane w systemie tradycyjnym przez człowieka (operatora), są realizowane przez system automatyczny. System taki może pracować w sposób ciągły bez ingerencji człowieka, co znacznie zwiększa jego wydajność i niezawodność. Elastyczność systemu wynika zasadniczo z dwóch faktów: (1) możliwości równoległej realizacji wielu zadań produkcyjnych o różnym charakterze, (2) umiejętności szybkiej adaptacji (przeprogramowania) poszczególnych podsystemów, pozwalającej na częste zmiany profilu produkcji. Elementem kluczowym dla wydajności ESP jest zdolność do szybkiej konstrukcji efektywnych (optymalnych bądź prawie optymalnych) harmonogramów pracy. Eliminacja człowieka, najbardziej zawodnego czynnika w systemie, pozwala rozpatrywać ESP w kategoriach deterministycznych systemów dyskretnych. Stąd, właściwym narzędziem do analizy i optymalizacji w ESP może być deterministyczna teoria szeregowania zadań, użyta w rozprawie.

Do podstawowych modeli rzeczywistych systemów produkcyjnych zalicza się tzw. problem gniazdowy oraz problem przepływowy (także hybrydowy). Kolejne uogólnienia oraz rozszerzenia wymienionych problemów pozwalają na coraz doskonalsze modelowanie pracy coraz większej grupy rzeczywistych systemów produkcyjnych, a co za tym idzie, na wzrost wydajności ich pracy. Wśród najbardziej interesujących rozszerzeń są dodatkowe ograniczenia czasowe i zasobowe, takie jak: czasy przezbrojeń maszyn,

skończona liczba i pojemność buforów międzystanowiskowych, ograniczenia czasów oczekiwania, ograniczenia składowania, skończona liczba palet, ograniczony czas transportu międzystanowiskowego. Jeden z ważnych kierunków badań, będący aktualnie w fazie rozwoju, dotyczy czasów transportu przy jednoczesnym uwzględnieniu skończonej liczby wykorzystywanych środków transportu. W systemach ESP odpowiada to polityce zarządzania automatycznym transportem realizowanym na przykład przy użyciu wózków AGV (ang. *automated guided vehicle*). Właśnie ten ostatni kierunek badań jest rozwijany w niniejszej rozprawie.

Deterministyczne problemy szeregowania zadań można zasadniczo podzielić na dwie grupy. Do pierwszej z nich, stosunkowo nielicznej, należą relatywnie proste problemy posiadające algorytmy wyznaczania rozwiązania optymalnego o wielomianowej złożoności obliczeniowej. Drugą, zdecydowanie większą grupę generowaną przez praktykę przemysłową, stanowią problemy NP-trudne, które, przy obecnym stanie wiedzy, wymagają znacznego czasu do wyznaczenia rozwiązania optymalnego. Potrzeby praktyki w tej drugiej grupie są tradycyjnie rozwiązywane przy użyciu jednego z dwóch ogólnych podejść: (a) poprzez użycie algorytmów dokładnych o wykładniczej złożoności obliczeniowej (np. schemat podziału i ograniczeń, schemat programowania dynamicznego), mających zastosowanie tylko do problemów o niewielkich rozmiarach, (b) poprzez zastosowanie algorytmów przybliżonych. Algorytmy klasy (b) posiadają zwykle (choć niekoniecznie) wielomianową złożoność obliczeniową oferując stosunkowo krótki czas pracy, co powoduje, że cieszą się niesłabnącym zainteresowaniem praktyków. Algorytmy przybliżone nie dają gwarancji odnalezienia rozwiązania optymalnego problemu, nie oznacza to jednak, że odnalezienie takiego rozwiązania przez te algorytmy nie jest możliwe. Najnowsze algorytmy przybliżone pozwalają elastycznie kształtować kompromis pomiędzy czasem obliczeń a dokładnością przybliżenia. Dla poprawy własności numerycznych algorytmów klasy (a) często wykorzystuje się specyficzne własności problemu, pozwalające zaprojektować specjalizowany algorytm dokładny, szybciej zbiegający do rozwiązania optymalnego, niż algorytm ogólny tej klasy. Specjalizowane w ten sposób algorytmy klasy (a) pozwalają rozwiązywać „nieco" większe przykłady problemów, jednak generalnie nie są w stanie uniknąć „eksplozji obliczeń" charakterystycznej dla problemów NP-trudnych. Dotychczasowe wyniki badań pokazują, że zastosowanie tych samych własności do projektowania algorytmów w klasie (b) dostarcza metod rozwiązywania o niespotykanie dobrych cechach numerycznych. Dodatkowo możliwe jest sformułowanie innych, nowych własności wspierających algorytmy tej klasy. To ostatnie podejście będzie twórczo rozwijane w niniejszej rozprawie.

1.1 Tezy pracy

Uwzględnienie dodatkowych ograniczeń w problemie szeregowania zadań wiąże się ze wzrostem skomplikowania samego problemu jak również pociąga za sobą konieczność modyfikacji istniejących bądź zaprojektowania nowych metod jego rozwiązywania. W związku z tym wysuwam następujące tezy:

1. możliwa jest modyfikacja istniejących modeli matematycznych problemów szeregowania zadań, umożliwiająca bardziej precyzyjne zamodelowanie zjawisk związanych z transportem w większej grupie elastycznych systemów produkcyjnych, przy jednocześnie niewielkim wzroście ich skomplikowania,
2. możliwe jest wychwycenie nowych własności problemów w rozpatrywanej klasie bądź wykazanie prawdziwości własności analogicznych do niektórych własności znanych dla problemów klasycznych,
3. dzięki wykorzystaniu specyficznych własności rozważanych problemów możliwa jest modyfikacja istniejących, bądź konstrukcja nowych, efektywnych czasowo algorytmów przybliżonych, generujących rozwiązania wysokiej jakości.

Prawdziwość powyżej postawionych tez zostanie wykazana poprzez:

a. zaproponowanie modeli matematycznych i reprezentacji permutacyjno-grafowych problemów gniazdowych oraz przepływowych z transportem; w modelach tych uwzględnia się ograniczenia technologiczne związane ze skończoną liczbą środków transportu,
b. wykrycie i udowodnienie specyficznych własności wspomnianych problemów,
c. implementację przybliżonych algorytmów rozwiązywania badanych problemów,
d. przeprowadzenie badań numerycznych, ocenę jakości zaimplementowanych algorytmów i przedstawienie wniosków wynikających z przeprowadzonych badań.

Rolę środków transportu (maszyn transportowych) mogą pełnić roboty manipulacyjne, dźwigi, taśmociągi, wózki widłowe, suwnice bramowe, układnice regałowe, etc. Jednakże, ze względu na ich praktyczność i rosnącą popularność, rozważania na temat środków transportu będą prowadzone głównie w kontekście wózków AGV. Konsekwencją udowodnienia powyższych tez jest możliwość zwiększenia wydajności pracy systemów produkcyjnych o strukturze przepływowej bądź gniazdowej, w których czasy transportu międzystanowiskowego mają istotny wpływ na całokształt procesu technologicznego.

1.2 Zakres pracy

Na pracę składa się 7 rozdziałów. W rozdziale 2 znajduje się wprowadzenie do problematyki rozpatrywanej w rozdziałach 3-7. Zawiera podstawowe definicje, opis symboli, modele matematyczne oraz permutacyjno-grafowe problemów klasycznych, które podlegają uogólnieniu w dalszych rozdziałach pracy. W rozdziale tym, poza opisem podstawowych technik przybliżonego rozwiązywania problemów, znajduje się też przegląd literatury pod kątem osiągnięć w zakresie problematyki szeregowania zadań z transportem. Pozostałe rozdziały stanowią zasadniczą część pracy, w której prezentuje się całkowicie oryginalne rezultaty uzyskane przez autora.

W rozdziale 3 rozważa się ESP o strukturze gniazdowej ze zbiorem dedykowanych wózków AGV. W modelu matematycznym i reprezentacji permutacyjno-grafowej sformułowanego problemu uwzględnia się szczególnie praktyczny tryb pracy wózków „zabierz i zostaw". Następnie prezentowane są własności będące wynikiem wychwycenia podobieństw i różnic pomiędzy badanym problemem a problemem klasycznym. Dalej, w kontekście rozważań wykazujących obecność wielkiej doliny w krajobrazie przestrzeni rozwiązań problemu, prezentowana jest nowa miara odległości pomiędzy poszczególnymi rozwiązaniami. W rozdziale 4 prezentuje się algorytmy przybliżonego rozwiązywania problemu sformułowanego w rozdziale trzecim. Między innymi, zaprezentowano nowy quasi-operator krzyżowania wykorzystujący, wspomnianą powyżej, nową miarę odległości. W dalszej części rozdziału prezentowane są wyniki badań numerycznych proponowanych algorytmów oraz poprawione wartości górnych ograniczeń wyznaczonych dla literaturowych instancji testowych. W rozdziale 5 poddano analizie ESP o strukturze gniazdowej ze swobodnym przydziałem wózków AGV do poszczególnych transportów. Dalej, podobnie jak w rozdziałach wcześniejszych, przedstawione są własności sformułowanego problemu, algorytmy rozwiązywania, wyniki badań numerycznych oraz poprawione wartości górnych ograniczeń wyznaczonych dla literaturowych instancji testowych. W rozdziale 6 rozważany jest permutacyjny problem przepływowy z transportem i jednym wózkiem AGV. Problem modeluje szczególnie praktyczny przypadek elastycznego systemu produkcyjnego, w którym maszyny produkcyjne zorganizowane są w układ typu pętla, zaś na wózku wymusza się jednokierunkowy tryb pracy. Ze względu na większy stopień skomplikowania modelu matematycznego w stosunku do problemów gniazdowych z rozdziałów 3-5, problem ten prezentowany jest jako ostatni. Podobnie jak w rozdziałach 3-5, w rozdziale szóstym przedstawia się własności problemu, proponuje się algorytmy rozwiązywania i prezentuje się wyniki badań numerycznych.

Rozdział 2

Problemy szeregowania zadań

Teoria szeregowania zadań, obejmująca część problematyki związanej z optymalizacją kombinatoryczną, jest tematem intensywnych badań od początku lat pięćdziesiątych dwudziestego wieku. Początkowo problematyka ta miała swoje zastosowanie w optymalizacji pracy systemów komputerowych i produkcyjnych. Z czasem zakres zastosowań praktycznych szybko objął wiele innych dziedzin życia, takich jak np. zarządzanie zasobami ludzkimi, opieka medyczna, rolnictwo, budownictwo, czy transport. Do najważniejszych i najintensywniej badanych problemów szeregowania zadań, ze względu na bogactwo zastosowań praktycznych, można zaliczyć problemy jednomaszynowe, przepływowe i gniazdowe.

Dużą część uwagi badaczy pochłonęła analiza problemów szeregowania zadań z punktu widzenia teorii złożoności obliczeniowej. Zgodnie z tą teorią, wszystkie problemy szeregowania zadań można zasadniczo zaliczyć do jednej z trzech podstawowych grup. Pierwszą grupę stanowią tzw. problemy wielomianowe. Każdy problem z tej grupy może być rozwiązany optymalnie w czasie wielomianowo-zależnym od jego rozmiaru. Drugą, zdecydowanie większą grupę stanowią problemy NP-trudne. W przypadku problemów należących do tej grupy znalezienie rozwiązania optymalnego w czasie wielomianowym nie jest możliwe. W trzeciej grupie znajdują się problemy, których nie udało się jeszcze zakwalifikować do grupy pierwszej bądź drugiej.

NP-trudność danego problemu wiąże się z koniecznością wyboru odpowiedniej metody jego rozwiązywania. Wszystkie metody można podzielić na dwie podstawowe grupy: dokładne i heurystyczne (przybliżone). Algorytmy dokładne, poprzez umiejętnie prowadzony przegląd wszystkich rozwią-

zań, gwarantują odnalezienie rozwiązania optymalnego w skończonym czasie. Wykładnicza złożoność obliczeniowa tych algorytmów jednak sprawia, że czas poszukiwań rozwiązania optymalnego (nawet dla problemów o stosunkowo niewielkim rozmiarze) często nie jest akceptowalny dla praktyków. Stosunkowo dużych trudności nastręcza również projektowanie i implementacja takich algorytmów. Powyższe kłopoty sprawiły, że począwszy od lat siedemdziesiątych ubiegłego stulecia uwaga badaczy i praktyków stopniowo zaczęła się skupiać wokół metod przybliżonych. Cechą charakterystyczną tych metod jest rezygnacja z wymogu odnalezienia rozwiązania optymalnego na rzecz rozwiązania bliskiego (w sensie wartości funkcji kryterialnej) rozwiązaniu optymalnemu. Zysk z takiego podejścia objawia się w zmniejszonej, wielomianowej złożoności obliczeniowej algorytmów i, w konsekwencji, znacznie skróconym czasie pracy.

Postęp wiedzy w zakresie teorii szeregowania zadań, ogromny postęp technologiczny i upowszechnienie się komputerów osobistych sprawiły, że zarówno badacze jak i firmy produkujące oprogramowanie komercyjne ponownie zainteresowały się tematem technik dokładnych. Tutaj należy podkreślić, że użycie algorytmu dokładnego wciąż nie gwarantuje odnalezienia rozwiązania optymalnego w satysfakcjonującym czasie – poszukiwania można jednak przerwać, zadowalając się rozwiązaniem przybliżonym. W konsekwencji rozwoju obu nurtów powstał szereg pakietów optymalizacyjnych, takich jak Ilog CPLEX czy Lindo, głównie dostępnych pod postacią bibliotek dla popularnych języków programowania oraz solverów dla popularnych arkuszy kalkulacyjnych.

2.1 Klasyfikacja problemów

Instancję każdego problemu szeregowania zadań można opisać jako parę (X, F), gdzie $X \subseteq X^0$ jest zbiorem rozwiązań dopuszczalnych problemu, X^0 jest zbiorem wszystkich rozwiązań (przestrzenią rozwiązań), zaś $F : X \to R$ jest przyjętą funkcją celu (kryterium optymalizacji). Problem polega na odnalezieniu rozwiązania globalnie optymalnego $x^* \in X$ takiego, że

$$F(x^*) = \min_{x \in X} F(x), \tag{2.1}$$

gdzie x jest rozwiązaniem problemu. Instancja (X, F) praktycznie nigdy nie jest dana jawnie (tzn. jako lista rozwiązań i odpowiadających im wartości funkcji celu). Przestrzeń rozwiązań z reguły jest reprezentowana przez zestaw zmiennych niezależnych, mogących przyjmować wartości z pewnych zakresów. Te zmienne mogą bezpośrednio odnosić się do modelu matematycznego formułującego problem.

Dalsze rozważania w tej pracy będą ograniczone do tzw. „problemów kolejnościowych" (ang. *shop-scheduling problems*). Problemy z tej grupy należą do najbardziej praktycznych, jednocześnie najtrudniejszych problemów szeregowania zadań. W ich sformułowaniu matematycznym zawsze występuje zbiór m maszyn, który poniżej będzie oznaczany przez $M = \{1, 2, \ldots, m\}$, oraz zbiór r zadań produkcyjnych, który będzie oznaczany przez $J = \{1, 2, \ldots, r\}$. W problemach kolejnościowych elementem charakterystycznym jest podział zadań na operacje, przy czym operacje należące do tego samego zadania nie mogą być wykonywane równolegle. Zbiór wszystkich operacji będzie oznaczany przez $O = \{1, 2, \ldots, n\}$, gdzie $n = \sum_{k \in J} n_k$ jest liczbą wszystkich operacji, zaś n_k oznacza liczbę operacji należących do zadania k. Klasycznie, w problemach kolejnościowych zakłada się nielimitowaną pojemność buforów oraz jednostkową przepustowość każdej z maszyn. Zakłada się również, że wykonanie raz rozpoczętej operacji nie może być przerwane.

Jedną z pierwszych prób usystematyzowania problemów szeregowania zadań podjęto w roku 1979 w pracy [37] wprowadzając trójpolową notację $\alpha|\beta|\gamma$. Notację tą nazwano – od nazwiska jednego z autorów – notacją Grahama. W latach późniejszych notacja ta uległa dość znacznej rozbudowie. Na przykład, w pracach [43, 50] proponuje się rozszerzenie notacji o elementy związane z transportem. Omówienie owej notacji, ograniczone do grupy problemów kolejnościowych z uwzględnieniem transportu, znajduje się poniżej.

Pole $\alpha = \alpha_1, \alpha_2$ notacji Grahama determinuje typ problemu (park maszynowy), gdzie

1. $\alpha_1 = G$ oznacza problem ogólny (ang. *general-shop problem*); każde zadanie wykonywane jest według indywidualnej marszruty technologicznej, relacje kolejnościowe pomiędzy operacjami poszczególnych zadań nie są dane.

2. $\alpha_1 = O$ oznacza problem otwarty (ang. *open-shop problem*); wszystkie zadania są wykonywane według tej samej marszruty technologicznej, każda operacja danego zadania jest wykonywana na innej maszynie, relacje kolejnościowe pomiędzy operacjami poszczególnych zadań nie są dane.

3. $\alpha_1 = J$ oznacza problem gniazdowy (ang. *job-shop problem*); każde zadanie wykonywane jest według indywidualnej marszruty technologicznej, relacje kolejnościowe pomiędzy operacjami poszczególnych zadań są narzucone.

4. $\alpha_1 = F$ oznacza problem przepływowy (ang. *flow-shop problem*); wszystkie zadania są wykonywane według tej samej marszruty technologicznej, każda operacja danego zadania jest wykonywana na innej maszynie, relacje kolejnościowe pomiędzy operacjami poszczególnych zadań są narzucone.

5. $\alpha_1 = FP$ oznacza permutacyjny problem przepływowy (ang. *permutation flow-shop problem*); wszystkie zadania są wykonywane według tej samej marszruty technologicznej, każda operacja zadania wykonywana jest na innej maszynie, relacje kolejnościowe pomiędzy operacjami poszczególnych zadań są narzucone. Dodatkowo, kolejność wykonania zadań przez każdą z maszyn jest taka sama.

Pole $\alpha_2 = 1, 2, \ldots$ występuje w notacji nieobowiązkowo (może pozostać puste) i oznacza liczbę maszyn produkcyjnych. W pracach [43, 50] proponuje się rozszerzenie pola α o dwa dodatkowe pola α_3, α_4, określające konfigurację maszyn transportowych[1] (środków transportu). Jeżeli zakłada się nieograniczoną liczbę identycznych maszyn transportowych – pola α_3, α_4 pozostają puste. W przypadku ograniczonej liczby identycznych maszyn (gdy każde zadanie może być transportowane przez dowolną z maszyn) przyjmuje się $\alpha_3 = R$. W przypadku, w którym zadania mogą być transportowane przez podzbiór maszyn, przyjmuje się $\alpha_3 = MPR$ (od ang. *multipurpose robots* – roboty wielozadaniowe). W przypadku dedykowanych maszyn transportowych (każde zadanie może być transportowane tylko przez jedną, konkretną maszynę) przyjmuje się $\alpha_3 = DR$ (od ang. *dedicated robots*). Pole $\alpha_4 = 1, 2, \ldots$ występuje w notacji nieobowiązkowo i oznacza liczbę maszyn transportowych.

Pole β określa obecność bądź brak pewnych dodatkowych ograniczeń technologicznych (charakterystykę zadania). Poniżej zostaną omówione tylko ograniczenia związane z czasami transportu zadań i przejazdów[2] bez załadunku (przejazdów pustych) poszczególnych maszyn transportowych, wyszczególnione w pracach [43, 50]. Pole β może przyjmować poniższe wartości.

1. t_{jkl} oznacza zadaniowo- i maszynowo-zależne czasy transportu,

2. $t_{jkl} = t_{jlk}$ oznacza zadaniowo- i maszynowo-zależne, symetryczne czasy transportu,

[1] Oryginalnie, w cytowanych pracach używano terminu „roboty", przez co rozumiane były zarówno roboty stacjonarne jak i mobilne.

[2] W przypadku robotów stacjonarnych należy mówić o „ruchach bez załadunku" bądź „ruchach pustych" (od ang. *empty moves*).

3. t_j oznacza zadaniowo-zależne czasy transportu, niezależne od maszyn,
4. $t_j \in \{T_1, T_2\}$ oznacza zadaniowo-zależne czasy transportu, niezależne od maszyn, które mogą przyjmować tylko jedną z dwóch wartości T_1, T_2,
5. t_{kl} oznacza maszynowo-zależne czasy transportu, niezależne od zadań,
6. $t_{kl} = t_{lk}$ oznacza maszynowo-zależne, niezależne od zadań, symetryczne czasy transportu,
7. $t_{jkl} = T$ oznacza, że wszystkie czasy transportu są równe wartości T,
8. t'_{kl} oznacza maszynowo-zależne czasy przejazdów (ruchów) pustych,
9. $t'_{kl} = T'$ oznacza, że wszystkie czasy przejazdów (ruchów) pustych są równe wartości T'.

Puste pole β oznacza brak dodatkowych ograniczeń technologicznych, gdzie czasy wszystkich transportów i przejazdów pustych są zerowe.

Pole γ określa kształt funkcji celu. Niech f_k będzie funkcją związaną z każdym zadaniem k zależną od momentu zakończenia jego wykonywania C_k, $k \in J$. Wyróżnia się dwa typy kryteriów optymalizacji:

1. kryteria minimaksowe, $f_{\max} = \max_{k \in J} f_k(C_k)$ oraz
2. kryteria addytywne, $\sum f_k = \sum_{k \in J} f_k(C_k)$.

Jednym z najbardziej praktycznych kryteriów minimaksowych jest moment zakończenia wykonywania wszystkich zadań (zwany również długością uszeregowania), równy $C_{\max} = \max_{k \in J} C_k$. Praktyczność kryterium wynika z przynajmniej dwóch faktów. Po pierwsze, kryterium to jest równoważne niektórym kryteriom związanym z właściwym wykorzystaniem maszyn, takim jak czas przestoju wszystkich maszyn, ważona suma czasów przestoju wszystkich maszyn czy średni stopień wykorzystania maszyn. Równoważność dwóch kryteriów oznacza tutaj, że rozwiązanie optymalne ze względu na jedno z nich pozostaje optymalne po przyjęciu kryterium drugiego. Po drugie, dysponując rozwiązaniem optymalnym ze względu na wiele innych kryteriów, takich jak np. maksymalna terminowość wykonania zadań czy suma opóźnień wszystkich zadań, dysponujemy również rozwiązaniem optymalnym dla kryterium $C_{\max}$.

Notacja Grahama z pewnością przyczyniła się do przekształcenia teorii szeregowania zadań z dość bezładnego zbioru pomysłów w systematyczną gałąź wiedzy. Dzięki tej notacji możliwe było zaklasyfikowanie wielu problemów do konkretnych grup, jak np. problemy przepływowe czy gniazdowe.

Ze względu na bogactwo uogólnień i praktycznych zastosowań, permutacyjny problem przepływowy i problem gniazdowy warte są dokładniejszego omówienia.

2.1.1 Permutacyjny problem przepływowy

Omawiany poniżej permutacyjny problem przepływowy w notacji Grahama oznacza się jako problem $FP||C_{\max}$. Problem pozwala zamodelować systemy wytwórcze, w których proces produkcyjny jest oparty o przepływ różnych zadań z jednakowymi marszrutami przejścia. Model matematyczny problemu można przedstawić następująco. Dany jest zbiór m maszyn $M = \{1, 2, \ldots, m\}$ i zbiór r zadań $J = \{1, 2, \ldots, r\}$. Każde zadanie $k \in J$ podzielone jest na m operacji, które muszą być wykonane kolejno na maszynach $1, 2, \ldots, m$. Operacja l zadania k jest wykonywana w czasie $p_{l,k} > 0$. Zbiór wszystkich operacji będzie oznaczany przez $O = \{(l, k) : l \in M, k \in J\}$. Nietrudno zauważyć, że moc zbioru O wynosi $n = |O| = m \cdot r$. Przyjmuje się, że

1. w każdej chwili czasu każda maszyna może wykonywać co najwyżej jedną operację,
2. w każdej chwili czasu można wykonywać co najwyżej jedną operację każdego zadania,
3. raz rozpoczęta operacja nie może być przerwana.

Niech Π będzie zbiorem wszystkich permutacji zbioru J. Uszeregowanie definiuje się jako zestaw czasów rozpoczęcia wykonywania $S_{l,k}$, $l \in M$, $k \in J$, poszczególnych operacji. Uszeregowanie jest dopuszczalne gdy wszystkie powyżej opisywane ograniczenia są spełnione, czego wyrazem jest prawdziwość poniższych nierówności:

$$S_{1,\pi(1)} \geqslant 0, \tag{2.2}$$

$$S_{l,\pi(k-1)} + p_{l,\pi(k-1)} \leqslant S_{l,\pi(k)}, \qquad 1 < k \leqslant r, l \in M, \tag{2.3}$$

$$S_{l-1,\pi(k)} + p_{l-1,\pi(k)} \leqslant S_{l,\pi(k)}, \qquad k \in J, 1 < l \leqslant m, \tag{2.4}$$

Niech $C_{l,k} = S_{l,k} + p_{l,k}$ będzie momentem zakończenia wykonywania operacji $(l, k) \in O$. Problem polega na odnalezieniu takiego uszeregowania dopuszczalnego, dla którego moment zakończenia wykonywania procesu technologicznego, równy $C_{\max} = \max_{k \in J} C_{m,k}$, przyjmuje wartość minimalną. Problem jest silnie NP-trudny.

W wielu praktycznych sytuacjach, jako narzędzie umożliwiające wygodne i szybkie wyznaczenie wartości funkcji celu, warto wykorzystać model permutacyjno-grafowy problemu, przedstawiony poniżej. Niech

$G(\pi) = (O, E)$, $\pi \in \Pi$, będzie skierowanym grafem ze zbiorem obciążonych wierzchołków O i zbiorem nieobciążonych łuków $E = E^T \cup E^K$, gdzie

$$E^T = \bigcup_{l=1}^{m} \bigcup_{k=1}^{r-1} \{((l,k),(l,k+1))\} \tag{2.5}$$

jest zbiorem łuków technologicznych, określających kolejność wykonania operacji w zadaniach, zaś

$$E^K = \bigcup_{l=1}^{m-1} \bigcup_{k=1}^{r} \{((l,k),(l+1,k))\} \tag{2.6}$$

to zbiór łuków kolejnościowych, określających kolejność wykonania operacji przez poszczególne maszyny. Każdy wierzchołek $(l,k) \in O$ reprezentuje operację i przyjmuje obciążenie $p_{l,\pi(k)}$. Należy zauważyć, że długość najdłuższej ścieżki dochodzącej do każdego wierzchołka (l,k) (bez jego obciążenia $p_{l,\pi(k)}$) jest równa najwcześniejszemu możliwemu momentowi rozpoczęcia wykonania reprezentowanej operacji w uszeregowaniu reprezentowanym przez permutację π. Wyznaczenie długości najdłuższych ścieżek dochodzących do poszczególnych wierzchołków implikuje zatem możliwość szybkiego wyznaczenia dowolnej wartości funkcji celu związanej z czasami zakończenia operacji. W szczególności, długość $C_{\max}(\pi)$ najdłuższej ścieżki (ścieżki krytycznej) w grafie $G(\pi)$ jest równa przyjętej wartości funkcji kryterialnej, $C_{\max}(\pi) = \max_{k \in J} C_{m,k}$. Zatem, problem sprowadza się do odnalezienia takiej permutacji $\pi \in \Pi$, dla której długość ścieżki krytycznej w grafie $G(\pi)$ przyjmuje wartość minimalną.

2.1.2 Problem gniazdowy

Omawiany poniżej problem gniazdowy w notacji Grahama oznacza się jako problem $J||C_{\max}$. Problem pozwala na zamodelowanie konwencjonalnych i automatycznych systemów wytwórczych o dowolnej strukturze stanowisk z procesem wytwarzania danym w formie niezależnych zadań o ustalonych (lecz różnych) marszrutach technologicznych. Model matematyczny problemu można przedstawić następująco. Dany jest zbiór m maszyn $M = \{1, 2, \ldots, m\}$ i zbiór r zadań $J = \{1, 2, \ldots, r\}$. Każde zadnie $k \in J$ podzielone jest na n_k operacji indeksowanych przez $j_k+1, j_k+2, \ldots, j_k+n_k$, $j_k = \sum_{i=1}^{k-1} n_i$, $j_1 = 0$, które powinny być wykonane w tej kolejności. Zbiór wszystkich operacji oznacza się przez $O = \{1, 2, \ldots, n\}$, gdzie $n = \sum_{k \in J} n_k$. Każda operacja $i \in O$ musi być wykonana na maszynie $m_i \in M$ w czasie $p_i > 0$. Zakłada się, że

1. w każdej chwili czasu każda maszyna może wykonywać co najwyżej jedną operację,
2. w każdej chwili czasu można wykonywać co najwyżej jedną operację każdego zadania,
3. raz rozpoczęta operacja nie może być przerwana.

Uszeregowanie definiuje się poprzez wektor $S = (S_1, S_2, \ldots, S_n)$ czasów rozpoczęcia wykonywania S_i poszczególnych operacji $i \in O$. Uszeregowanie jest dopuszczalne gdy wszystkie powyższe ograniczenia są spełnione, czego wyrazem jest prawdziwość poniższych nierówności:

$$S_{j_k+1} \geqslant 0 \qquad k \in J, \tag{2.7}$$

$$C_i \leqslant S_{i+1}, \qquad j_k + 1 \leqslant i < j_k + n_k, k \in J, \tag{2.8}$$

$$(C_i \leqslant S_j) \dot{\vee} (C_j \leqslant S_i), \qquad i, j \in O, i \neq j, m_i = m_j, \tag{2.9}$$

gdzie $C_i = S_i + p_i$ jest momentem zakończenia wykonywania operacji $i \in O$. Za kryterium optymalizacji przyjmuje się moment zakończenia wykonywania wszystkich zadań, równy $C_{\max} = \max_{i \in O} C_i$. Problem polega na odnalezieniu takiego uszeregowania dopuszczalnego, by kryterium optymalizacji przyjęło wartość minimalną. Problem jest silnie NP-trudny.

Powyżej przedstawiony model przedziałowy jest jednym z wielu modeli problemu znanych w literaturze. Przykładowo, zamiana warunku dysjunktywnego, danego równaniem (2.9), na dwa warunki ze zmiennymi binarnymi prowadzi do binarno-ciągłego zadania programowania liniowego. Kolejnym modelem jest model dysjunktywny. W grafie dysjunktywnym warunek (2.9) reprezentowany jest przez nieskierowane łuki pomiędzy każdą parą operacji wykonywanych na tej samej maszynie. Ustalenie zwrotu łuków jest równoznaczne z określeniem kolejności wykonania poszczególnych operacji. W modelu listowym problemu jako zmienną niezależną wykorzystuje się permutację zbioru (listę) wszystkich operacji. Zaletą podejścia jest dopuszczalność wszystkich rozwiązań, wadą – ogromna redundancja przestrzeni rozwiązań.

Jak napisano w pracy [93] „wybór modelu jest kompromisem pomiędzy dopuszczalnością, redundancją, dostępnymi własnościami oraz rodzajem projektowanego algorytmu." Pod wieloma względami, w wielu praktycznych sytuacjach najlepszy wydaje się model permutacyjno-grafowy, przedstawiony poniżej. Należy zauważyć, że zbiór operacji można podzielić na m rozłącznych podzbiorów $O_1, O_2, \ldots, O_m$, gdzie podzbiór $O_l = \{i \in O : m_i = l\}$ reprezentuje operacje, które należy wykonać na maszynie l. Kolejność wykonania operacji ze zbiorów $O_1, \ldots, O_m$ na poszczególnych maszynach określa zestaw permutacji $\pi = (\pi_1, \pi_2, \ldots, \pi_m)$, gdzie

$\pi_l = (\pi_l(1), \pi_l(2), \ldots, \pi_l(o_l))$, $o_l = |O_l|$, jest permutacją zbioru O_l. Niech Π_l oznacza zbiór wszystkich permutacji zbioru O_l. Wtedy zbiór wszystkich zestawów permutacji określa się przez $\Pi^0 = \Pi_1 \times \Pi_2 \times \cdots \times \Pi_m$. Dla uproszczenia, zestaw permutacji $\pi \in \Pi$ będzie nazywany permutacją bądź, po prostu, rozwiązaniem problemu. Niech $G(\pi) = (O, E(\pi))$, $\pi \in \Pi^0$, będzie skierowanym grafem ze zbiorem obciążonych wierzchołków O i zbiorem nieobciążonych łuków $E(\pi) = E^T \cup E^K(\pi)$, gdzie

$$E^T = \bigcup_{i=1}^{r} \bigcup_{j=1}^{n_i-1} \{(j, j+1)\} \tag{2.10}$$

jest zbiorem łuków technologicznych, określających relacje kolejnościowe pomiędzy operacjami w poszczególnych zadaniach, zaś

$$E^K(\pi) = \bigcup_{l=1}^{m} \bigcup_{j=1}^{o_l-1} \{(\pi_l(j), \pi_l(j+1))\} \tag{2.11}$$

to zbiór łuków kolejnościowych, określających relacje kolejnościowe pomiędzy operacjami wykonywanymi na poszczególnych maszynach. Każdy wierzchołek $i \in O$ reprezentuje operację i przyjmuje obciążenie równe p_i.

Permutacja π reprezentująca uszeregowanie dopuszczalne będzie nazywana permutacją dopuszczalną (rozwiązaniem dopuszczalnym), zaś zbiór wszystkich permutacji dopuszczalnych będzie oznaczany przez $\Pi \subset \Pi^0$. Z własności grafu wynika, że graf $G(\pi)$ jest acykliczny wtedy i tylko wtedy, gdy $\pi \in \Pi$. Nie trudno zauważyć, że w acyklicznym grafie $G(\pi)$ długość najdłuższej ścieżki dochodzącej do wierzchołka i (bez jego obciążenia p_i) jest równa najwcześniejszemu możliwemu momentowi rozpoczęcia S_i operacji $i \in O$. Wyznaczenie długości najdłuższych ścieżek dochodzących do poszczególnych wierzchołków implikuje zatem możliwość szybkiego wyznaczenia dowolnej wartości funkcji celu związanej z czasami zakończenia wykonania operacji. W szczególności, długość $C_{\max}(\pi)$ ścieżki krytycznej w grafie $G(\pi)$ jest równa wartości przyjętej funkcji kryterialnej, $C_{\max}(\pi) = \max_{i \in O} C_i$. Wobec tego, problem sprowadza się do odnalezienia takiej permutacji $\pi \in \Pi$, dla której długość ścieżki krytycznej w grafie $G(\pi)$ przyjmuje wartość minimalną.

2.2 Algorytmy rozwiązywania problemów szeregowania zadań

Jak już wspomniano, wszystkie metody szeregowania zadań można podzielić na metody dokładne i przybliżone. Wśród metod dokładnych można wyróżnić m. in. metody oparte o schemat podziału i ograniczeń (ang.

Branch and Bound), metody przeglądu sterowanego, metody płaszczyzn odcinających, metody oparte o schemat programowania dynamicznego, czy metody subgradientowe. Ze względu na powszechnie znane wady – algorytmy dokładne nie będą dalej omawiane.

Wśród algorytmów przybliżonych można wyszczególnić dwie duże grupy: algorytmy konstrukcyjne, które, jak mówi sama nazwa – konstruują rozwiązanie problemu, oraz algorytmy lokalnych poszukiwań (ang. *local search algorithms*). Schemat działania algorytmów konstrukcyjnych jest w dużym stopniu zależny od specyfiki problemu. Wśród tych algorytmów można jednak wyszczególnić dwie duże grupy: algorytmy priorytetowe oraz algorytmy wykorzystujące tzw. „technikę wstawień". Algorytmy priorytetowe szeregują zadania (operacje) względem niemalejących priorytetów określanych przy pomocy całej gamy reguł priorytetowych. Algorytmy typu wstaw, przed uszeregowaniem danego zadania (operacji), tworzą zbiór rozwiązań „próbnych" poprzez próbne wstawienie zadania (operacji) na różne pozycje w rozwiązaniu bieżącym, utworzonym w poprzedniej iteracji algorytmu. Spośród rozwiązań próbnych wybierane jest rozwiązanie najlepsze (w sensie wartości funkcji celu), które staje się rozwiązaniem bieżącym w następnej iteracji algorytmu.

Algorytmy lokalnych poszukiwań, rozpoczynając pracę od pewnego rozwiązania początkowego (dostarczonego przez algorytm konstrukcyjny), „starają się" iteracyjnie poprawiać rozwiązanie, wykorzystując w tym celu różnego rodzaju techniki przeglądu przestrzeni rozwiązań. Wśród najlepszych algorytmów w tej grupie można wyszczególnić algorytmy symulowanego wyżarzania (ang. *simulated annealing*), algorytmy genetyczne (ang. *genetic algorithms*), algorytmy poszukiwania rozproszonego (ang. *scatter search*) oraz algorytmy oparte na metodzie poszukiwań z zabronieniami (ang. *tabu search*).

Jednym z podstawowych pojęć związanych z algorytmami lokalnych poszukiwań jest pojęcie ruchu. Ruch można opisać jako czynność polegającą na przejściu pomiędzy dwoma rozwiązaniami. Bardziej precyzyjnie, ruch można przedstawić jako funkcję $v(x) : X \to X$, gdzie $v(x) = x_v \in X$, $x_v \neq x$. Rozwiązanie x_v, powstałe w wyniku „wykonania ruchu" v będzie nazywane rozwiązaniem sąsiednim, bądź po prostu sąsiadem rozwiązania x. Dla każdego rozwiązania $x \in X$ można zdefiniować zbiór ruchów $V(x)$, który generuje sąsiedztwo

$$N(x) = \{x_v : v \in V(x)\} \tag{2.12}$$

rozwiązania x. Definicja zbioru ruchów (i sąsiedztwa) jest zależna od specyfiki analizowanego problemu oraz algorytmu jego rozwiązywania.

Kolejnym ważnym elementem jest lokalna optymalność rozwiązania. Dana jest instancja problemu (X, F). Rozwiązanie $x \in X$ jest lokalnie optymalne w odniesieniu do sąsiedztwa $N(x)$ wtedy i tylko wtedy gdy zachodzi nierówność

$$F(x) \leqslant F(y), \qquad y \in N(x). \tag{2.13}$$

W literaturze (a także w tej pracy), w kontekście algorytmów lokalnego poszukiwania często pojawia się termin „najlepsze rozwiązanie" odnalezione przez dany algorytm. Pojęcie to jest oczywiście wynikiem pewnego skrótu myślowego wskazującego na rozwiązanie generujące najmniejszą wartość funkcji celu spośród wszystkich rozwiązań przeglądniętych przez dany algorytm.

2.2.1 Algorytmy poszukiwań z zabronieniami

Technika poszukiwań z zabronieniami została zaproponowana po raz pierwszy w pracy Glovera [28] w 1989 r. Już pierwsze eksperymenty wykazały, że technika ta może konkurować z niemal wszystkimi innymi znanymi technikami. Z czasem technika uległa wielu modyfikacjom i rozbudowie. Algorytmy tabu search zostały wyposażone w dodatkowe mechanizmy intensyfikacji i dywersyfikacji obliczeń i z powodzeniem znalazły szerokie zastosowanie w rozwiązywaniu problemów kombinatorycznych.

W każdej iteracji algorytmu z sąsiedztwa rozwiązania bieżącego wybierane jest rozwiązanie najlepsze (w odniesieniu do wartości funkcji celu), po czym rozwiązanie to staje się rozwiązaniem bieżącym w następnej iteracji algorytmu. W przeciwieństwie do algorytmów zstępujących, w momencie odnalezienia rozwiązania lokalnie optymalnego algorytm tabu kontynuuje poszukiwania. Takie podejście zwiększa szansę odnalezienia rozwiązania globalnie optymalnego, jednakże, wraz z taką strategią pojawia się ryzyko cyklicznego powracania do rozwiązań odwiedzonych w poprzednich iteracjach algorytmu. Charakterystycznym elementem metody, zabezpieczającym algorytm przed wpadnięciem w taki cykl, jest krótkoterminowa pamięć, zwana listą tabu. Na liście tej zapamiętuje się rozwiązania (częściej jednak ruchy, bądź atrybuty rozwiązań lub ruchów) pozwalające na zidentyfikowanie „wędrówki" algorytmu w ciągu określonej liczby ostatnich iteracji. Rozwiązania identyfikowane przez listę tabu mają status zabronionych i nie mogą stać się rozwiązaniami bieżącymi.

Ogólny schemat algorytmu tabu został przedstawiony na rys. 2.1. W pierwszym kroku algorytmu konstruowane jest rozwiązanie początkowe. W kroku 4 tworzony jest zbiór $N'(x)$ poprzez usunięcie z sąsiedztwa $N(x)$ rozwiązań zabronionych przez listę tabu T. W kroku 5 ze zbioru $N'(x)$ wybierane jest rozwiązanie najlepsze, które zapamiętywane jest na liście

1. Generuj rozwiązanie początkowe x.
2. Połóż $x^* := x$.
3. Powtarzaj kroki 4-7.
4. Wyznacz zbiór $N'(x) \in N(x)$ rozwiązań niezabronionych przez listę T.
5. Znajdź rozwiązanie $y \in N'(x)$ takie, że $F(y) = \min_{z \in N'(x)} F(z)$. Jeżeli $F(y) < F(x^*)$ to podstaw $x^* := y$.
6. Umieść rozwiązanie y na liście T i podstaw $x := y$.
7. Jeżeli spełniony jest warunek końca, to zwróć rozwiązanie x^* i STOP.

Rysunek 2.1: Schemat algorytmu tabu search

tabu i staje się rozwiązaniem bieżącym w następnej iteracji algorytmu. Do najbardziej typowych warunków końca w algorytmach tabu zalicza się ograniczenia liczby iteracji algorytmu oraz ograniczenia czasowe.

2.2.2 Algorytmy symulowanego wyżarzania

Algorytmy symulowanego wyżarzania należą do grupy metod lokalnych poszukiwań, znanej jako algorytmy progowe. Do ich głównych zalet można zaliczyć uniwersalność, łatwość implementacji i szybkość działania. Podstawowe idee algorytmu pochodzą z termodynamiki, a konkretnie wykorzystują analogie do procesu wyżarzania ciała stałego. Algorytmy te, zaproponowane po raz pierwszy w pracy Kirkpatricka i in. [49] w 1983 r., znalazły szerokie zastosowanie w rozwiązywaniu takich zagadnień, jak problem komiwojażera czy problem podziału grafu [2].

Ogólny schemat algorytmu został przedstawiony na rys. 2.2. W pierwszym kroku algorytmu tworzone jest rozwiązanie początkowe. W k-tej iteracji części zasadniczej algorytmu (kroki 4-7) z sąsiedztwa $N(x)$ rozwiązania bieżącego x losowane jest jedno rozwiązanie sąsiednie, które akceptowane jest z pewnym prawdopodobieństwem $P(c_k)$, zależnym między innymi od aktualnej wartości parametru zwanego temperaturą c_k. Jeżeli rozwiązanie zostanie zaakceptowane, staje się rozwiązaniem bieżącym w kolejnej iteracji algorytmu. W przeciwnym przypadku rozwiązanie bieżące nie ulega zmianie.

W kolejnych iteracjach algorytmu temperatura zmienia się zgodnie z przyjętym schematem chłodzenia, poczynając od pewnej temperatury

1. Generuj rozwiązanie początkowe x.
2. Połóż $x^* := x$.
3. Dla $k = 0, 1, 2, ...$ wykonuj kroki 4-7.
4. Wylosuj rozwiązanie $y \in N(x)$.
5. Jeżeli $F(y) < F(x^*)$, to połóż $x^* := y$.
6. Połóż $x := y$ z prawdopodobieństwem $P(c_k)$.
7. Jeżeli spełniony jest warunek końca, to zwróć rozwiązanie x^* i STOP.

Rysunek 2.2: Schemat algorytmu symulowanego wyżarzania

początkowej i kończąc na pewnej temperaturze końcowej. W literaturze zasadniczo rozważa się dwa schematy chłodzenia:

1. schemat geometryczny, gdzie $c_{k+1} = \lambda c_k$, $0 < \lambda < 1$,
2. schemat logarytmiczny, gdzie $c_{k+1} = c_k/(1 + \lambda c_k)$, $0 < \lambda < 1$.

Temperatura początkowa i końcowa są zazwyczaj parametrami, które mogą być wyznaczone przez algorytm „automatycznie" na etapie strojenia. Etap ten wykonywany jest przed zasadniczą częścią algorytmu. Parametry te można też wyznaczyć doświadczalnie.

Oryginalnie, w pracy Kirkpatricka [49] prawdopodobieństwo $P(c_k)$ zaakceptowania rozwiązania $y \in N(x)$ wynosi 1, gdy $F(y) \leqslant F(x)$ oraz $e^{(F(x)-F(y))/c_k}$ w przeciwnym razie. Kryterium akceptacji w powyższej postaci jest powszechnie stosowane do dziś.

2.2.3 Algorytmy genetyczne

Algorytmy genetyczne są algorytmami lokalnych poszukiwań, w których wykorzystuje się analogie do procesów ewolucji zachodzących w przyrodzie. Za prekursora w tej dziedzinie uważa się Hollanda [42]. Pomimo prostoty ogólnej idei, specyfika konkretnego problemu zmusza projektanta algorytmu do rozstrzygnięcia szeregu kwestii takich jak wybór odpowiedniej reprezentacji rozwiązania, sposobu wyznaczania funkcji przystosowania czy dobór odpowiedniego schematu selekcji. Trafność doboru (i często modyfikacji) owych elementów ma nietrywialny wpływ na zachodzący proces ewolucji i jakość algorytmu jako całości.

Schemat algorytmu genetycznego przedstawiono na rys. 2.3. W kroku 1 tworzony jest zbiór P_0 (populacja) zawierający p osobników reprezentujących konkretne rozwiązania problemu. Wielkość p jest z reguły parametrem.

Algorytm rozpoczyna działanie z ustaloną wartością parametru p.

1. Generuj populację początkową $P_0 = \{x^1, x^2, \ldots, x^p\}$.
2. Dla $k = 0, 1, 2, \ldots$ wykonuj kroki 3-5.
3. Wyselekcjonuj z populacji P_k zbiór P'_k osobników.
4. Osobniki ze zbioru P'_k poddaj operacjom krzyżowania i mutacji i utwórz populację P_{k+1}.
5. Jeżeli spełniony jest warunek końca, to zwróć najlepsze znalezione rozwiązanie x^* i STOP.

Rysunek 2.3: Schemat algorytmu genetycznego

Następnie, w zasadniczej części algorytmu (kroki 3-5), populację poddaje się procesowi ewolucji, na którą składają się cyklicznie wykonywane trzy operacje: selekcja, krzyżowanie i mutacja. W k-tej iteracji algorytmu, w kroku 3, z populacji P_k selekcjonuje się określoną liczbę osobników zgodnie z przyjętym schematem selekcji. Selekcja odbywa się w oparciu o funkcję przystosowania, której wartość dla poszczególnych osobników jest zależna od wartości funkcji celu reprezentowanych rozwiązań. W literaturze można zasadniczo spotkać dwa schematy selekcji. W myśl pierwszego z nich, selekcji ruletkowej, każdego osobnika w populacji wybiera się z prawdopodobieństwem proporcjonalnym do jego przystosowania. Drugi schemat, selekcja twarda, polega na wybraniu z populacji $q < p$ najlepiej przystosowanych osobników, gdzie q jest z reguły parametrem. W kroku 4 osobniki z wyselekcjonowanego zbioru P'_k poddaje się operacjom krzyżowania i mutacji w celu utworzenia potomstwa (nowej populacji P_{k+1}). Operacje krzyżowania dwóch lub więcej osobników przeprowadzane są w ten sposób, by powstałe osobniki potomne posiadały cechy pochodzące od każdego z rodziców. Mutacja polega na losowym „zaburzeniu" osobnika, wykonywanym z niewielkim prawdopodobieństwem i mającym na celu zapobieganie stagnacji procesu poszukiwań.

Zaletą wskazującą na uniwersalność algorytmów genetycznych jest minimum wiedzy o problemie, jakie jest wymagane w trakcie obliczeń. Projektant algorytmu jest zwolniony z obowiązku definiowania zbioru ruchów, sąsiedztwa, czy innych elementów zależnych od specyfiki rozwiązywanego problemu. Dlatego algorytmy genetyczne z powodzeniem znajdują zastosowanie w rozwiązywaniu problemów, dla których nie są znane specyficzne własności pozwalające na implementację innych wyrafinowanych technik rozwiązywania.

Algorytm rozpoczyna działanie z ustalonymi wartościami parametrów $maxP$, $maxR \leqslant maxP$, $maxS$.

1. Połóż $P := \emptyset$. Użyj *metody zróżnicowanego generowania* do skonstruowania rozwiązania x i zastosuj do niego *metodę ulepszania rozwiązań*. Jeżeli $x \notin P$, to połóż $P := P \cup \{x\}$. Powtarzaj krok 1 dopóki $|P| < maxP$.
2. Użyj *metody uaktualniania zbioru referencyjnego* w celu zbudowania zbioru rozwiązań referencyjnych $R = \{x^1, \ldots, x^{maxR}\}$ zawierającego $maxR$ najlepszych rozwiązań ze zbioru P. Połóż $z := TRUE$.
3. Dopóki $z = TRUE$ wykonuj kroki 4-5.
4. Wykorzystując zbiór R utwórz zbiór podzbiorów rozwiązań referencyjnych $S = \{s^1, \ldots, s^{maxS}\}$ używając *metody generowania podzbiorów*. Połóż $z := FALSE$.
5. Dopóki zbiór $S \neq \emptyset$ wykonuj kroki 6-7.
6. Wybierz podzbiór rozwiązań $s \in S$. Zastosuj *metodę tworzenia kombinacji rozwiązań* do podzbioru s w celu wygenerowania jednego, lub więcej rozwiązań próbnych x i zastosuj do nich *metodę ulepszania rozwiązań*.
7. Użyj *metody uaktualniania zbioru referencyjnego* w celu uaktualnienia zbioru R. Jeżeli zbiór R został uaktualniony, połóż $z := TRUE$. Połóż $S := S \setminus \{s\}$.
8. Zwróć najlepsze znalezione rozwiązanie x^* i STOP.

Rysunek 2.4: Schemat algorytmu poszukiwania rozproszonego

2.2.4 Algorytmy poszukiwania rozproszonego

Poszukiwanie rozproszone, zapoczątkowane w 1977 przez Glovera [27], jest bardziej metodologią prowadzenia lokalnych poszukiwań, niż konkretną techniką. Metodologia ta charakteryzuje się dużą elastycznością; każdy z jej elementów jest nieobowiązkowy i może być zrealizowany na wiele sposobów z różnym stopniem skomplikowania. Algorytmy poszukiwania rozproszonego systematycznie ukierunkowują swoje poszukiwania w odniesieniu do zbioru punktów referencyjnych, stanowiących z reguły rozwiązania dobrej jakości, uzyskane w trakcie wcześniejszych poszukiwań. Kryterium „rozwiązania dobrej jakości" nie musi odnosić się tu do wartości funkcji celu i ma zastosowanie bardziej do podzbiorów, niż pojedynczych rozwiązań.

Schemat algorytmu poszukiwania rozproszonego przedstawiono na rys. 2.4. W powyższym algorytmie został użyty szereg metod:

- *metoda zróżnicowanego generowania* generuje zbiór różnorodnych rozwiązań próbnych przy użyciu arbitralnie wybranego rozwiązania początkowego,
- *metoda ulepszania rozwiązań* przekształca rozwiązanie próbne w rozwiązanie wyższej jakości,
- *metoda uaktualniania zbioru referencyjnego* konstruuje i podtrzymuje zbiór referencyjny, przechowujący najlepsze znalezione rozwiązania. Rozwiązania dodawane są do zbioru dzięki ich jakości albo różnorodności. Zbiór zorganizowany jest w ten sposób, by mógł być efektywnie wykorzystywany przez inne metody wchodzące w skład algorytmu,
- *metoda generowania podzbiorów* tworzy podzbiory zbioru rozwiązań referencyjnych, wykorzystywane do utworzenia rozwiązań będących kombinacją rozwiązań referencyjnych,
- *metoda tworzenia kombinacji rozwiązań* przekształca dany podzbiór rozwiązań referencyjnych w kombinacje rozwiązań referencyjnych.

Jak już wspomniano, każda z powyższych metod może być zrealizowana na wiele sposobów i jest nieobowiązkowa. Przy ich projektowaniu zastosowanie mogą znaleźć zarówno techniki krzyżowania i mutacji wykorzystywane w algorytmach genetycznych, idea „ścieżki łączącej" wykorzystywana w kontekście wielkiej doliny, często obecnej w przestrzeni rozwiązań wielu problemów kombinatorycznych, czy też metody konstruowania zbioru ruchów i sąsiedztwa wykorzystywane w np. algorytmach poszukiwania z zabronieniami. Co więcej, każdą z powyższych metod może stanowić osobny algorytm zdolny do prowadzenia samodzielnych, niezależnych obliczeń. Zaletą algorytmów poszukiwania rozproszonego jest niewątpliwie wysoka jakość dostarczanych rozwiązań. Do ich wad można zaliczyć duże nakłady obliczeniowe oraz problemy implementacyjne związane z dużą liczbą elementów składowych.

2.3 Metody badania jakości algorytmów heurystycznych

W literaturze można spotkać się z wieloma pomysłami dotyczącymi badania jakości algorytmów przybliżonych. Wśród metod analitycznych można wymienić analizę najgorszego przypadku i analizę probabilistyczną.

Tą drogą uzyskano jednak stosunkowo niewiele rezultatów. Studia analityczne są przez badaczy podejmowanie niechętnie zarówno ze względu na złożoność wywodu matematycznego jak i duże prawdopodobieństwo poniesienia porażki. Przykładowo, algorytmy, dla których znany jest współczynnik najgorszego przypadku (dotyczy to głównie algorytmów konstrukcyjnych), bądź przynajmniej jego górne i dolne oszacowanie, należą do rzadkości. Zatem, na temat jakości algorytmów wnioskuje się głównie na podstawie analizy eksperymentalnej. Wszystkie warianty metody eksperymentalnej zawsze w pewnym stopniu odnoszą się do wartości funkcji celu najlepszych rozwiązań wyznaczonych przez algorytmy w kontekście czasu ich pracy. Badania prowadzi się przy użyciu zestawów instancji testowych (dostępnych publicznie, bądź generowanych losowo), stanowiących jedynie niewielką próbkę reprezentatywną zbioru wszystkich instancji, przez co porównanie algorytmów tą drogą nigdy nie jest do końca precyzyjne.

Z reguły, wartość funkcji celu najlepszego rozwiązania wyznaczonego przez porównywany algorytm jest odnoszona do wartości referencyjnej poprzez wyznaczenie błędu względnego bądź względnej poprawy. Błąd względny (nazywany też względną procentową odległością) pomiędzy wartością F^A wyznaczoną przez pewien algorytm A i wartością referencyjną F^B wyznacza się z równania

$$\Delta_A^B = 100\% \cdot \frac{F^A - F^B}{F^B}. \tag{2.14}$$

Względną poprawę wyznacza się z równania

$$\Phi_A^B = 100\% \cdot \frac{F^B - F^A}{F^B}. \tag{2.15}$$

Wartość referencyjną F^B może tu stanowić dolne ograniczenie wartości funkcji celu danej instancji problemu (ang. *lower bound*), optymalna wartość funkcji celu (jeśli jest znana) bądź najlepsze rozwiązanie wyznaczone przez inne algorytmy, które stanowią bazę do porównań.

Jak już wspomniano, względny błąd, tudzież względną poprawę często wyznacza się w kontekście czasu pracy porównywanych ze sobą algorytmów. Porównanie całkowitego czasu pracy algorytmów bądź czasu potrzebnego na odnalezienie rozwiązania najlepszego jest kłopotliwe, choćby ze względu na różnice w oprogramowaniu i platformach sprzętowych używanych przez poszczególnych badaczy. Dotychczas nie znaleziono skutecznego rozwiązania tego problemu. Jednym z pomysłów była próba ustandaryzowania czasu obliczeń, w myśl której czasy pracy algorytmów miały być przeliczane według uniwersalnego wzorca. Uzyskane w ten sposób wyniki miały szansę

być niezależne od platformy sprzętowej. Można jednak powiedzieć, że do tej pory pomysł ten nie znalazł szerokiego odzewu w literaturze. Poza tym, ze względu na niedeterministyczny charakter systemu operacyjnego i drobne różnice w częstotliwości taktowania procesora, nawet pomiary czasu wykonywane na tym samym komputerze nie są w pełni wiarygodne. Znacznie bardziej popularnym pomysłem wydaje się być porównywanie algorytmów poprzez zliczanie liczby wyznaczeń wartości funkcji celu czy liczby wykonywanych iteracji. Ten pomysł również nie gwarantuje precyzyjnej oceny, choćby ze względu na różnice w złożonościach obliczeniowych algorytmów i czasie trwania poszczególnych iteracji nawet w obrębie działania jednego algorytmu. Poza tym, w wielu algorytmach nie wyznacza się wartości funkcji kryterialnej explicite, lecz stosuje się stosunkowo tanie obliczeniowo oszacowanie owej wartości.

Zdaniem autora tej rozprawy, jedną z najlepszych metod porównania jakości algorytmów jest analiza ich zbieżności do rozwiązania optymalnego. W praktyce analizy zbieżności dokonuje się poprzez odnotowanie zmian najlepszej (lub średniej) wartości funkcji celu w funkcji czasu pracy (bądź liczby iteracji) algorytmu. Do wad metody należą kłopoty związane z prezentacją jej wyników. Ze względu na czasochłonność, badanie takie może zostać powtórzone dla stosunkowo niewielkiej liczby instancji danego problemu. Wspomniana metoda jest jednak często niezastąpiona na etapie projektowania czy strojenia danego algorytmu.

2.4 Problematyka transportu w elastycznych systemach produkcyjnych

W elastycznych systemach produkcyjnych zastosowanie może znaleźć wiele typów środków transportu (roboty manipulacyjne, dźwigi, taśmociągi, wózki widłowe, suwnice bramowe, układnice regałowe, etc). Jednakże, w ostatnich latach, szczególnie w elastycznych systemach produkcyjnych, rosnącą popularnością cieszą się wózki AGV. Elastyczne systemy produkcyjne wykorzystujące wózki AGV charakteryzują się zwiększoną elastycznością, lepszym wykorzystaniem przestrzeni roboczej, zwiększonym bezpieczeństwem i niższym kosztem eksploatacji w stosunku do systemów wykorzystujących inne formy transportu międzystanowiskowego. Właśnie dlatego poniższe rozważania (jak również rozważania w dalszej części tej rozprawy) na temat transportów w elastycznych systemach produkcyjnych będą głównie prowadzone w kontekście wózków AGV. Należy jednak zaznaczyć, że dyskusja ta jest często prawdziwa dla innych środków transportu.

Wózki AGV po raz pierwszy na wielką skalę zostały użyte w roku 1974 w fabryce Volvo we Szwecji. W czasie niewiele dłuższym, niż 10 lat na świecie istniało już około 3300 fabryk wykorzystujących ponad 15000 wózków. Największy system produkcyjny, wykorzystujący ponad 1000 wózków AGV, należy do koncernu General Motors i znajduje się w Kanadzie. Jednakże, zastosowanie wózków AGV nie ogranicza się do systemów produkcyjnych i montażowych. Na przykład, w centrum dystrybucji koncernu Kodak w Nowym Jorku wózki AGV wykorzystuje się do transportu filmów, papieru fotograficznego i chemikaliów. Również *The New York Times* wykorzystuje wózki do transportu gazet z drukarni do magazynu. Wózki AGV powszechnie są stosowane na lotniskach do transportu bagaży. Wydaje się jednak, że pod względem wykorzystania tychże maszyn króluje Japonia. Tylko w roku 1989 przedsiębiorstwa japońskie zakupiły ponad 5000 wózków AGV, podczas gdy w Europie zakupiono ich 3000 a w Stanach Zjednoczonych zaledwie 500 [26].

Pokaźną część wszystkich elastycznych systemów wytwarzania stanowią systemy o strukturze przepływowej i gniazdowej. Systemy tego typu, po uwzględnieniu ograniczeń związanych z transportem, można ogólnie opisać następująco. W systemie takim znajduje się określona liczba maszyn (jedno- bądź wielomaszynowych gniazd) produkcyjnych i określona liczba wózków AGV. W systemie należy wykonać ustaloną liczbę zadań. Każde zadanie podzielone jest na operacje produkcyjne i transportowe, które należy wykonać w ściśle określonej kolejności (porządku technologicznym) na maszynach ze ściśle określonych podzbiorów maszyn. Wykonanie operacji produkcyjnej zadania składa się z trzech etapów:

1. z bufora wejściowego maszyny produkcyjnej pobierana jest paleta z detalem (wykonywanym w ramach zadania),
2. detal jest poddawany obróbce na maszynie przez ustalony okres czasu,
3. paleta wraz z detalem trafia do bufora wyjściowego maszyny.

Zakłada się, że pojemność buforów wejściowych i wyjściowych wszystkich maszyn produkcyjnych jest wystarczająco duża, więc nie będzie brana pod uwagę w dalszych rozważaniach. Pomiędzy każdą parą operacji produkcyjnych każdego zadania wykonywana jest operacja transportowa przez jeden z wózków AGV. Operacja transportowa polega na:

1. pobraniu palety z detalem z bufora wyjściowego maszyny,
2. transporcie palety w określonym czasie w kierunku kolejnej maszyny produkcyjnej, zgodnej z przyjętym porządkiem technologicznym zadania,
3. umieszczeniu palety w buforze wejściowym maszyny.

Czasy transportu zależą od odległości dzielącej poszczególne maszyny produkcyjne, transportowanej palety oraz specyfikacji wózka wybranego do wykonania operacji transportowej.

Wśród systemów wykorzystujących wózki AGV można dokonać szeregu podziałów, zarówno pod względem stosowanych rozwiązań technologicznych, sposobów modelowania jak i podejść do optymalizacji harmonogramu produkcji. Pierwszych podziałów można dokonać ze względu na podejście do optymalizacji pracy systemu. O szeregowaniu zadań w czasie rzeczywistym (ang *real time scheduling*) jest mowa w sytuacji, gdy w momencie rozpoczęcia szeregowania nie jest dostępna pełna wiedza na temat zadań, które będą uszeregowane. Problem transportów przy takim podejściu rozwiązuje się stosując np. szereg priorytetowych reguł „on-line" w sposób równoległy kontrolujących pracę maszyn i wózków AGV (patrz. np. [5,6,87]). Podejście to, ze względu na charakter tej rozprawy, nie będzie dalej omawiane. Drugie podejście nazywane jest szeregowaniem deterministycznym bądź szeregowaniem „off-line". W tym przypadku zakłada się, że wiedza zarówno na temat systemu jak i zadań do wykonania jest całkowicie deterministyczna i znana a priori.

Kolejnego podziału można dokonać ze względu na sposób modelowania systemu i liczbę uwzględnianych w modelu środków transportu. W literaturze rozpatruje się dwa przypadki. W przypadku pierwszym zakłada się nieograniczoną liczbę środków transportu oraz, w konsekwencji, że transport zadania rozpoczyna się natychmiast po zakończeniu operacji produkcyjnej. Założenie takie ma szansę być zrealizowane w sytuacji praktycznej, gdy po pierwsze, liczba wózków w systemie jest równa lub większa od liczby zadań, po drugie, wszystkie operacje transportowe danego zadania wykonywane są przez ten sam wózek. Podejście takie jest jedną z najstarszych prób uwzględnienia ograniczeń związanych z transportem w problematyce szeregowania zadań. W modelach matematycznych tego typu transporty modeluje się poprzez wprowadzenie tzw. opóźnień (ang. *time lags*) pomiędzy poszczególnymi operacjami produkcyjnymi każdego zadania. Problematyka szeregowania zadań z opóźnieniami jest stosunkowo dobrze opisana w literaturze (patrz. np. [22, 61, 83]) i, również ze względu na stosunkowo mało precyzyjny sposób modelowania rzeczywistości, nie będzie dalej omawiana.

W przypadku drugim w modelach matematycznych systemów produkcyjnych uwzględnia się ograniczoną liczbę środków transportu. Podejście to znacznie lepiej odzwierciedla rzeczywistą pracę systemu, jednakże problematyka szeregowania zadań jest wtedy znacznie bardziej skomplikowana. W większości opisów literaturowych tego przypadku do szeregowania zadań na maszynach produkcyjnych i konstrukcji trasy przejazdu wózków pod-

chodzi się dwuetapowo. Etap pierwszy polega na uszeregowaniu zadań na maszynach produkcyjnych z pominięciem (lub częściowym uwzględnieniem) ograniczeń związanych z transportem. W drugim etapie rozwiązuje się problem doboru trasy przejazdu wózków (ang. *vehicle routing problem*) w odniesieniu do uszeregowania skonstruowanego w etapie pierwszym (patrz. np. [33]). W zaledwie kilku pracach (np. [7, 45, 50, 109]) porusza się temat jednoczesnego szeregowania zadań na maszynach i doboru trasy przejazdu wózków pomimo oczywistej wyższości takiego podejścia nad szeregowaniem dwuetapowym. W pracy [36] dowodzi się, że problem gniazdowy z ograniczoną liczbą wózków AGV może być rozpatrywany jako szczególny przypadek problemu gniazdowego z przezbrojeniami. Spostrzeżenie to miało istotny wpływ na całokształt rozważań prowadzonych w tej rozprawie.

Klasyfikacji systemów wykorzystujących wózki AGV można dokonać również ze względu na tryb pracy wózków. W literaturze rozpatruje dwa tryby pracy wózków:

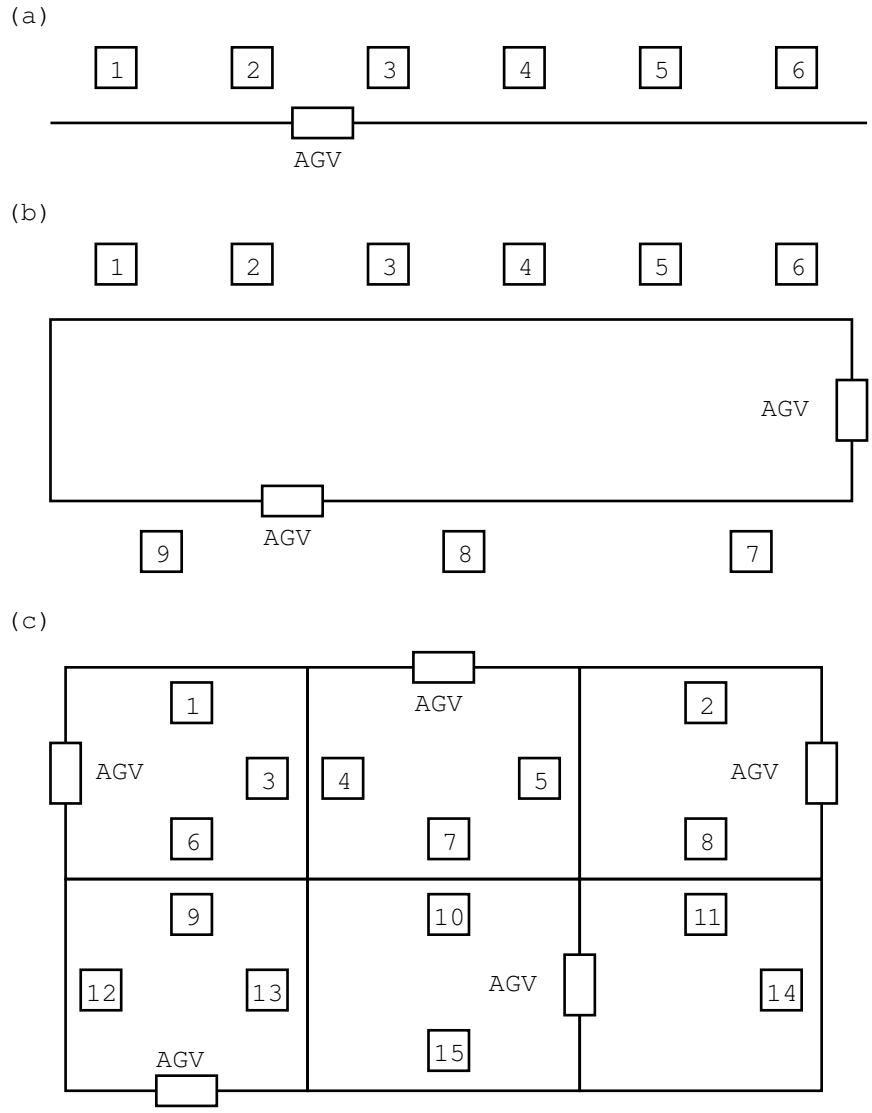
1. tryb *czekaj i jedź* (ang. *stop and go*),
2. tryb *zabierz i zostaw* (ang. *pick and drop*).

Wózek pracujący w trybie pierwszym związany jest z konkretnym zadaniem przez cały okres jego pobytu w systemie. Wykonując operację transportową wózek *jedzie* w kierunku odpowiedniej maszyny, po czym, przed rozpoczęciem kolejnej operacji transportowej, wózek stoi i *czeka* przy maszynie na zakończenie się operacji produkcyjnej. Zdecydowanie bardziej praktycznym jest drugi tryb pracy. Wózek pracujący w tym trybie, po *zostawieniu* transportowanej palety przy danej maszynie, może wykonać tzw. przejazd pusty (przejazd bez załadunku) w kierunku innej maszyny i tam *zabrać* inną paletę, rozpoczynając w ten sposób kolejną operację transportową innego zadania. Warto też zauważyć, że tryb pracy *czekaj i jedź* jest szczególnym przypadkiem trybu *zabierz i zostaw*.

Systemy wykorzystujące wózki AGV różnicuje się też ze względu na układ maszyn produkcyjnych. Wyróżnia się trzy podstawowe rozmieszczenia maszyn:

1. linia (ang. *single line*),
2. pętla (ang. *single loop*),
3. siatka (ang. *complex network*).

Układy te zostały przedstawione na rys. 2.5. System cechowany liniowym układem maszyn charakteryzuje się prostotą, jednakże nie wykorzystuje w pełni potencjału niesionego przez wózki. W przypadku układów typu siatka podstawowym problemem wydaje się powstawanie zatorów (lub innych konfliktów transportowych). Jednym ze sposobów zapobiegania zatorom jest dekompozycja takiego układu w zbiór niezależnych systemów



Rysunek 2.5: Najbardziej typowe układy maszyn: (a) linia, (b) pętla, (c) siatka

typu pętla. Rozwiązanie tego typu rozważane jest np. w pracach [12, 13]. Z reguły, w każdym systemie typu pętla znajduje się stacja załadowcza, stacja wyładowcza, jeden lub więcej wózków AGV i określona liczba maszyn. Aby uniknąć konfliktów transportowych, na system narzuca się jednokie-

runkowy przepływ zadań, zaś na wózkach wymusza się cykliczny, jednokierunkowy tryb pracy. W poszczególnych cyklach pracy każdego wózka do systemu wprowadzane jest jedno zadanie przez stację załadowczą, jedno zadanie opuszcza system przez stację wyładowczą, zaś wszystkie pozostałe zadania są transportowane w kierunku kolejnych maszyn w pętli (systemy takie maja strukturę permutacyjną przepływową). Systemy typu pętla łączą w sobie prostotę układów typu linia i zalety układów typu siatka, co czyni je szczególnie interesującymi dla badaczy.

W elastycznych systemach produkcyjnych, w ramach poszczególnych zadań, wytwarzane są z reguły detale różniące się między sobą i wymagające wykonania różnych operacji na poszczególnych maszynach. Detale te mogą również wymagać różnych metod transportu, co w konsekwencji może wiązać się z koniecznością zróżnicowania środków transportu w systemie. Systemy produkcyjne można zatem podzielić na systemy z:

1. identycznymi środkami transportu,
2. nieidentycznymi środkami transportu.

W systemach należących do grupy pierwszej każde zadanie produkcyjne może być transportowane przez dowolny z wózków. W przypadku drugim każde z zadań może być transportowane albo przez wózek z podzbioru wózków albo przez jeden, konkretny wózek dedykowany. Optymalizacja harmonogramu pracy w systemach wykorzystujących wózki identyczne jest znacznie trudniejsza niż w przypadku wózków dedykowanych; poza kolejnością wykonania operacji transportowych należy określić również przydział środków transportu do poszczególnych operacji.

W szczególnym przypadku, w systemie może znajdować się tylko jedna maszyna transportowa, wykonująca wszystkie operacje transportowe. Wtedy funkcję maszyny transportowej najczęściej pełni robot stacjonarny, zaś maszyny produkcyjne ustawione są w półkole. Systemy produkcyjne tego typu nazywa się komórkami robotycznymi (ang. *robotic cells*), zaś zagadnienia optymalizacji produkcji w takich systemach wiążą się z koniecznością rozwiązania problemu szeregowania w komórkach robotycznych (ang. *robotic cell scheduling problem*, patrz. np. [38, 91]) bądź jego szczególnego przypadku, nazywanego robotycznym problemem przepływowym (ang. *robotic flow-shop*). Z matematycznego punktu widzenia problematyka ta jest zbliżona do problematyki harmonogramowania pracy w elastycznym systemie produkcyjnym typu linia z jednym wózkiem AGV.

Podsumowując, problematyka szeregowania zadań z uwzględnieniem transportu jest bardzo szeroką, jednocześnie wciąż słabo zbadaną dziedziną wiedzy. Wydajność pracy elastycznych systemów produkcyjnych wciąż ograniczona jest przez niedoskonałość modeli matematycznych, zaś część

rozwiązań teoretycznych nie jest bezpośrednio możliwa do zrealizowania w praktyce. Należy również dodać, że problem optymalizacji pracy rzeczywistego systemu produkcyjnego oczywiście nie ogranicza się jedynie do problematyki harmonogramowania zadań produkcyjnych poprzez projektowanie coraz lepszych algorytmów czy doskonalszych modeli matematycznych, lepiej odzwierciedlających rzeczywistość. Problem ten nierozerwalnie związany jest z całym szeregiem innych problemów, wśród których tylko nieliczną część stanowią problemy takie jak problem podziału powierzchni (ang. *facility layout problem*), problem doboru liczby wózków (ang. *fleet sizing problem*) czy problem projektowania ścieżki przepływu (ang. *flow-path design problem*). Opis osiągnięć w tym zakresie można znaleźć np. w pracach [25, 94, 107].

Rozdział 3

Własności problemu gniazdowego z ustalonym przydziałem wózków AGV

Dzięki wysokiej elastyczności, lepszym wykorzystaniem przestrzeni roboczej i stosunkowo niskiemu kosztowi pracy, elastyczne systemy produkcyjne wykorzystujące wózki AGV cieszą się coraz większą popularnością na całym świecie. W wielu systemach planowanie produkcji odbywa się dwuetapowo. W pierwszym etapie konstruuje się uszeregowanie operacji na maszynach z całkowitym lub częściowym pominięciem ograniczeń związanych z transportem. W przypadku systemów o strukturze gniazdowej można tego dokonać poprzez rozwiązanie klasycznego problemu gniazdowego przy użyciu dowolnego z wielu algorytmów literaturowych. W fazie drugiej planowanie trasy przejazdu wózków odbywa się np. w oparciu o zbiór reguł priorytetowych w kontekście ustalonego uszeregowania operacji na maszynach. Szansę na zwiększenie wydajności takich systemów można upatrywać w zintegrowaniu fazy szeregowania zadań produkcyjnych i fazy planowania trasy przejazdu wózków poprzez rozwiązanie problemu gniazdowego z transportem.

W tym rozdziale rozpatruje się praktyczne uogólnienie klasycznego problemu gniazdowego polegające na uwzględnieniu czasów transportu zadań pomiędzy poszczególnymi maszynami produkcyjnymi; przydział wszystkich transportów do poszczególnych wózków AGV dany jest a priori, rozważa się maszynowo-zależne i zadaniowo-zależne czasy transportu oraz maszynowo-zależne czasy przejazdów bez załadunku. Zakłada się również, że wózki pracują w trybie „zabierz i zostaw". Za kryterium optymalizacji przyjmuje się moment zakończenia wykonywania wszystkich zadań. Po przedstawie-

niu problemu i jego modeli matematycznych prezentowany jest zarówno szereg własności analitycznych jak i rezultatów empirycznych, przydatnych na etapie projektowania algorytmów rozwiązywania problemu.

Omawiany problem, stosując rozszerzoną notację Grahama z prac [43, 50] (naszkicowaną również w rozdziale 2), można zaklasyfikować jako problem $J, DR|t_{jkl}, t'_{kl}|C_{\max}$, gdzie zakłada się, że każde z zadań może być transportowane przez jeden, konkretny środek transportu. Jednakże, problem $J, DR|t_{jkl}, t'_{kl}|C_{\max}$ jest szczególnym przypadkiem problemu omawianego w tym rozdziale i nie obejmuje wszystkich sytuacji, w których przydział operacji transportowych do poszczególnych środków transportu dany jest a priori. Po pierwsze, taki przydział można rozpatrywać w sytuacji, w której podzbiór maszyn produkcyjnych obsługiwany jest przez jedną maszynę transportową. W konsekwencji, nawet gdy zbiór środków transportu w systemie jest identyczny, transporty danego zadania pomiędzy konkretną parą maszyn mogą być wykonane przez tylko jedną maszynę transportową, inną dla różnych par maszyn. Po drugie, omawiany przydział może być wynikiem działania fazy wstępnej algorytmu rozwiązywania problemów $J, R|t_{jkl}, t'_{kl}|C_{\max}$, $J, MPR|t_{jkl}, t'_{kl}|C_{\max}$, tzn. możliwe jest rozwiązanie wspomnianych problemów poprzez sprowadzenie ich do problemu z ustalonym przydziałem środków transportu. W takiej sytuacji poszczególne operacje transportowe zadania są wykonywane przez dedykowane, lecz zazwyczaj różne wózki AGV.

3.1 Model matematyczny

Problem gniazdowy z ustalonym przydziałem wózków AGV można sformułować następująco. Dany jest elastyczny system produkcyjny ze zbiorem m^p maszyn produkcyjnych $M^p = \{1, \ldots, m^p\}$ i zbiorem m^t maszyn transportowych (wózków AGV) $M^t = \{m^p + 1, m^p + 2, \ldots, m^p + m^t\}$; $M = M^p \cup M^t$, $m = |M| = m^p + m^t$. W systemie należy wykonać r zadań ze zbioru $J = \{1, \ldots, r\}$. Każde z zadań odpowiada detalowi, który ma być poddany obróbce na maszynach produkcyjnych zgodnie z jego indywidualnym porządkiem technologicznym. Zatem, każde zadanie $k \in J$ składa się z sekwencji n_k^p operacji produkcyjnych, indeksowanych przez $j_k + 1, j_k + 2, \ldots, j_k + n_k^p$, które powinny być wykonane w tej kolejności; wielkość $j_k = \sum_{i=1}^{k-1} n_i^p$, $j_1 = 0$, jest całkowitą liczbą operacji pierwszych $k - 1$ zadań. Operację $x = j_k + i$, $k \in J$, $1 \leqslant i \leqslant n_k^p$, należy wykonywać na maszynie $m_x \in M^p$ w czasie $p_x > 0$. Operacja produkcyjna x składa się z trzech kroków:

1. z bufora wejściowego maszyny m_x pobierana jest paleta z detalem wykonywanym w ramach zadania k,
2. detal jest poddany obróbce na maszynie,
3. paleta z detalem umieszczana jest buforze wyjściowym maszyny.

Nie uwzględnia się ograniczeń związanych z pojemnością buforów maszyn. Zbiór wszystkich operacji produkcyjnych będzie oznaczany przez $O^p = \{1, 2, \ldots, n^p\}$, gdzie $n^p = \sum_{k \in J} n_k^p$. Zakłada się, że każde dwie kolejne operacje produkcyjne tego samego zadania będą wykonywane na różnych maszynach produkcyjnych.

Transport zadania $k \in J$ (tzn. transport palety z detalem wykonywanym w ramach zadania k realizowany przez wózek AGV) i przejazd pusty maszyny transportowej $h \in M^t$ (tzn. przejazd bez załadunku) pomiędzy maszynami $l', l'' \in M^p$, $l' \neq l''$, wymaga odpowiednio $t_k(l', l'') > 0$ i $e(l', l'') > 0$ czasu; $t_k(l', l') = e(l', l') = 0$. Zakłada się, że czasy transportu i czasy przejazdów pustych spełniają tzw. silny warunek trójkąta oraz czasy transportu są nie krótsze niż czasy przejazdów pustych. Powyższe założenia opisują nierówności

$$t_k(l', l'') \leqslant t_k(l', l) + t_k(l, l''), \tag{3.1}$$

$$e(l', l'') \leqslant e(l', l) + e(l, l''), \tag{3.2}$$

$$t_k(l', l'') \geqslant e(l', l''), \tag{3.3}$$

dla każdej maszyny $l, l', l'' \in M^p$ i zadania $k \in J$. Powyższe założenia są zawsze prawdziwe w większości rzeczywistych sytuacji produkcyjnych; jeżeli czas transportu $t_k(l', l'')$ nie spełnia nierówności (3.1), to zakłada się, że możliwy jest transport zadania k pomiędzy maszynami l', l'' „obok" maszyny l, na który potrzebne jest $t_k(l', l) + t_k(l, l'')$ czasu. Podobnie można uzasadnić nierówność (3.2), zaś nierówność (3.3) jest oczywista.

Pomiędzy każdą parą sąsiednich operacji produkcyjnych $j_k + i$ oraz $j_k + i + 1$, $1 \leqslant i < n_k^p$, zadania $k \in J$ należy wykonać operację transportową. Taka operacja będzie oznaczana przez $x = [j_k + i]$. Operacja x jest wykonywana przez maszynę transportową $m_x \in M^t$ w czasie $p_x = t_k(l', l'')$, gdzie $l' = m_{j_k+i}$ oraz $l'' = m_{j_k+i+1}$. Operacja ta polega na:

1. zabraniu palety z detalem wykonywanym w ramach zadania k z bufora wyjściowego maszyny l',
2. przetransportowaniu palety,
3. pozostawieniu palety w buforze wejściowym[1] maszyny l''.

[1]Czasy związane z zabraniem i pozostawieniem transportowanej palety w odpowiednich buforach można uwzględnić poprzez dodanie ich do czasu wykonania p_x operacji transportowej x.

Zakłada się, że przydział maszyn do operacji transportowych dany jest a priori. Zbiór wszystkich operacji transportowych będzie oznaczany przez $O^t = \bigcup_{k\in J} \bigcup_{i=1}^{n_k^p-1} \{[j_k + i]\}$. Całkowita liczba operacji transportowych wynosi $n^t = |O^t| = \sum_{k\in J} n_k^t = n^p - r$, gdzie $n_k^t = n_k^p - 1$ jest liczbą operacji transportowych w zadaniu $k \in J$. W celu wprowadzenia kompletnej notacji definiuje się liczbę wszystkich operacji $n_k = n_k^p + n_k^t$ w zadaniu $k \in J$, zbiór wszystkich operacji $O = O^p \cup O^t$ oraz liczbę wszystkich operacji $n = \sum_{k\in J} n_k$.

Dana jest operacja transportowa $x' = [j_g + i]$, $1 \leqslant i \leqslant n_g^t$, $g \in J$ (transport zadania g od maszyny m_{j_g+i} do maszyny $l' = m_{j_g+i+1}$), i operacja $x'' = [j_k + h]$, $1 \leqslant h \leqslant n_k^t$, $k \in J$ (transport zadania k od maszyny $l'' = m_{j_k+h}$ do maszyny m_{j_k+h+1}) wykonywane kolejno przez tą samą maszynę transportową $m_{x'} = m_{x''} \in M^t$. Po wykonaniu operacji x' i przed wykonaniem operacji x'' maszyna transportowa wykonuje przejazd pusty pomiędzy maszynami l', l'', który trwa $e(l', l'')$ czasu. Przejazd pusty może być utożsamiony z przezbrojeniem $s(x', x'') = e(l', l'')$ maszyny transportowej, wykonywanym pomiędzy operacjami x', x''. Z równań (3.1)-(3.3) wynika, że czasy przezbrojeń spełniają słaby warunek trójkąta, tj.

$$s(x', x'') \leqslant s(x', x) + p_x + s(x, x''), \qquad x, x', x'' \in O^t. \tag{3.4}$$

W celu uproszczenia dalszej notacji definiuje się również czasy przezbrojeń $s(x', x'')$ pomiędzy każdą parą operacji produkcyjnych $x', x'' \in O^p$, równe $s(x', x'') = 0$ oraz czasy $s(0, 0) = s(i, 0) = s(0, i) = 0$, $i \in O$.

Podobnie jak w przypadku klasycznego problemu gniazdowego (patrz. np. sekcja 2.1.2) zakłada się, że:

1. raz rozpoczęta operacja nie może być przerwana,
2. w każdej chwili czasu można wykonywać co najwyżej jedną operację każdego zadania,
3. w każdej chwili czasu każda maszyna (produkcyjna i transportowa) może wykonywać co najwyżej jedną operację.

Uszeregowanie definiuje się poprzez wektor $S = (S_1, S_2, \ldots, S_n)$ czasów rozpoczęcia wykonywania S_i poszczególnych operacji $i \in O$. Uszeregowanie jest dopuszczalne, gdy wszystkie powyższe ograniczenia są spełnione, czego wyrazem jest prawdziwość nierówności (2.7), (2.8) oraz dodatkowo nierówności

$$(C_i + s(i, j) \leqslant S_j) \,\dot\vee\, (C_j + s(j, i) \leqslant S_i), \qquad i, j \in O, i \neq j, m_i = m_j, \tag{3.5}$$

gdzie $C_i = S_i + p_i$ jest momentem zakończenia wykonywania operacji $i \in O$. Za kryterium optymalizacji przyjmuje się moment zakończenia wykonywa-

nia wszystkich zadań, równy $C_{\max} = \max_{i \in O} C_i$. Problem polega na odnalezieniu takiego uszeregowania dopuszczalnego, by kryterium optymalizacji przyjęło wartość minimalną. Problem jest silnie NP-trudny.

W tym miejscu należy zauważyć, że model matematyczny problemu w postaci podobnej do powyższej (podobnie jak prezentowany poniżej model permutacyjno-grafowy) był już prezentowany w książce [36]. Jedna z podstawowych różnic polega na tym, że w cytowanej książce rozpatrywano czasy transportu niezależne od przewożonych zadań, równe czasom przejazdu bez załadunku.

3.1.1 Model permutacyjno-grafowy

Poniżej zostanie przedstawiony model permutacyjno-grafowy problemu. Niech $O_l = \{i \in O : m_i = l\}$ będzie zbiorem wszystkich operacji wykonywanych na maszynie $l \in M$. Niech $\pi = (\pi_1, \ldots, \pi_m)$ będzie zestawem permutacji, gdzie $\pi_l = (\pi_l(1), \ldots, \pi_l(o_l))$, $o_l = |O_l|$, jest permutacją zbioru O_l, określającą kolejność wykonania operacji z tego zbioru na maszynie l. Niech Π_l oznacza zbiór wszystkich permutacji zbioru O_l. Wtedy zbiór wszystkich zestawów permutacji (przestrzeń rozwiązań problemu) określa się przez $\Pi^0 = \Pi_1 \times \Pi_2 \times \cdots \times \Pi_m$. Dla uproszczenia, w dalszej części rozdziału zestaw permutacji $\pi \in \Pi^0$ będzie nazywany permutacją bądź, po prostu, rozwiązaniem problemu. Permutacja π reprezentująca uszeregowanie dopuszczalne będzie nazywana permutacją dopuszczalną (rozwiązaniem dopuszczalnym), zaś zbiór wszystkich permutacji dopuszczalnych (przestrzeń rozwiązań dopuszczalnych) będzie oznaczany przez $\Pi \subset \Pi^0$.

Dla dowolnej permutacji $\pi \in \Pi^0$ można skonstruować skierowany graf $G(\pi) = (O, E(\pi))$ ze zbiorem obciążonych wierzchołków O i zbiorem łuków $E(\pi) = E^T \cup E^K(\pi)$, gdzie

$$E^T = \bigcup_{i=1}^{r} \bigcup_{j=1}^{n_i - 1} \{(j, [j]), ([j], j+1)\}, \tag{3.6}$$

zaś zbiór $E^K(\pi)$ dany jest równaniem (2.11). Każdy wierzchołek $i \in O$ grafu reprezentuje operację i przyjmuje obciążenie p_i. Zbiór E^T jest zbiorem nieobciążonych łuków technologicznych, określających relacje kolejnościowe pomiędzy operacjami w poszczególnych zadaniach. Zbiór $E^K(\pi)$ jest zbiorem łuków kolejnościowych, określających relacje kolejnościowe pomiędzy operacjami wykonywanymi na poszczególnych maszynach. Obciążenie każdego łuku $(i, j) \in E^K(\pi)$ wynosi $s(i, j)$.

Z własności grafu wynika, że graf $G(\pi)$ jest acykliczny wtedy i tylko wtedy, gdy $\pi \in \Pi$. Przez r_i^π oraz q_i^π będzie oznaczana długość najdłuższej ścieżki odpowiednio dochodzącej i wychodzącej z wierzchołka $i \in O$ (bez jego

obciążenia p_i) w acyklicznym grafie $G(\pi)$; $r_0^\pi = q_0^\pi = 0$. Nie trudno zauważyć, że w takim grafie wielkość r_i^π jest równa najwcześniejszemu możliwemu momentowi rozpoczęcia S_i operacji $i \in O$. Długość dowolnej ścieżki krytycznej, oznaczanej symbolem $C_{\max}(\pi)$, w grafie $G(\pi)$ jest zatem równa wartości przyjętej funkcji kryterialnej, tj. $C_{\max}(\pi) = \max_{i \in O} C_i = \max_{i \in O}(r_i^\pi + p_i)$. Wobec tego, problem sprowadza się do odnalezienia takiej permutacji π^*, że $C_{\max}(\pi^*) = \min_{\pi \in \Pi} C_{\max}(\pi)$.

W acyklicznym grafie $G(\pi)$, $\pi \in \Pi$, przez b_i^T (a_i^T) oraz $b_i^K(\pi)$ ($a_i^K(\pi)$) będzie oznaczony odpowiednio bezpośredni poprzednik (następnik) technologiczny oraz bezpośredni poprzednik (następnik) kolejnościowy wierzchołka $i \in O$. Bezpośredni poprzednik i następnik technologiczny wierzchołka i w grafie $G(\pi)$ oznacza wierzchołek reprezentujący odpowiednio operację bezpośrednio poprzedzającą i następującą po operacji i w zadaniu, do którego ta operacja należy. Wierzchołki te połączone są ze sobą łukami ze zbioru E^T. Analogicznie, bezpośredni poprzednik i następnik kolejnościowy wierzchołka i w grafie $G(\pi)$ oznacza wierzchołek reprezentujący odpowiednio operację bezpośrednio poprzedzającą i następującą po operacji i na maszynie m_i. Wierzchołki te połączone są ze sobą łukami ze zbioru $E^K(\pi)$. Jeśli dany wierzchołek nie posiada któregoś z poprzedników lub następników, to przyjmuje się odpowiednio $b_i^T = a_i^T = b_i^K(\pi) = a_i^K(\pi) = 0$. Symbolem $ZB_i(\pi)$ oraz $ZA_i(\pi)$ będzie oznaczany odpowiednio zbiór wszystkich poprzedników i następników wierzchołka i.

Warto zauważyć, że podobnie jak w klasycznym problemie gniazdowym, wielkości r_i^π i q_i^π, $i \in O$ (a tym samym długość ścieżki krytycznej i wartość funkcji celu), w acyklicznym grafie $G(\pi)$ mogą być wyznaczone rekurencyjnie przy użyciu poniższych równań

$$r_i^\pi = \max\{r_a^\pi + p_a, r_b^\pi + p_b + s(b,i)\}, \tag{3.7}$$

$$q_i^\pi = \max\{q_c^\pi + p_c, q_d^\pi + p_d + s(i,d)\}, \tag{3.8}$$

gdzie $a = b_i^T$, $b = b_i^K(\pi)$, $c = a_i^T$, $d = a_i^K(\pi)$. Przyjmuje się też, że $r_0^\pi = p_0 = q_0^\pi = 0$.

3.1.2 Przykład ilustracyjny

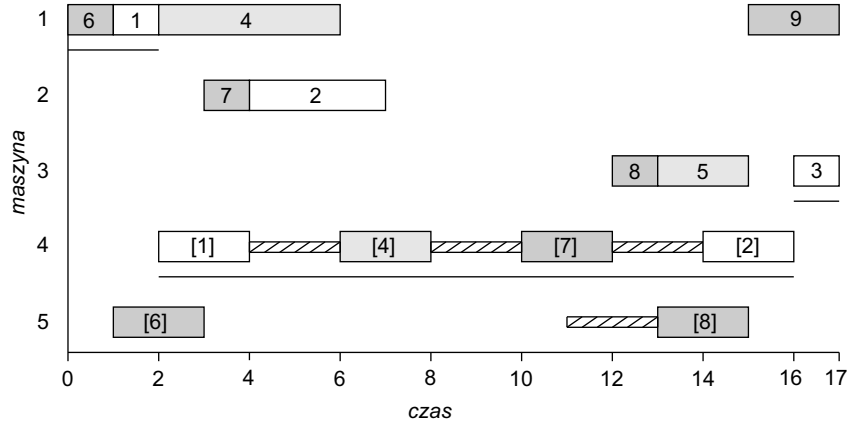
Poniżej znajduje się szczegółowa analiza przykładowej instancji rozpatrywanego problemu.

Przykład 3.1 Dany jest system produkcyjny ze zbiorem $m^p = 3$ maszyn produkcyjnych $M^p = \{1, 2, 3\}$ i zbiorem $m^t = 2$ maszyn transportowych $M^t = \{4, 5\}$. Zakłada się, że czas przejazdu bez załadunku każdego wózka $v \in M^t$ pomiędzy każdą parą maszyn produkcyjnych wy-

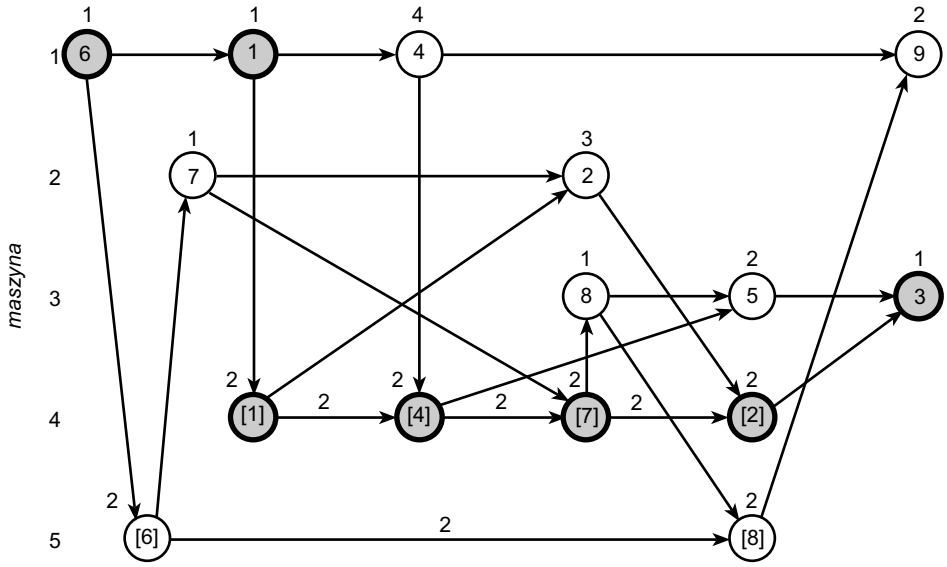
Tabela 3.1: Przydział maszyn, czasy wykonania oraz długości najdłuższych ścieżek dla poszczególnych operacji z przykładu 3.1

i	1	[1]	2	[2]	3	4	[4]	5
m_i	1	4	2	4	3	1	4	3
p_i	1	2	3	2	1	4	2	2
b_i^T	0	1	[1]	2	[2]	0	4	[4]
a_i^T	[1]	2	[2]	3	0	[4]	5	0
$b_i^K(\pi)$	6	0	7	[7]	5	1	[1]	8
$a_i^K(\pi)$	4	[4]	0	0	0	9	[7]	3
r_i^π	1	2	4	14	16	2	6	13
q_i^π	15	13	3	1	0	11	9	1

i	6	[6]	7	[7]	8	[8]	9
m_i	1	5	2	4	3	5	1
p_i	1	2	1	2	1	2	2
b_i^T	0	6	[6]	7	[7]	8	[8]
a_i^T	[6]	7	[7]	8	[8]	9	0
$b_i^K(\pi)$	0	0	0	[4]	0	[6]	4
$a_i^K(\pi)$	1	[8]	2	[2]	5	0	0
r_i^π	0	1	3	10	12	13	15
q_i^π	16	8	7	5	4	2	0



Rysunek 3.1: Wykres Gantta uszeregowania reprezentowanego przez permutację π z przykładu 3.1



Rysunek 3.2: Graf $G(\pi)$ dla permutacji π z przykładu 3.1

nosi $e(l', l'') = 2$, $l', l'' \in M^p$. W systemie należy wykonać $r = 3$ zadania ze zbioru $J = \{1, 2, 3\}$. Zadanie 1 składa się z sekwencji operacji $1, [1], 2, [2], 3$ na zadanie 2 składa się sekwencja operacji $4, [4], 5$, zaś zadanie trzecie stanowią operacje $6, [6], 7, [7], 8, [8], 9$ ($n_1 = n_1^p + n_1^t = 3 + 2 = 5$, $n_2 = n_2^p + n_2^t = 2 + 1 = 3$, $n_3 = n_3^p + n_3^t = 4 + 3 = 7$). Zbiór operacji produkcyjnych i transportowych jest zatem równy odpowiednio $O^p = \{1, 2, \ldots, 9\}$ i $O^t = \{[1], [2], [4], [6], [7], [8]\}$ ($n^p = n_1^p + n_2^p + n_3^p = 3 + 2 + 4 = 9$, $n^t = n_1^t + n_2^t + n_3^t = 2 + 1 + 3 = 6$, $n = n^p + n^t = 9 + 6 = 15$). Przydział maszyn do operacji oraz czasy ich wykonania dane są w tabeli 3.1. Czasy wykonania operacji produkcyjnych określone są niezależnie, zaś czasy wykonania operacji transportowych (jak również czasy przejazdów pustych) wynikają z rozmieszczenia dróg oraz odległości dzielących poszczególne maszyny produkcyjne. Czasy wykonania operacji transportowych zostały określone przy założeniu, że czasy transportów poszczególnych zadań pomiędzy maszynami wynoszą $t_k(l', l'') = e(l', l'') = 2$, $l', l'' \in M^p$, $k \in J$. Zatem, zgodnie z definicją, $p_{[1]} = t_1(1, 2) = 2$, $p_{[2]} = t_1(2, 3) = 2$, $p_{[4]} = t_2(1, 3)$, itd.

Dana jest permutacja dopuszczalna $\pi = (\pi_1, \ldots, \pi_5)$, gdzie $\pi_1 = (6, 1, 4, 9)$, $\pi_2 = (7, 2)$, $\pi_3 = (8, 5, 3)$, $\pi_4 = ([1], [4], [7], [2])$, $\pi_5 = ([6], [8])$ określająca kolejność wykonania operacji ze zbioru $O = O^p \cup O^t$ oraz acykliczny graf $G(\pi)$. Wykres Gantta uszeregowania reprezentowanego przez permutację π przedstawiony jest na rysunku 3.1. Operacje należące do jednej ze ścieżek krytycznych w grafie $G(\pi)$ zostały podkreślone. Graf $G(\pi)$

przedstawiony jest na rysunku 3.2. Obciążenia wierzchołków naniesione są nad wierzchołkami, łuki bez podanego obciążenia mają wagę zero. Wykorzystując graf przedstawiony na rysunku 3.2 nie trudno określić wielkości b_i^T, a_i^T, $b_i^K(\pi)$, $a_i^K(\pi)$, $i \in O$. Na podstawie tych wielkości oraz przy użyciu równań (3.7), (3.8) nie trudno też wyznaczyć długości najdłuższych ścieżek r_i^π, q_i^π. Wielkości te zostały podane w tabeli 3.1. Wartość funkcji celu rozwiązania π, równa długości ścieżki krytycznej w grafie $G(\pi)$, wynosi $C_{\max}(\pi) = r_3^\pi + p_3 = 17$.

3.2 Zbiory ruchów i sąsiedztwa

Jednym z podstawowych pojęć związanych z algorytmami lokalnych poszukiwań jest pojęcie ruchu. Ruch można opisać jako czynność polegającą na przejściu pomiędzy dwoma rozwiązaniami. W przypadku rozwiązań reprezentowanych przez permutacje, w literaturze, obok wielu innych, najczęściej wymienia się:

1. ruchy typu zamień (ang. *swap moves*),
2. ruchy typu zamień sąsiednie elementy (ang. *adjacent swap moves*),
3. ruchy typu wstaw (ang. *insert moves*).

Pierwszy typ ruchów polega na zamianie miejscami dwóch elementów w permutacji, drugi typ ruchów na zamianie miejscami dwóch sąsiadujących ze sobą elementów w permutacji, trzeci typ ruchów polega na usunięciu elementu z zajmowanej przez niego pozycji i wstawieniu go w inne miejsce w permutacji. Z punktu widzenia algorytmów rozwiązywania badanego problemu szczególnie interesujące wydają się ruchy typu zamień sąsiednie operacje, zarówno ze względu na możliwość redukcji rozmiaru generowanego sąsiedztwa jak i znajomość efektywnych metod wyznaczania wartości funkcji celu rozwiązań sąsiednich.

W celu precyzyjnego opisu zbiorów ruchów, sąsiedztw i ich własności zostanie użyte podejście blokowe, zapoczątkowane w pracy [35] i rozwijane w dalszych pracach, np. [36, 65, 70, 71]. Niech ciąg wierzchołków $u = (u_1, u_2, \ldots, u_{lu})$, $u_i \in O$, $1 \leqslant i \leqslant lu$, będzie arbitralnie wybraną ścieżką krytyczną w grafie $G(\pi)$, $\pi \in \Pi$, gdzie lu jest liczbą wierzchołków ścieżki. Ścieżkę u w jednoznaczny sposób można podzielić na lb rozłącznych ciągów $u = (B_1, B_2, \ldots, B_{lb})$, gdzie $B_h = (\pi_{l_h}(e_h), \pi_{l_h}(e_h + 1), \ldots, \pi_{l_h}(f_h))$, $1 \leqslant e_h \leqslant f_h \leqslant o_{l_h}$, $1 \leqslant h \leqslant lb$ jest h-tym blokiem operacji na ścieżce krytycznej u. Każdy blok B_h jest ciągiem

1. zawierającym operacje wykonywane na tej samej maszynie (oznaczonej przez $l_h \in M$) oraz

2. operacje w bloku B_h są wykonywane na maszynie innej, niż operacje w bloku B_{h-1}, tj. $l_{h-1} \neq l_h$, $h = 2, \ldots, lb$.

Pozycje e_h, f_h oraz pozycje $e_h + 1, \ldots, f_h - 1$ będą nazywane odpowiednio zewnętrznymi oraz wewnętrznymi pozycjami bloku B_h, $h = 1, \ldots, lb$. Blok B_h zbudowany z operacji produkcyjnych i transportowych, tzn. taki, że odpowiednio $l_h \in M^p$ i $l_h \in M^t$, będzie nazywany odpowiednio blokiem operacji produkcyjnych i blokiem operacji transportowych. Z własności grafu wynika, że pomiędzy każdą parą bloków operacji produkcyjnych znajduje się jeden i tylko jeden blok operacji transportowych; pierwszym i ostatnim blokiem na ścieżce krytycznej jest blok operacji produkcyjnych.

Niech $av_{l,z+1} = (\pi_l(z), \pi_l(z+1))$ będzie ruchem typu zamień sąsiednie operacje polegającym na zamianie miejscami dwóch sąsiednich operacji $\pi_l(z), \pi_l(z+1)$, $l \in M$, $1 \leqslant z < o_l$, w permutacji $\pi \in \Pi$. Najbardziej podstawowe, jednocześnie najstarsze zbiory ruchów typu zamień sąsiednie operacje zostały opisane (dla klasycznego problemu gniazdowego) w pracy [54] i będą dalej oznaczone przez $AV^0(\pi)$, $AV^1(\pi)$, $\pi \in \Pi$. Zbiór $AV^0(\pi)$ zawiera wszystkie możliwe ruchy typu zamień sąsiednie operacje. Zbiór ten zawiera $\sum_{l \in M}(o_l - 1)$ elementów. Zbiór $AV^1(\pi)$ zawiera wszystkie możliwe ruchy polegające na zamianie dwóch sąsiednich operacji na wszystkich ścieżkach krytycznych w grafie $G(\pi)$. Jego rozmiar zależy od liczby ścieżek krytycznych w grafie oraz liczby operacji na każdej ze ścieżek. Zakładając, że graf $G(\pi)$ posiada tylko jedną ścieżkę krytyczną, zbiór ten zawiera $\sum_{h=1}^{lb}(|B_h - 1|)$ elementów. Sąsiedztwem danej permutacji nazywany jest zbiór zawierający permutacje otrzymane po zastosowaniu wszystkich ruchów z danego zbioru ruchów do tejże permutacji. Bardziej precyzyjnie, sąsiedztwem permutacji π będzie nazywany zbiór permutacji $AN^i(\pi) = \{\pi^v : v \in AV^i(\pi)\}$, $i \in \{1, 2\}$, gdzie π^v oznacza permutację sąsiednią, otrzymaną w wyniku wykonania ruchu v. W cytowanej już pracy [54] dowodzi się, że w przypadku klasycznego problemu gniazdowego każde rozwiązanie β ze zbioru $AN^0(\pi) \backslash AN^1(\pi)$ jest albo niedopuszczalne albo spełnia nierówność $C_{\max}(\beta) \geqslant C_{\max}(\pi)$. Dla rozważanego problemu ten wniosek pozostaje prawdziwy. Można również łatwo udowodnić, że podobnie jak dla problemu klasycznego, każde rozwiązanie w zbiorze $AN^1(\pi)$ jest rozwiązaniem dopuszczalnym. Pomimo dość znacznej redukcji jego rozmiaru, sąsiedztwo $AN^1(\pi)$ wciąż zawiera relatywnie dużo elementów. W pracy [70] proponuje się dalsze redukcje rozmiaru sąsiedztwa poprzez wprowadzenie tak zwanego zredukowanego zbioru ruchów $AV^2(\pi)$ (i odpowiadającego mu zredukowanego sąsiedztwa $AN^2(\pi)$), w którym zamieniane są tylko dwie pierwsze i dwie ostatnie operacje należące do każdego z bloków operacji. Innymi słowy, zbiór $AV^2(\pi)$ otrzymuje się ze zbioru $AV^1(\pi)$ poprzez arbitralny wybór jednej ze ścieżek krytycz-

nych w grafie $G(\pi)$ i usunięcie ruchów polegających na zamianie operacji na wewnętrznych pozycjach poszczególnych bloków operacji. Bardziej precyzyjnie, $AV^2(\pi) = \bigcup_{h=1}^{lb} AV_h^2(\pi)$, gdzie

$$AV_1^2(\pi) = \begin{cases} \{av_{l_1,f_1}\}, & e_1 < f_1, lb > 1, \\ \emptyset & \text{w przeciwnym przypadku,} \end{cases} \tag{3.9}$$

$$AV_h^2(\pi) = \begin{cases} \{av_{l_h,e_h+1}, av_{l_h,f_h}\}, & e_h < f_h, \\ \emptyset & \text{w przeciwnym przypadku,} \end{cases} \tag{3.10}$$

$$h = 2, ..., lb-1,$$

$$AV_{lb}^2(\pi) = \begin{cases} \{av_{l_{lb},e_{lb+1}}\}, & e_{lb} < f_{lb}, lb > 1, \\ \emptyset & \text{w przeciwnym przypadku.} \end{cases} \tag{3.11}$$

Zbiór ten zawiera

$$|AV^2(\pi)| = \min\{1, |B_1|-1\} + \sum_{h=2}^{lb-1} \min\{2, |B_h|-1\} + \min\{1, |B_{lb}|-1\} \tag{3.12}$$

ruchów, gdzie $|B_h| = f_h - e_h + 1$ oznacza liczbę operacji w bloku B_h.

W pracy [70] dowodzi się również, że po pierwsze, dla klasycznego problemu gniazdowego i każdego rozwiązania $\beta \in AN^1(\pi) \setminus AN^2(\pi)$ spełniona jest nierówność $C_{\max}(\beta) \geqslant C_{\max}(\pi)$, po drugie, jeżeli $AV^2(\pi) = \emptyset$, to rozwiązanie π jest rozwiązaniem optymalnym. W przypadku problemu omawianego w tym rozdziale oba powyższe twierdzenia nie są prawdziwe. Dzieje się tak za sprawą różnic we własnościach bloków operacji produkcyjnych i transportowych. Ze słabego warunku trójkąta, danego nierównością (3.4), oraz niezerowych obciążeń łuków kolejnościowych wynika, że zamiana miejscami dwóch operacji na pozycjach wewnętrznych bloku operacji transportowych może wiązać się z natychmiastową poprawą wartości funkcji celu. Dlatego, w algorytmach opisywanych w dalszej części pracy, głównie używany jest zbiór ruchów $AV^3(\pi) = \bigcup_{h=1}^{lb} AV_h^3(\pi)$ (oraz sąsiedztwo $AN^3(\pi)$), gdzie

$$AV_1^3(\pi) = \begin{cases} \{av_{l_1,f_1}\} & e_1 < f_1, lb > 1, \\ \emptyset & \text{w przeciwnym przypadku,} \end{cases} \tag{3.13}$$

$$AV_h^3(\pi) = \begin{cases} \{av_{l_h,e_h+1}, av_{l_h,f_h}\}, & e_h < f_h, l_h \in M^p, \\ \{av_{l_h,i} : e_h < i \leqslant f_h\}, & e_h < f_h, l_h \in M^t, \\ \emptyset & \text{w przeciwnym przypadku,} \end{cases} \tag{3.14}$$

$$h = 2, ..., lb-1,$$

$$AV^3_{lb}(\pi) = \begin{cases} \{av_{l_{lb},e_{lb}+1}\}, & e_{lb} < f_{lb}, lb > 1, \\ \emptyset & \text{w przeciwnym przypadku.} \end{cases} \tag{3.15}$$

Zbiór ten zawiera

$$|AV^3(\pi)| = \min\{1, |B_1| - 1\} + \sum_{h=2}^{lb-1} z_h + \min\{1, |B_{lb}| - 1\} \tag{3.16}$$

ruchów, gdzie

$$z_h = \begin{cases} \min\{2, |B_h| - 1\}, & l_h \in M^p, \\ |B_h| - 1 & l_h, \in M^t, \end{cases} \tag{3.17}$$

zaś $|B_h| = f_h - e_h + 1$ oznacza liczbę operacji w bloku B_h.

Po wprowadzeniu powyższych definicji możliwe jest przeformułowanie własności prezentowanych w pracach [54,70] i zastosowanie ich do omawianego problemu. Na początku będzie wykazana prawdziwość pewnej własności pomocniczej, wykorzystanej w dowodzie własności 3.2.

Własność 3.1 *Dana jest permutacja $\pi \in \Pi$, acykliczny graf $G(\pi)$, zbiór ruchów $AV^1(\pi)$ oraz dowolny ruch $(x, y) \in AV^1(\pi)$. Operacje x, y połączone są tylko jednym łukiem kolejnościowym $(x, y) \in E^K(\pi)$.*

Dowód. Z definicji zbioru $AV^1(\pi)$ wynika, że operacje x, y należą do ścieżki krytycznej i są połączone łukiem kolejnościowym $(x, y) \in E^K(\pi)$ o obciążeniu $s(x, y) = e(m_{x'}, m_{y'})$, gdzie $x' = a^T_x$, $y' = b^T_y$. Załóżmy, że istnieje ścieżka pomiędzy wierzchołkiem x i wierzchołkiem y nie przechodząca przez łuk (x, y), której długość jest równa ρ. Ścieżka ta musiałaby rozpoczynać się łukiem technologicznym $(x, x') \in E^T$ i kończyć łukiem technologicznym $(y', y) \in E^T$. Z założenia $p_{x'} > 0$, $p_{y'} > 0$ oraz równań (3.1)-(3.3) wynika, że $\rho > s(x, y)$. W takiej sytuacji operacje x, y nie mogłyby być sąsiednimi operacjami na ścieżce krytycznej i tym samym ruch (x, y) nie mógłby należeć do zbioru $AV^1(\pi)$. Powyższe spostrzeżenie kończy dowód ■

Własność 3.2 *Dana jest permutacja $\pi \in \Pi$, acykliczny graf $G(\pi)$ i sąsiedztwo $AN^1(\pi)$. Każda permutacja $\beta \in AN^1(\pi)$ jest dopuszczalna.*

Dowód. Wykazanie dopuszczalności β sprowadza się do pokazania, że graf $G(\beta)$ jest acykliczny. Niech $v = (x, y) \in AV^1(\pi)$ będzie ruchem przeprowadzającym rozwiązanie π w rozwiązanie β, tzn. takim, że $\pi^v = \beta$. Należy zauważyć, że graf $G(\beta)$ można uzyskać z grafu $G(\pi)$ poprzez usunięcie co najwyżej trzech łuków kolejnościowych $(b^K_x(\pi), x)$, (x, y), $(y, a^K_y(\pi))$ i dodanie co najwyżej trzech łuków kolejnościowych $(b^K_x(\pi), y)$, (y, x), $(y, a^K_y(\pi))$. Cykl w grafie $G(\beta)$ może mieć zatem jedną z trzech poniższych postaci

1. po przejściu z wierzchołka $b_x^K(\pi)$ do wierzchołka y po nowo utworzonym łuku możliwy jest powrót do wierzchołka $b_x^K(\pi)$ po ścieżce istniejącej już w grafie $G(\pi)$,

2. po przejściu z wierzchołka y do wierzchołka x po nowo utworzonym łuku możliwy jest powrót do wierzchołka y po ścieżce istniejącej już w grafie $G(\pi)$,

3. po przejściu z wierzchołka x do wierzchołka $a_y^K(\pi)$ po nowo utworzonym łuku możliwy jest powrót do wierzchołka x po ścieżce istniejącej już w grafie $G(\pi)$.

Z acykliczności grafu $G(\pi)$ wynika, że nie istnieje ścieżka z wierzchołka y do wierzchołka $b_x^K(\pi)$, podobnie, nie istnieje ścieżka z wierzchołka $a_y^K(\pi)$ do wierzchołka x. Z własności 3.1 wynika, że w grafie $G(\pi)$ wierzchołki x, y są połączone tylko jednym łukiem kolejnościowym $(x, y) \in E^K(\pi)$, nie występującym w grafie $G(\beta)$, zatem ścieżka z wierzchołka x do wierzchołka y również nie istnieje. Powyższe spostrzeżenia kończą dowód ■

Powyższe dwie własności są oczywiście prawdziwe również dla zbiorów $AV^2(\pi)$ i $AV^3(\pi)$, ponieważ $AV^2(\pi) \subseteq AV^3(\pi) \subseteq AV^1(\pi)$.

Własność 3.3 *Dana jest permutacja $\pi \in \Pi$, sąsiedztwo $AN^1(\pi)$ i sąsiedztwo $AN^3(\pi)$. Dla każdej permutacji $\beta \in AN^1(\pi) \setminus AN^3(\pi)$ zachodzi nierówność $C_{\max}(\beta) \geqslant C_{\max}(\pi)$.*

Ze względu na podobieństwo do dowodu analogicznej własności z pracy [70], dowód powyższej własności może być pominięty. W celu przeprowadzenia dowodu wystarczy zauważyć, że w grafie $G(\beta)$ istnieje ścieżka, której długość jest nie krótsza od długości ścieżki krytycznej w grafie $G(\pi)$.

Własność 3.4 *Dana jest permutacja $\pi \in \Pi$ i zbiór ruchów $AV^3(\pi)$. Jeżeli $AV^3(\pi) = \emptyset$, to permutacja π jest permutacją optymalną.*

Ze względu na podobieństwo do dowodu analogicznej własności z pracy [70], dowód powyższej własności może być pominięty. W celu przeprowadzenia dowodu wystarczy zauważyć, że zbiór $AV^3(\pi)$ może być pusty tylko w dwóch sytuacjach:

1. wszystkie wierzchołki ścieżki krytycznej w grafie $G(\pi)$ należą do jednego bloku operacji produkcyjnych,
2. każdy wierzchołek ścieżki krytycznej należy do innego bloku.

W przypadku pierwszym długość ścieżki krytycznej jest równa dolnemu ograniczeniu generowanemu przez maszyny (ang. *machine-based bound*).

W przypadku drugim długość ścieżki krytycznej jest równa dolnemu ograniczeniu generowanemu przez zadania (ang. *job-based bound*).

Zbiór $AV^3(\pi)$ charakteryzuje się znacznie mniejszym rozmiarem od zbioru $AV^1(\pi)$, $\pi \in \Pi$. Dzięki własności 3.3, ze zbioru tego została usunięta znaczna część ruchów nie mogących spowodować natychmiastowej poprawy wartości funkcji celu $C_{\max}(\pi)$. Należy jednak pamiętać, że po pierwsze, zbiór $AV^3(\pi)$ został wygenerowany na podstawie jednej, arbitralnie wybranej ścieżki krytycznej w grafie $G(\pi)$, po drugie, wraz z wydłużeniem się czasów wykonania operacji transportowych liczba elementów zbioru może drastycznie wzrosnąć. Ma to związek z tym, że poza przejazdami bez załadunku, maszyny transportowe wykonują blisko 50% wszystkich operacji w systemie. Jeżeli liczba maszyn transportowych jest stosunkowo niewielka – maszyny te, pracując niemal bez przerwy, stają się „wąskimi gardłami", zaś wśród bloków operacji na ścieżce krytycznej dominują „długie" bloki operacji transportowych. W takiej sytuacji może okazać się korzystna dalsza redukcja zbioru ruchów $AV^3(\pi)$, na przykład poprzez usunięcie kolejnych ruchów nie powodujących natychmiastowej poprawy wielkości $C_{\max}(\pi)$. Taka redukcja możliwa jest dzięki poniższej, prezentowanej po raz pierwszy, własności, w której wykorzystuje się zbiór ZU oznaczający zbiór wszystkich ścieżek krytycznych w grafie $G(\pi)$, $\pi \in \Pi$, oraz zbiór ZW dany równaniem

$$ZW = \{i \in O : \forall_{u \in ZU}\ i \in u\}. \tag{3.18}$$

Zbiór ZW zawiera tylko te wierzchołki, które należą do każdej ścieżki krytycznej w grafie $G(\pi)$.

Własność 3.5 *Dana jest permutacja $\pi \in \Pi$, acykliczny graf $G(\pi)$, zbiór wszystkich ścieżek krytycznych ZU w grafie $G(\pi)$, zbiór ruchów $AV^1(\pi)$ oraz zbiór ZW dany równaniem (3.18). Niech β będzie permutacją powstałą w wyniku wykonania ruchu $(x, y) \in AV^1(\pi)$. Jeżeli przynajmniej jedna z operacji x, y nie należy do zbioru ZW, to zachodzi nierówność*

$$C_{\max}(\beta) \geqslant C_{\max}(\pi). \tag{3.19}$$

Dowód. Dla uproszczenia notacji, w dowodzie przyjmuje się następujące oznaczenia: $b(\alpha) = b_y^K(\alpha)$, $a(\alpha) = a_y^K(\alpha)$, $\alpha \in \Pi$, oraz $y' = b_y^T$, $y'' = a_y^T$. Dla ustalonego ruchu (x, y) można poczynić następujące spostrzeżenia:

Fakt 1: graf $G(\beta)$ można uzyskać z grafu $G(\pi)$ poprzez usunięcie co najwyżej trzech łuków kolejnościowych $(b_x^K(\pi), x)$, (x, y), $(y, a(\pi))$ i dodanie co najwyżej trzech łuków kolejnościowych $(b_x^K(\pi), y)$, (y, x), $(x, a(\pi))$.

Fakt 2: Jeżeli wierzchołek y posiada bezpośredniego poprzednika technologicznego, tzn. jeśli $y' > 0$, to $r_{y'}^{\pi} = r_{y'}^{\beta}$ (fakt ten jest bezpośrednią konsekwencją faktu 1 oraz własności 3.1).

Fakt 3: Jeżeli wierzchołek y posiada bezpośredniego następnika kolejnościowego, tzn. jeśli $a(\pi) > 0$, to $q^{\pi}_{a(\pi)} = q^{\beta}_{a(\pi)}$ (fakt ten jest bezpośrednią konsekwencją faktu 1).

W celu wykazania prawdziwości nierówności (3.19) należy rozpatrzyć trzy przypadki:

(i) $x \notin ZW$, $y \notin ZW$,
(ii) $x \notin ZW$, $y \in ZW$,
(iii) $x \in ZW$, $y \notin ZW$.

Przypadek (i). Z definicji zbioru ruchów $AV^1(\pi)$ wynika, że w grafie $G(\pi)$ istnieje ścieżka krytyczna $u \in ZU$ o długości $\rho = C_{\max}(\pi)$, nie zawierająca wierzchołków x, y. Z faktu 1 wynika, że w grafie $G(\beta)$ istnieje ścieżka $u' = u$ (nie koniecznie krytyczna), której długość wynosi $\rho' = \rho$, co kończy dowód przypadku (i).

Przypadek (ii). Niech $u = (u_1, u_2, \ldots, u_{lu}) \in ZU$ będzie ścieżką krytyczną o długości ρ taką, że $x \notin u$. Z definicji zbioru ruchów $AV^1(\pi)$ wynika, że $x = b(\pi)$. Wtedy wierzchołek y musi być poprzedzony wierzchołkiem y' na ścieżce u. Zakładając, że tak nie jest – możliwe są dwa przypadki:

(1) Wierzchołek y jest pierwszym wierzchołkiem na ścieżce u, tzn. $u_1 = y$,
(2) Wierzchołek y jest poprzedzony wierzchołkiem x na ścieżce u.

W przypadku (1) ścieżka u nie mogłaby być ścieżką krytyczną ponieważ posiada bezpośredniego poprzednika technologicznego $x = b(\pi)$. W przypadku (2) powstaje sprzeczność z założeniem $x \notin u$. Ścieżka u może mieć zatem jedną z dwóch postaci:

(a) $u = (u_1, \ldots, y', u_k = y, y'', \ldots, u_{lu})$, $1 < k \leqslant lu$ lub
(b) $u = (u_1, \ldots, y', u_k = y, a(\pi), \ldots, u_{lu})$, $1 < k \leqslant lu$.

Sytuacja (a). Z faktu 1 wynika, że w grafie $G(\beta)$ istnieje ścieżka $u' = u$ (nie koniecznie krytyczna), której długość wynosi $\rho' = \rho$.

Sytuacja (b). W tej sytuacji długość ścieżki u wynosi

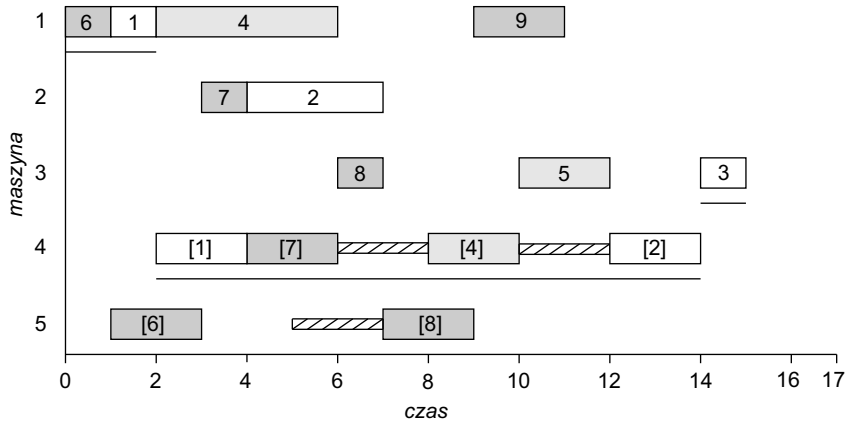
$$\rho = r^{\pi}_{y} + p_y + s(y, a(\pi)) + p_{a(\pi)} + q^{\pi}_{a(\pi)}. \tag{3.20}$$

Ponieważ zarówno wierzchołek y', jak i wierzchołek x (z definicji zbioru $AV^1(\pi)$) należy do jednej ze ścieżek krytycznych w grafie $G(\pi)$ mamy

$$r^{\pi}_{y} = r^{\pi}_{y'} + p_{y'} = r^{\pi}_{x} + p_x + s(x, y). \tag{3.21}$$

Z faktu 1 wynika, że w grafie $G(\beta)$ istnieje (nie koniecznie krytyczna) ścieżka $u' = (u_1, \ldots, y', y, x, a(\pi), \ldots, u_{lu})$, której długość wynosi

$$\rho' = r^{\beta}_{y} + p_y + s(y, x) + p_x + s(x, a(\pi)) + p_{a(\pi)} + q^{\beta}_{a(\pi)}. \tag{3.22}$$



Rysunek 3.3: Wykres Gantta uszeregowania reprezentowanego przez permutację β z przykładu 3.2

Z równania

$$r_y^\beta = \max\{r_{y'}^\beta + p_{y'}, r_{b(\beta)}^\beta + p_{b(\beta)} + s(b(\beta), y)\} \tag{3.23}$$

oraz faktu 2 mamy, że

$$r_y^\beta \geqslant r_y^\pi. \tag{3.24}$$

Ze słabego warunku trójkąta danego nierównością (3.4) mamy

$$s(y, x) + p_x + s(x, a(\pi)) \geqslant s(y, a(\pi)). \tag{3.25}$$

Z powyższego, nierówności (3.24) oraz faktu 3 wynika, że $\rho' \geqslant \rho$, co kończy dowód przypadku (ii).

Przypadek (iii). Dowód tego przypadku, ze względu na symetrię do przypadku (ii), może zostać pominięty ■

Poniżej zilustrowane są różnice pomiędzy sąsiedztwami $AN^1(\pi)$, $AN^2(\pi)$, $AN^3(\pi)$ oraz praktyczne znaczenie własności 3.5 przy użyciu instancji problemu oraz permutacji π z przykładu 3.1.

Przykład 3.2 Dana jest instancja problemu oraz permutacja dopuszczalna $\pi = (\pi_1, \ldots, \pi_5)$, gdzie $\pi_1 = (6, 1, 4, 9)$, $\pi_2 = (7, 2)$, $\pi_3 = (8, 5, 3)$, $\pi_4 = ([1], [4], [7], [2])$, $\pi_5 = ([6], [8])$, rozpatrywana w przykładzie 3.1 (wykres Gantta uszeregowania reprezentowanego przez tą permutację oraz graf $G(\pi)$ zostały przedstawione na odpowiednich rysunkach 3.1 oraz 3.2). Zbiór wszystkich ścieżek krytycznych $ZU = \{u^1, \ldots, u^4\}$ w grafie $G(\pi)$ zawiera 4 ścieżki $u^1 = (6,\ 1,\ [1],\ [4],\ [7],\ [2],\ 3)$,

$u^2 = (6,\ 1,\ 4,\ [4],\ [7],\ [2],\ 3)$, $u^3 = (6,\ 1,\ [1],\ [4],\ [7],\ 8,\ [8],\ 9)$ oraz $u^4 = (6,\ 1,\ 4,\ [4],\ [7],\ 8,\ [8],\ 9)$. Zgodnie z definicją, zbiór $AV^1(\pi)$ zawiera 5 ruchów $AV^1(\pi) = \{(6,1), (1,4), ([1],[4]), ([4],[7]), ([7],[2])\}$. Arbitralnie wybrana ścieżka krytyczna u^1 generuje $lb = 3$ bloki operacji, $u^1 = (B_1,\ B_2,\ B_3)$, gdzie $B_1 = (6,1)$ $(l_1 = 1,\ e_1 = 1,\ f_1 = 2)$, $B_2 = ([1],[4],[7],[2])$ $(l_2 = 4,\ e_2 = 1,\ f_2 = 4)$, $B_3 = (3)$ $(l_3 = 3,\ e_3 = 3,\ f_3 = 3)$. Zbiór $AV^2(\pi)$ zawiera 3 ruchy $AV^2(\pi) = \{(6,1), ([1],[4]), ([7],[2])\}$, zaś zbiór $AV^3(\pi) = AV^2(\pi) \cup ([4],[7])$ zawiera 4 elementy. Zgodnie z własnością 3.3 wykonanie ruchu $(1,4)$ nie może zatem spowodować natychmiastowej poprawy wartości $C_{\max}(\pi) = 17$. W zbiorze ZW, zawierającym wierzchołki należące do wszystkich ścieżek krytycznych ze zbioru ZU, znajdują się zaledwie 4 elementy, $ZW = \{6, 1, [4], [7]\}$. Zgodnie z własnością 3.5 jedynie wykonanie ruchu $(6,1)$ oraz ruchu $([4],[7])$ może się wiązać z natychmiastową poprawą wielkości $C_{\max}(\pi)$.

Dana jest permutacja $\beta = \pi^v$ powstała w wyniku zastosowania ruchu $v = ([4],[7])$ do permutacji π. Uszeregowanie zgodne z tą permutacją zostało przedstawione na rys. 3.3. Jak widać, wykonanie ruchu v spowodowało poprawę wartości funkcji celu rozwiązania π (ponieważ $C_{\max}(\beta) = 15$). Ruch ten nie należy do zbioru $AV^2(\pi)$ (operacje [4], [7] znajdują się na pozycjach wewnętrznych bloku operacji transportowych). Fakt ten dowodzi, że dyskutowana już powyżej, zamiana miejscami dwóch operacji na pozycjach wewnętrznych bloku operacji transportowych może wiązać się z natychmiastową poprawą wartości funkcji celu.

3.2.1 Efektywna metoda przeglądu sąsiedztwa

Najbardziej czasochłonnym elementem wielu algorytmów lokalnych poszukiwań jest wyznaczenie wartości funkcji celu wszystkich rozwiązań w danym sąsiedztwie. W pracy [72] opisano pewne własności akcelerujące, umożliwiające dokonania przeglądu (wyznaczenia wartości funkcji celu wszystkich rozwiązań sąsiednich) sąsiedztwa $AN^2(\pi)$, $\pi \in \Pi$, dla klasycznego problemu gniazdowego w relatywnie krótkim czasie. Poniżej dowodzi się, że analogicznych własności można użyć w celu przyspieszenia przeglądu sąsiedztwa $AN^3(\pi)$, $\pi \in \Pi$, dla problemu gniazdowego z transportem, omawianego w tym rozdziale. W tym celu należy wprowadzić kilka dodatkowych oznaczeń.

Niech $F^\pi = (F^\pi(1), F^\pi(2), \ldots, F^\pi(n))$ będzie porządkiem topologicznym wierzchołków w grafie $G(\pi)$, $\pi \in \Pi$, stanowiącym uporządkowaną listę wszystkich elementów ze zbioru O. Wtedy $f^\pi(x)$ będzie oznaczać pozycję zajmowaną przez wierzchołek $x \in O$ na liście F^π. Lista F^π jest upo-

rządkowana w ten sposób, że dla każdej pary $x, y \in O$ połączonej łukiem $(x, y) \in E(\pi)$ spełniona jest nierówność $f^\pi(x) < f^\pi(y)$.

Poniżej definiuje się zbiory ścieżek. Ścieżka $w^\pi = (w_1^\pi, w_2^\pi, \ldots, w_{lw}^\pi)$, $w_i^\pi \in O$, $1 \leqslant i \leqslant lw$, w grafie $G(\pi)$ jest listą wierzchołków, gdzie $w_{i-1}^\pi \in \{b_{w_i^\pi}^T, b_{w_i^\pi}^K(\pi)\}$, $1 < i \leqslant lw$, zaś lw jest liczbą wierzchołków na liście. Dane są dwa wierzchołki $x, y \in O$. Przez $P^\delta(x)$ i $P^\delta(\overline{x})$ będzie oznaczany zbiór wszystkich ścieżek odpowiednio zawierających i nie zawierających wierzchołka x w grafie $G(\delta)$, $\delta \in \Pi$. Przez $P^\delta(x \vee y)$ będzie oznaczany zbiór wszystkich ścieżek zawierających wierzchołek x lub wierzchołek y. Podobnie, przez $P^\delta(x \wedge y)$ ($P^\delta(\overline{x} \wedge y)$) będzie oznaczany zbiór wszystkich ścieżek zawierających wierzchołki x i y (nie zawierających wierzchołka x i zawierających wierzchołek y). Symbolem $D[P]$ będzie oznaczana długość najdłuższej ścieżki w zbiorze P.

Własność 3.6 *Dla dowolnej permutacji $\pi \in \Pi$, acyklicznego grafu $G(\pi)$ i rozwiązania sąsiedniego $\delta = \pi^v$, gdzie $v = (x, y) \in AV^3(\pi)$, spełniona jest nierówność*

$$D[P^\pi(\overline{x} \wedge y)] \leqslant D[P^\delta(x \vee y)] \tag{3.26}$$

Dowód. Graf $G(\delta)$ różni się od grafu $G(\pi)$ tylko trzema łukami. Łuki $(b_x^K(\pi), x)$, (x, y), $(y, a_y^K(\pi))$ w grafie $G(\pi)$ zastąpione są łukami $(b_x^K(\pi), y)$, (y, x) i $(x, a_y^K(\pi))$ w grafie $G(\delta)$. Należy rozważyć dwa przypadki:

1. $x, y \in O^p$,
2. $x, y \in O^t$.

Przypadek 1. Należy zauważyć, że dla każdej ścieżki ze zbioru $P^\pi(\overline{x} \wedge y)$ istnieje ścieżka w zbiorze $P^\delta(x \vee y)$ zawierająca wszystkie jej wierzchołki i obciążone łuki, co implikuje nierówność (3.26).

Przypadek 2. Należy zauważyć, że zbiór $P^\pi(\overline{x} \wedge y)$ jest sumą dwóch rozłącznych zbiorów $P^\pi(\overline{x} \wedge y \wedge a_y^T)$ i $P^\pi(\overline{x} \wedge y \wedge a_y^K(\pi))$. Podobnie jak w przypadku 1, dla każdej ścieżki ze zbioru $P^\pi(\overline{x} \wedge y \wedge a_y^T)$ istnieje ścieżka w zbiorze $P^\delta(x \vee y)$ zawierająca wszystkie jej wierzchołki i obciążone łuki. Ponadto, dla każdej ścieżki $\rho \in P^\pi(\overline{x} \wedge y \wedge a_y^K(\pi))$ istnieje ścieżka $\sigma \in P^\delta(x \vee y)$, która zawiera wszystkie jej wierzchołki. Ścieżka ρ zawiera jeden obciążony łuk $(y, a_y^K(\pi))$ nie występujący na ścieżce σ, zaś ścieżka σ zawiera dwa obciążone łuki (y, x), $(x, a_y^K(\pi))$ i wierzchołek x pomiędzy nimi, które nie występują na ścieżce ρ. Z porównania obciążeń owych łuków i wierzchołka oraz nierówności (3.4) wynika nierówność

$$s(y, a_y^K(\pi)) \leqslant s(y, x) + p_x + s(x, a_y^K(\pi)), \tag{3.27}$$

która implikuje prawdziwość nierówności $D[\{\rho\}] \leqslant D[\{\sigma\}]$, kończąc w ten sposób dowód ■

Własność 3.7 *Dla dowolnej permutacji $\pi \in \Pi$, acyklicznego grafu $G(\pi)$ i permutacji $\delta = \pi^v$, gdzie $v = (x, y) \in AV^3(\pi)$, spełniona jest nierówność*

$$C_{\max}(\delta) = \max\{D[P^\delta(x \vee y)], D[P^\pi(\overline{x})]\} \tag{3.28}$$

Dowód. Graf $G(\delta)$ różni się od grafu $G(\pi)$ tylko trzema łukami. Łuki $(b_x^K(\pi), x)$, (x, y), $(y, a_y^K(\pi))$ w grafie $G(\pi)$ zastąpione są łukami $(b_x^K(\pi), y)$, (y, x) i $(x, a_y^K(\pi))$ w grafie $G(\delta)$. Z powyższego wynika następujące równanie

$$P^\delta(\overline{x} \wedge \overline{y}) = P^\pi(\overline{x} \wedge \overline{y}). \tag{3.29}$$

Prawdziwość nierówności (3.26) i równania (3.29) implikuje prawdziwość równania

$$\begin{array}{rcl} C_{\max}(\delta) & = & \max\{D[P^\delta(x \vee y)], D[P^\delta(\overline{x} \wedge \overline{y})]\} \\ & = & \max\{D[P^\delta(x \vee y)], D[P^\pi(\overline{x} \wedge \overline{y})]\} \\ & = & \max\{D[P^\delta(x \vee y)], D[P^\pi(\overline{x} \wedge \overline{y})], D[P^\pi(\overline{x} \wedge y)]\} \\ & = & \max\{D[P^\delta(x \vee y)], D[P^\pi(\overline{x})]\}, \end{array} \tag{3.30}$$

która dowodzi równania (3.28) i kończy dowód ■

Można zauważyć, że wielkość $D[P^\delta(x \vee y)]$ może być wyznaczona w czasie $O(1)$ z równania

$$\begin{array}{rcl} D[P^\delta(x \vee y)] & = & \max\{\Delta + q^\pi_{a_y^T}, \max\{\Delta + s(y, x), r^\pi_{b_x^T}\} + p_x \\ & + & \max\{q^\pi_{a_x^T}, q^\pi_{a_y^K(\pi)} + s(x, a_y^K(\pi))\}\}, \end{array} \tag{3.31}$$

gdzie

$$\Delta = \max\{r^\pi_{b_y^T}, r^\pi_{b_x^K(\pi)} + s(b_x^K(\pi), y)\} + p_y. \tag{3.32}$$

Powyższe równanie jest dolnym ograniczeniem (LB) wartości $C_{\max}(\delta)$ zaproponowanym w pracy [105], w którym uwzględniono czasy przejazdów pustych. Obliczenie wartości $D[P^\pi(\overline{x})]$ jest nieco bardziej skomplikowane.

Własność 3.8 *Dana jest permutacja $\pi \in \Pi$, acykliczny graf $G(\pi)$ i porządek topologiczny F^π. Dla każdej permutacji $\delta = \pi^v$, gdzie $v = (x, y) \in AV^3(\pi)$, spełnione jest równanie*

$$D[P^\pi(\overline{x})] = \max\{RQ,\ R',\ R'',\ Q',\ Q''\}, \tag{3.33}$$

gdzie

$$\begin{array}{rcl} RQ & = & \max\{r_a^\pi + s(a, b) + q_b^\pi : \\ & & f^\pi(a) < f^\pi(x) < f^\pi(b), (a, b) \in E^T \cup E^K(\pi)\}, \end{array} \tag{3.34}$$

$$R' = \max\{r_a^\pi : f^\pi(a) < f^\pi(x), a \in A^0\}, \tag{3.35}$$

$$R'' = \max\{r_a^\pi : a \in ZB_x(\pi)\}, \tag{3.36}$$

$$Q' = \max\{q_a^\pi : f^\pi(x) < f^\pi(a), a \in B^0\}, \tag{3.37}$$

$$Q'' = \max\{q_a^\pi : a \in ZA_x(\pi)\}, \tag{3.38}$$

$$A^0 = \{j \in O : a_j^T = 0\}, \qquad B^0 = \{j \in O : b_j^T = 0\}. \tag{3.39}$$

Dowód własności jest podobny do dowodu analogicznej własności z pracy [72] i może być pominięty. Można zauważyć, że dysponując wartością $C_{\max}(\pi)$ nie ma potrzeby wyznaczania wartości R'', Q'', ponieważ $C_{\max}(\pi) \geqslant \max\{R'', Q''\}$. Z własności 3.7, 3.8 wynika, że

$$C_{\max}(\delta) = \begin{cases} LB, & LB \geqslant C_{\max}(\pi), \\ \max\{RQ, R', Q', LB\}, & \text{w przeciwnym przypadku.} \end{cases} \tag{3.40}$$

Twierdzenie 3.1 *Dana jest permutacja $\pi \in \Pi$, acykliczny graf $G(\pi)$, sąsiedztwo $AN^3(\pi)$, porządek topologiczny F^π oraz wartości r_i^π, q_i^π, $i \in O$. Dla każdego rozwiązania sąsiedniego $\delta \in AN^3(\pi)$ wartość funkcji celu $C_{\max}(\delta)$ można wyznaczyć w czasie $O(\max\{\sum_{k\in J} \log n_k, \sum_{l\in M} \log o_l\})$.*

Dowód. Wartości LB, R', Q' mogą być wyznaczone w czasie odpowiednio $O(1)$, $O(r)$ i $O(r)$. Formalnie, wielkość RQ można wyznaczyć w czasie $O(n)$, ponieważ $|E^K(\pi)| = \sum_{k\in M}(o_k - 1) = n - m$, $|E^T| = \sum_{k\in J}(n_k - 1) = n - r$. Jednakże, w zbiorze łuków $\bigcup_{i=1}^{o_l-1}\{(\pi_l(i), \pi_l(i+1))\}$, związanych z maszyną $l \in M$, istnieje co najwyżej jeden łuk (a, b) taki, że $f^\pi(a) < f^\pi(x) < f^\pi(b)$. Odnalezienie tego łuku (bądź odpowiedź, że ten łuk nie istnieje) możliwe jest dzięki poszukiwaniu binarnemu w czasie $O(\log o_l)$ w rosnącej sekwencji $f^\pi(\pi_l(1)), \ldots, f^\pi(\pi_l(o_l))$. Podobnie jest w przypadku zbioru łuków $\bigcup_{i=1}^{n_k-1}\{(j_k + i, j_k + i + 1)\}$, związanych z zadaniem $k \in J$. Zatem, łuk (a, b) można odnaleźć dzięki poszukiwaniu binarnemu w czasie $O(\log n_k)$ w rosnącej sekwencji $f^\pi(j_k + 1), \ldots, f^\pi(j_k + n_k)$. Z powyższego wynika, że wartość $D[RQ]$ można wyznaczyć w czasie $O(\max\{\sum_{k\in J} \log n_k, \sum_{l\in M} \log o_l\})$, co implikuje możliwość wyznaczenia wartości $C_{\max}(\delta)$ w tym samym czasie i kończy dowód. ■

Powyższy dowód przeprowadza się identycznie jak dowód analogicznego twierdzenia z pracy [72] i w zasadzie mógłby zostać pominięty. Dowód ten został jednak zamieszczony ze względu na jego kluczową rolę dla całości rozważań prowadzonych w tej sekcji. Można też zauważyć, że twierdzenie 3.1, jak i własności 3.6-3.8 są również prawdziwe dla sąsiedztwa $AN^2(\pi)$, $\pi \in \Pi$, w kontekście badanego problemu, ponieważ $AN^2(\pi) \subseteq AN^3(\pi)$.

3.3 Własności grafu i permutacji częściowej

W tej sekcji przedstawione są pewne specyficzne własności problemu, które są wykorzystywane przy konstrukcji algorytmów typu wstaw, opisanych w dalszej części pracy. Ze względu jednak na ich ogólny charakter i możliwość wykorzystania w innych algorytmach (na przykład wykorzystujących sąsiedztwa generowane przy użyciu ruchów typu wstaw), zamieszczone są w oddzielnej sekcji. Przed ich omówieniem niezbędne jest wprowadzenie kilku dodatkowych pojęć. Niech $U \subseteq O$ oznacza zbiór operacji uszeregowanych, stanowiący dowolny podzbiór operacji O. Poprzez permutację częściową β_l, $l \in M$, rozumiana będzie permutacja, w której nie koniecznie wszystkie operacje ze zbioru O_l zostały uszeregowane, tj. permutacja zbioru $U_l = \{i \in \{U \cap O_l\}\}$. Zestawem permutacji częściowych (dla uproszczenia również nazywanym permutacją częściową) będzie nazywany taki zestaw permutacji, w którym przynajmniej jedna permutacja jest permutacją częściową. Zbiór wszystkich permutacji częściowych zbioru U będzie oznaczany symbolem $\Pi^{0,U}$. Dla dowolnej permutacji częściowej $\beta \in \Pi^{0,U}$ można skonstruować skierowany graf częściowy $G(\beta) = (O, E(\beta))$ ze zbiorem obciążonych wierzchołków O i zbiorem łuków $E(\beta) = E^T \cup E^K(\beta)$. Graf częściowy $G(\beta)$ różni się od grafu „kompletnego" $G(\pi)$, $\pi \in \Pi$ (prezentowanego w sekcji 3.1.1), jedynie postacią zbioru łuków kolejnościowych

$$E^K(\beta) = \bigcup_{l=1}^{m} \bigcup_{j=1}^{|\beta_l|-1} \{(\beta_l(j), \beta_l(j+1))\}, \tag{3.41}$$

gdzie $|\beta_l|$ jest liczbą operacji w permutacji β_l, $l \in M$. W tym miejscu warto zauważyć, że graf częściowy $G(\beta)$, $\beta \in \Pi^{0,U}$, jest również poprawnie określony, gdy $U = \emptyset$. W tej sytuacji zbiór łuków kolejnościowych $E^K(\beta)$ jest zbiorem pustym. Dopuszczalną permutacją częściową będzie nazywana taka permutacja $\beta \in \Pi^{0,U}$, dla której graf $G(\beta)$ jest acykliczny. Zbiór wszystkich dopuszczalnych permutacji częściowych zbioru U będzie oznaczany symbolem Π^U.

Niech wielkość d_i^β będzie miarą długości najdłuższej ścieżki przechodzącej przez wierzchołek $i \in O$ w grafie $G(\beta)$, $\beta \in \Pi^{0,U}$, dla ustalonego zbioru U. W przypadku acyklicznego grafu $G(\beta)$ przyjmuje się, że wielkość ta jest równa długości najdłuższej ścieżki przechodzącej przez wierzchołek i, czyli

$$d_i^\beta = r_i^\beta + p_i + q_i^\beta. \tag{3.42}$$

Miara długości d_i^β jest wykorzystywana we wspomnianych już wyżej algorytmach typu wstaw do oszacowania jakości próbnego wstawienia danej

operacji i. Dokładniej, niech $Z(\beta,i) = \{\alpha^1, \alpha^2, \ldots, \alpha^{|\beta_l|+1}\}$ określa zbiór $|\beta_l|+1$ permutacji częściowych, które zostały utworzone z permutacji częściowej β poprzez wstawienie nie uszeregowanej operacji $i \in O \setminus U$ na pozycje $z = 1, 2, \ldots, |\beta_l|+1$ w permutacji β_l na maszynie $l = m_i$;

$$\begin{array}{l} Z(\beta,i) = \{(\beta_1, \beta_2, \ldots, \beta_{l-1}, \omega, \beta_{l+1}, \ldots, \beta_m) : \\ \omega = (\beta_l(1), \ldots, \beta_l(z-1), i, \beta_l(z), \ldots, \beta_l(|\beta_l|)), 1 \leqslant z \leqslant |\beta_l|+1\}. \end{array} \tag{3.43}$$

We wspomnianych algorytmach konstrukcyjnych typu wstaw należy odnaleźć najlepszą permutację δ w zbiorze $Z(\beta,i)$, tzn. taką, że $C_{\max}(\delta) = \min_{\alpha \in Z(\beta,i)} C_{\max}(\alpha)$, gdzie $C_{\max}(\alpha)$ jest długością najdłuższej ścieżki w grafie częściowym $G(\alpha)$. Wiąże się to z koniecznością wyznaczenia wartości d_i^α dla każdej permutacji częściowej $\alpha \in Z(\beta,i)$, dla której graf $G(\alpha)$ jest acykliczny. Procedura ta jest bardzo pracochłonna, ponieważ wyznaczenie jednej wartości d_i^α (a konkretnie wartości r_i^α, q_i^α) lub stwierdzenie, że graf $G(\alpha)$ jest cykliczny wymaga aż $O(n)$ czasu. Korzystając jednak z następujących trzech własności można to zrobić w czasie znacznie krótszym.

Własność 3.9 *Dany jest zbiór operacji uszeregowanych $U \subset O$, permutacja częściowa $\beta \in \Pi^U$, nieuszeregowana operacja $i \in O \setminus U$ oraz zbiór $Z(\beta,i)$. Jeżeli graf $G(\alpha)$, $\alpha \in Z(\beta,i)$, jest acykliczny, to długość najdłuższej ścieżki przechodzącej przez wierzchołek i w tym grafie, a tym samym wartość d_i^α, można wyznaczyć przy użyciu poniższych równań:*

$$d_i^\alpha = \max\{r_i^\beta, r_x^\beta + p_x\} + p_i + \max\{q_i^\beta, q_y^\beta + p_y\}, \qquad i \in O^p, \tag{3.44}$$

$$\begin{array}{rcll} d_i^\alpha & = & \max\{r_i^\beta, r_x^\beta + p_x + s(x,i)\} + p_i & \\ & + & \max\{q_i^\beta, q_y^\beta + p_y + s(i,y)\}, & i \in O^t, \end{array} \tag{3.45}$$

gdzie $x = b_i^K(\alpha)$, $y = a_i^K(\alpha)$, $p_0 = 0$.

Ze względu na podobieństwo do dowodu analogicznej własności dla klasycznego problemu gniazdowego, opisywanego m.in. w pracy [36], dowód powyższej własności będzie pominięty. Dowód powyższych równań opiera się na spostrzeżeniu, że długości najdłuższych ścieżek dochodzących i wychodzących z niektórych wierzchołków w grafach $G(\beta)$ i $G(\alpha)$ są sobie równe, tj. $r_j^\alpha = r_j^\beta$, $j \in ZB_i(\beta)$ oraz $q_k^\alpha = q_k^\beta$, $k \in ZA_i(\beta)$.

Należy zauważyć, że dzięki równaniom (3.44), (3.45) (a konkretnie, dzięki wykorzystaniu wartości r_j^β, q_j^β, $j \in O$, wyliczonych już dla grafu $G(\beta)$) możliwe są dwie rzeczy. Po pierwsze, czas wyznaczenia najdłuższej ścieżki przechodzącej przez wierzchołek $i \in O$ w acyklicznym grafie $G(\alpha)$, $\alpha \in Z(\beta,i)$ (a tym samym wielkości d_i^α), redukuje się z $O(n)$ do $O(1)$.

Po drugie, wielkość d_i^α można wyznaczyć dla każdego (nie koniecznie acyklicznego) grafu $G(\alpha)$, $\alpha \in Z(\beta, i)$, również w czasie $O(1)$. Bezpośrednią konsekwencją prawdziwości powyższej własności jest również prawdziwość równania

$$C_{\max}(\alpha) = \max\{C_{\max}(\beta), d_i^\alpha\} \tag{3.46}$$

dla każdej permutacji $\alpha \in Z(\beta, i)$ nie generującej cyklu w grafie $G(\alpha)$. To oznacza, że odnalezienie w zbiorze $Z(\beta, i)$ permutacji δ takiej, że $d_i^\delta = d_{\min}$, gdzie

$$d_{\min} = \min_{\alpha \in Z(\beta,i)} d_i^\alpha, \tag{3.47}$$

implikuje prawdziwość równania

$$C_{\max}(\delta) = \min_{\alpha \in Z(\beta,i)} C_{\max}(\alpha). \tag{3.48}$$

Pozostaje jeszcze kwestia znalezienia efektywnej metody określania czy graf $G(\alpha)$, $\alpha \in Z(\beta, i)$, jest acykliczny. Umożliwiają to poniższe dwie własności; własność 3.10 jest własnością analogiczną do własności „klasycznej", zaś własność 3.11 jest specyficzną własnością problemu, wykrytą przez autora tej rozprawy. W obu własnościach wykorzystuje się zbiór permutacji

$$ZD(\beta, i) = \{\alpha \in Z(\beta, i) : d_i^\alpha = d_{\min}\}, \tag{3.49}$$

gdzie wartość $d_{\min}$ dana jest równaniem (3.47), zaś wartości d_i^α dane są odpowiednimi równaniami (3.44), (3.45). W poniższych własnościach wykorzystuje się też pozycje $e \in ZE$, $f \in ZF$, gdzie

$$\begin{aligned} ZE \;=\; & \max\{1 \leqslant j \leqslant |\beta_l| : \text{istnieje ścieżka} \\ & \text{z wierzchołka } \beta_l(j) \text{ do } i \text{ w grafie } G(\beta)\}, \end{aligned} \tag{3.50}$$

$$\begin{aligned} ZF \;=\; & \min\{1 \leqslant j \leqslant |\beta_l| : \text{istnieje ścieżka} \\ & \text{z wierzchołka } i \text{ do } \beta_l(j) \text{ w grafie } G(\beta)\}. \end{aligned} \tag{3.51}$$

Własność 3.10 *Dany jest zbiór operacji uszeregowanych $U \subset O$, permutacja częściowa $\beta \in \Pi^U$, nieuszeregowana operacja produkcyjna $i \in O^p \setminus U$ oraz zbiór $Z(\beta, i)$. Wtedy dla każdej permutacji $\alpha \in ZD(\beta, i)$ graf $G(\alpha)$ jest acykliczny.*

Ze względu na podobieństwo do dowodu analogicznej własności dla klasycznego problemu gniazdowego, prezentowanego m.in. w pracach [36, 65], dowód powyższej własności będzie pominięty.

Własność 3.11 *Dany jest zbiór operacji uszeregowanych $U \subset O$, permutacja częściowa $\beta \in \Pi^U$, nieuszeregowana operacja transportowa $i \in O^t \setminus U$, maszyna $l = m_i$ oraz zbiór $Z(\beta, i)$. Wtedy zbiór $ZD(\beta, i)$ zawiera przynajmniej jedną taką permutację α, że graf $G(\alpha)$ jest acykliczny.*

Dowód. Dla uproszczenia notacji, dla ustalonego zbioru $U \subset O$ i ustalonej operacji transportowej $i \in O^t \setminus U$ w dowodzie przyjmuje się następujące oznaczenia: $b(\alpha) = b_i^K(\alpha)$, $a(\alpha) = a_i^K(\alpha)$, $\alpha \in \Pi^U \cup \Pi^{U\cup\{i\}}$. Dla permutacji częściowej β_l określa się dwie pozycje $e \in ZE$, $f \in ZF$, gdzie zbiory ZE, ZF dane są odpowiednimi równaniami (3.50), (3.51). Zbiory ZE, ZF są co najwyżej jednoelementowe. Jeżeli któryś ze zbiorów jest pusty, to przyjmuje się odpowiednio $e = 0$, $f = |\beta_l| + 1$, $\beta_l(0) = \beta_l(|\beta_l| + 1) = 0$. Z własności grafu acyklicznego wynika, że $e < f$. Z definicji zbiorów ZE, ZF wynika, że istnieje ścieżka z wierzchołka $\beta_l(e)$ do i oraz z wierzchołka i do $\beta_l(f)$. Zatem, istnieje ścieżka z wierzchołka $\beta_l(e)$ do $\beta_l(f)$. Niech $\alpha \in Z(\beta, i)$ oznacza permutację powstałą z permutacji β po wstawieniu operacji i na pozycję $1 \leqslant z \leqslant |\beta_l| + 1$. Poniżej wykazuje się, że w przypadku, gdy $e < z \leqslant f$, graf $G(\alpha)$ jest acykliczny.

Należy zauważyć, że graf $G(\alpha)$, a dokładniej zbiór łuków kolejnościowych $E^K(\alpha)$ w tym grafie ma postać

$$E^K(\alpha) = \begin{cases} E^K(\beta) \cup \{(i, x)\}, & z = 1, \\ E^K(\beta) \setminus \{(y, x)\} \cup \{(y, i), (i, x)\}, & 1 < z \leqslant |\beta_l|, \\ E^K(\beta) \cup \{y, i)\}, & z = |\beta_l| + 1, \end{cases} \tag{3.52}$$

gdzie $y = \beta_l(z-1)$, $x = \beta_l(z)$, $l = m_i$. Wierzchołki, jak i łuki technologiczne nie ulegają zmianie. Zatem, graf $G(\alpha)$ można utworzyć z grafu $G(\beta)$ poprzez dodanie co najwyżej dwóch łuków kolejnościowych $(b(\alpha), i)$, $(i, a(\alpha))$, gdzie $(b(\alpha) = \beta_l(z-1), a(\alpha) = \beta_l(z)$. Cykl w grafie może mieć tylko dwie postacie:

1. po przejściu z wierzchołka $b(\alpha)$ do wierzchołka i po nowo utworzonym łuku możliwy jest powrót do wierzchołka $b(\alpha)$ po ścieżce istniejącej już w grafie $G(\beta)$ lub

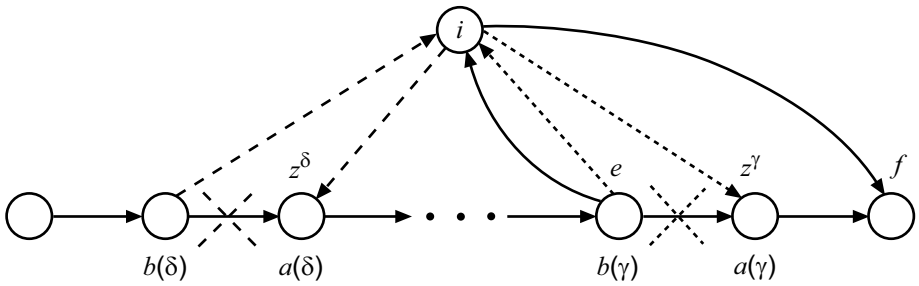
2. po przejściu z wierzchołka i do wierzchołka $a(\alpha)$ po nowo utworzonym łuku możliwy jest powrót do wierzchołka i po ścieżce istniejącej już w grafie $G(\beta)$.

Przy założeniu, że graf $G(\beta)$ jest acykliczny, więc przy warunku $e < z^\alpha \leqslant f$, w obu przypadkach ścieżka z wierzchołka $a(\alpha)$ do $b(\alpha)$ nie istnieje.

Niech $\delta \in Z(\beta, i)$ będzie permutacją powstałą z permutacji β po wstawieniu operacji i na pozycję $1 \leqslant z^\delta \leqslant |\beta_l| + 1$ taką, że graf $G(\delta)$ jest cykliczny. Stąd wynika, że zachodzi jedna z dwóch poniższych sytuacji

1. $z^\delta \leqslant e$,
2. $f < z^\delta$.

Poniżej każda z tych sytuacji będzie rozpatrzona osobno.



Rysunek 3.4: Modyfikacje grafu $G(\beta)$

Sytuacja 1. Niech $\gamma \in Z(\beta, i)$ będzie permutacją otrzymaną po wstawieniu operacji transportowej i na pozycję $z^\gamma = e + 1$ w permutacji β_l. Sytuację tą ilustruje rysunek 3.4. Łuki zaznaczone liniami ciągłymi są wspólne dla wszystkich rozpatrywanych grafów. Łuki wyrysowane linią przerywaną bądź skreślone linią przerywaną odpowiednio istnieją, bądź nie istnieją w grafie $G(\delta)$. Analogicznie należy analizować łuki narysowane linią punktowaną dla grafu $G(\gamma)$. Na podstawie definicji (3.45), miary długości dla poszczególnych grafów mają postać

$$\begin{aligned} d_i^\delta &= \max\{r_i^\beta, r_{b(\delta)}^\beta + p_{b(\delta)} + s(b(\delta), i)\} \\ &\quad + p_i + \max\{q_i^\beta, q_{a(\delta)}^\beta + p_{a(\delta)} + s(i, a(\delta))\}; \end{aligned} \tag{3.53}$$

$$\begin{aligned} d_i^\gamma &= \max\{r_i^\beta, r_{b(\gamma)}^\beta + p_{b(\gamma)} + s(b(\gamma), i)\} \\ &\quad + p_i + \max\{q_i^\beta, q_{a(\gamma)}^\beta + p_{a(\gamma)} + s(i, a(\gamma))\}. \end{aligned} \tag{3.54}$$

Z definicji e wynika, że w grafie $G(\beta)$ istnieje ścieżka z wierzchołka $b(\gamma) = \beta_l(e)$ do i, więc

$$r_i^\beta \geqslant r_{b(\gamma)}^\beta + p_{b(\gamma)} + s(b(\gamma), i), \tag{3.55}$$

Można zauważyć, że gdy $z^\delta = e$, to w grafie $G(\beta)$ wierzchołek $a(\delta) = b(\gamma)$ jest bezpośrednim poprzednikiem kolejnościowym wierzchołka $a(\gamma)$ i prawdziwa jest nierówność (3.56). W przypadku, gdy $z^\delta < e$, to pomiędzy wierzchołkami $a(\delta)$ i $a(\gamma)$ w grafie $G(\beta)$ istnieje ścieżka zbudowana z łuków kolejnościowych, przechodząca przez operacje transportowe, których łączne obciążenie (wynikające ze słabego warunku trójkąta danego równaniem (3.4)) jest nie mniejsze, niż czas przejazdu pustego $s(a(\delta), a(\gamma))$, co również implikuje prawdziwość nierówności (3.56).

$$q_{a(\delta)}^\beta \geqslant s(a(\delta), a(\gamma)) + p_{a(\gamma)} + q_{a(\gamma)}^\beta. \tag{3.56}$$

Można zauważyć, że ze słabego warunku trójkąta (nierówność (3.4)) wynika prawdziwość nierówności

$$s(i, a(\delta)) + p_{a(\delta)} + s(a(\delta), a(\gamma)) \geqslant s(i, a(\gamma)). \tag{3.57}$$

Po obustronnym dodaniu wartości $p_{a(\gamma)} + q^{\beta}_{a(\gamma)}$ powyższa nierówność przyjmuje postać

$$\begin{gathered} s(i, a(\delta)) + p_{a(\delta)} + s(a(\delta), a(\gamma)) + p_{a(\gamma)} + q^{\beta}_{a(\gamma)} \geqslant \\ s(i, a(\gamma)) + p_{a(\gamma)} + q^{\beta}_{a(\gamma)}. \end{gathered} \tag{3.58}$$

Po uwzględnieniu nierówności (3.56) powyższa nierówność przyjmuje postać

$$q^{\beta}_{a(\delta)} + p_{a(\delta)} + s(i, a(\delta)) \geqslant q^{\beta}_{a(\gamma)} + p_{a(\gamma)} + s(i, a(\gamma)). \tag{3.59}$$

Jak już wspomniano, gdy $z^{\delta} = e$, to w grafie $G(\beta)$ wierzchołek $a(\delta) = b(\gamma)$. Natomiast jeżeli $z^{\delta} < e$, to wierzchołek $a(\delta)$ jest poprzednikiem technologicznym wierzchołka $b(\gamma)$ w grafie $G(\beta)$. Stąd w obu przypadkach prawdziwe są nierówności

$$q^{\beta}_{a(\delta)} + p_{a(\delta)} + s(i, a(\delta)) > q^{\beta}_{a(\delta)} \geqslant q^{\beta}_{b(\gamma)} \geqslant q^{\beta}_{i} + p_i + s(b(\gamma), i) > q^{\beta}_{i}. \tag{3.60}$$

Prawdziwość nierówności (3.55), (3.59) oraz (3.60) implikuje prawdziwość nierówności

$$d^{\delta}_{i} \geqslant d^{\gamma}_{i}, \tag{3.61}$$

co kończy dowód sytuacji 1. Analiza sytuacji 2 też prowadzi do nierówności (3.61). Ze względu na symetrię, wyprowadzenie to będzie pominięte ■

Z reguły, w zbiorze $ZD(\beta, i)$, $i \in O^t \backslash U$, znajdują się jedynie permutacje dopuszczalne. Aby w zbiorze tym znalazły się permutacje niedopuszczalne (co odpowiada przejściu nierówności (3.61) w równość), musi być spełniony cały szereg warunków, który w praktyce spełniony jest rzadko. Przykład 3.3 ilustruje sytuację, w której w zbiorze $D(\beta, i)$ znajdują się permutacje zarówno dopuszczalne, jak i niedopuszczalne.

Przykład 3.3 Dany jest system produkcyjny ze zbiorem $m^p = 6$ maszyn produkcyjnych $M^p = \{1, 2, \ldots, 6\}$ i $m^t = 1$ maszyną transportową $M^t = \{7\}$. W systemie należy wykonać $r = 2$ zadania ze zbioru $J = \{1, 2\}$. Zadanie 1 składa się z sekwencji operacji $1, [1], 2, [2], 3, [3], 4$ na zadanie 2 składa się sekwencja operacji $5, [5], 6$ ($n_1 = n^p_1 + n^t_1 = 4 + 3 = 7$, $n_2 = n^p_2 + n^t_2 = 2 + 1 = 3$). Zbiór operacji produkcyjnych i transportowych

jest zatem równy odpowiednio $O^p = \{1, 2, \ldots, 6\}$ i $O^t = \{[1], [2], [3], [5]\}$ ($n^p = n_1^p + n_2^p = 4+2 = 6$, $n^t = n_1^t + n_2^t = 4+1 = 4$, $n = n^p + n^t = 6+4 = 10$). Przydział maszyn do operacji oraz czasy ich wykonania dane są w tabeli 3.2. Rozmieszczenie maszyn produkcyjnych (i dróg) przedstawione jest na rysunku 3.5. Zakłada się jednostkowe czasy przejazdów pustych maszyny transportowej pomiędzy dowolną parą sąsiadujących ze sobą maszyn produkcyjnych (z wyjątkiem maszyn 3, 5, gdzie $e(3,5) = 2$). Wynikające z czasów przejazdów pustych czasy przezbrojeń maszyny transportowej pomiędzy poszczególnymi operacjami transportowymi dane są w tabeli 3.3.

Tabela 3.2: Maszyny dedykowane, czasy wykonywania i długości najdłuższych ścieżek

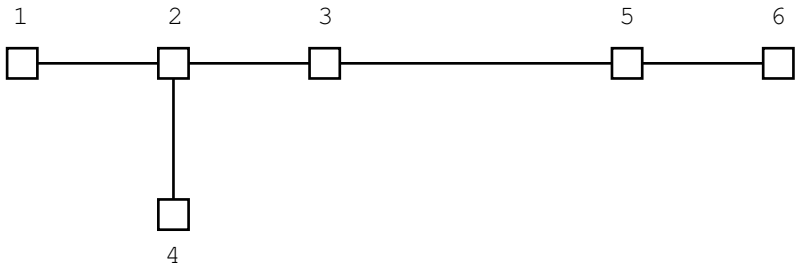
i	1	[1]	2	[2]	3	[3]	4	5	[5]	6
m_i	1	7	2	7	3	7	4	6	7	5
r_i^β	0	1	2	3	4	5	7	0	7	8
p_i	1	1	1	1	1	2	2	1	1	1
q_i^β	8	7	6	5	4	2	0	2	1	0

Tabela 3.3: Czasy przezbrojeń pomiędzy operacjami

$s(i,k)$	[1]	[2]	[3]	[5]
[1]	1	0	1	4
[2]	2	1	0	3
[3]	2	1	2	5
[5]	4	3	2	1

Dany jest zbiór $U = \{1, \ldots, 6, [1], [2], [5]\}$, permutacja częściowa $\beta = (\beta_1, \ldots, \beta_7) = ((1), (2), (3), (4), (6), (5), ([1], [2], [5])) \in \Pi^U$ oraz operacja transportowa $i = [3] \in O^t \setminus U$. Długości najdłuższych ścieżek dochodzących i wychodzących z poszczególnych wierzchołków grafu $G(\beta)$ są przedstawione w tabeli 3.2. Poniżej będzie rozpatrywana maszyna $l = m_i = 7$. W zbiorze $Z(\beta, i)$ znajdują się $|\beta_l| + 1 = 4$ rozwiązania $\alpha^1, \ldots, \alpha^4$, powstałe poprzez wstawienie operacji i na odpowiednie pozycje $z = 1, \ldots, 4$ w permutacji β_l. Poszczególne miary długości przyjmują wartości

$$d_i^{\alpha^1} = \max(5, 0+0+0) + 2 + \max(2, 7+1+2) = 5 + 2 + 10 = 17, \quad (3.62)$$



Rysunek 3.5: Rozmieszczenie maszyn i dróg pomiędzy maszynami

$$d_i^{\alpha^2} = \max(5, 1+1+1) + 2 + \max(2, 5+1+1) = 5+2+7 = 14, \quad (3.63)$$

$$d_i^{\alpha^3} = \max(5, 3+1+0) + 2 + \max(2, 1+1+5) = 5+2+7 = 14, \quad (3.64)$$

$$d_i^{\alpha^4} = \max(5, 7+1+2) + 2 + \max\{2, 0+0+0\} = 10+2+2 = 14. \quad (3.65)$$

Wartość $d_{\min} = \min_{\alpha \in Z(\beta,i)} d_i^{\alpha} = 14$. W zbiorze $ZD(\beta, i)$ znajdują się zatem 3 rozwiązania, $ZD(\beta, i) = \{\alpha^2, \alpha^3, \alpha^4\}$. Rozwiązanie α^2 jest rozwiązaniem niedopuszczalnym, ponieważ w rozwiązaniu tym operacja [3] została uszeregowana przed operacją [2], będącą jej poprzednikiem technologicznym.

Siła powyżej przedstawionych własności polega na tym, że w celu wyznaczenia najlepszej pozycji dla wstawianej operacji $i \in O$, tzn. wyznaczenia najlepszej permutacji w zbiorze $Z(\beta, i)$ (takiej, że graf $G(\delta)$ jest acykliczny oraz $d_i^{\delta} = \min\{d_i^{\alpha} : \alpha \in Z(\beta, i)\}$), nie potrzeba sprawdzać acykliczności wszystkich grafów $G(\alpha)$, $\alpha \in Z(\beta, i)$. W przypadku operacji produkcyjnych nie trzeba robić tego wcale (wyznaczenie najlepszej pozycji wymaga wtedy $O(|\beta_{m_i}| + 1)$, $i \in O^p$, czasu), zaś dla operacji transportowych $i \in O^t \setminus U$, gdy $|ZD(\beta, i)| > 1$, sprawdzenie wystarczy ograniczyć tylko do $|ZD(\beta, i)|$ grafów; gdy $|ZD(\beta, i)| = 1$ sprawdzenie acykliczności grafu $G(\alpha)$, $\alpha \in ZD(\beta, i)$ również nie jest potrzebne.

3.3.1 Efektywna metoda wyznaczania najlepszej pozycji

Mając na uwadze własności przedstawione w poprzedniej sekcji oraz postępując podobnie jak w monografii [65] (dla problemu gniazdowego z przezbrojeniami o dowolnym czasie trwania) poniżej zostanie naszkicowana metoda wyznaczania najlepszej pozycji dla wstawianej operacji $i \in O^t \setminus U$. Bez straty ogólności, dalsze rozważania mogą być ograniczone do ustalonego zbioru operacji uszeregowanych $U \subset O$, permutacji częściowej $\beta \in \Pi^U$, acyklicznego grafu $G(\beta)$, nie uszeregowanej operacji transportowej $i \in O^t \setminus U$

Procedura rozpoczyna działanie z permutacją częściową β, acyklicznym grafem $G(\beta)$ i nie uszeregowaną operacją transportową i. Procedura zwraca najlepszą pozycję z^* dla wstawianej operacji i.

1. wyznacz zbiory $ZB_j(\beta)$, $ZA_k(\beta)$ dla $j = b_i^T$, $k = a_i^T$,
2. wyznacz pozycje $e \in ZE$, $f \in ZF$, gdzie zbiory ZE, ZF dane są odpowiednimi równaniami (3.50), (3.51),
3. wyznacz wartości er_z, eq_z, $1 \leqslant z \leqslant |\beta_l|$ oraz r_i^β, q_i^β,
4. oblicz wartości ed_z, $e < z \leqslant f$ i wśród nich znajdź wartość najmniejszą ed_{z^*}. Zwróć pozycję z^* i STOP.

Rysunek 3.6: Metoda wyznaczania najlepszej pozycji dla wstawianej operacji i

i maszyny $l = m_i$. Niech $er_z = r_x^\beta$, $eq_z = q_x^\beta$, $ep_z = p_x$ oznacza odpowiednio długość najdłuższej ścieżki dochodzącej i wychodzącej z wierzchołka $x = \beta_l(z)$, $1 \leqslant z \leqslant |\beta_l|$, oraz jego obciążenie w grafie $G(\beta)$. Przyjmuje się, że $er_0 = ep_0 = ep_{|\beta_l|+1} = eq_{|\beta_l|+1} = 0$. Wprowadza się też nieco bardziej „wygodną" definicję miary długości dla poniższych rozważań. Niech

$$\begin{aligned} ed_z &= d_i^{\alpha^z} = \max\{r_i^\beta, er_{z-1} + ep_{z-1} + s(y,i)\} + p_i \\ &+ \max\{q_i^\beta, eq_z + ep_z + s(i,x)\}, \end{aligned} \tag{3.66}$$

będzie miarą długości najdłuższej ścieżki przechodzącej przez wierzchołek i w grafie $G(\alpha^z)$, $\alpha^z \in \{\alpha \in Z(\beta,i) : \alpha_l(z) = i\}$, $1 \leqslant z \leqslant |\beta_l| + 1$, gdzie $y = \beta_l(z-1)$, $x = \beta_l(z)$, zaś zbiór $Z(\beta,i)$ dany jest równaniem (3.43). Jak już wspominano, każde rozwiązanie α^z takie, że $e < z \leqslant f$, $e \in ZE$, $f \in ZF$, jest rozwiązaniem dopuszczalnym, gdzie zbiory ZE, ZF dane są odpowiednimi równaniami (3.50), (3.51). Metoda wyznaczenia najlepszej pozycji dla wstawianej operacji i sprowadza się do wyznaczenia takiej pozycji z^* z przedziału $e+1, \ldots, f$, że $ed_{z^*} = \min_{e<z\leqslant f} ed_z$. Pozycję z^* spełniającą powyższe założenia wyznacza procedura przedstawiona na rys. 3.6, składająca się z czterech kroków.

Oczywiście, jak już wspominano, stosowanie powyższej procedury ma sens jedynie w sytuacji, gdy $|ZD(\beta,i)| > 1$. Wyznaczona pozycja z^* jest najlepszą pozycją dla wstawianej operacji i, zaś odpowiadające jej rozwiązanie α^{z^*} jest najlepszym rozwiązaniem w zbiorze $Z(\beta,i)$. Wartość $C_{\max}(\alpha^{z^*})$ można wyznaczyć z równania (3.46). Kolejne kroki procedury wymagają odpowiednio $O(n)$, $O(|\beta_l|)$, $O(n)$ oraz $O(|\beta_l|)$ czasu. Zatem, cała procedura ma złożoność obliczeniową $O(n)$. Wykorzystując oryginalnie zaproponowane, opisane poniżej własności można jednak pokazać, że w celu wyznaczenia

pozycji z^* nie jest konieczne wyznaczanie pozycji e, f, a zatem i zbiorów $ZB_j(\beta)$, $ZA_k(\beta)$. Wiąże się to z dalszą redukcją praktycznej złożoności obliczeniowej procedury. W tym celu należy zauważyć, że prawdziwość nierówności (3.61) oznacza prawdziwość poniższych nierówności

$$ed_z \geqslant ed_{e+1}, \qquad z = 1, \ldots, e, \tag{3.67}$$

$$ed_z \geqslant ed_f, \qquad z = f+1, \ldots, |\beta_l| + 1. \tag{3.68}$$

Istotną rolę w dalszych rozważaniach będą też pełniły dwie pozycje

$$a = \min\{1 \leqslant z \leqslant |\beta_l| + 1 : ed_z = d_{\min}\}, \tag{3.69}$$

$$b = \max\{1 \leqslant z \leqslant |\beta_l| + 1 : ed_z = d_{\min}\}, \tag{3.70}$$

gdzie wielkość $d_{\min}$ dana jest równaniem (3.47). Warto również zauważyć, że $d_{\min} = \min_{1 \leqslant z \leqslant |\beta_l|+1} ed_z$, gdzie wartości ed_z dane są równaniem (3.66). Z nierówności (3.67), (3.68) i definicji pozycji a, b dodatkowo wynika prawdziwość nierówności

$$a \leqslant f, \tag{3.71}$$

$$e < b. \tag{3.72}$$

Prawdziwa jest także następująca własność.

Własność 3.12 *Dany jest zbiór operacji uszeregowanych $U \subset O$, permutacja częściowa $\beta \in \Pi^U$, acykliczny graf $G(\beta)$, nieuszeregowana operacja transportowa $i \in O^t \setminus U$ i maszyna $l = m_i$. Niech*

$$c = \max\{1 \leqslant z \leqslant |\beta_l| + 1 : er_{z-1} + ep_{z-1} \leqslant r_i^{\beta}\}. \tag{3.73}$$

Wtedy $e < c \leqslant f$, czyli graf $G(\alpha^c)$ jest acykliczny.

Dowód. Z równania (3.50) wynika, że $\beta_l(e) \in ZB_i(\beta)$, co implikuje nierówność $er_e + ep_e \leqslant r_i^{\beta}$, która dowodzi, że $e < c$. Podobnie, z równania (3.51) wynika, że $\beta_l(z-1) \in ZA_i(\beta)$ dla $f \leqslant z - 1 \leqslant |\beta_l|$, co implikuje nierówności $er_{z-1} + ep_{z-1} \geqslant er_f + ep_f > er_f > r_i^{\beta}$, która dowodzi nierówności $c \leqslant f$ i kończy dowód ■

Z powyższych rozważań oraz nierówności (3.67), (3.68) wynika, że prawdziwe jest następujące twierdzenie, które stanowi podstawę przy formułowaniu ostatecznej postaci procedury wyznaczającej najlepsze rozwiązanie w zbiorze $Z(\beta, i)$.

Procedura rozpoczyna działanie z ustaloną dopuszczalną permutacją częściową β, nie uszeregowaną operacją transportową i oraz maszyną l. Procedura zwraca najlepszą pozycję z^* dla wstawianej operacji i.

1. wyznacz wartości r_i^β, q_i^β oraz wartości er_z, qr_z, $1 \leqslant z \leqslant |\beta_l|$,
2. wyznacz wartości ed_z, $1 \leqslant z \leqslant |\beta_l| + 1$, pozycje a, b, c, dane odpowiednimi równaniami (3.69), (3.70), (3.73) oraz określ pozycję z^* wykorzystując równanie (3.74). Zwróć pozycję z^* i STOP.

Rysunek 3.7: Schemat procedury $PWNP1$

Twierdzenie 3.2 *Dany jest zbiór operacji uszeregowanych $U \subset O$, permutacja częściowa $\beta \in \Pi^U$, acykliczny graf $G(\beta)$, nieuszeregowana operacja transportowa $i \in O^t \setminus U$ i maszyna $l = m_i$. Niech*

$$z^* = \begin{cases} \min\{c \leqslant z \leqslant b : ed_z = d_{\min}\}, & a < c < b, \\ b, & b \leqslant c, \\ a, & c \leqslant a. \end{cases} \tag{3.74}$$

Wtedy graf $G(\alpha^{z^*})$, $\alpha^{z^*} \in \{\alpha \in Z(\beta, i) : \alpha_l(z^*) = i\}$, jest acykliczny oraz $ed_{z^*} = d_{\min}$.

Dowód. Dowód sprowadza się do wykazania, że $e < z^* \leqslant f$; wtedy równość $ed_{z^*} = d_{\min}$ jest oczywista. Poniżej będą rozpatrywane następujące przypadki:

1. $a < c < b$,
2. $b \leqslant c$,
3. $c \leqslant a$.

Przypadek 1. Z równania (3.70) wynika, że $ed_f \geqslant ed_b = d_{\min}$. Jeżeli $ed_f > d_{\min}$, to z nierówności (3.68) wynika, że spełniona jest nierówność $b < f$. W tej sytuacji, z własności 3.12 i równania (3.74) wynika, że $e < c \leqslant z^* \leqslant b < f$. Natomiast, jeżeli $ed_f = d_{\min}$, to z nierówności (3.68) wynika, że spełniona jest nierówność $f \leqslant b$ i ostatecznie $e < c \leqslant z^* \leqslant f \leqslant b$, co kończy dowód tego przypadku.

Przypadek 2. Z równania (3.70) wynika, że $ed_{e+1} \geqslant ed_b = d_{\min}$. Z nierówności (3.72) wynika, że $e < b$. Z kolei, z własności 3.12 wynika, że $c \leqslant f$, zatem $e < z^* = b \leqslant c \leqslant f$, co kończy dowód tego przypadku.

Przypadek 3. Ze względu na symetrię do przypadku 2 dowód tego przypadku zostanie pominięty ■

Procedurę przyspieszonego wyznaczania najlepszej pozycji ($PWNP1$), wykorzystującą powyższe twierdzenie, przedstawiono na rys. 3.7. W celu wyznaczenia najlepszej pozycji dla operacji i, procedurę należy zastosować do permutacji β, operacji i oraz maszyny m_i. Krok 1 procedury wykonywany jest w czasie $O(n)$, krok 2 wymaga $O(|\beta_l| + 1)$ czasu. Wyznaczona pozycja z^* jest najlepszą pozycją dla wstawianej operacji i, zaś odpowiadające jej rozwiązanie α^{z^*} jest najlepszym rozwiązaniem w zbiorze $Z(\beta, i)$. Wartość $C_{\max}(\alpha^{z^*})$ można wyznaczyć z równania (3.46). Nie trudno zauważyć, że w powyższej procedurze – jak już podkreślano – nie ma potrzeby wyznaczania pozycji $e \in ZE$, $f \in ZF$ (wymagałoby to kolejnych $O(n)$ jednostek czasu), co znacznie przyspiesza ogólną metodę z pracy [65].

3.4 Własności przestrzeni rozwiązań

W ostatnich latach, w literaturze coraz więcej uwagi poświęca się badaniom własności przestrzeni rozwiązań różnych problemów kombinatorycznych. W badaniach tych można wyspecyfikować dwa kierunki. Pierwszy kierunek dotyczy badań mających na celu znalezienie ewentualnej, istotnie dodatniej korelacji pomiędzy odległościami dzielącymi poszczególne rozwiązania w przestrzeni (wykorzystując odpowiednie miary) a ich wartościami funkcji celu. Istnienie takiej korelacji dla danego problemu (świadczącej o obecności tzw. „Wielkiej Doliny") pozwala, między innymi, na modyfikację i rozbudowę istniejących algorytmów heurystycznych w celu zwiększenia jakości dostarczanych przez nie rozwiązań. Obecność wielkiej doliny została już wykazana dla takich problemów jak, np. problem komiwojażera [11], problem przepływowy [85], czy problem gniazdowy [73].

W większości problemów kombinatorycznych liczba wszystkich rozwiązań dowolnej jego instancji rośnie wykładniczo w zależności od jej rozmiaru. Jednak, tylko niewielką część rozwiązań stanowią rozwiązania dopuszczalne, czyli rozwiązania spełniające wszystkie narzucone ograniczenia technologiczne. Nie oznacza to jednak, że liczba tych ostatnich jest mała, ze względu na ogromną liczbę wszystkich rozwiązań. Jest rzeczą oczywistą, że analityczne próby oszacowania frakcji rozwiązań dopuszczalnych lub innych własności dotyczących „kształtu" przestrzeni rozwiązań dopuszczalnych są z góry skazane na niepowodzenie. Możliwe jest jednak przeprowadzenie badań statystycznych, które dostarczają pewnych parametrów, na których podstawie można wnioskować o kształcie i rozmiarze przestrzeni rozwiązań dopuszczalnych. Znajomość wspomnianych parametrów dla danej instancji pozwala również na określenie stopnia oraz przyczyn jej „trudności" i może się również okazać pomocna przy konstrukcji algorytmów heurystycznych.

Badania tego typu stanowią, wspomniany wcześniej, drugi kierunek badań przestrzeni rozwiązań.

3.4.1 Miary odległości

Istnieje wiele miar odległości pomiędzy rozwiązaniami reprezentowanymi przez permutacje. Wśród nich można wymienić miarę stopową (ang. *footrule*), stopień korelacji Spearmana (ang. *Spearman's rank correlation*), miarę Hamminga, Cayleya czy Ulama. Jedną z najbardziej interesujących, ze względu na związek z metodami lokalnego poszukiwania, jest miara tau Kendalla (ang. *Kendall's tau*). Odległość pomiędzy dwoma rozwiązaniami $\pi, \delta \in \Pi^0$ wyrażona w tej mierze jest równa minimalnej liczbie ruchów typu zamień sąsiednie operacje, jaką należy wykonać, by przejść z rozwiązania π do rozwiązania δ. Formalnie, miarę tą można zapisać w postaci

$$h(\pi, \delta) = \sum_{l \in M} |h_l(\pi_l, \delta_l)|, \tag{3.75}$$

gdzie

$$\begin{aligned} h_l(\pi_l, \delta_l) \quad = \quad & \{(\pi_l(j), \pi_l(k)) : \\ & \delta^{-1}(\pi_l(j)) > \delta^{-1}(\pi_l(k)),\ 1 \leqslant j < k \leqslant o_l\}, \end{aligned} \tag{3.76}$$

zaś symbol π^{-1} oznacza permutację odwrotną do permutacji π, tj. spełnia równanie $\pi_{m_i}(\pi^{-1}(i)) = i$ dla każdej operacji $i \in O$. Łatwo zauważyć, że odległość maksymalna jest równa

$$h_{\max} = \sum_{l \in M} \frac{o_l(o_l - 1)}{2}. \tag{3.77}$$

Rozkład odległości $h(\pi, \delta)$ dla ustalonej permutacji π i równomiernego rozkładu δ na przestrzeni rozwiązań Π^0 można wyznaczyć wykorzystując formułę rekurencyjną, opisaną w pracy [51]. Wartość średnia rozkładu odległości jest równa $h_{\max}/2$, zaś odchylenie standardowe wynosi

$$\sigma h = \sqrt{\frac{\sum_{l \in M}(o_l(o_l - 1))(2o_l + 5)}{72}}. \tag{3.78}$$

Poniżej proponuje się zupełnie inną miarę odległości, opracowaną przez autora tej rozprawy, bazującą na reprezentacji permutacji w n-wymiarowej przestrzeni Euklidesowej. Mianowicie, dla dowolnej permutacji $\pi \in \Pi$ można określić punkt $A = (a_1, a_2, \ldots, a_n) \in \mathbb{R}^n$ taki, że $a_{\phi(i)} = \pi^{-1}(i)$, $i \in O$,

gdzie $\phi : O \to \{1, 2, \ldots, n\}$ jest dowolną funkcją odwzorowującą zbiór operacji w podzbiór zbioru liczb naturalnych. Możliwe jest zatem zdefiniowanie miary odległości

$$eu(\pi, \delta) = ||AB|| = \sqrt{\sum_{i=1}^{n}(a_i - b_i)^2}, \tag{3.79}$$

pomiędzy permutacjami $\pi, \delta \in \Pi^0$, równej odległości Euklidesowej pomiędzy punktami $A = (a_1, a_2, \ldots, a_n)$, $B = (b_1, b_2, \ldots, b_n)$, reprezentującymi owe permutacje w przestrzeni $\mathbb{R}^n$.

Wiele własności nowo zaproponowanej miary pozostaje nieznane. Tym niemniej, faktem jest istnienie istotnej dodatniej korelacji pomiędzy, prezentowaną powyżej, miarą geometryczną i miarą tau Kendalla. Istnienie tej korelacji zostało wykazane doświadczalnie przy użyciu 80 instancji testowych Taillarda [104], zaproponowanych dla klasycznego problemu gniazdowego. Dla każdej instancji zostało wygenerowane $g = 2000$ (dopuszczalnych) permutacji $\pi^1, \pi^2, \ldots, \pi^g$. Dla każdego zestawu par $(h(\pi^i, \pi^{i+g/2}), eu(\pi^i, \pi^{i+g/2}))$, $i = 1, 2, \ldots, g/2$, został wyznaczony współczynnik korelacji. Wartość tego współczynnika w każdym przypadku przekraczała 0.99. Można zatem się spodziewać, że wiele rezultatów prawdziwych dla miary tau Kendalla będzie również prawdziwe dla miary geometrycznej.

3.4.2 Wielka dolina

O obecności wielkiej doliny w krajobrazie przestrzeni rozwiązań dopuszczalnych świadczy istotna dodatnia korelacja pomiędzy odległościami dzielącymi poszczególne rozwiązania w przestrzeni a ich wartościami funkcji celu. W celu zbadania tej korelacji dla rozważanego problemu, przeanalizowano 30 instancji testowych[2], zaproponowanych w pracach [45, 50]. Instancje te zostały wygenerowane na bazie instancji FT6 (36 operacji, 6 zadań i 6 maszyn produkcyjnych) i FT10 (100 operacji, 10 zadań i 10 maszyn produkcyjnych), zaproponowanych w roku 1963 przez Fishera i Thompsona [62] dla klasycznego problemu gniazdowego; w obu tych instancjach każda maszyna produkcyjna wykonuje każde zadanie dokładnie raz. Każda z 30 instancji została utworzona poprzez dodanie jednej maszyny transportowej i różnych kombinacji czasów transportów i przejazdów pustych. Nazwa każdej instancji, która będzie przedstawiana w postaci $HKa/b/c/d$,

[2]Oryginalnie, instancje te zostały zaprojektowane dla problemu gniazdowego z transportem i robotem transportowym, jednakże, z matematycznego punktu widzenia problem ten jest szczególnym przypadkiem problemu rozważanego w tym rozdziale.

w pełni identyfikuje jej parametry. Zmienna $a \in \{1, 2\}$ określa numer bazowej instancji. Zmienna $0 < b < 1$ jest opcjonalnym współczynnikiem skalującym; jej obecność w nazwie instancji oznacza, że wszystkie wyjściowe czasy wykonywania p_k operacji $k \in O^p$, (z wyjątkiem ostatnich operacji poszczególnych zadań) zostały zastąpione przez wartości $\lceil b \cdot p_k \rceil$. Trzecia zmienna określa typ transportu i może przybierać jedną z czterech wartości. Jeśli $c = t_{jkl}$ lub $c = t_{kl}$, to czasy wykonywania operacji transportowych zostały wylosowane z przedziału $[1, 10]$, przy czym czasy są odpowiednio zależne lub niezależne od transportowanego zadania. Jeżeli $c = Di$, to czas transportu pomiędzy maszynami produkcyjnymi $l, l' \in M^p$ wynosi $i \cdot |l - l'|$, zaś gdy $c = Ti$, to czas ten nie zależy od maszyn i równa się i. Zmienna d określa sposób doboru czasów przejazdów pustych; d może być równe t'_{kl}, di lub ti, co oznacza, że czasy przejazdów pustych zostały dobrane w ten sam sposób jak czasy wykonywania operacji transportowych, oznaczone odpowiednio przez t_{kl}, Di lub Ti.

Po zastosowaniu podejścia analogicznego jak w pracy [73], dla każdej z 30 instancji zostało wygenerowane $g+1 = 201$ rozwiązań (dopuszczalnych) lokalnie optymalnych, oznaczonych przez $\pi^0, \pi^1, \ldots \pi^g$. Niech π^0 oznacza najlepsze rozwiązanie spośród wszystkich $g + 1$ rozwiązań, które będzie dalej nazywane rozwiązaniem referencyjnym. Następnie zostały wyznaczone znormalizowane odległości

$$H(\pi^i, \pi^j) = 100\% \cdot \frac{h(\pi^i, \pi^j)}{h_{\max}}, \qquad i, j = 0, \ldots, g, \tag{3.80}$$

pomiędzy rozwiązaniami, odległości średnie

$$H_{av}(\pi^i) = \sum_{j=0, j \neq i}^{g} \frac{H(\pi^i, \pi^j)}{g}, \qquad i = 0, \ldots, g, \tag{3.81}$$

i różnice wartości funkcji celu

$$\Delta C^i = C_{\max}(\pi^i) - C_{\max}(\pi^0), \qquad i = 0, \ldots, g. \tag{3.82}$$

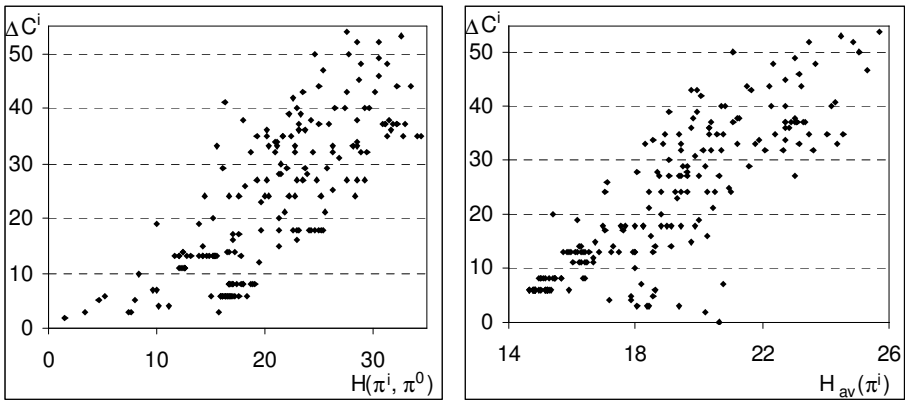
Dla zestawu par $(H(\pi^i, \pi^0), \Delta C^i)$, $i = 1, \ldots, g$ oraz $(H_{av}(\pi^i), \Delta C^i)$, $i = 0, \ldots, g$, zostały wyznaczone współczynniki korelacji, oznaczone odpowiednio ρ_{min}, ρ_{av}, które zostały zamieszczone w tabeli 3.4. Dodatkowo, dwa przykładowe zestawy par $(H(\pi^i, \pi^0), \Delta C^i)$ i $(H_{av}(\pi^i), \Delta C^i)$ dla instancji o numerze 27 zostały przedstawione na rysunku 3.8. Dla każdego ze współczynników korelacji przeprowadzono test istotności [58], na którego podstawie można stwierdzić, że z wyjątkiem współczynnika ρ_{min} dla instancji 2 i ρ_{av} dla instancji 16, wszystkie współczynniki korelacji są statystycznie istotne na poziomie $0,1\%$. W tabeli 3.4 zostały również zamieszczone

Tabela 3.4: Współczynniki korelacji i inne parametry dla poszczególnych instancji

Instancja		$C_{\max}$	Korelacja		$H(\pi^i, \pi^0)$		$H(\pi^i, \pi^j)$	
Nr	Nazwa	(π^0)	ρ_{min}	ρ_{av}	max	av	max	av
1	$HK1\_t_{jkl}\_t'_{kl}.1$	134	0,33	0,34	26,1	14,4	34,9	15,8
2	$HK1\_t_{jkl}\_t'_{kl}.2$	129	0,09	0,55	24,6	17,0	32,0	14,1
3	$HK1\_t_{jkl}\_t'_{kl}.3$	143	0,56	0,65	29,1	14,6	36,4	16,6
4	$HK1\_D1\_d1$	87	0,57	0,58	29,7	15,3	39,6	15,8
5	$HK1\_D1\_t1$	80	0,50	0,56	17,7	9,3	23,8	10,8
6	$HK1\_D2\_d1$	148	0,46	0,57	28,2	16,4	39,0	17,8
7	$HK1\_D3\_d1$	214	0,47	0,27	32,0	21,0	40,8	20,8
8	$HK1\_t_{kl}\_t'_{kl}.1$	136	0,72	0,56	30,1	16,3	32,6	17,2
9	$HK1\_T2\_t1$	74	0,32	0,56	17,9	11,1	28,6	9,5
10	$HK1\_T3\_t0$	92	0,36	0,48	21,7	11,9	26,5	12,8
Średnia			0,44	0,51	25,7	14,7	33,4	15,1
11	$HK2\_D1\_d1$	957	0,57	0,72	30,9	17,5	36,3	19,3
12	$HK2\_D1\_t0$	955	0,61	0,66	35,1	19,4	36,7	20,3
13	$HK2\_D1\_t1$	960	0,66	0,81	29,0	16,4	36,9	19,1
14	$HK2\_D2\_d1$	983	0,74	0,65	30,1	17,1	34,5	18,8
15	$HK2\_D3\_d1$	1026	0,65	0,57	29,6	15,5	31,3	17,4
16	$HK2\_D5\_t2$	1292	0,55	0,16	26,5	16,1	32,9	16,7
17	$HK2\_T1\_t1$	945	0,50	0,74	33,7	21,8	38,8	19,8
18	$HK2\_T2\_t1$	941	0,72	0,77	34,9	20,0	38,9	19,7
19	$HK2\_T5\_t2$	974	0,81	0,76	31,3	16,8	34,5	19,2
20	$HK2\_t_{jkl}\_t'_{kl}.1$	957	0,71	0,77	33,8	19,5	37,3	19,5
21	$HK2\_t_{jkl}\_t'_{kl}.2$	962	0,55	0,72	32,8	20,2	36,8	19,2
22	$HK2\_t_{jkl}\_t'_{kl}.3$	955	0,70	0,66	31,5	19,6	37,1	20,3
23	$HK2\_t_{jkl}\_t'_{kl}.4$	961	0,71	0,70	32,6	20,0	35,7	19,8
24	$HK2\_t_{kl}\_t'_{kl}.1$	965	0,55	0,68	31,8	21,4	35,0	19,8
25	$HK2\_t_{kl}\_t'_{kl}.2$	972	0,65	0,67	32,8	17,1	35,4	19,8
26	$HK2f0.5\_D1\_d1$	536	0,74	0,71	28,5	14,9	33,4	18,1
27	$HK2f0.5\_D1\_t1$	520	0,78	0,84	34,4	20,6	36,6	19,1
28	$HK2f0.5\_D2\_d1$	625	0,44	0,69	23,7	14,8	29,7	16,3
29	$HK2f0.5\_D2\_t0$	557	0,83	0,79	30,8	14,8	34,2	17,5
30	$HK2f0.5\_D2\_t1$	574	0,77	0,64	25,1	14,3	31,2	16,7
Średnia			0,66	0,68	31,0	17,9	35,2	18,8

wartości maksymalne (kolumny max) oraz średnie (kolumny av) zbiorów wartości $H(\pi^i, \pi^0)$, $H(\pi^i, \pi^j)$.

Z analizy tabeli 3.4 (i rysunku 3.8) wynika, że krajobraz przestrzeni rozwiązań cechuje istotnie dodatnia korelacja pomiędzy odległościami dzielą-



Rysunek 3.8: Wykres par $(H(\pi^i, \pi^0), \Delta C^i)$ oraz $(H_{av}(\pi^i), \Delta C^i)$ dla instancji testowej nr 27

cymi poszczególne rozwiązania lokalnie optymalne a ich wartościami funkcji celu. Dodatkowo, po analizie wartości maksymalnych i średnich odległości $(H(\pi^i, \pi^0)$, $H_{av}(\pi^i)$ można wysnuć wniosek, że rozwiązanie referencyjne znajduje się w centrum wszystkich pozostałych g rozwiązań.

3.4.3 Inne własności przestrzeni rozwiązań dopuszczalnych

Poniżej przedstawiono wyniki badań mających na celu dostarczenie dodatkowych parametrów charakteryzujących przestrzeń rozwiązań dopuszczalnych. W tym celu dla każdej z 30 instancji zostało wygenerowane $g = 5000$ losowych rozwiązań dopuszczalnych $\gamma^1, \gamma^2, \ldots, \gamma^g$. Następnie, podobnie jak w sekcji 3.4.1, dla każdej pary rozwiązań wyznaczono znormalizowaną odległość $H(\gamma^i, \gamma^j)$, $i, j = 1, \ldots, h$. Wartość maksymalną $MaxH$, średnią AvH) i odchylenie standardowe σH zbioru tych odległości przedstawiono w tabeli 3.5. Analizie poddano również wartości funkcji celu poszczególnych rozwiązań, wyznaczając dla każdego z nich błąd względny

$$RE(\gamma^i, \pi^0) = 100\% \cdot \frac{C_{\max}(\gamma^i) - C_{\max}(\pi^0)}{C_{\max}(\pi^0)}, \qquad i = 1, \ldots, g, \qquad (3.83)$$

gdzie π^0 jest rozwiązaniem referencyjnym. Wartość maksymalna $MaxRE$, średnia $AvRE$ i minimalna $MinRE$ spośród tych wartości dla każdej instancji została przedstawiona w tabeli 3.5.

Po analizie tabeli 3.5 nasuwa się kilka wniosków. Po pierwsze, średnia odległość $AvRE$ różni się bardzo niewiele pomiędzy poszczególnymi instancjami. Dotyczy to nawet instancji o różnych rozmiarach (instancje

Tabela 3.5: Odległości i błędy względne dopuszczalnych permutacji losowych

Instancja		$H(\gamma^i, \gamma^j)$			$RE(\gamma^i, \gamma^0)$		
Nr	Nazwa	$MaxH$	AvH	σH	$MaxRE$	$AvRE$	$MinRE$
1	$HK1\_t_{jkl}\_t'_{kl}.1$	61,1	22,4	6,4	38,1	13,8	1,5
2	$HK1\_t_{jkl}\_t'_{kl}.2$	56,4	22,0	6,7	36,4	13,6	0,8
3	$HK1\_t_{jkl}\_t'_{kl}.3$	56,8	22,8	6,8	32,9	11,9	0,0
4	$HK1\_D1\_d1$	53,1	20,8	7,4	56,3	20,8	2,3
5	$HK1\_D1\_t1$	51,2	20,2	6,0	61,3	23,4	5,0
6	$HK1\_D2\_d1$	53,5	21,9	6,9	33,8	13,3	2,0
7	$HK1\_D3\_d1$	54,3	23,5	7,5	22,4	8,3	1,4
8	$HK1\_t_{kl}\_t'_{kl}.1$	44,4	18,9	5,9	27,2	8,6	0,0
9	$HK1\_T2\_t1$	50,5	19,5	5,9	68,9	23,5	1,4
10	$HK1\_T3\_t0$	47,8	19,7	5,3	47,8	14,2	0,0
Średnia		52,9	21,2	6,5	42,5	15,1	1,4
11	$HK2\_D1\_d1$	44,1	23,0	4,8	64,2	31,5	8,2
12	$HK2\_D1\_t0$	44,5	23,0	4,8	68,3	30,9	11,0
13	$HK2\_D1\_t1$	46,3	22,9	4,8	68,1	30,4	9,2
14	$HK2\_D2\_d1$	44,6	22,5	4,7	61,9	32,0	12,1
15	$HK2\_D3\_d1$	43,9	22,4	4,8	59,3	32,5	12,9
16	$HK2\_D5\_t2$	42,1	21,2	4,4	44,6	22,0	8,4
17	$HK2\_T1\_t1$	46,2	23,3	4,9	61,2	30,8	11,3
18	$HK2\_T2\_t1$	44,9	23,2	4,9	63,4	32,3	12,1
19	$HK2\_T5\_t2$	42,5	22,1	4,7	60,9	31,2	11,3
20	$HK2\_t_{jkl}\_t'_{kl}.1$	43,8	22,6	4,8	65,2	32,1	9,4
21	$HK2\_t_{jkl}\_t'_{kl}.2$	44,1	22,5	4,8	64,4	30,9	8,6
22	$HK2\_t_{jkl}\_t'_{kl}.3$	45,4	22,7	4,8	58,7	31,9	9,9
23	$HK2\_t_{jkl}\_t'_{kl}.4$	44,9	22,5	4,7	63,1	31,7	10,9
24	$HK2\_t_{kl}\_t'_{kl}.1$	43,5	22,5	4,8	62,8	31,2	11,7
25	$HK2\_t_{kl}\_t'_{kl}.2$	42,9	22,2	4,7	59,7	30,7	11,7
26	$HK2f0.5\_D1\_d1$	45,9	23,1	4,9	65,7	37,0	14,4
27	$HK2f0.5\_D1\_t1$	43,7	22,3	4,7	67,3	36,8	11,7
28	$HK2f0.5\_D2\_d1$	48,0	24,4	5,4	62,1	34,3	10,6
29	$HK2f0.5\_D2\_t0$	41,2	20,7	4,4	70,2	35,9	14,0
30	$HK2f0.5\_D2\_t1$	40,7	21,1	4,4	64,8	35,2	16,9
Średnia		44,2	22,5	4,8	62,8	32,1	11,3

1-10 w porównaniu do instancji 11-30). Odległość ta (na poziomie 22%) jest istotnie mniejsza od średniej odległości wszystkich rozwiązań (niekoniecznie dopuszczalnych), która jest równa 50%. Po drugie, średni błąd względny jest znacznie mniejszy dla instancji małych 1-10, niż dla instancji

większych 11-30. Ten sam trend mają minimalne i maksymalne wartości błędu względnego. Po trzecie, wśród instancji o małym rozmiarze znajdują się dwie instancje wyjątkowo łatwe, dotyczy to instancji 7 i 8 ze średnim błędem względnym na poziomie $8,5\%$.

Wyniki z tabeli 3.5 mogą być także pośrednio wykorzystane do bardziej zaawansowanej analizy przestrzeni rozwiązań. Przykładowo, czyniąc założenie, iż rozkład odległości $H(\gamma^i, \gamma^j)$ jest aproksymowalny krzywą Gaussa, można wykorzystać wartości średnie AvH i odchylenia standardowe σH (przy dodatkowej znajomości teoretycznego rozkładu odległości [51] i wartości teoretycznych $h_{\max}$, σh z sekcji 3.4.1) do oszacowania górnego ograniczenia frakcji rozwiązań dopuszczalnych. Ponadto, informacje z tabeli 3.5 mogą być również wykorzystane jako elementy wskaźników „trudności" poszczególnych instancji.

3.5 Wnioski i uwagi

W tym rozdziale został rozważony elastyczny system produkcyjny, w którym transporty międzystanowiskowe były realizowane przy użyciu zbioru dedykowanych wózków AGV. W modelu matematycznym problemu uwzględniono szczególnie praktyczny tryb pracy wózków „zabierz i zostaw" oraz maszynowo- i zadaniowo-zależne czasy wykonania operacji transportowych. Za kryterium optymalizacji przyjęto moment zakończenia wykonywania wszystkich zadań. Posługując się reprezentacją permutacyjno-grafową zaproponowano szereg ogólnych i specyficznych własności problemu. Własności wykryte dla tzw. rozwiązania i grafu częściowego przyczyniły się do powstania tzw. „efektywnej metody wyznaczania najlepszej pozycji". Metoda ta znajduje zastosowanie w algorytmach konstrukcyjnych wykorzystujących tzw. „technikę wstawień" oraz w efektywnych metodach przeglądu sąsiedztwa generowanego w oparciu o ruchy typu wstaw.

W rozdziale zaproponowano też sąsiedztwo generowane w oparciu o zbiór ruchów typu zamień sąsiednie operacje. Przedstawiony zbiór ruchów posiada wiele cech analogicznych do niektórych zbiorów wykorzystywanych w algorytmach rozwiązywania klasycznego problemu gniazdowego. Po pierwsze, każdy ruch nie należący do zbioru ruchów prowadzi albo do rozwiązania niedopuszczalnego, albo nie może spowodować natychmiastowej poprawy wartości funkcji celu. Po drugie, każde rozwiązanie, na podstawie którego został wygenerowany pusty zbiór ruchów, jest rozwiązaniem optymalnym. Dla zaproponowanego sąsiedztwa przedstawiono też efektywną metodę jego przeglądu w oparciu o własności analogiczne do własności znanych dla klasycznego problemu gniazdowego.

W niniejszym rozdziale wykazano też istotną dodatnią korelację pomiędzy wartością funkcji celu rozwiązań a ich odległością wyrażoną w mierze tau Kendalla. Fakt ten niezbicie świadczy o obecności wielkiej doliny w krajobrazie przestrzeni rozwiązań problemu. Zaproponowano również zupełnie nową miarę odległości. Odległość dzieląca dwa rozwiązania w tej mierze jest równa długości odcinka łączącego dwa punkty reprezentujące owe rozwiązania w n-wymiarowej przestrzeni Euklidesowej. Wykazano istotną dodatnią korelację pomiędzy odległościami wyrażonymi w mierze geometrycznej i mierze tau Kendalla. Nowo zaproponowana miara odległości przyczyniła się do powstania nowego quasi-operatora krzyżowania, opisanego w następnym rozdziale.

Część własności i technik prezentowanych w tym rozdziale można stosunkowo łatwo poddać modyfikacjom i zastosować w przypadku innych uogólnień klasycznego problemu gniazdowego. Na przykład, dla problemu gniazdowego z przezbrojeniami o dowolnym czasie trwania prawdziwa będzie własność 3.5 prezentowana w sekcji 3.2. Podobnie ma się rzecz z efektywną metodą przeglądu sąsiedztwa (sekcja 3.2.1). Jednakże, bezpośrednie zastosowanie efektywnej metody wyznaczania najlepszej pozycji dla wstawianej operacji (sekcja 3.3.1) dla wspomnianego problemu z przezbrojeniami zakończy się połowicznym sukcesem. Wstawienie operacji na pozycję wyznaczoną w wyniku działania procedury gwarantuje jedynie acykliczność skonstruowanego w ten sposób grafu.

Rozdział 4

Algorytmy rozwiązywania problemu gniazdowego z ustalonym przydziałem wózków AGV

Ogólnie, algorytmy przybliżone można podzielić na dwie klasy. W klasie pierwszej znajdują się tzw. algorytmy konstrukcyjne, o dużej szybkości działania, ale dające rozwiązania stosunkowo niskiej jakości. Rozwiązania generowane przez algorytmy konstrukcyjne są często traktowane jako początkowe dla drugiej klasy algorytmów – algorytmów popraw, które „starają się" poprawić dane rozwiązanie. Ich zaletą jest to, że produkują z reguły istotnie lepsze rozwiązania niż algorytmy konstrukcyjne. Możliwość zakończenia działania w praktycznie dowolnym momencie, poprzez odpowiedni dobór jednego z kryteriów stopu, jest jeszcze jedną ich pozytywną cechą.

Poniżej prezentuje się szereg algorytmów rozwiązywania problemu $J, DR|t_{jkl}, t'_{kl}|C_{\max}$, omawianego w rozdziale 3. Zarówno algorytmy konstrukcyjne jak i algorytmy popraw zostały zrealizowane przy użyciu zupełnie odmiennych technik. Wśród algorytmów konstrukcyjnych można wymienić algorytmy priorytetowe i algorytmy wykorzystujące tzw. „technikę wstawień". Algorytmy popraw zostały zrealizowane przy użyciu różnych technik lokalnych poszukiwań, takich jak technika poszukiwań z zabronieniami, poszukiwania rozproszonego, symulowane wyżarzanie oraz podejście genetyczne. Pod koniec rozdziału prezentuje się wyniki badań numerycznych, przeprowadzonych przy użyciu szeregu własnych, jak i literaturowych instancji testowych. Dostarczone wyniki wykorzystuje się w celu wyłonienia algorytmów najlepszych.

4.1 Algorytmy konstrukcyjne

Opisane poniżej algorytmy konstrukcyjne stanowią pewną modyfikację algorytmów znanych dla klasycznego problemu gniazdowego. Prezentowane są dwa typy algorytmów: algorytmy priorytetowe oraz algorytmy typu wstaw, bazujące na popularnym algorytmie $INSA$ (patrz. np. [36,65,113]). Algorytmy priorytetowe, w przypadku omawianego problemu, są algorytmami jednofazowymi; każda operacja wybierana jest według pewnego priorytetu i szeregowana za ostatnią dotychczas uszeregowaną operacją na jednej z maszyn. Zaletą tych algorytmów jest bardzo duża szybkość działania, natomiast wadą niska jakość generowanych rozwiązań. Z kolei, algorytmy typu wstaw są algorytmami dwufazowymi. W fazie pierwszej tworzona jest lista operacji, w której wszystkie operacje zostają posortowane według pewnych priorytetów. Następnie, w każdej iteracji fazy drugiej, kolejna operacja z listy wstawiana jest próbnie na wszystkie możliwie pozycje w uszeregowaniu zbudowanym w poprzednich iteracjach. Ostatecznie, operację wstawia się na najlepszą pozycję, wybraną na podstawie wartości pewnej pomocniczej funkcji, wyliczanej przy każdym wstawieniu.

4.1.1 Algorytmy priorytetowe

Podstawowe modyfikacje algorytmów priorytetowych dla klasycznych problemów gniazdowych, pozwalające na ich zastosowanie do konstrukcji rozwiązań badanego zagadnienia, będą nazywane algorytmami $AP(R)$; R jest tu regułą priorytetową.

Algorytm $AP(R)$ wykonuje n iteracji. W każdej iteracji określa się zbiór operacji gotowych do uszeregowania

$$OG = \{i \in O \setminus U : ZB_i(\beta) \subseteq U\}, \tag{4.1}$$

tzn. dla danej dopuszczalnej permutacji częściowej $\beta \in \Pi^U$ oraz grafu częściowego $G(\beta)$ wyznacza się zbiór operacji i, które nie zostały jeszcze uszeregowane, ale wszystkie ich poprzedniki (ze zbioru $ZB_i(\beta)$) są już uszeregowane; zbiór $U \subset O$ jest zbiorem operacji uszeregowanych. Następnie wyznaczana jest chwila czasowa

$$\Delta = \min_{j \in OG} r(j), \tag{4.2}$$

oznaczająca najwcześniejszą możliwą chwilę rozpoczęcia wykonywania przynajmniej jednej operacji gotowej do uszeregowania, gdzie wielkość $r(j)$ dana jest równaniem

$$r(j) = \max\{\max_{i \in ZB_j(\beta)}(S_i + p_i), t_{m_j} + s(\beta_{m_j}(|\beta_{m_j}|), j)\}. \tag{4.3}$$

Wielkość t_{m_j} oznacza tutaj chwilę zakończenia wykonywania ostatniej operacji uszeregowanej na maszynie m_j. Po wyznaczeniu momentu czasowego Δ wyznaczany jest zbiór maszyn

$$M' = \{l \in M : \exists_{j \in OG}\, m_j = l, r(j) = \Delta\}, \tag{4.4}$$

na których mogą być wykonywane operacje ze zbioru OG, których najwcześniejszy możliwy moment rozpoczęcia jest równy Δ. Ze zbioru M' wybiera się losowo jedną maszynę $l \in M'$ oraz określa się tzw. zbiór operacji konfliktowych

$$OK_l = \{j \in OG : m_j = l, r(j) = \Delta\}. \tag{4.5}$$

Ostatecznie, stosując regułę priorytetową R wybiera się operację $j \in OK_l$ i kładzie się $\beta_l := \beta_l, j$, $U := U \cup \{j\}$, $S_j := r(j)$; $t_l := S_j + p_j$. Złożoność obliczeniowa algorytmów $AP(R)$ jest zależna od złożoności obliczeniowej zastosowanych reguł priorytetowych. Najczęściej stosowane reguły mają złożoność $O(n)$, zatem złożoność całego algorytmu z reguły wynosi $O(n^2)$.

W niektórych źródłach literaturowych niektóre z powyższych składników definiuje się nieco inaczej. W szczególności, wielkość Δ i zbiory M', OK_l mogą być zastąpione wielkością Δ' i odpowiednimi zbiorami M'', OK'_l, gdzie

$$\Delta' = \min_{j \in OG} r(j) + p_j, \tag{4.6}$$

$$M'' = \{l \in M : \exists_{j \in OG}\, m_j = l, r(j) + p_j = \Delta\}, \tag{4.7}$$

$$OK'_l = \{j \in OG : m_j = l, r(j) < \Delta\}. \tag{4.8}$$

W przypadku badanego zagadnienia takich modyfikacji jednak się nie wprowadza. Już we wstępnej fazie badań numerycznych okazuje się, że zastosowanie powyższych modyfikacji w algorytmach wiąże się z drastycznym spadkiem jakości generowanych rozwiązań.

Na koniec tej sekcji krótko zostaną omówione najczęściej stosowane reguły priorytetowe. Do najbardziej popularnych reguł należą reguły o nazwach STT, LTT, SPT, LPT, $SRPT$, $LRPT$, RAN. W myśl pierwszych dwóch reguł ze zbioru OK_l wybierana jest operacja z odpowiednio najkrótszym i najdłuższym czasem wykonania. W myśl kolejnych dwóch reguł ze zbioru OK_l wybierane są operacje zadania odpowiednio o najkrótszym i najdłuższym czasie wykonania. W myśl reguł o nazwie $SRPT$, $LRPT$ wyborowi ze zbioru operacji konfliktowych podlegają operacje zadania odpowiednio o najkrótszym i najdłuższym pozostałym czasie wykonania. Reguła RAN polega na wyborze operacji konfliktowej w sposób losowy.

Pewną szczególną regułą priorytetową jest reguła $R = PWK(\pi)$, $\pi \in \Pi^0$. Reguła ta została opracowana specjalnie na potrzeby algorytmów wykorzystywanych w algorytmach genetycznych, opisywanych w sekcji 4.2.3. Zgodnie z regułą $R = PWK(\pi)$, najwyższy priorytet ma operacja znajdująca się najbliżej początku permutacji π_l, $l \in M$. Bardziej precyzyjnie, niech OK_l będzie zbiorem operacji konfliktowych na maszynie $l \in M$. Wtedy, zgodnie z powyższą regułą, najwyższy priorytet w tym zbiorze przyjmuje operacja i należąca do zbioru $\{j \in OK_l : \pi^{-1}(j) = \min_{k \in OK_l} \pi^{-1}(k)\}$, gdzie π^{-1} jest permutacją odwrotną (definiowaną w sekcji 3.4.1).

4.1.2 Algorytm typu wstaw

Poniżej przedstawia się algorytm typu wstaw, zwany dalej algorytmem $INT3$, będący modyfikacją, znanego z literatury, algorytmu $INSA$. Algorytm ten – podobnie jak algorytm $INSA$ – jest algorytmem dwufazowym. W pierwszej fazie tworzona jest lista operacji ψ, w której wszystkie operacje posortowane są według nierosnących wartości priorytetów $\omega(\psi(i)) \geqslant \omega(\psi(i+1))$, $i = 1, \ldots, n-1$, gdzie priorytet jest równy czasowi wykonania operacji, $\omega(i) = p_i$, $i \in O$. W fazie drugiej algorytm wykonuje n iteracji. W k-tej iteracji z listy wybierana jest operacja $i = \psi(k)$ i próbnie wstawiana na każdą pozycję w permutacji bieżącej β_{m_i}; permutacja bieżąca β jest permutacją częściową utworzoną w iteracji $k-1$. Zbiór wszystkich powstałych w ten sposób permutacji określony jest symbolem $Z(\beta, i)$ i dany jest równaniem (3.43). Następnie, dla każdej permutacji $\alpha \in Z(\beta, i)$ oblicza się wartość d_i^α, daną odpowiednimi równaniami (3.44), (3.45), oraz wybiera się permutację najlepszą, tzn. taką permutację $\delta \in Z(\beta, i)$, że graf $G(\delta)$ jest acykliczny oraz $d_i^\delta = d_{\min}$, gdzie wielkość $d_{\min}$ dana jest równaniem (3.47). W przypadku operacji produkcyjnych, na mocy twierdzenia 3.10, najlepszą permutacją w zbiorze $Z(\beta, i)$ jest każda permutacja ze zbioru $ZD(\beta, i)$, danego równaniem (3.49). W przypadku operacji transportowych, najlepszą permutację odnajduje się za pomocą procedury przyspieszonego wyznaczania najlepszej pozycji dla wstawianej operacji i, przedstawionej na rys. 3.7. Najlepsza permutacja w zbiorze $Z(\beta, i)$ staje się permutacją bieżącą w $k+1$ iteracji algorytmu. Schemat algorytmu $INT3$ został szczegółowo przedstawiony na rys. 4.1.

4.2 Algorytmy lokalnych poszukiwań

NP-trudność problemu omawianego w tym rozdziale sprawia, że odnalezienie rozwiązania optymalnego nie jest możliwe w czasie wielomianowo-

1. Połóż $\beta^0 := \emptyset$. Utwórz listę ψ taką, że $\omega(\psi(i)) \geqslant \omega(\psi(i+1))$, $i = 1, \ldots, n-1$.
2. Dla każdego $k = 1, 2, \ldots, n$ wykonuj krok 3 i krok 4
3. Połóż $i := \psi(k)$, $l := m_i$, $\beta := \beta^{k-1}$.
4. Jeżeli $i \in O^p$ to wykonaj krok 5. W przeciwnym przypadku wykonaj krok 6.
5. Wyznacz wartości r_i^β, q_i^β w grafie $G(\beta)$. Określ zbiory permutacji $Z(\beta, i)$, $ZD(\beta, i)$, dane odpowiednimi równaniami (3.43), (3.49). Wybierz dowolną permutację $\delta \in ZD(\beta, i)$ i połóż $\beta^k := \delta$.
6. Określ zbiór permutacji $Z(\beta, i)$, dany równaniem (3.43). Stosując procedurę $PWNP1$, prezentowaną na rys. 3.7, do permutacji β, operacji i oraz maszyny l wyznacz pozycję z^*. Wybierz permutację $\delta \in Z(\beta, i)$ taką, że $\delta_l(z^*) = i$. Połóż $\beta^k := \delta$.

Rysunek 4.1: Schemat algorytmu $INT3$

zależnym od rozmiaru problemu. Dlatego, w przypadku problemów tego typu, ze względu na długi czas pracy, często rezygnuje się z algorytmów dokładnych na rzecz technik przybliżonych. W szczególności, wykorzystuje się algorytmy lokalnych poszukiwań, takich jak algorytmy poszukiwań z zabronieniami i poszukiwania rozproszonego, algorytmy symulowanego wyżarzania, czy algorytmy genetyczne. Przedstawiciele algorytmów każdej z wymienionych grup omówione są poniżej.

4.2.1 Algorytmy poszukiwań z zabronieniami i poszukiwania rozproszonego

Ogólny schemat algorytmu poszukiwań z zabronieniami (algorytmu tabu) został już przedstawiony w sekcji 2.2.1. Jednym z najlepszych algorytmów tego typu dla klasycznego problemu gniazdowego jest algorytm o nazwie $TSAB$, który został przedstawiony po raz pierwszy w pracy [70]. Algorytm ten, w stosunku do typowych algorytmów tabu, został wyposażony w dodatkowy mechanizm tzw. skoków powrotnych. Mechanizm ten umożliwia algorytmowi powrót do tych fragmentów przestrzeni rozwiązań, które wydają się szczególnie obiecujące. W skrócie, algorytm $TSAB$ można opisać następująco. Algorytm rozpoczyna działanie z dowolną permutacją $\pi \in \Pi$ (np. skonstruowaną przez algorytm typu wstaw) oraz po zainicjo-

waniu zmiennych $\pi^* := \pi$, $C^* := C_{\max}(\pi)$, przechowujących najlepszą permutację i najlepszą wartość funkcji celu znalezioną w trakcie poszukiwań. Algorytm wymaga również ustalenia wartości parametrów $maxiter$, max_t, $max\delta$ i $maxc$. Parametr $maxiter$ określa maksymalną liczbę iteracji bez poprawy wartości funkcji celu, max_t określa długość listy tabu, parametry $max\delta$ i $maxc$ są używane w detektorze cykli. Detektor cykli jest pewnym dodatkowym mechanizmem zabezpieczającym algorytm przed cyklicznym „odwiedzaniem" tych samych rozwiązań. W każdej iteracji algorytmu, przy użyciu procedury przeszukiwania sąsiedztwa NSP (od ang. *neighborhood searching procedure*), ze zbioru ruchów $AV^2(\pi)$ (nie zabronionych przez aktualną listę tabu) wybierany jest ruch v', rozwiązanie sąsiednie π' oraz modyfikowana jest lista tabu. Jeżeli w ciągu $maxiter$ iteracji wartość zmiennej C^* nie zostanie poprawiona, wykonywany jest skok powrotny do jednego z najlepszych rozwiązań odnalezionych w trakcie wcześniejszych iteracji algorytmu. Rozwiązania te są przechowywane na specjalnej liście L, przechowującej maksymalnie $maxl$ elementów, gdzie $maxl$ jest parametrem. Algorytm kończy swoje działanie, gdy lista L jest pusta.

W celu przybliżonego rozwiązania omawianego problemu gniazdowego z transportem, poniżej przedstawia się modyfikacje algorytmu $TSAB$, które zachowują wszystkie zalety oryginalnego algorytmu. Autor tej pracy zdecydował się ograniczyć prezentację algorytmów jedynie do tych składników, którymi należy zastąpić oryginalne składniki algorytmu $TSAB$ w celu utworzenia wspomnianych modyfikacji.

Omawiane poniżej algorytmy w trakcie swojej pracy wykorzystują sąsiedztwo $AN^3(\pi)$, $\pi \in \Pi$. Z wykorzystaniem notacji podobnej jak w pracy [70], opis algorytmów będzie rozpoczęty od koncepcji modyfikacji listy tabu. Lista tabu jest krótkoterminową pamięcią (zwykle o stałym rozmiarze), która zabezpiecza algorytm przed cyklicznym powrotem do rozwiązań przeglądniętych we wcześniejszych iteracjach algorytmu. Bardziej precyzyjnie, każdorazowo po zastosowaniu ruchu $v = (x, y) \in AV^3(\pi)$ do dowolnej permutacji $\pi \in \Pi$, na liście tabu zapamiętywany jest ruch odwrotny, notowany jako para $\overline{v} = (y, x)$. Każdy ruch na liście będzie nazywany ruchem zabronionym i nie może być zastosowany do żadnej permutacji.

Dana jest permutacja $\pi \in \Pi$. Każdy ruch $(x, y) \in AV^3(\pi)$ taki, że $m_x \in M^p$ i $m_x \in M^t$ będzie nazywany odpowiednio ruchem produkcyjnym i transportowym. Zatem, zbiór ruchów $AV^3(\pi)$ można podzielić na dwa rozłączne podzbiory: zbiór ruchów produkcyjnych $AV_p^3(\pi) = \{(x, y) \in AV^3(\pi) : m_x \in M^p\}$ oraz zbiór ruchów transportowych $AV_t^3(\pi) = \{(x, y) \in AV^3(\pi) : m_x \in \mathcal{M}^t\}$. Ponieważ moc zbioru $AV_t^3(\pi)$ jest zwykle znacznie większa od mocy zbioru $AV_p^3(\pi)$, dobrym pomy-

słem wydaje się użycie dwóch list tabu. Pierwsza, lista tabu produkcyjna $T^p = (T^p_1, T^p_2, \ldots, T^p_{maxp})$, o stałej długości $maxp$, nie zawiera ruchów transportowych, tzn. na liście mogą być zapamiętane ruchy produkcyjne i ruchy sztuczne. ruchem sztucznym będzie nazywany ruch $dv = (0, 0)$. Podobnie, lista tabu transportowa $T^t = (T^t_1, T^t_2, \ldots, T^t_{maxt})$, o stałej długości $maxt$, nie zawiera ruchów produkcyjnych. Dodanie ruchu $\overline{v}$, $v \in AV^3_p(\pi)$, notowane jako $T^p \oplus \overline{v}$, polega na położeniu $T^p_j := T^p_{j+1}$, $j = 1, \ldots, maxp - 1$ oraz $T^p_{maxp} := \overline{v}$. W ten sam sposób dodawany jest ruch $\overline{v}$, $v \in AV^3_t(\pi)$ do listy T^t. Czynność ta będzie notowana jako $T^t \oplus \overline{v}$. Jeżeli któraś z list tabu będzie pusta, tzn. będzie zawierać jedynie ruchy sztuczne, fakt ten będzie oznaczany odpowiednio przez $T^p = \emptyset$ lub $T^t = \emptyset$.

Główna idea zastosowania listy tabu o stałej długości polega na tym, że ruch zabroniony znajduje się na liście stałą liczbę iteracji, równą długości listy tabu. Aby idea ta była zachowana w przypadku dwóch list, za każdym razem gdy do jednej z list dodawany jest ruch odwrotny, do drugiej listy oddawany jest ruch sztuczny. Bardziej precyzyjnie, jeżeli kładzie się $T^p \oplus \overline{v}$, to kładzie się też $T^t \oplus dv$. Symetrycznie, jeżeli kładzie się $T^t \oplus \overline{v}$, to kładzie się też $T^p \oplus dv$.

Ruch zabroniony jest perspektywiczny, jeżeli spełnia przyjęte kryterium aspiracji, tzn. jeżeli prowadzi do rozwiązania lepszego od najlepszego rozwiązania znalezionego do tej pory w trakcie pojedynczego uruchomienia algorytmu. Zbiór wszystkich takich ruchów będzie oznaczany przez

$$ZP = \{v \in AV^3(\pi) \cap T : C_{\max}(\pi^v) < C^*\}, \tag{4.9}$$

gdzie $T = T^p \cup T^t$, zaś C^* jest wartością funkcji celu najlepszego znalezionego rozwiązania. Ostatecznie, zbiór ruchów niezabronionych i zabronionych perspektywicznych można zapisać następująco:

$$ZR = (AV^3(\pi) \setminus T) \cup ZP. \tag{4.10}$$

Najprostszy algorytm poszukiwań z zabronieniami powstaje poprzez iterowanie procedury przeszukiwania sąsiedztwa (PPS) przedstawionej na rys. 4.2. Jednakże, najlepsze rezultaty przynosi osadzenie procedury PPS w schemacie omawianego już algorytmu $TSAB$. Zatem, pierwszy z algorytmów rozwiązywania problemu omawianego w tym rozdziale, nazwany algorytmem $TSAB_{AGV}$, uzyskuje się poprzez zastąpienie oryginalnego sąsiedztwa $AN^2(\pi)$ sąsiedztwem $AN^3(\pi)$ i oryginalnej procedury NSP procedurą PPS. Algorytm $TSAB_{AGV}$ wykorzystujący „klasyczne" sąsiedztwo $AN^2(\pi)$ będzie nazywany algorytmem $TSAB'_{AGV}$. Podobnie jak oryginalny algorytm, algorytmy $TSAB_{AGV}$, $TSAB'_{AGV}$ wymagają określenia wartości parametrów $maxiter$, $maxl$, $max\delta$, $maxc$ (omawianych na początku tej

Procedura rozpoczyna działanie z permutacją π, aktualną listą tabu $T = T^p \cup T^t$, niepustym zbiorem ruchów $AV^3(\pi)$ i najlepszą znalezioną wartością funkcji celu C^*. Procedura zwraca ruch v', zmodyfikowaną listę tabu T' i nową permutację π'.

1. Dla każdego ruchu $v \in AV^3(\pi)$ wyznacz i zapamiętaj wartość $C_{\max}(\pi^v)$ używając równania (3.40).
2. Wyznacz zbiór ZR, dany równaniem (4.10).
3. Jeżeli $ZR \neq \emptyset$, to wyznacz ruch $v' \in ZR$ taki, że $C_{\max}(\pi^{v'}) = \min\{C_{\max}(\pi^v) : v \in ZR\}$ i idź do kroku 12.
4. Jeżeli $|AV^3(\pi)| = 1$, to wybierz ruch $v' \in AV^3(\pi)$ i idź do kroku 12.
5. Jeżeli $T^p \neq \emptyset$, to znajdź $i = \min\{1 \leqslant j \leqslant maxp : T^p_j \neq dv\}$. W przeciwnym przypadku idź do kroku 8.
6. Jeżeli $i \neq maxp$, to powtarzaj $T^p := T^p \oplus T^p_{maxp}$ dopóki $AV^3_p(\pi) \setminus T^p = \emptyset$ i idź do kroku 11.
7. Powtarzaj $T^p := T^p \oplus dv$ dopóki $AV^3_p(\pi) \setminus T^p = \emptyset$ i idź do kroku 11.
8. Znajdź $i = \min\{1 \leqslant j \leqslant maxt : T^t_j \neq dv\}$.
9. Jeżeli $i \neq maxt$, to powtarzaj $T^t := T^t \oplus T^t_{maxt}$ dopóki $AV^3_t(\pi) \setminus T^t = \emptyset$ i idź do kroku 11.
10. Powtarzaj $T^t := T^t \oplus dv$ dopóki $AV^3_t(\pi) \setminus T^t = \emptyset$.
11. Wybierz ruch $v' \in AV^3(\pi) \setminus \{T^p \cup T^t\}$.
12. Jeżeli $v' \in AV^3_p(\pi)$, to połóż $T^p := T^p \oplus \overline{v'}$ oraz $T^t := T^t \oplus dv$. W przeciwnym przypadku połóż $T^t := T^t \oplus \overline{v'}$ oraz $T^p := T^p \oplus dv$. Połóż $\pi' := \pi^{v'}$ oraz $T' := T^p \cup T^t$.

Rysunek 4.2: Schemat procedury PPS

sekcji); oryginalny parametr określający długość listy tabu został zastąpiony parametrami $maxp$ i $maxt$, omawianymi powyżej.

W celu przybliżonego rozwiązania problemu gniazdowego z transportem, modyfikacjom został również poddany algorytm poszukiwania rozproszonego (ogólny schemat algorytmów tego typu został omówiony w sek-

cji 2.2.4) o nazwie *i–TSAB*. Oryginalnie, algorytm ten został zaprojektowany dla klasycznego problemu gniazdowego i opisany w pracy [72]. Poniżej zamieszczony jest krótki opis jego działania. Algorytm wykorzystuje procedurę generowania nowych rozwiązań początkowych NIS (od ang. *new initial solution generator*). Procedura rozpoczyna działanie z dwoma permutacjami α, β. Procedura zwraca permutację ϕ, której odległość w mierze tau Kendalla (omawianej w sekcji 3.4.1) od permutacji α i β wynosi odpowiednio $h(\alpha,\beta)\cdot maxV\%$ i $h(\alpha,\beta)\cdot(100\%-maxV\%)$, gdzie $maxV$ jest parametrem.

Algorytm *i–TSAB* jest algorytmem dwufazowym. W fazie wstępnej algorytm rozpoczyna działanie z dowolną permutacją początkową $\pi^0 \in \Pi$. Przy użyciu algorytmu $TSAB$, później procedury NIS generowane jest $maxE$ rozwiązań elitarnych, zapamiętywanych na liście, gdzie $maxE$ jest parametrem. W każdej iteracji fazy zasadniczej z listy rozwiązań elitarnych wybierane jest rozwiązanie najlepsze π^* oraz – najbardziej od niego oddalone w sensie miary tau Kendalla – rozwiązanie π^d. Następnie, w efekcie użycia procedury NIS, później algorytmu $TSAB$, generowane jest rozwiązanie π', które zastępuje rozwiązanie π^d na liście rozwiązań elitarnych. Algorytm *i–TSAB* kończy działanie, jeśli odległość pomiędzy rozwiązaniem π^* i π^d jest mniejsza niż $maxD$, gdzie $maxD$ jest parametrem.

Modyfikacja algorytmu *i–TSAB*, nazwana algorytmem $i\text{–}TSAB_{AGV}$, polega na zastąpieniu algorytmu $TSAB$ w fazie wstępnej oraz zasadniczej odpowiednio algorytmem $TSAB'_{AGV}$ i $TSAB_{AGV}$. Ponieważ czas pracy algorytmu $i\text{–}TSAB_{AGV}$ silnie zależy od poszczególnych instancji problemu, poza parametrami $maxV$, $maxE$, $maxD$ stosuje się również parametry $maxR$ oraz $maxT$, stanowiące dodatkowe kryteria stopu. Pierwsze z kryteriów określa maksymalną liczbę wywołań algorytmów $TSAB_{AGV}$, $TSAB'_{AGV}$ bez poprawy najlepszego rozwiązania π^*, zaś drugie – maksymalny czas pracy algorytmu $i\text{–}TSAB_{AGV}$.

Najbardziej czasochłonnym krokiem każdej iteracji wszystkich algorytmów jest przegląd sąsiedztwa dokonywany przez procedurę PPS. Dzięki zastosowaniu w niej własności 3.6-3.8 i twierdzenia 3.1, omawianych w sekcji 3.2.1, złożoność obliczeniowa procedury PPS (i tym samym poszczególnych iteracji wszystkich algorytmów) wynosi $O(|AN^i(\pi)|\cdot n\log m)$, $i\in\{2,3\}$, przy założeniu, że $n_k = m$, $k\in J$, $o_l = n$, $l\in M$, $n \geqslant m$.

4.2.2 Algorytm symulowanego wyżarzania

Algorytmy symulowanego wyżarzania zostały ogólnie omówione w sekcji 2.2.2. Algorytm symulowanego wyżarzania prezentowany w tej sekcji (zwany dalej algorytmem SW) w skrócie można opisać następująco. Algo-

rytm rozpoczyna działanie z dowolną permutacją początkową $\pi \in \Pi$. Praca algorytmu przebiega dwufazowo. W pierwszej fazie, zwanej strojeniem, określa się parametry mające wpływ na czas i jakość pracy algorytmu, takie jak *temperatura początkowa*, *temperatura końcowa* oraz *średnia liczba rozwiązań sąsiednich*. W każdej iteracji fazy drugiej (zasadniczej) z sąsiedztwa $AN^3(\pi)$ rozwiązania bieżącego π losowane jest jedno rozwiązanie sąsiednie, które akceptowane jest z pewnym prawdopodobieństwem, zależnym między innymi od parametru, zwanego *temperaturą*. Jeżeli rozwiązanie zostanie zaakceptowane, staje się rozwiązaniem bieżącym w kolejnej iteracji algorytmu. W przeciwnym przypadku rozwiązanie bieżące nie ulega zmianie. W początkowych iteracjach fazy zasadniczej algorytmu temperatura jest równa temperaturze początkowej. Po wykonaniu określonej liczby iteracji, zwanej *zakresem temperatury*, temperatura ulega obniżeniu zgodnie z przyjętym *schematem chłodzenia*, po czym proces powtarza się. Algorytm kończy swoje działanie, gdy temperatura zrówna się z temperaturą końcową.

Przed przedstawieniem szczegółowego opisu algorytmu, poniżej zostaną omówione poszczególne elementy, z których zbudowany jest algorytm. Pierwszym z elementów algorytmu SW wymagającym szczegółowego omówienia jest prawdopodobieństwo akceptacji rozwiązania sąsiedniego. Dane jest dowolne rozwiązanie bieżące $\pi \in \Pi$ oraz dowolne, wybrane w sposób losowy, rozwiązanie sąsiednie $\pi' \in AN^3(\pi)$. Prawdopodobieństwo $P(\pi, \pi')$ akceptacji rozwiązania sąsiedniego π' (tzn. prawdopodobieństwo przejścia z rozwiązania π do π') przez algorytm SW określany jest zależnością

$$P(\pi, \pi') = \begin{cases} 1, & \Delta \leqslant 0, \\ \exp(\frac{-\Delta}{c}), & \Delta > 0, \end{cases} \tag{4.11}$$

gdzie

$$\Delta = C_{\max}(\pi') - C_{\max}(\pi), \tag{4.12}$$

zaś parametr c jest temperaturą. Zatem, dla określonej temperatury c, rozwiązania lepsze od rozwiązania bieżącego akceptowane są z prawdopodobieństwem 1, natomiast każde rozwiązanie sąsiednie π', gorsze od rozwiązania π, akceptowane jest z pewnym niezerowym prawdopodobieństwem, które jest tym większe, im mniejsza jest wartość $C_{\max}(\pi')$.

Kolejnymi istotnymi elementami algorytmu SW są zakres temperatury i schemat chłodzenia. Z reguły, zakres temperatury w algorytmach symulowanego wyżarzania jest parametrem ustalonym tak, by w przybliżeniu był równy liczbie rozwiązań sąsiednich w przyjętym sąsiedztwie. Jednakże, w problemie rozpatrywanym w tym rozdziale, liczba rozwiązań sąsiednich

silnie zależy od danej instancji problemu i waha się w dość znacznym zakresie. Dlatego w proponowanym algorytmie zakres temperatury, który będzie oznaczany przez dt, nie jest parametrem, lecz funkcją $dt = \phi \cdot rs$, gdzie ϕ jest parametrem, zaś

$$rs = \frac{\sum_{k=1}^{n^2/2} |AN^3(\pi^k)|}{n^2/2} \tag{4.13}$$

jest średnią liczbą rozwiązań sąsiednich. W równaniu (4.13) każde rozwiązanie $\pi^{k+1} \in AN^3(\pi^k)$, $1 \leqslant k < n^2/2$, wybierane jest w sposób losowy z sąsiedztwa $AN^3(\pi^k)$, zaś rozwiązanie π^1 jest rozwiązaniem początkowym, dostarczonym przez algorytm konstrukcyjny.

W trakcie pracy algorytmu temperatura c maleje zgodnie z geometrycznym schematem chłodzenia, począwszy od temperatury początkowej cp i kończąc na temperaturze końcowej ck. W celu wyznaczenia wartości cp, ck należy znać minimalną $\Delta_{\min}$ i maksymalną $\Delta_{\max}$ możliwą zmianę wartości funkcji celu przy przejściu pomiędzy dwoma sąsiednimi rozwiązaniami. Ponieważ czasy wykonania poszczególnych operacji w omawianym problemie zawsze można sprowadzić do liczb całkowitych, to przyjmuje się, że $\Delta_{\min} = 1$. Ze względu na czasy przezbrojeń pomiędzy poszczególnymi operacjami transportowymi, dokładne wyznaczenie wartości $\Delta_{\max}$ wymaga dość dużego nakładu obliczeń. Dlatego w proponowanym algorytmie wartość $\Delta_{\max}$ wyznacza się doświadczalnie, korzystając z równania

$$\Delta_{\max} = \max_{1 \leqslant k < n^2/2} \{C_{\max}(\pi^{k+1}) - C_{\max}(\pi^k)\}, \tag{4.14}$$

gdzie każde rozwiązanie $\pi^{k+1} \in AN^3(\pi^k)$, $1 \leqslant k < n^2/2$, wybierane jest w sposób losowy z sąsiedztwa $AN^3(\pi^k)$, zaś rozwiązanie π^1 jest rozwiązaniem początkowym. Temperaturę początkową cp określa się tak, by przy założeniu $c = cp$, prawdopodobieństwo zaakceptowania rozwiązania sąsiedniego, gorszego o wartość $\Delta_{\max}$ od dowolnego rozwiązania bieżącego $\pi \in \Pi$ wynosiło σp, gdzie $\sigma p < 1$ jest parametrem. Analogicznie określa się temperaturę końcową ck, zatem

$$cp = \frac{-\Delta_{\max}}{\ln(\sigma p)}, \tag{4.15}$$

$$ck = \frac{-\Delta_{\min}}{\ln(\sigma k)}, \tag{4.16}$$

gdzie $\sigma k < 1$ jest parametrem.

Jak już wspomniano, temperatura maleje zgodnie z geometrycznym schematem chłodzenia. To oznacza, że w fazie zasadniczej algorytmu po

Algorytm rozpoczyna działanie z permutacją π i ustalonymi parametrami σp, σk, λ oraz ϕ. Algorytm zwraca najlepsze znalezione rozwiązanie π^* oraz jego wartość funkcji celu C^*.

Faza1:

1. Połóż $\pi^* := \pi$, $C^* = C_{\max}(\pi)$.
2. Wyznacz wartości rs i $\Delta_{\max}$ dane odpowiednio równaniami (4.13), (4.14).
3. Wyznacz wartości cp i ck dane odpowiednio równaniami (4.15), (4.16).
4. Połóż $c := cp$ oraz $dt := \phi \cdot rs$.

Faza2:

5. Dla każdego $k = 1, 2, \ldots, dt$ wykonuj kroki 6-9.
6. Wyznacz zbiór $AV^3(\pi)$. Jeżeli $AV^3(\pi) \neq \emptyset$, to wylosuj ruch $v \in AV^3(\pi)$ i wyznacz $C_{\max}(\pi^v)$. W przeciwnym przypadku połóż $\pi^* := \pi$, zwróć rozwiązanie optymalne π^*, wartość c^* i STOP.
7. Połóż $\Delta := C_{\max}(\pi^v) - C_{\max}(\pi)$.
8. Jeżeli $\Delta \leqslant 0$, to połóż $\pi := \pi^v$. Jeżeli $C^* > C_{\max}(\pi^v)$, to połóż $\pi^* := \pi^v$, $C^* := C_{\max}(\pi^v)$.
9. Jeżeli $\Delta > 0$, to wylosuj liczbę x z przedziału $[0, 1]$. Jeżeli $x < \exp(-\Delta/c)$, to połóż $\pi := \pi^v$.
10. Jeżeli $c > ck$, to połóż $c := \lambda \cdot c$ i idź do kroku 5. W przeciwnym przypadku zwróć π^*, wartość C^* i STOP.

Rysunek 4.3: Schemat algorytmu SW

każdorazowym wykonaniu dt iteracji aktualna temperatura c zmniejszana jest zgodnie z równaniem $c = \lambda \cdot c$, gdzie $\lambda < 1$ jest parametrem.

Na rysunku 4.3 przedstawiony jest schemat algorytmu SW. Pewnego omówienia wymaga krok 6 algorytmu. W kroku tym wykorzystuje się własność 3.4, w myśl której rozwiązanie π jest rozwiązaniem optymalnym, jeżeli $AV^3(\pi) = \emptyset$. Dzięki wykorzystaniu własności 3.6-3.8 i twierdzenia 3.1, obliczenie wielkości $C_{\max}(\pi^v)$ dla dowolnego rozwiązania π^v, $v \in AV^3(\pi)$ odbywa się w czasie $O(n \log m)$ przy założeniu, że $n_k = m$, $k \in J$, $o_l = n$, $l \in M$, $n \geqslant m$. Taka jest też złożoność obliczeniowa poszczególnych iteracji algorytmu.

4.2.3 Algorytmy genetyczne

Ogólna idea algorytmów genetycznych została omówiona w sekcji 2.2.3. W ostatnich latach algorytmy genetyczne dla problemów gniazdowych z kryterium zakończenia wykonywania procesu technologicznego były szeroko dyskutowane w wielu publikacjach, których przegląd znajduje się m.in. w pracach [17,18]. Dodatkowo warta uwagi jest praca [19], w której omawia się algorytm genetyczny w połączeniu z innymi technikami przybliżonymi dla problemu gniazdowego z sekwencyjnie zależnymi czasami przezbrojeń.

W tej sekcji przedstawiono dwa algorytmy genetyczne. Pierwszy z nich, algorytm $GADS$, jest algorytmem opartym na dość typowym schemacie, nie odbiegającym zasadniczo od schematu przedstawionego w sekcji 2.2.3; stosowana jest selekcja ruletkowa i modyfikacje typowych operatorów krzyżowania i mutacji. W algorytmie drugim, nazwanym algorytmem $GAGX$, stosuje się nowy quasi-operator krzyżowania GX, który posiada również cechy algorytmu lokalnych poszukiwań zdolnego do eksploracji wielkiej doliny, omawianej w sekcji 3.4.1. Szczegółowe omówienie poszczególnych elementów obu algorytmów zamieszczono poniżej.

W algorytmach genetycznych przestrzeń rozwiązań jest odwzorowywana w zbiór ciągów kodowych skończonej długości (zwanych osobnikami bądź chromosomami). Dla problemu gniazdowego znane jest przynajmniej dziewięć reprezentacji rozwiązań [17], spośród których autor tej pracy wybrał reprezentację bazującą na listach preferencyjnych. Reprezentacja ta po raz pierwszy została zaproponowana przez Davisa w pracy [21]. W podejściu tym chromosom składa się z zestawu sub-chromosomów, gdzie każdy z nich przedstawia preferowaną kolejność wykonania poszczególnych operacji na maszynach (listę preferencyjną). Nie trudno się tu zorientować, że taki chromosom jest tożsamy z permutacją, zaś geny wchodzące w skład chromosomu odpowiadają poszczególnym operacjom. Ponieważ chromosom stanowi z reguły permutację niedopuszczalną, konieczne jest jego zdekodowanie przy użyciu odpowiedniej procedury dekodującej. Procedura ta, na podstawie dowolnego chromosomu konstruuje permutację dopuszczalną i wyznacza jej wartość funkcji celu. Bardziej precyzyjnie, niech $P = \{\pi^1, \pi^2, \ldots, \pi^{maxp}\}$, $\pi \in \Pi^0$ będzie populacją złożoną z $maxp$ osobników. Do dekodowania chromosomów będzie użyty algorytm priorytetowy $AP(PWK(\pi))$ z regułą priorytetową $R = PWK(\pi)$, opisany w sekcji 4.1.1. Przez $(\alpha, C_{\max}(\alpha)) := AP(PWK(\pi))$ należy rozumieć operację dekodowania chromosomu $\pi \in P$; permutacja $\alpha \in \Pi$ oznacza permutację dopuszczalną z wartością funkcji celu $C_{\max}(\alpha)$, powstałą w wyniku działania algorytmu.

Przez $C(\pi) = C_{\max}(\alpha)$ będzie oznaczana wartość funkcji celu rozwiązania dopuszczalnego α (powstałego w wyniku dekodowania chromosomu π) reprezentowanego przez chromosom $\pi \in P$. Dla każdego z chromosomów $\pi \in P$ określa się wskaźnik przystosowania $f(\pi)$, który wykorzystywany jest na etapie selekcji, w myśl której osobniki najlepiej przystosowane mają największą szansę na wydanie potomstwa. Jednakże, omawiany problem szeregowania zadań jest problemem minimalizacyjnym. Wskaźnik przystosowania $f(\pi)$ osobnika π nie może być zatem tożsamy z wartością $C(\pi)$. Autor tej pracy proponuje jedno z najbardziej powszechnych przekształceń funkcji celu w funkcję przystosowania, w myśl której dla każdego $\pi \in P$ wskaźnik przystosowania jest równy

$$f(\pi) = \frac{Cw - C(\pi)}{maxp \cdot Cw - \sum_{\alpha \in P} C(\alpha)}, \tag{4.17}$$

gdzie $maxp$ jest liczbą elementów w populacji P, zaś

$$Cw = \max_{\alpha \in P}\{C(\alpha)\} + 1. \tag{4.18}$$

Powyższe przekształcenie gwarantuje, że dla każdego chromosomu $\pi \in P$ spełniona jest zależność $0 < f(\pi) < 1$ oraz $\sum_{\alpha \in P} f(\alpha) = 1$.

Dana jest pula rodzicielska $P' \subseteq P$. Z puli P' losuje się określoną liczbę chromosomów i poddaje operacjom krzyżowania. Podczas krzyżowania, z dwóch chromosomów macierzystych formowane są dwa chromosomy potomne wskutek wymiany fragmentów sub-chromosomów. Następnie chromosomy potomne podlegają mutacji. Operacje rekombinacji muszą być przeprowadzone w ten sposób, by chromosomy potomne były legalne, tzn. wciąż stanowiły permutacje. Legalność potomków, w przypadku reprezentacji rozwiązań bazującej na permutacji, zagwarantowana jest przez użycie odpowiednich operatorów. Operatory te powinny być też dobrane w ten sposób, by ich użycie pozwoliło na dostateczną eksplorację przestrzeni rozwiązań. Wśród operatorów mutacji można wymienić operator inwersji I i inwersji podciągu SI. Do najpopularniejszych operatorów krzyżowania zalicza się operatory:

1. SX (ang. *single crossover*),
2. CX (ang. *cycle crossover*),
3. PMX (ang. *partially matched crossover*),
4. PBX (ang. *position-based crossover*),
5. OBX (ang. *order-based crossover*),
6. OX (ang. *order crossover*),
7. LOX (ang. *linear order crossover*).

Ich omówienie znajduje się między innymi w pracy [93]. W przypadku problemów szeregowania zadań dobrą alternatywą wydaje się stosowanie quasi-operatorów rekombinacji wykorzystujących ideę „ścieżki łączącej" (ang. *path relinking*). W myśl tej idei, operator krzyżowania posiada cechy algorytmu lokalnego poszukiwania poszukującego najlepszych rozwiązań znajdujących się na ścieżce łączącej dwa krzyżowane ze sobą chromosomy macierzyste. Ścieżka łącząca może być zdefiniowana na wiele sposobów, jak również można stosować wiele strategii jej przemierzania. W literaturze wymienia się dwa operatory tego typu [84, 86]:

1. quasi-operator krzyżowania $MSXF$ (ang. *multi-step crossover fusion*) oraz
2. quasi-operator mutacji $MSMF$ (ang. *multi-step mutation fusion*).

Zaletą powyższych operatorów jest to, że w trakcie swojej pracy generują tylko permutacje dopuszczalne. Każdy chromosom jest zatem tożsamy z reprezentowanym rozwiązaniem i nie ma potrzeby jego dekodowania. Jednakże, użycie tych operatorów wiąże się z koniecznością doboru odpowiednich parametrów i elementów składowych, zależnych od specyfiki problemu. Konieczność ta znacznie utrudnia użycie operatorów w praktyce, a w przypadku niektórych problemów czyni operatory stosunkowo mało efektywnymi. Dlatego uzasadnione są próby projektowania nowych operatorów o podobnej zasadzie działania, których efektywność nie jest determinowana przez dobór elementów zależnych od specyfiki problemu.

Na rysunku 4.4 przedstawiony jest nowy quasi-operator krzyżowania, nazwany operatorem GX. Operator ten jest jedynym operatorem krzyżowania użytym w algorytmie $GAGX$. W operatorze tym, podobnie jak w operatorach $MSXF$ i $MSMF$, wykorzystuje się ideę „ścieżki łączącej". Wykorzystuje się też geometryczną miarę odległości pomiędzy permutacjami, daną równaniem (3.79). Dzięki istnieniu istotnej dodatniej korelacji (opisywanej w sekcji 3.4.1) pomiędzy miarą geometryczną i miarą tau Kendalla, operator GX umożliwia eksplorację wielkiej doliny, obecnej w przestrzeni rozwiązań problemu. Pomiędzy operatorem GX i operatorami $MSXF$, $MSMF$ występuje jednak szereg różnic. Między innymi, operator GX nie wymaga określenia struktury sąsiedztwa, ale w trakcie swojej pracy generuje chromosomy stanowiące permutacje niedopuszczalne. Wiąże się to z koniecznością ich dekodowania. Operator GX rozpoczyna działanie z ustalonymi chromosomami α, β i parametrem $maxiter$. W kroku 1 procedury tworzone są punkty $A = (a_1, \ldots, a_n)$, $B = (b_1, \ldots, b_n)$ odpowiadające permutacjom α, β, skonstruowane w myśl zasady przedstawionej w sekcji 3.4.1. Bardziej precyzyjnie, kładzie się $a_{\phi(i)} := \alpha^{-1}(i)$, $b_{\phi(i)} := \beta^{-1}(i)$, $i \in O$, gdzie α^{-1}, β^{-1} są permutacjami odwrotnymi, zaś $\phi : O \to \{1, 2, \ldots, n\}$ jest

Operator rozpoczyna działanie z ustalonymi chromosomami α, β i parametrem *maxiter*. Operator zwraca chromosom π^*, będący permutacją dopuszczalną, i wartość funkcji celu C^*.

1. Znajdź punkty $A, B \in \mathbb{R}^n$, odpowiadające odpowiednim permutacjom α, β. Połóż $A' := A$, $B' := B$, $\pi^* := \emptyset$, $C^* := \inf$ oraz $iter := 0$.
2. Utwórz zbiór $W = \{W_1, W_2, \ldots, W_m\}$, gdzie $W_l = \{1, 2, \ldots, o_l\}$, $l \in M$.
3. Utwórz listę $\gamma = (\gamma(1), \gamma(2), \ldots, \gamma(n))$ taką, że $\gamma(i) = 0$, $i = 1, 2, \ldots, n$.
4. Wylosuj liczbę r z przedziału $(0, 1)$. Znajdź punkt $C = (c_1, c_2, \ldots, c_n)$ taki, że $||AB|| = ||AC|| + ||CB||$ oraz $||AC|| = r \cdot ||AB||$. Dla każdego $k = 1, 2, \ldots, n$ wykonuj krok 5.
5. Połóż $l := m_k$. Znajdź $t = \min_{i \in W_l} |c_k - i|$. Wylosuj element z ze zbioru $\{i \in W_l : |c_k - i| = t\}$. Połóż $\gamma(k) := z$ oraz $W_l := W_l \setminus \{z\}$.
6. Utwórz permutację δ taką, że $\delta^{-1} = \gamma$.
7. Połóż $(\omega, C_{\max}(\omega)) := AP(PWK(\delta))$. Jeżeli $C_{\max}(\omega) < C^*$ wtedy połóż $\pi^* := \omega$, $C^* := C_{\max}(\omega)$.
8. Znajdź punkt $A \in \mathbb{R}^n$, odpowiadający permutacji ω. Wylosuj liczbę r z przedziału $[0, 1]$. Jeżeli $r \leqslant 0,5$, to połóż $B := A'$. W przeciwnym przypadku połóż $B := B'$.
9. Połóż $iter := iter + 1$. Jeżeli $iter < maxiter$ to idź do kroku 2. W przeciwnym przypadku zwróć permutację π^* i wartość funkcji celu C^* i STOP.

Rysunek 4.4: Schemat operatora *GX*

dowolną funkcją odwzorowującą zbiór operacji w zbiór $\{1, \ldots, n\}$. W kroku 2 tworzony jest zbiór $W = \{W_1, W_2, \ldots, W_m\}$, gdzie W_l, $l \in M$, jest zbiorem pozycji, na których mogą się znajdować operacje wykonywane na maszynie l w permutacji odwrotnej γ (inicjowanej w kroku 3). W kroku 4 wyznaczany jest punkt C, leżący na odcinku pomiędzy punktami A, B, oddalony o $r\%$ od punktu A, gdzie r jest liczbą losowaną z przedziału $(0, 1)$. W kroku 5, na podstawie punktu C, konstruowana jest permutacja odwrotna γ poprzez odpowiedni wybór elementów ze zbioru W. W kroku 7, na podstawie permutacji δ utworzonej w kroku 6, konstruuje się permuta-

cję dopuszczalną ω przy użyciu algorytmu priorytetowego. Operator zwraca najlepszą permutację dopuszczalną π^*, odnalezioną w trakcie *maxiter* iteracji kroków 2-9, gdzie *maxiter* jest parametrem. Należy podkreślić, że poza chromosomami α, β, parametr *maxiter* jest jedynym parametrem wymaganym przez operator GX. Poprzez zapis $(\delta, C_{\max}(\delta)) := GX(\alpha, \beta)$ należy rozumieć zastosowanie operatora GX do chromosomów α, β, gdzie δ jest permutacją dopuszczalną z wartością funkcji celu $C_{\max}(\delta)$, powstałą w wyniku działania operatora.

W algorytmie $GADS$ również został użyty tylko jeden operator krzyżowania, nazwany operatorem $RPMX$, i operator mutacji I. Działanie operatora $RPMX$, będącego modyfikacją operatora PMX, można opisać następująco. Dane jest dwoje rodziców $\alpha = (\alpha_1, \ldots, \alpha_m), \beta = (\beta_1, \ldots, \beta_m)$, $\alpha, \beta \in P'$. W każdym z sub-chromosomów α_l, β_l losuje się z równym prawdopodobieństwem g_l genów (znajdujących się na tych samych pozycjach w obu sub-chromosomach α_l, β_l). Liczba g_l każdorazowo losowana jest ze zbioru

$$G = \begin{cases} \{\lfloor o_l \cdot s_1 \rfloor, \lfloor o_l \cdot s_1 \rfloor + 1, \ldots, \lfloor o_l \cdot s_2 \rfloor\}, & l \in M^p, \\ \{\lfloor o_l \cdot s_3 \rfloor, \lfloor o_l \cdot s_3 \rfloor + 1, \ldots, \lfloor o_l \cdot s_4 \rfloor\}, & l \in M^t, \end{cases} \tag{4.19}$$

gdzie s_1, s_2, s_3, s_4 są parametrami dobieranymi doświadczalnie. Wylosowane geny są kopiowane do chromosomów potomnych z zachowaniem ich pozycji. Pozostała część operacji krzyżowania przebiega analogicznie jak w przypadku operatora PMX. Poprzez zapis $(\gamma, \delta) := RPMX(\alpha, \beta)$ należy rozumieć operację krzyżowania osobników $\alpha, \beta \in P'$, gdzie γ, δ jest parą chromosomów potomnych.

Poprzez zapis $\gamma := I(\alpha, pm)$ należy rozumieć operację mutacji, gdzie γ jest chromosomem powstałym w wyniku mutacji chromosomu α, zaś pm jest prawdopodobieństwem mutacji. W wyniku mutacji chromosomu α każdy z genów $\alpha_l(i)$, $i = 1, 2, \ldots, o_l$, $l \in M$, jest wymieniany z prawdopodobieństwem pm z genem $\alpha_l(j)$, znajdującym się na losowo wybranej pozycji $1 \leqslant j \leqslant o_l$, $j \neq i$.

W większości przypadków praktycznych, algorytmy genetyczne wyposażone są w dodatkowy mechanizm poprawy rozwiązań dopuszczalnych, generowanych na poszczególnych etapach swojej pracy. Wtedy algorytmy takie nazywa się algorytmami genetycznymi *hybrydowymi*. W przypadku algorytmu $GADS$, do poprawy wspomnianych rozwiązań został użyty algorytm zstępujący (ang. *descent search*) z sąsiedztwem $AN^3(\pi)$, $\pi \in \Pi$, zaopatrzony w implementację efektywnej metody przeglądu sąsiedztwa, omawianą w sekcji 3.2.1. Algorytm ten z sąsiedztwa $AN^3(\pi)$ wybiera każdorazowo rozwiązanie najlepsze i kończy działanie w momencie odnalezienia rozwiązania lokalnie optymalnego. Poprzez zapis $(\delta, C_{\max}(\delta)) := DS(\alpha)$ będzie

Algorytm rozpoczyna działanie z ustalonymi parametrami s_1, s_2, s_3, s_4, pm, $maxgen$ oraz parzystą liczbą $maxp$. Algorytm zwraca najlepsze znalezione rozwiązanie π^* oraz jego wartość funkcji celu C^*.

1. Generuj populację $P = (\pi^1, \pi^2, \ldots, \pi^{2 \cdot maxp})$ kładąc $(\pi^i, C(\pi^i)) := AP(RAN)$, $i = 1, 2, \ldots, maxp$ oraz $\pi^i := \emptyset$, $C(\pi^i) := 0$, $i = maxp + 1, \ldots, 2 \cdot maxp$. Wyznacz wskaźniki przystosowania $f(\pi^i)$, $i = 1, 2, \ldots, maxp$. Połóż $iter := 0$.
2. Połóż $t := maxp + 1$.
3. Stosując schemat selekcji ruletkowej wylosuj parę osobników $\pi^i, \pi^j \in P$, $i, j \leqslant maxp$, $i \neq j$. Połóż $\sigma^1 := \pi^i$, $\sigma^2 := \pi^j$.
4. Połóż $(\alpha^1, \alpha^2) := RPMX(\sigma^1, \sigma^2)$. Połóż $\beta^i := I(\alpha^i, pm)$, $i = 1, 2$.
5. Połóż $(\gamma^i, C_{\max}(\gamma^i)) := AP(PWK(\beta^i))$, $(\delta^i, C_{\max}(\delta^i)) := DS(\gamma^i)$, $i = 1, 2$.
6. Jeżeli $C_{\max}(\delta^1) < C^*$, połóż $\pi^* := \delta^1$, $C^* := C_{\max}(\delta^1)$. Jeżeli $C_{\max}(\delta^2) < C^*$, połóż $\pi^* := \delta^2$, $C^* := C_{\max}(\delta^2)$.
7. Połóż $\pi^t := \beta^1$, $\pi^{t+1} := \beta^2$. Połóż $C(\pi^t) := C_{\max}(\delta^1)$, $C(\pi^{t+1}) := C_{\max}(\delta^2)$. Wyznacz wskaźniki przystosowania $f(\pi^t)$, $f(\pi^{t+1})$.
8. Połóż $t := t+2$. Jeżeli $t < 2 \cdot maxp$, to idź do kroku 3.
9. Uszereguj elementy na liście P znajdujące się na pozycjach $1, 2, \ldots, 3 \cdot maxp/2$ zgodnie z nierosnącymi wskaźnikami przystosowań. Połóż $\pi^i := \pi^{maxp+i}$, $C(\pi^i) := C(\pi^{maxp+i})$, $\pi^{maxp+i} := \emptyset$, $C(\pi^{maxp+i}) := 0$ dla $i = maxp/2+1, maxp/2+2, \ldots, maxp$. Połóż $iter := iter+1$.
10. Jeżeli $iter < maxiter$ to idź do kroku 2. W przeciwnym przypadku zwróć rozwiązanie π^*, wartość C^* i STOP.

Rysunek 4.5: Schemat algorytmu *GADS*

rozumiane zastosowanie algorytmu zstępującego do rozwiązania dopuszczalnego α, gdzie rozwiązanie δ jest rozwiązaniem lokalnie optymalnym, powstałym w wyniku działania algorytmu. W przypadku algorytmu *GAGX* do poprawy wspomnianych rozwiązań został użyty algorytm symulowanego wyżarzania SW', będący modyfikacją algorytmu SW opisywanego w sekcji 4.2.2. Zasadnicza różnica polega na tym, że z algorytmu SW' została

Algorytm rozpoczyna działanie z parametrem $maxp$, parametrem $maxiter$ (dla operatora GX) i parametrami λ, dt, σp, σk (dla algorytmu SW'). Algorytm zwraca najlepsze znalezione rozwiązanie dopuszczalne i jego wartość funkcji celu.

1. Generuj populację $P = (\pi^1, \pi^2, \ldots, \pi^{maxp})$ kładąc $(\pi^i, C_{\max}(\pi^i)) := AP(RAN)$.
2. Uszereguj osobniki na liście P w ten sposób, by $C_{\max}(\pi^i) \leqslant C_{\max}(\pi^{i+1})$, $i = 1, 2, \ldots, maxp - 1$.
3. Wylosuj dwa osobniki π^i, π^j, $i \neq j$, z prawdopodobieństwem odwrotnie proporcjonalnym do ich pozycji i, j na liście P.
4. Połóż $(\gamma, C_{\max}(\gamma)) := GX(\pi^i, \pi^j)$.
5. Połóż $(\delta, C_{\max}(\delta)) := SW'(\gamma)$.
6. Jeżeli $C_{\max}(\delta) < C_{\max}(\pi^{maxp})$ oraz $C_{\max}(\delta) \neq C_{\max}(\pi^i)$, $i = 1, 2, \ldots, maxp$, to połóż $\pi^{maxp} := \delta$.
7. Testuj kryterium stopu. Jeżeli kryterium stopu nie jest spełnione, to idź do kroku 2. W przeciwnym przypadku zwróć najlepsze rozwiązanie na liście P, jego wartość funkcji celu i STOP.

Rysunek 4.6: Schemat algorytmu $GAGX$

usunięta faza strojenia (a konkretnie krok 2 fazy 1 algorytmu SW), co wiąże się z koniecznością arbitralnego przyjęcia uprzednio wyznaczanej wielkości $\Delta_{\max} = 1$. Rezygnuje się też z niepotrzebnego już parametru ϕ; parametrem natomiast staje się zmienna dt, nazywana zakresem temperatury. Sposób wyznaczania temperatury początkowej i końcowej, przy użyciu parametrów σp, σk, nie ulega zmianie. Poprzez zapis $(\delta, C_{\max}(\delta)) := SW'(\alpha)$ należy rozumieć zastosowanie algorytmu SW' do rozwiązania dopuszczalnego α, gdzie rozwiązanie δ jest rozwiązaniem powstałym w wyniku działania algorytmu.

Schemat algorytmu $GADS$ został przedstawiony na rys. 4.5. W kroku 1 algorytmu tworzona jest populacja początkowa, licząca $maxp$ osobników, gdzie $maxp$ jest parametrem, będącym parzystą liczbą naturalną. W tym celu użyty został algorytm priorytetowy $AP(RAN)$, lecz technika generowania poszczególnych osobników może być dowolna. Należy jednak pamiętać o konieczności wyznaczenia poszczególnych wielkości $C(\pi^i)$, $i = 1, \ldots, maxp$. Następnie, w kroku 3, zgodnie ze schematem selekcji

ruletkowej, z populacji selekcjonowana jest para rozwiązań, która zostaje poddana rekombinacji i dekodowaniu w krokach 4, 5. Powstała w ten sposób para rozwiązań potomnych dodawana jest do populacji w kroku 7. Kroki 3-7 powtarzane są aż do momentu, w którym liczebność populacji osiągnie $2 \cdot maxp$ osobników. Wtedy, w kroku 10, z populacji usuwa się $maxp$ osobników w taki sposób, by w populacji pozostały zarówno osobniki o najwyższym przystosowaniu jak i część tych, które powstały bezpośrednio w trakcie rekombinacji w kroku 4. Takie podejście gwarantuje utrzymanie zarówno wysokiego przystosowania populacji jak i jej różnorodności. Jednakże, przy takim podejściu istnieje ryzyko propagowania się osobników identycznych. By temu zapobiec, w kroku 4 osobniki krzyżowane są z prawdopodobieństwem, równym 1. Algorytm kończy swoje działanie po wygenerowaniu $maxgen$ populacji, gdzie $maxgen$ jest parametrem.

Schemat algorytmu $GAGX$ został przedstawiony na rysunku 4.6. Schemat selekcji (krok 3) oraz usuwania (krok 6) osobników z populacji wzorowany jest na pomyśle przedstawionym w pracy [84]. W kroku 1 populacja P została wygenerowana przez algorytm priorytetowy $AP(RAN)$, lecz, podobnie jak w przypadku algorytmu $GADS$, technika generowania poszczególnych osobników może być dowolna. W kroku 7 zasadniczo rozważa się tylko jedno kryterium stopu – maksymalny czas pracy algorytmu. Pozostałe kroki algorytmu nie wymagają komentarza.

4.3 Wyniki badań numerycznych

Celem przeprowadzonych badań numerycznych było określenie relacji czasowych i jakościowych pomiędzy algorytmami prezentowanymi zarówno w poprzednich sekcjach jak i literaturze. Badania można podzielić na dwa etapy. Pierwszy etap miał na celu wyłonienie najlepszych, spośród algorytmów prezentowanych w tym rozdziale. Ze względu na brak literaturowych instancji testowych i algorytmów rozwiązywania problemu $J, DR|t_{jkl}, t'_{kl}|C_{\max}$ (z wyjątkiem przypadku szczególnego, tj. problemu $J, DR1|t_{jkl}, t'_{kl}|C_{\max}$) etap pierwszy badań był przeprowadzony przy użyciu instancji testowych wygenerowanych przez autora tej rozprawy. Drugi etap badań polegał na porównaniu jakości pracy prezentowanych algorytmów z algorytmami znanymi z literatury, zaprojektowanymi do rozwiązywania przypadku szczególnego problemu gniazdowego z transportem – problemu $J, DR1|t_{jkl}, t'_{kl}|C_{\max}$. Realizacja tego etapu polegała na rozwiązaniu instancji testowych z prac [45, 50] i porównaniu dostarczonych wyników z wynikami prezentowanymi również w pracach [45,50]. Wspomniane instancje testowe zostały też szczegółowo opisane w sekcji 3.4.1. Należy dodać,

że wszystkie algorytmy opisywane w tym rozdziale były zaimplementowane w Delphi 6 i uruchamiane na komputerze z procesorem AMD Athlon XP 2500+ (1833 MHz 7568 MIPS, 2888 MFLOPS[1]).

4.3.1 Instancje testowe

Zaprojektowane przez autora tej pracy instancje testowe dla problemu gniazdowego z transportem można opisać następująco. Instancje te zostały wygenerowane na bazie 40 instancji testowych Lawrence'a ($LA01$-$LA40$) dla klasycznego problemu gniazdowego, prezentowanych w pracy [55]. Instancje Lawrence'a są podzielone na 8 zestawów po 5 instancji o rozmiarach $r \times m^p = 10 \times 5$, 15×5, 20×5, 10×10, 15×10, 20×10, 30×10, 15×15. Poprzez uwzględnienie różnych układów maszyn produkcyjnych, dodanie różnej liczby wózków AGV, różnych kombinacji czasów transportu oraz przejazdów pustych zostało wygenerowane 480 instancji podzielonych na 40 zestawów, oznaczanych przez $TR01, TR02, \ldots, TR40$, po 12 instancji w każdym zestawie. Każda instancja ma swoją unikalną nazwę w formacie $TRa/m^t/b/c/d$, gdzie a, b, c, d są zmiennymi, zaś $m^t \in \{2, 4\}$ określa liczbę wózków AGV. Zmienna $a \in \{1, \ldots, 40\}$ oznacza numer instancji Lawrence'a, która podlega modyfikacji. Bardziej precyzyjnie, dla każdej instancji z grupy TRa przyjmuje się liczbę maszyn produkcyjnych, liczbę zadań i operacji produkcyjnych, czasy wykonania oraz przydział maszyn do operacji produkcyjnych taki sam jak dla instancji LAa. Zmienna $b \in \{1, 2\}$ określa typ układu maszyn produkcyjnych. Rozpatrywane były dwa, spośród najbardziej popularnych układów maszyn. Dla $b = 1$ oraz $b = 2$ przyjęto odpowiednio układ typu pętla oraz układ typu siatka. Zmienne c i d są współczynnikami skalującymi odpowiednio czasów przejazdów pustych i czasów transportu; mogą przyjmować jedną z trzech par wartości, $(c, d) \in \{(2, 2), (2, 5), (5, 5)\}$. Ostatecznie, dla każdej pary maszyn produkcyjnych $l', l'' \in M^p$ przyjmuje się $e(l', l'') = c \cdot g(l', l'')$ oraz $t_k(l', l'') = d \cdot g(l', l'')$, $k \in J$, gdzie $g(l', l'')$ jest funkcją określającą odległość pomiędzy maszynami l', l''. Dla $b = 1$ przyjmuje się

$$g(l', l'') = \begin{cases} |l' - l''|, & |l' - l''| \leqslant \left\lfloor \frac{m^p}{2} \right\rfloor, \\ m^p - |l' - l''|, & \text{w przeciwnym przypadku.} \end{cases} \tag{4.20}$$

Dla $b = 2$ funkcja $g(l', l'')$ przyjmuje postać

$$g(l', l'') = |\, t' - t''| + \left| l' - t' - l'' + \frac{t''}{5} \right|, \tag{4.21}$$

[1]Pomiar wykonano przy użyciu programu SiSoftware Sandra Lite Unicode (Win32 x86) 2005.2.10.50. Program jest programem typu Shareware i jest dostępny w Internecie.

gdzie

$$t' = 5 \cdot \left(\frac{l' - 1}{5} - \left\lfloor \frac{l' - 1}{5} \right\rfloor \right) + 1, \tag{4.22}$$

$$t'' = 5 \cdot \left(\frac{l'' - 1}{5} - \left\lfloor \frac{l'' - 1}{5} \right\rfloor \right) + 1. \tag{4.23}$$

Przydział maszyn transportowych do operacji transportowych określony jest równaniem

$$m_i = \min \left\{ m^t, \left\lfloor \frac{(z_i - 1) \cdot m^t}{r} \right\rfloor \right\}, \tag{4.24}$$

gdzie z_i jest numerem zadania, w ramach którego wykonywana jest operacja transportowa $i \in O^t$.

Spośród wszystkich zestawów instancji najbardziej interesujące wydają się zestawy $TR16$-25, oraz $TR36$-40 (w sumie 180 instancji) i głównie do nich będą ograniczone dalsze rozważania. Powodem tego zainteresowania jest stosunkowo niewielka liczba instancji rozwiązanych optymalnie w tych zestawach. Fakt ten będzie szerzej omawiany przy okazji prezentacji wyników badań numerycznych algorytmów popraw. Zestawy te można też traktować jako „trudne" z punktu widzenia rozważań prowadzonych w sekcji 3.4.3.

4.3.2 Wyniki badań algorytmów konstrukcyjnych

Celem badań numerycznych algorytmów konstrukcyjnych było wyłącznie wyłonienie algorytmu generującego rozwiązania najlepsze. Celowo też nie prowadzi się dyskusji na temat czasów pracy poszczególnych algorytmów; dla ogromnej większości badanych instancji testowych czas ten nie przekraczał setnej części sekundy. Najbardziej czasochłonnym algorytmem jest algorytm $INT3$, opisywany w sekcji 4.1.2. Dla największych instancji czas jego pracy nie przekraczał jednak 0,06 sekundy.

Algorytmy priorytetowe $AP(R)$ opisywane w sekcji 4.1.1 dla każdej instancji z zestawu $TR16$-25, $TR36$-40 były uruchamiane z regułami priorytetowymi $R \in Z = \{STT, LTT, SPT, LPT, RAN\}$. Wyznaczone w ten sposób wartości funkcji celu będą oznaczane odpowiednio przez C^{STT}, C^{LTT}, C^{SPT}, C^{LPT} i C^{RAN}. Na podstawie tych wartości wyznaczono również wartość minimalną, równą

$$C^{MIN} = \min_{A \in Z} \{C^A\}. \tag{4.25}$$

Algorytm konstrukcyjny $INT3$ nie wymaga sprecyzowania parametrów przed jego uruchomieniem. Wartość funkcji celu wyznaczonej przez ten algorytm dla każdej instancji będzie oznaczana przez C^{IN}. Na podstawie

Tabela 4.1: Średnie arytmetyczne poprawy $av\Phi_A^{MIN}$ dla zestawów $TR16$-25, $TR36$-40

TRa	$av\Phi_{STT}^{MIN}$ [%]	$av\Phi_{LTT}^{MIN}$ [%]	$av\Phi_{SPT}^{MIN}$ [%]	$av\Phi_{LPT}^{MIN}$ [%]	$av\Phi_{RAN}^{MIN}$ [%]	$av\Phi_{IN}^{MIN}$ [%]
$TR16$	-6,06	-9,43	-5,50	**-4,46**	-5,42	7,03
$TR17$	-2,48	-5,2	**-1,96**	-7,81	-6,52	7,49
$TR18$	**-2,68**	-11,60	-13,83	-13,94	-4,52	5,99
$TR19$	**-0,82**	-12,24	-7,95	-16,60	-4,23	-1,02
$TR20$	-1,83	-22,86	-26,98	**-1,12**	-12,72	1,46
$TR21$	**-0,78**	-13,34	-10,43	-6,19	-6,33	3,68
$TR22$	**-2,79**	-16,59	-11,19	-17,49	-5,90	5,29
$TR23$	**-0,11**	-15,83	-10,35	-22,38	-6,88	0,96
$TR24$	-8,93	**-3,42**	-7,59	-21,19	-3,92	9,07
$TR25$	-5,73	-7,71	-10,52	-9,02	**-1,99**	10,04
$TR36$	-6,69	-6,26	-5,17	-12,14	**-2,99**	4,70
$TR37$	**-0,77**	-10,37	-7,55	-5,52	-6,89	-0,15
$TR38$	**-2,55**	-9,73	-8,03	-4,94	-3,76	3,32
$TR39$	**-1,84**	-8,11	-2,83	-8,66	-4,78	2,04
$TR40$	**-1,85**	-11,60	-6,91	-9,61	-3,48	0,49
średnia	**-3,06**	-10,95	-9,12	-10,74	-5,36	4,03

wyznaczonych wartości obliczono względną poprawę

$$\Phi_A^{MIN} = 100\% \cdot \frac{C^{MIN} - C^A}{C^{MIN}} \tag{4.26}$$

wartości minimalnej C^{MIN} przez odpowiedni z algorytmów, gdzie $A \in Z \cup \{IN\}$. Ostatecznie, dla każdego zestawu $TR16$-25, $TR36$-40 wyliczono średnie arytmetyczne poprawy $av\Phi_A^{MIN}$ i przedstawiono w tabeli 4.1. Największe wielkości $av\Phi_A^{MIN}$, $A \in Z$, dla każdego z zestawów zostały pogrubione.

Jak wynika z tabeli 4.1, wśród testowanych algorytmów priorytetowych najlepszym okazał się algorytm z regułą priorytetową STT. Średnia arytmetyczna poprawa generowana przez ten algorytm dla poszczególnych zestawów instancji nie była gorsza, niż $-8,93\%$. Po uszeregowaniu pozostałych algorytmów priorytetowych pod względem jakości generowanych rozwiązań mamy algorytm z regułą RAN, SPT, LPT, LTT. Kolejność ta, niestety, jest zachowana bardzo rzadko w odniesieniu do indywidualnych zestawów instancji, co wskazuje na ewidentną słabość tych algorytmów. Przykładem może być zestaw $TR24$, dla którego najlepsze rozwiązania ge-

Tabela 4.2: Wartości funkcji celu rozwiązań wygenerowanych przez algorytm $INT3$ dla zestawów $TR16$-25, $TR36$-40

TRa	$TRa/2/1/c/d$			$TRa/2/2/c/d$		
	2/2	2/5	5/5	2/2	2/5	5/5
$TR16$	1122	1193	1288	1112	1075	1194
$TR17$	851	979	1121	849	926	1038
$TR18$	958	1050	1163	945	1014	1123
$TR19$	959	1094	1195	971	1059	1122
$TR20$	1003	1051	1173	1001	1063	1103
$TR21$	1224	1393	1544	1208	1347	1488
$TR22$	1088	1318	1491	1139	1239	1426
$TR23$	1160	1335	1571	1161	1284	1420
$TR24$	1076	1384	1610	1028	1212	1430
$TR25$	1112	1324	1481	1161	1244	1416
$TR36$	1683	2795	3299	1493	2028	2356
$TR37$	1796	2653	3164	1667	2062	2435
$TR38$	1657	2718	3154	1408	1927	2281
$TR39$	1567	2568	3007	1495	1938	2359
$TR40$	1686	2672	3096	1459	1936	2383

TRa	$TRa/4/1/c/d$			$TRa/4/2/c/d$		
	2/2	2/5	5/5	2/2	2/5	5/5
$TR16$	1122	1169	1186	1112	1069	1079
$TR17$	851	911	942	849	897	915
$TR18$	958	980	1033	944	1001	1022
$TR19$	959	1073	1102	971	1030	1050
$TR20$	1002	1040	1040	1001	1058	1069
$TR21$	1210	1320	1352	1205	1274	1299
$TR22$	1082	1147	1172	1139	1198	1232
$TR23$	1164	1213	1267	1153	1193	1215
$TR24$	1076	1147	1224	1028	1088	1131
$TR25$	1098	1170	1213	1161	1217	1239
$TR36$	1480	1813	2152	1466	1597	1737
$TR37$	1726	1918	2127	1654	1785	1890
$TR38$	1441	1829	2103	1383	1506	1611
$TR39$	1486	1863	2140	1478	1577	1718
$TR40$	1500	1747	2019	1431	1545	1720

nerowały kolejno algorytmy z regułą LTT, RAN, SPT, STT, LPT. Dla tego zestawu najlepsze rozwiązania wygenerowane zostały przy użyciu reguły ogólnie najgorszej.

Jakość poszczególnych algorytmów priorytetowych można też określić na podstawie liczby instancji, dla których determinowały one wielkość C^{MIN}. I tutaj kolejność pozostaje niezmieniona: najlepszy okazał się algorytm z regułą STT determinując wielkość C^{MIN} dla 77 instancji, później algorytm z regułą RAN (48 instancji), dalej kolejno algorytmy z regułą SPT (28 instancji), LPT (16 instancji) oraz LTT (11 instancji).

Najlepszym, spośród testowanych algorytmów konstrukcyjnych, okazał się algorytm $INT3$. Algorytm ten poprawił wielkość C^{MIN} dla 137, spośród 180 rozpatrywanych instancji testowych, dostarczając rozwiązań średnio lepszych o 4% od najlepszych rozwiązań generowanych przez algorytmy priorytetowe. Wartości funkcji celu rozwiązań dostarczonych przez ten algorytm dla wszystkich 180 instancji zostały przedstawione w tabeli 4.2. Analizując rozwiązania dostarczone przez algorytm $INT3$ pod kątem czasów transportu, liczby wózków i typu układu maszyn produkcyjnych można poczynić kilka dodatkowych obserwacji. Po pierwsze, wraz ze wzrostem czasów transportu i przejazdów pustych (zwiększeniem współczynników skalujących c, d) przewaga algorytmu $INT3$ (w sensie jakości generowanych rozwiązań) nad algorytmami priorytetowymi maleje. Podobnie jest w przypadku zastąpienia układu maszyn typu siatka układem typu pętla (zmiana wartości współczynnika b z 2 na 1) oraz zmniejszenia liczby maszyn transportowych. Każda z tych zmian powoduje wydłużenie się czasów transportu i przejazdów pustych oraz wzrost obciążenia maszyn transportowych. Zatem, instancjami „najtrudniejszymi" i „najłatwiejszymi" dla algorytmu $INT3$ wydają się instancje z odpowiednio najkrótszym i najdłuższym stosunkowym czasem transportu. Są to instancje odpowiednio $TRa/2/1/5/5$ oraz $TRa/4/2/2/2$, $a \in \{16, \ldots, 25, 36, \ldots, 40\}$. I rzeczywiście, najmniejszą poprawę Φ^{MIN}_{INT3}, wynoszącą $-10,3\%$, zanotowano dla instancji $TR36/2/1/5/5$, zaś największą, wynoszącą $20,13\%$, dla instancji $TR36/4/1/2/2$. Analogicznych spostrzeżeń nie można poczynić obserwując zachowanie się algorytmów priorytetowych. Jakość pracy tych algorytmów wydaje się niezależna od obciążeń maszyn transportowych.

4.3.3 Wyniki badań algorytmów popraw – instancje TR

Pierwsza część badań numerycznych opisywanych w tym rozdziale algorytmów popraw będzie przeprowadzona przy użyciu zestawów instancji $TR01$-$TR40$ (ze szczególnym uwzględnieniem zestawów $TR16$-25, $TR36$-40). Badania te mają na celu porównanie algorytmów poprzez

zestawienie czasów ich pracy oraz jakości dostarczonych rozwiązań. W trakcie testów każdy z algorytmów rozpoczynał swoje działanie z rozwiązaniem dostarczonym przez algorytm $INT3$. Wyjątkiem są algorytmy genetyczne $GADS$ oraz $GAGX$, w których populacja początkowa była generowana w oparciu o algorytm priorytetowy $AP(R)$ z regułą priorytetową $R = RAN$.

Algorytm $TSAB_{AGV}$ był uruchomiony dla każdej instancji dwukrotnie z wartościami parametrów $maxl = 5$, $max\delta = 100$, $maxc = 2$, $maxp = 13$ i $maxt = 30$ ($maxp = 6$ i $maxt = 12$ w trakcie drugiego uruchomienia). W trakcie pierwszego uruchomienia maksymalna liczba iteracji była równa $maxiter = 20000$ po starcie algorytmu i po każdej poprawie wartości funkcji celu oraz $maxiter = 10000$ po wykonaniu skoku powrotnego. W trakcie drugiego uruchomienia została przyjęta stała wartość $maxiter = 16000$. Algorytm i–$TSAB_{AGV}$ również został uruchomiony dla każdej instancji dwukrotnie z tymi samymi wartościami parametrów co $TSAB_{AGV}$ i dodatkowymi parametrami o wartościach $maxE = 8$, $maxV = 0,5$ $maxD = 5$, $maxR = 20$ oraz $maxT = 10$ minut. Dla każdej z instancji algorytm SW uruchomiono 10 razy z wartościami parametrów $\sigma p = 0,97$, $\sigma k = 0,001$, $\lambda = 0,999$ oraz $\phi = 10$. Dla każdej instancji algorytm $GADS$ został uruchomiony dziesięciokrotnie z ustalonymi wartościami $s_1 = 20\%$, $s_2 = 40\%$, $s_3 = 17\%$, $s_4 = 23\%$, $pm = 0,05$, $maxgen = 1500$ oraz $maxp = 50$. Przed uruchomieniem algorytmu $GAGX$ konieczne było określenie parametru $maxp = 30$, parametru $maxiter = 100$ wymaganego przez operator GX oraz parametrów $\lambda = 0,98$, $dt = 500$, $\sigma p = 0,99$, $\sigma k = 0,1$, wymaganych przez algorytm SW'. Algorytm $GAGX$ dla każdej instancji został uruchomiony trzykrotnie z ograniczeniem czasowym, wynoszącym odpowiednio 200, 400 i 600 sekund. Dla uproszczenia, algorytm ten, dla odpowiednich ograniczeń czasowych, będzie notowany jako algorytm $GAGX1$, $GAGX2$ oraz $GAGX3$.

Wartość funkcji celu najlepszego rozwiązania znalezionego przez algorytm $TSAB_{AGV}$, i–$TSAB_{AGV}$, SW, $GADS$, $GAGX1$, $GAGX2$, $GAGX3$ dla każdej instancji będzie oznaczana odpowiednio symbolem C^{TS}, C^{IT}, C^{SW}, C^{DS}, C^{GX1}, C^{GX2} i C^{GX3}. Dodatkowo, zostały wyznaczone średnie arytmetyczne wartości funkcji celu uzyskanych po dziesięciokrotnym algorytmu SW i $GADS$ dla każdej instancji. Wielkości te zostały oznaczone odpowiednio przez C^{SWS} oraz C^{DSS}. Sumaryczny czas pracy algorytmu $TSAB_{AGV}$, i–$TSAB_{AGV}$, SW oraz $GADS$ dla poszczególnych instancji będzie oznaczany odpowiednio symbolem T^{TS}, T^{IT}, T^{SW}, oraz T^{DS}. Średnia arytmetyczna wartości T^A, $A \in \{TS, IT, SW, DS\}$, wyznaczona dla poszczególnych zestawów instancji będzie oznaczana symbolem

T^A_{av}. Czas pracy algorytmu $GAGX$ i średnia arytmetyczna czasu pracy, z przyczyn oczywistych, nie będzie oznaczana.

Dzięki własności 3.4 wiadomo, że powyższe algorytmy rozwiązały optymalnie większość instancji w zestawach TR01-15, $TR$26-35. W dodatku, liczba instancji rozwiązanych optymalnie praktyczne nie zależy od użytego algorytmu. Na tej podstawie rozważane instancje można uznać za „łatwe" i usunąć z dalszych rozważań. Wśród pozostałych zestawów, optymalność rozwiązania została udowodniona jedynie dla 4 instancji. Są to instancje $TR$23/2/1/2/2, $TR$23/2/2/2/2, $TR$23/4/1/2/2, $TR$23/4/2/2/2.

Wszystkie algorytmy popraw zostały porównane z najlepszym spośród badanych algorytmów konstrukcyjnych – algorytmem $INT3$. Dla każdej instancji z grupy $TR$16-25, $TR$36-40, korzystając z równania (4.26), została wyznaczona względna poprawa Φ^{IN}_A wielkości C^{IN} przez poszczególne algorytmy, gdzie $A \in Z = \{TS, IT, SW, SWS, DS, DSS, GX1, GX2, GX3\}$. Następnie, dla każdego rozważanego zestawu zostały wyznaczone średnie arytmetyczne poprawy $av\Phi^{IN}_A$. Średnie poprawy, jak i średnie czasy pracy poszczególnych algorytmów zostały przedstawione w tabelach 4.3, 4.4; największe wyznaczone wielkości $av\Phi^{IN}_A$, $A \in Z$, zostały pogrubione.

Bezapelacyjnie najgorszym, spośród testowanych algorytmów, okazał się algorytm genetyczny $GADS$, zarówno pod względem czasu pracy jak i jakości generowanych rozwiązań. Nawet najlepsze rozwiązania wygenerowane przez ten algorytm są znacznie gorsze od średniej rozwiązań zwracanych przez algorytm SW. Algorytm ten nie będzie dalej analizowany w kontekście zestawów instancji testowych TR. Z dalszych rozważań można również usunąć algorytm SW, ponieważ sumaryczny czas jego pracy w każdym przypadku był dłuższy niż czas pracy algorytmu $TSAB_{AGV}$, zaś średnie poprawy $av\Phi^{IN}_{SW}$, z wyjątkiem zestawu $TR$39, były mniejsze od poprawy $av\Phi^{IN}_{TS}$. Niewielka różnica pomiędzy wartościami $av\Phi^{IN}_{SW}$, $av\Phi^{IN}_{TS}$ i stosunkowo krótki sumaryczny czas pracy algorytmu $TSAB_{AGV}$ może nieco dziwić. Należy jednak pamiętać, że algorytm ten był testowany bardziej jako składnik algorytmu i–$TSAB_{AGV}$ niż jako w pełni niezależny algorytm; czas pracy algorytmu i tym samym jakość generowanych rozwiązań można poprawić poprzez odpowiedni dobór parametrów sterujących. Najlepszym, pod względem generowanych rozwiązań, jest algorytm i–$TSAB_{AGV}$, choć pod względem średniego czasu pracy, algorytm ten jest niewiele lepszy do algorytmu $GADS$. W przypadku zestawów instancji $TR$16-25 algorytmem pracującym najkrócej jest algorytm $TSAB_{AGV}$. Jednocześnie, rozwiązania generowane przez algorytm dla tych zestawów ustępują jakością tylko algorytmom i–$TSAB_{AGV}$, $GAGX2$, $GAGX3$ i są porównywalne jakościowo z rozwiązaniami wygenerowanymi przez algorytm $GAGX1$. W przypadku

Tabela 4.3: Średnie arytmetyczne poprawy $av\Phi_A^{IN}$ oraz średnie czasy pracy algorytmów popraw dla zestawów $TR16$-25, $TR36$-40

TRa	$av\Phi_{TS}^{IN}$ [%]	$av\Phi_{IT}^{IN}$ [%]	$av\Phi_{SW}^{IN}$ [%]	$av\Phi_{SWS}^{IN}$ [%]	T_{av}^{TS} [s]	T_{av}^{IT} [s]	T_{av}^{SW} [s]
$TR16$	10,12	**10,38**	9,96	8,98	13,3	238,1	42,6
$TR17$	**7,74**	**7,74**	7,58	7,08	18,9	336,0	56,3
$TR18$	8,72	**8,92**	8,46	7,99	15,9	296,5	49,5
$TR19$	11,11	**11,40**	11,13	10,41	19,3	330,2	52,4
$TR20$	7,87	**7,89**	7,67	7,38	11,7	224,7	48,0
$TR21$	12,70	**13,16**	12,37	11,52	40,5	625,5	114,8
$TR22$	14,68	**15,37**	14,33	13,16	47,7	693,9	134,0
$TR23$	12,93	**13,21**	12,53	11,71	47,3	563,0	121,4
$TR24$	9,14	**9,99**	8,86	7,55	63,0	740,8	138,5
$TR25$	11,66	**12,72**	11,61	10,05	44,6	661,5	119,7
$TR36$	9,37	9,99	9,09	8,03	238,4	1093,1	399,2
$TR37$	11,13	**12,29**	11,13	10,05	207,9	1124,1	357,3
$TR38$	8,95	**9,61**	8,82	7,47	240,9	1142,2	402,5
$TR39$	11,88	**13,11**	12,34	11,08	203,4	1103,3	377,5
$TR40$	10,55	**11,80**	10,90	9,57	213,2	1135,2	399,5
średnia	10,57	**11,17**	10,45	9,47	95,1	687,2	187,6

zestawów $TR36$-40 algorytm $GAGX1$ zdobywa jednak przewagę czasową i jakościową nad algorytmem $TSAB_{AGV}$. Nieproporcjonalnie duży, w stosunku do wzrostu rozmiaru poszczególnych instancji, wzrost czasu pracy algorytmu $TSAB_{AGV}$ ma związek ze wzrostem rozmiaru sąsiedztwa. Przyczyny takiego zjawiska należy upatrywać w wydłużeniu się czasów transportu we wspomnianych instancjach i, co za tym idzie, wydłużeniu się bloków operacji transportowych. Analizując algorytmy $GAGX1$-3 pod kątem czasu pracy można zauważyć, że zwiększenie limitu czasu pracy powyżej 200 sekund w bardzo niewielkim stopniu wpływa na jakość generowanych rozwiązań. Maksymalna różnica pomiędzy wielkością $av\Phi_{GX3}^{IN}$ i wielkością $av\Phi_{GX1}^{IN}$ (notowana dla zestawu $TR40$) nie przekracza $0,5$ punktu procentowego.

W przypadku algorytmu $TSAB_{AGV}$ najmniejszą i największą poprawę, wynoszącą odpowiednio $\Phi_{TS}^{IN} = 3,6\%$ oraz $\Phi_{TS}^{IN} = 18,3\%$, zanotowano odpowiednio dla instancji $TR24/2/2/2/2$ oraz instancji $TR22/4/2/5/5$. W przypadku algorytmu i–$TSAB_{AGV}$ poprawę najmniejszą $\Phi_{IT}^{IN} = 4,8\%$ i największą $\Phi_{IT}^{IN} = 20,0\%$ zanotowano odpowiednio dla

Tabela 4.4: Średnie arytmetyczne poprawy $av\Phi_A^{IN}$ oraz średnie czasy pracy algorytmów popraw dla zestawów $TR16$-25, $TR36$-40, c.d.

TRa	$av\Phi_{DS}^{IN}$ [%]	$av\Phi_{DSS}^{IN}$ [%]	$av\Phi_{GX1}^{IN}$ [%]	$av\Phi_{GX2}^{IN}$ [%]	$av\Phi_{GX3}^{IN}$ [%]	T_{av}^{DS} [s]
$TR16$	7,99	6,99	10,33	10,36	10,37	383,9
$TR17$	5,45	5,06	7,55	7,62	7,66	380,4
$TR18$	6,95	6,62	8,82	8,84	8,84	384,3
$TR19$	8,82	8,64	11,31	11,32	11,33	380,4
$TR20$	5,91	5,89	7,84	7,85	7,85	380,6
$TR21$	10,66	9,63	12,22	12,36	12,43	759,8
$TR22$	12,43	11,06	14,11	14,37	14,45	763,7
$TR23$	11,82	10,91	12,26	12,32	12,40	686,3
$TR24$	7,59	6,56	9,28	9,30	9,42	756,3
$TR25$	9,40	8,06	11,38	11,60	11,66	761,5
$TR36$	8,89	8,04	9,77	9,99	**10,02**	1526,7
$TR37$	10,80	9,77	11,45	11,67	11,73	1533,8
$TR38$	7,46	6,47	8,64	8,83	8,86	1537,6
$TR39$	11,01	9,97	12,69	12,84	12,89	1536,7
$TR40$	10,20	9,28	11,00	11,29	11,48	1542,0
średnia	9,02	8,20	10,58	10,70	10,76	887,6

instancji $TR16/2/2/2/5$ oraz instancji $TR22/4/2/5/5$. Podobnie ma się rzecz z algorytmami $GAGX1$-3. Najmniejszą poprawę $\Phi_{GX1}^{IN} = 4,7\%$ zanotowano dla instancji $TR16/2/2/2/5$, zaś największą $\Phi_{GX3}^{IN} = 20,0\%$ dla instancji $TR22/4/2/5/5$.

Analizując poprawy wyznaczone przez poszczególne algorytmy można zauważyć tendencję odwrotną jak w przypadku algorytmu konstrukcyjnego $INT3$; wraz ze wzrostem czasów transportu i przejazdów pustych (zwiększeniem współczynników skalujących c, d) można zaobserwować wzrost poszczególnych wielkości Φ_A^{IN}, $A \in Z$. Można zatem zaryzykować stwierdzenie, że algorytmy popraw „nadrabiają zaległości" pozostawione przez algorytm $INT3$. Powyższe spostrzeżenie oznacza też, że jakość rozwiązań dostarczanych przez algorytmy $TSAB_{AGV}$, i–$TSAB_{AGV}$ w niewielkim tylko stopniu zależy od jakości rozwiązań początkowych.

Ostatnią prezentowaną tabelą w tej sekcji jest tabela 4.5. W tabeli tej naniesione są wartości funkcji celu najlepszych rozwiązań znalezionych przez wszystkie algorytmy; wielkości wygenerowane przez algorytm

Tabela 4.5: Wartości funkcji celu najlepszych rozwiązań wygenerowanych przez wszystkie algorytmy dla zestawów $TR$16-25, $TR$36-40

TRa	$TRa/2/1/c/d$			$TRa/2/2/c/d$		
	2/2	2/5	5/5	2/2	2/5	5/5
$TR$16	***978***	***1052***	**1113**	*980*	***1023***	***1043***
$TR$17	***805***	**900**	**980**	***801***	***849***	**898**
$TR$18	*884*	***954***	**1015**	**876**	***930***	**952**
$TR$19	***875***	***948***	**1013**	***878***	*947*	**997**
$TR$20	***936***	***986***	**1028**	***935***	***976***	***983***
$TR$21	*1084*	**1178**	**1290**	**1081**	**1167**	**1261**
$TR$22	***953***	**1131**	**1247**	***947***	**1056**	**1173**
$TR$23	***1032***	**1142**	**1293**	***1032***	**1116**	1224
$TR$24	**967**	**1220**	1353	*967*	*1114*	**1215**
$TR$25	***1001***	**1136**	**1249**	***1003***	**1102**	**1195**
$TR$36	**1481**	2592	2835	*1347*	1844	2052
$TR$37	**1558**	2436	2712	**1485**	1810	2018
$TR$38	1434	2533	2754	**1285**	1810	2030
$TR$39	**1408**	2287	2543	**1281**	1698	1926
$TR$40	**1394**	2471	2713	**1288**	1757	1985

TRa	$TRa/4/1/c/d$			$TRa/4/2/c/d$		
	2/2	2/5	5/5	2/2	2/5	5/5
$TR$16	***977***	***1042***	***1052***	*978*	***1017***	***1021***
$TR$17	***805***	***859***	***875***	***801***	***830***	***842***
$TR$18	*881*	***931***	***940***	*874*	***919***	***920***
$TR$19	***874***	***920***	*933*	***876***	***929***	***939***
$TR$20	***931***	***977***	***980***	***934***	***967***	***967***
$TR$21	***1082***	***1129***	**1137**	1079	**1123**	***1135***
$TR$22	***953***	**1002**	***1027***	***947***	*977*	***986***
$TR$23	***1032***	*1043*	**1057**	***1032***	***1040***	**1052**
$TR$24	***967***	**1032**	**1057**	*964*	**1017**	**1035**
$TR$25	*1001*	**1057**	**1081**	***998***	**1044**	**1056**
$TR$36	**1369**	**1626**	**1814**	***1331***	**1433**	**1508**
$TR$37	**1495**	**1693**	**1862**	**1457**	**1555**	**1599**
$TR$38	**1302**	**1599**	**1778**	**1256**	**1365**	**1455**
$TR$39	**1317**	**1549**	**1753**	**1274**	**1377**	**1443**
$TR$40	**1299**	**1539**	**1740**	**1267**	**1370**	**1446**

i–$TSAB_{AGV}$ oraz algorytm $GAGX3$ zostały odpowiednio pogrubione bądź oznaczone kursywą. Jak widać, algorytm i–$TSAB_{AGV}$ dostarczył najlepszych rozwiązań dla 141 spośród 180 badanych instancji, podczas gdy rozwiązania algorytmu $GAGX3$ były najlepsze tylko 77 razy, w tym wielkość C^{GX3} była równa wielkości C^{IT} dla 62 instancji. Fakt ten jest nieco zaskakujący, zwłaszcza że różnica pomiędzy wartościami $av\Phi^{IN}_{IT}$, $av\Phi^{IN}_{GX3}$ dla niewielu tylko zestawów instancji przekroczyła $0,5$ punktu procentowego. Jedynie dla 22 badanych instancji najlepsze rozwiązanie zostało dostarczone przez algorytm inny, niż algorytm i–$TSAB_{AGV}$ czy $GAGX3$.

Na podstawie analizy tabeli 4.5 można poczynić jeszcze jedno spostrzeżenie. Widoczna jest wyraźna przewaga algorytmu i–$TSAB_{AGV}$ nad algorytmem $GAGX3$ dla instancji o większym rozmiarze. Fakt ten jest szczególnie widoczny w przypadku instancji $TR36/4/1/2/2 - TR40/4/2/5/5$. Dla pozostałych instancji przewaga ta nie jest już tak wyraźnie zarysowana. Można również odnieść wrażenie, że algorytm $GAGX3$ generuje gorsze rozwiązania dla instancji, gdzie czasy wykonania operacji transportowych są stosunkowo długie.

4.3.4 Wyniki badań algorytmów popraw – instancje HK

Poniżej prezentuje się wyniki drugiego etapu badań numerycznych. Etap ten polega na porównaniu wyników wygenerowanych przez algorytmy prezentowane w tym rozdziale z wynikami dostarczonymi przez algorytmy opisywane w literaturze, zaprojektowane do rozwiązywania problemu $J, DR1|t_{jkl}, t'_{kl}|C_{\max}$. Badania będą przeprowadzone przy użyciu 30 instancji testowych zaproponowanych przez Hurinka i Knust w pracach [45, 50]. Instancje te, od nazwisk autorów będą dalej nazywane instancjami HK. Wspomniane instancje zostały też szczegółowo opisane w sekcji 3.4.1.

W pracach [45,50] przeprowadzono badania numeryczne trzech różnych algorytmów bazujących na technice poszukiwań z zabronieniami. Z różnymi wartościami parametrów kontrolnych, wszystkie algorytmy były uruchomione łącznie 24 razy na stacji roboczej Sun Ultra 2 z ograniczeniem czasowym, równym 10 minut. Dodatkowo, najlepszy spośród testowanych algorytmów został uruchomiony z ograniczeniem czasowym, równym jednej godzinie. Przez C^{HK} będzie notowana wartość funkcji celu najlepszego rozwiązania odnalezionego dla każdej instancji w trakcie wszystkich uruchomień wspomnianych algorytmów. Algorytmy opisywane w pracach [45, 50] będą nazywane algorytmami HK.

Badania numeryczne algorytmów $TSAB_{AGV}$, i–$TSAB_{AGV}$, SW, $GADS$, $GAGX$ dla omawianych instancji zostały przeprowadzone w sposób analogiczny jak w przypadku instancji TR. Algorytmy zo-

stały uruchomione z identyczną liczbą powtórzeń oraz identycznymi wartościami poszczególnych parametrów. Przyjmuje się też identyczne oznaczenia określające najlepsze wartości funkcji celu C^A, $A \in Z$, $Z = \{TS, IT, SW, DS, GX1, GX2, GX3\}$ (wielkości C^{SWS}, C^{DSS} nie będą analizowane) oraz sumaryczne czasy pracy T^B, $B \in \{TS, IT, SW, DS\}$; średnie arytmetyczne T^B_{av} poszczególnych wielkości T^B zostały wyznaczone dla instancji 1-10 oraz 11-30. Nie ulega też zmianie sposób generowania rozwiązań/populacji początkowych. Wyjątkiem są tu algorytmy $TSAB_{AGV}$ oraz i–$TSAB_{AGV}$. Rozwiązanie początkowe dla tych algorytmów było generowane przez algorytm konstrukcyjny $INT1$ opisany w pracy [75]. Algorytm $INT1$, w przeciwieństwie do algorytmu $INT3$, nie posiada mechanizmów efektywnego wyznaczania najlepszej pozycji dla wstawianej operacji (patrz. sekcja 3.3.1). Fakt ten jest przyczyną powstawania niewielkich różnic w rozwiązaniach konstruowanych przez oba algorytmy. Poza tą drobną, aczkolwiek istotną różnicą, idea działania obu algorytmów jest taka sama. Sumaryczne czasy pracy algorytmów popraw oraz ich średnie zostały umieszczone w tabeli 4.6. Poza wielkościami C^A, $A \in Z$, poniżej będzie również rozważana wielkość C^{IN}, oznaczająca wartość funkcji kryterialnej najlepszych rozwiązań dostarczonych dla poszczególnych instancji przez algorytm $INT3$.

Dla każdej instancji, korzystając z równania z równania (4.26), wyznaczona została poprawa Φ^{HK}_A wielkości C^{HK} przez poszczególne algorytmy, gdzie $A \in Z \cup \{IN\}$. Wyznaczone zostały też średnie arytmetyczne popraw dla instancji 1-10 oraz 11-30. Poszczególne wielkości C^{HK} oraz ich poprawy zostały zaprezentowane w tabeli 4.7; największe poprawy Φ^{HK}_A dla poszczególnych instancji zostały pogrubione.

Na początku będą przedyskutowane wyniki badań algorytmów popraw dla instancji 1-10. Generalnie, rozwiązania wygenerowane przez poszczególne algorytmy bardzo nieznacznie różnią się od siebie pod względem jakości. Wiąże się to najprawdopodobniej ze stosunkowo niewielkim rozmiarem poszczególnych instancji, gdzie zbiór operacji produkcyjnych zawiera zaledwie $n^p = 36$ operacji. W dominującej części przypadków wartość funkcji celu C^A, $A \in Z$, jest równa wielkości C^{HK}. Najmniejszą poprawę, wynoszącą $-2,24\%$, zanotowano dla instancji o numerze 1 i algorytmu SW. Poprawę największą zanotowano dla instancji 7; poprawa ta wynosi $\Phi^{HK}_{IT} = 1,39\%$.

Analizując jakość badanych algorytmów w kontekście instancji o numerach 1-10 można poczynić podobne spostrzeżenia jak dla instancji TR; najgorsze wyniki wygenerowały algorytmy SW oraz $GADS$. W tym przypadku algorytm $GADS$ okazał się jednak minimalnie lepszy (w kontekście popraw $av\Phi^{HK}_{DS}$, $av\Phi^{HK}_{SW}$) od algorytmu SW. Należy jednak pamiętać,

Tabela 4.6: Czasy pracy algorytmów popraw dla instancji HK

Instancja		T^{TS}	T^{IT}	T^{SW}	T^{DS}
Nr	Nazwa	$[s]$	$[s]$	$[s]$	$[s]$
1	$P1\_t_{jkl}\_t'_{kl}.1$	30,4	537,4	39,7	74,8
2	$P1\_t_{jkl}\_t'_{kl}.2$	28,5	400,2	37,1	73,6
3	$P1\_t_{jkl}\_t'_{kl}.3$	30,1	595,1	38,7	71,3
4	$P1\_D1\_d1$	27,3	276,0	33,5	71,6
5	$P1\_D1\_t1$	14,7	347,2	30,1	75,3
6	$P1\_D2\_d1$	23,8	414,4	40,3	73,6
7	$P1\_D3\_d1$	24,4	284,8	43,6	72,1
8	$P1\_t_{kl}\_t'_{kl}.1$	27,2	649,2	37,3	69,0
9	$P1\_T2\_t1$	13,5	242,8	30,1	75,9
10	$P1\_T3\_t0$	10,4	92,2	32,9	75,5
Średnia		23,0	383,9	36,3	73,3
11	$P2\_D1\_d1$	24,8	381,4	66,5	423,1
12	$P2\_D1\_t0$	12,1	294,1	57,3	416,8
13	$P2\_D1\_t1$	9,4	330,8	56,7	418,5
14	$P2\_D2\_d1$	15,4	378,4	83,1	429,3
15	$P2\_D3\_d1$	43,7	1200,0	126,6	438,2
16	$P2\_D5\_t2$	286,7	1200,0	226,4	459,3
17	$P2\_T1\_t1$	8,9	133,4	53,5	415,4
18	$P2\_T2\_t1$	12,3	168,0	55,7	415,9
19	$P2\_T5\_t2$	19,4	289,3	70,4	425,6
20	$P2\_t_{jkl}\_t'_{kl}.1$	8,1	176,5	55,9	421,2
21	$P2\_t_{jkl}\_t'_{kl}.2$	12,5	233,4	55,9	422,6
22	$P2\_t_{jkl}\_t'_{kl}.3$	16,9	166,2	52,4	419,5
23	$P2\_t_{jkl}\_t'_{kl}.4$	9,9	242,8	59,6	425,5
24	$P2\_t_{kl}\_t'_{kl}.1$	12,0	367,3	61,8	421,7
25	$P2\_t_{kl}\_t'_{kl}.2$	14,5	371,0	65,3	424,0
26	$P2f0.5\_D1\_d1$	25,8	638,7	121,1	427,1
27	$P2f0.5\_D1\_t1$	13,2	267,7	73,9	423,1
28	$P2f0.5\_D2\_d1$	168,5	1200,0	211,5	439,8
29	$P2f0.5\_D2\_t0$	33,4	512,8	121,3	441,3
30	$P2f0.5\_D2\_t1$	113,3	1200,0	163,5	447,7
Średnia		43,0	487,6	91,9	427,8

że dla każdej instancji sumaryczny czas pracy T^{DS} algorytmu genetycznego był prawie dwukrotnie dłuższy od czasu pracy T^{SW} algorytmu symulowanego wyżarzania. Algorytm SW wygenerował rozwiązanie wyższej jakości (świadczą o tym poprawy Φ_{SW}^{HK}) niż algorytmy HK jedynie dla instancji

Tabela 4.7: Wielkości C^{HK} oraz arytmetyczne poprawy Φ_A^{HK} dla instancji HK

Nr	C^{HK}	Φ_{IN}^{HK} [%]	Φ_{TS}^{HK} [%]	Φ_{IT}^{HK} [%]	Φ_{SW}^{HK} [%]	Φ_{DS}^{HK} [%]	Φ_{GX1}^{HK} [%]	Φ_{GX2}^{HK} [%]	Φ_{GX3}^{HK} [%]
1	134	-10,45	-0,75	**0,00**	-2,24	**0,00**	**0,00**	**0,00**	**0,00**
2	129	-10,08	**0,00**	**0,00**	**0,00**	**0,00**	**0,00**	**0,00**	**0,00**
3	144	-10,42	**0,69**	**0,69**	-0,69	**0,69**	**0,69**	**0,69**	**0,69**
4	87	-8,05	**0,00**	**0,00**	**0,00**	-2,30	**0,00**	**0,00**	**0,00**
5	81	-8,64	**1,23**	**1,23**	0,00	0,00	0,00	0,00	**1,23**
6	148	-5,41	**-0,68**	**-0,68**	-1,35	**-0,68**	**-0,68**	**-0,68**	**-0,68**
7	216	-2,78	0,00	**1,39**	0,00	0,00	0,00	0,46	0,46
8	137	-5,11	**0,73**	**0,73**	**0,73**	**0,73**	**0,73**	**0,73**	**0,73**
9	74	-6,76	**0,00**	**0,00**	**0,00**	**0,00**	**0,00**	**0,00**	**0,00**
10	92	-5,43	**0,00**	**0,00**	**0,00**	**0,00**	**0,00**	**0,00**	**0,00**
Średnia		-7,31	0,12	**0,34**	-0,36	-0,16	0,07	0,12	0,24
11	990	-10,00	2,02	**3,33**	2,42	1,21	**3,33**	**3,33**	**3,33**
12	989	-9,10	3,13	3,24	3,13	2,12	**3,44**	**3,44**	**3,44**
13	989	-9,10	2,12	2,83	2,83	0,61	**3,44**	**3,44**	**3,44**
14	993	-14,40	**1,31**	**1,31**	0,10	-2,72	0,20	1,01	1,21
15	1070	-11,31	2,71	**4,58**	2,43	-1,31	2,34	3,64	3,64
16	1325	-7,55	0,45	**3,62**	2,42	2,57	2,64	2,64	2,64
17	1006	-5,47	5,17	**6,86**	5,17	3,68	**6,86**	**6,86**	**6,86**
18	1015	-3,65	5,71	5,71	5,12	5,12	**7,29**	**7,29**	**7,29**
19	1020	-9,80	**4,51**	**4,51**	3,33	1,76	**4,51**	**4,51**	**4,51**
20	1027	-10,71	**5,94**	**5,94**	4,28	4,38	5,84	**5,94**	**5,94**
21	1033	-9,20	6,68	6,68	6,29	4,84	**6,87**	**6,87**	**6,87**
22	989	-9,50	2,93	2,93	2,22	0,91	**3,54**	**3,54**	**3,54**
23	997	-13,64	2,11	3,11	2,11	1,40	2,11	2,51	**3,61**
24	1018	-11,89	4,03	**5,21**	5,01	3,63	**5,21**	**5,21**	**5,21**
25	1014	-10,26	4,24	**5,03**	4,14	2,76	4,73	**5,03**	**5,03**
26	555	-8,29	2,34	**3,96**	3,24	0,18	2,88	3,42	3,60
27	542	-10,15	2,40	**4,06**	3,51	0,92	3,32	3,51	3,51
28	633	-14,53	-2,84	-1,11	**2,21**	-2,37	-0,79	0,95	0,95
29	578	-9,00	1,90	**4,33**	3,29	1,04	3,46	3,46	3,63
30	613	-6,04	0,00	**7,01**	5,87	2,77	5,22	5,22	5,71
Średnia		-9,68	2,84	4,16	3,46	1,68	3,82	4,09	**4,20**

nr 8, zaś gorszy okazał się trzykrotnie, dla instancji nr 1, 3 oraz 6. Algorytm $GADS$ okazał się dwukrotnie gorszy (instancje 4, 6) i dwukrotnie lepszy (instancje 3, 8) od algorytmów HK. Algorytmem pracującym średnio najkró-

cej i najdłużej jest odpowiednio algorytm $TASB_{AGV}$ i algorytm $GAGX3$. Średni czas pracy T_{av}^{IT} algorytmu i–$TSAB_{AGV}$ porównywalny jest z czasem pracy algorytmu $GAGX2$, przy jednocześnie wyższej jakości generowanych rozwiązań. Z kolei, jakość rozwiązań generowanych przez algorytm $TSAB_{AGV}$ jest porównywalna z jakością rozwiązań zwróconych przez algorytm $GAGX1$, przy czym pierwszy z wymienionych algorytmów pracował w czasie znacznie krótszym (średnio niemal dziesięciokrotnie krócej) niż algorytm drugi. W sumie, każdy z algorytmów i–$TSAB_{AGV}$ i $GAGX1$-3 dostarczył rozwiązania gorszego od algorytmów HK jedynie dla jednej (instancja numer 6) spośród 10 omawianych instancji. Algorytm $TSAB_{AGV}$ okazał się gorszy od algorytmów HK dla instancji 1 oraz 6.

Przejdźmy do omówienia instancji 11-30. W tym przypadku charakterystyczny jest fakt, że wszystkie wartości C^{HK} zostały poprawione. Co więcej, jedynie nieliczne, spośród wielkości Φ_A^{HK}, $A \in Z$, są ujemne. Podobnie jak w przypadku instancji TR, najgorszym okazał się algorytm $GADS$, który wygenerował znacznie gorsze rozwiązania niż algorytm SW czy $TSAB_{AGV}$. Jednocześnie, sumaryczny czas jego pracy T^{DS} jest znacznie dłuższy od czasów T^{TS}, T^{SW}. Jednakże, nawet ten algorytm okazał się lepszy od algorytmów HK, generując wartości funkcji celu gorsze od wielkości C^{HK} jedynie dla instancji o numerach 14, 15 i 28. Średnia arytmetyczna poprawa wynosi $av\Phi_{DS}^{HK} = 1,68\%$. Algorytmem pracującym najkrócej, jednocześnie generującym rozwiązania stosunkowo niskiej jakości, jest algorytm $TSAB_{AGV}$. W tym przypadku poprawa minimalna, maksymalna i średnia arytmetyczna popraw wyniosła odpowiednio $-2,84\%$, $5,94\%$ oraz $2,84\%$. Należy jednak pamiętać, że podobnie jak dla instancji TR, algorytm ten był bardziej testowany jako składnik algorytmu i–$TSAB_{AGV}$, niż niezależny algorytm. Na wyróżnienie zasługuje również algorytm SW, generujący rozwiązania stosunkowo wysokiej jakości w stosunkowo krótkim czasie.

Algorytmami generującymi najlepsze rozwiązania znów są algorytmy i–$TSAB_{AGV}$ oraz $GAGX3$. Średnia poprawa w obu przypadkach przekroczyła 4%. Średni czas pracy algorytmu i–$TSAB_{AGV}$ jest jednak nieco krótszy niż czas pracy algorytmu $GAGX3$. W przypadku instancji HK, dystans pomiędzy jakością rozwiązań generowanych przez oba algorytmy wyraźnie zmalał. Fakt ten można uzasadnić tendencją zaobserwowaną już w przypadku instancji TR, mianowicie spadkiem przewagi algorytmu i–$TSAB_{AGV}$ nad algorytmem $GAGX3$ dla instancji o niewielkim rozmiarze. W tym momencie należy przypomnieć, że zbiór operacji produkcyjnych analizowanych instancji zawiera jedynie $n^p = 100$ elementów. Instancje te są zatem znacznie mniejsze niż najmniejsza spośród instancji TR. Potwierdza się również druga z zaobserwowanych tendencji: algorytm $GAGX3$ generuje gorsze roz-

wiązania dla instancji, gdzie czasy wykonania operacji transportowych są stosunkowo długie. Do tej grupy zdecydowanie należą instancje $P2\_D2\_d1$-$P2\_D5\_t2$ oraz instancje $P2f0.5\_D1\_d1$-$P2f0.5\_D2\_t1$, dla których, z wyjątkiem instancji nr 28, algorytm i–$TSAB_{AGV}$ zwrócił rozwiązania lepsze niż algorytm $GAGX3$. Ogółem, algorytm i–$TSAB_{AGV}$ i algorytm $GAGX3$ dostarczył najlepszych rozwiązań odpowiednio dla 13 i 12 spośród 20 omawianych instancji, przy czym wielkości C^{IT}, C^{GX3} zrównały się jedynie sześć razy. Można zatem stwierdzić, że algorytmy te „uzupełniają się wzajemnie" w trakcie swojej pracy.

4.4 Wnioski i uwagi

W tym rozdziale zaprezentowano szereg algorytmów rozwiązywania problemu gniazdowego z dedykowanymi wózkami AGV. Wśród algorytmów konstrukcyjnych można wymienić algorytmy priorytetowe oraz wykorzystujące tzw. „technikę wstawień". Zaproponowane algorytmy popraw wykorzystują szereg najpopularniejszych technik lokalnych poszukiwań, wśród których można wymienić technikę symulowanego wyżarzania, poszukiwań z zabronieniami czy poszukiwania rozproszonego. Szczególnie interesujące wydaje się jednak podejście genetyczne. W jednym z zaproponowanych algorytmów tej klasy użyto nowego quasi-operatora krzyżowania, nazwanego operatorem GX. W większości algorytmów użyto sąsiedztwa oraz efektywnej metody jego przeglądu, prezentowanej w rozdziale 3. W trakcie pracy niektórych algorytmów wykorzystywane są też własności rozwiązania i grafu częściowego, omawiane w poprzednim rozdziale.

Algorytmami zwracającymi najlepsze rozwiązania rozważanego problemu okazały się algorytmy i–$TSAB_{AGV}$ oraz $GAGX$. Pierwszy z nich jest algorytmem opartym na schemacie poszukiwań rozproszonych, zaś drugi jest algorytmem genetycznym. Porównując jednak budowę obu algorytmów można dopatrzyć się szeregu podobieństw; oba algorytmy w trakcie poszukiwań wykorzystują zbiór rozwiązań. W przypadku algorytmu i–$TSAB_{AGV}$ jest to *zbiór rozwiązań elitarnych*, wygenerowanych przy użyciu algorytmu $TSAB'_{AGV}$ i procedury NIS. W przypadku algorytmu $GAGX$ *populacja* zawiera rozwiązania wygenerowane losowo. W każdej iteracji fazy zasadniczej algorytmu i–$TSAB_{AGV}$ ze zbioru rozwiązań wybierane jest rozwiązanie *najlepsze* oraz *najbardziej oddalone* od najlepszego w sensie miary tau Kendalla. W algorytmie $GAGX$ z populacji każdorazowo wybierane jest *jedno z najlepszych* rozwiązań. Rozwiązanie drugie wybierane jest *w sposób losowy*. W przypadku obu algorytmów generowane jest rozwiązanie nowe, leżące pomiędzy wcześniej wybranymi rozwiązaniami w sensie miary tau

Kendalla. Algorytm *i–*$TSAB_{AGV}$ wykorzystuje w tym celu procedurę NIS, zaś algorytm $GAGX$ – operator GX. Oba algorytmy wykorzystują też dodatkowe algorytmy poprawy nowo wygenerowanego rozwiązania. Pierwszy z nich wykorzystuje algorytm $TSAB_{AGV}$, drugi – algorytm SW', wykorzystujący *ten sam* zbiór ruchów, *to samo* sąsiedztwo i *te same* efektywne metody przeglądu sąsiedztwa, co algorytm $TSAB_{AGV}$. Ciekawym zagadnieniem badawczym wydaje się zatem próba wykorzystania komponentów obu algorytmów do konstrukcji algorytmu jeszcze bardziej efektywnego, nie posiadającego wad swoich poprzedników.

Niewątpliwie najprostszym i najbardziej „naiwnym" algorytmem poprawy, jaki można sobie wyobrazić jest algorytm przeszukujący przestrzeń rozwiązań w sposób losowy. Przykładowy algorytm tego typu, nazywany dalej algorytmem RS, można utworzyć poprzez iterowanie z ograniczeniem czasowym 10 minut algorytmu konstrukcyjnego $AP(RAN)$. Algorytm taki jest w stanie przeglądnąć od około 250 tysięcy do ponad 7 milionów rozwiązań, w zależności rozmiaru instancji. Należy się spodziewać, że rozwiązania uzyskane w ten sposób będą jakościowo znacznie gorsze od rozwiązań wygenerowanych przez bardziej zaawansowane algorytmy poprawy. I odwrotnie, każdy algorytm poprawy systematycznie generujący rozwiązania gorsze niż algorytm RS nie zasługuje na miano algorytmu „zaawansowanego". W ogólności, algorytm RS okazał się gorszy bądź znacznie gorszy od każdego z pozostałych badanych algorytmów zarówno dla instancji TR jak i HK. Jednakże, poszczególne algorytmy sporadycznie generowały rozwiązania jakościowo gorsze, niż algorytm RS. Przykładem może tu być instancja $TR36/2/1/5/5$ i algorytm $TSAB_{AGV}$, dla którego poprawa wartości funkcji celu zwróconej prze algorytm RS wyniosła zaledwie $-7,0\%$. Studium takich „kłopotliwych" instancji również może stanowić duży krok w kierunku dalszej eliminacji słabości badanych algorytmów.

Rozdział 5

Problem gniazdowy ze swobodnym przydziałem wózków AGV

W elastycznych systemach produkcyjnych, w ramach poszczególnych zadań, wytwarzane są z reguły detale różniące się między sobą i wymagające wykonania różnych operacji. Detale te mogą również wymagać różnych metod transportu, co w konsekwencji może wiązać się z koniecznością zróżnicowania środków transportu w systemie. Systemy produkcyjne wykorzystujące nieidentyczne wózki AGV można zasadniczo podzielić na dwie grupy. W grupie pierwszej znajdują się systemy wykorzystujące wózki dedykowane. Ta grupa systemów, w której każdy z detali może być transportowany przez jeden, z góry narzucony wózek, była szeroko dyskutowana w rozdziałach 3, 4. Do grupy drugiej należą systemy z wielozadaniowymi środkami transportu. To oznacza, że transport poszczególnych detali może być wykonany przez dowolny wózek z pewnego podzbioru wszystkich środków transportu.

W uogólnionym problemie gniazdowym, omawianym w tym rozdziale, uwzględnia się czasy transportu zadań pomiędzy poszczególnymi maszynami produkcyjnymi; detale wykonywane w ramach poszczególnych zadań mogą być transportowane przez dowolny wózek AGV z pewnego podzbioru wszystkich wózków. Rozważa się maszynowo-zależne i zadaniowo-zależne czasy transportu oraz maszynowo-zależne czasy przejazdów bez załadunku. Zakłada się również, że wózki pracują w trybie „zabierz i zostaw". Za kryterium optymalizacji przyjmuje się moment zakończenia wykonywania wszystkich zadań. W początkowych sekcjach rozdziału przedstawia się modele matematyczne oraz własności problemu. Dalej przedstawia się

algorytmy rozwiązywania problemu i wyniki badań numerycznych prezentowanych algorytmów.

Stosując omawianą w poprzednich rozdziałach rozszerzoną notację Grahama, badany problem można zanotować jako problem $J, MPR|t_{jkl}, t'_{kl}|C_{\max}$. Należy zauważyć, że rozważany w poprzednich rozdziałach problem $J, DR|t_{jkl}, t'_{kl}|C_{\max}$ jest przypadkiem szczególnym badanego problemu. W przypadku tym podzbiór wózków AGV mogących wykonać transport danego detalu każdorazowo zawiera jeden element. Ze względu na ten fakt, wiele rozważań i własności prezentowanych w rozdziałach 3, 4 jest również prawdziwe dla problemu $J, MPR|t_{jkl}, t'_{kl}|C_{\max}$ i wymaga jedynie odpowiedniego uogólnienia, do czego sprowadzają się obszerne fragmenty tego rozdziału. Co więcej, przypadkiem szczególnym rozważanego problemu jest również problem $J, R|t_{jkl}, t'_{kl}|C_{\max}$, w którym podzbiór wózków AGV mogących wykonać transport danego detalu jest równy zbiorowi wszystkich środków transportu. To sprawia, że poniższe rozważania są również prawdziwe dla systemów produkcyjnych wykorzystujących identyczne wózki AGV.

5.1 Model matematyczny

Przedstawiony model matematyczny jest zbliżony do modelu problemu omawianego w rozdziale 3. Dlatego poniżej przedstawia się tylko jego najważniejsze elementy. Dany jest zbiór m^p maszyn produkcyjnych $M^p = \{1, \ldots, m^p\}$ i zbiór m^t maszyn transportowych (wózków AGV) $M^t = \{m^p + 1, m^p + 2, \ldots, m^p + m^t\}$; $M = M^p \cup M^t$, $m = |M| = m^p + m^t$. Dany jest zbiór r zadań $J = \{1, \ldots, r\}$. Każde zadanie $k \in J$ składa się z sekwencji n_k^p operacji produkcyjnych, indeksowanych przez $j_k + 1, j_k + 2, \ldots, j_k + n_k^p$, które powinny być wykonane w tej kolejności; wielkość $j_k = \sum_{i=1}^{k-1} n_i^p$, $j_1 = 0$, określa całkowitą liczbę operacji pierwszych $k-1$ zadań. Przez $O^p = \{1, 2, \ldots, n^p\}$ będzie oznaczany zbiór wszystkich operacji produkcyjnych, gdzie $n^p = \sum_{k \in J} n_k^p$. Operacja $i \in O^p$, powinna być wykonana na dowolnej maszynie l ze zbioru $\mu_i \subseteq M^p$ w czasie $p_i > 0$, przy czym $|\mu_i| = 1$. Zakłada się, że każde dwie kolejne operacje produkcyjne tego samego zadania nie mogą być wykonywane na tej samej maszynie produkcyjnej.

Transport zadania $k \in J$ i przejazd pusty maszyny transportowej pomiędzy maszynami $l', l'' \in M^p$, $l' \neq l''$, wymaga odpowiednio $t_k(l', l'') > 0$ i $e(l', l'') > 0$ czasu; $t_k(l', l') = e(l', l') = 0$. Zakłada się, że czasy transportu i przejazdów pustych spełniają nierówności (3.1)-(3.3). Pomiędzy każdą parą sąsiednich operacji produkcyjnych $j_k + i$ oraz $j_k + i + 1$, $1 \leqslant i < n_k^p$,

zadania $k \in J$ jest wykonywana operacja transportowa, oznaczana symbolem $x = [j_k + i]$. Operacja x powinna być wykonana przez dowolną maszynę transportową l pochodzącą z podzbioru maszyn $\mu_x \subseteq M^t$ w czasie $p_x = t_k(l', l'')$, gdzie $l' = m_{j_k+i}$ oraz $l'' = m_{j_k+i+1}$. Zakłada się, że czas transportu p_x zadania x nie zależy od maszyny $l \in \mu_x$. Zbiór wszystkich operacji transportowych będzie oznaczany przez $O^t = \bigcup_{k \in J} \bigcup_{i=1}^{n_k^p - 1} \{[j_k + i]\}$. Całkowita liczba operacji transportowych wynosi $n^t = |O^t| = \sum_{k \in J} n_k^t = n^p - r$, gdzie $n_k^t = n_k^p - 1$ jest liczbą operacji transportowych w zadaniu $k \in J$. W celu wprowadzenia kompletnej notacji definiuje się liczbę wszystkich operacji $n_k = n_k^p + n_k^t$ w zadaniu $k \in J$, zbiór wszystkich operacji $O = O^p \cup O^t$ oraz liczbę wszystkich operacji $n = \sum_{k \in J} n_k$.

Po wykonaniu operacji $x' = [j_g + i]$, $1 \leqslant i \leqslant n_g^t$, $g \in J$, i przed wykonaniem operacji $x'' = [j_k + h]$, $1 \leqslant h \leqslant n_k^t$, $k \in J$, maszyna transportowa wykonuje przejazd pusty pomiędzy maszynami $l' = m_{j_g+i+1}$, $l'' = m_{j_k+h}$, który trwa $e(l', l'')$ jednostek czasu. Przejazd pusty wygodnie jest utożsamić z przezbrojeniem $s(x', x'') = e(l', l'')$ maszyny transportowej, wykonywanym pomiędzy operacjami x', x''. W ten sposób zdefiniowane czasy przezbrojeń spełniają słaby warunek trójkąta, tzn. spełniają nierówność (3.4). W celu uproszczenia dalszej notacji definiuje się również czasy przezbrojeń $s(x', x'')$ pomiędzy każdą parą operacji produkcyjnych $x', x'' \in O^p$, równe $s(x', x'') = 0$ oraz czasy $s(0, 0) = s(x, 0) = s(0, x) = 0$, $x \in O$.

Podobnie jak w klasycznym problemie gniazdowym (patrz. np. sekcja 2.1.2) zakłada się, że:

1. raz rozpoczęta operacja nie może być przerwana,
2. w każdej chwili czasu można wykonywać co najwyżej jedną operację każdego zadania,
3. w każdej chwili czasu każda maszyna (produkcyjna i transportowa) może wykonywać co najwyżej jedną operację.

Uszeregowanie stanowi wektor par $S = ((S_1, m_1), (S_2, m_2), \ldots, (S_n, m_n))$, gdzie $m_i \in \mu_i$ jest maszyną wybraną do wykonania operacji $i \in O$, zaś S_i jest czasem rozpoczęcia jej wykonywania. Uszeregowanie jest dopuszczalne, gdy wszystkie powyższe ograniczenia są spełnione, czego wyrazem jest prawdziwość nierówności (2.7), (2.8), (3.5); wielkość $C_i = S_i + p_i$ oznacza moment zakończenia wykonywania operacji i. Za kryterium optymalizacji przyjmuje się moment zakończenia wykonywania wszystkich zadań, równy $C_{\max} = \max_{i \in O} C_i$. Problem polega na odnalezieniu takiego uszeregowania dopuszczalnego, by kryterium optymalizacji przyjęło wartość minimalną. Problem jest silnie NP-trudny.

5.1.1 Model permutacyjno-grafowy

Poniżej zostanie przedstawiony model permutacyjno-grafowy problemu. W tym celu należy zauważyć, że zbiór operacji O można podzielić na m rozłącznych podzbiorów. Niech $\theta = \{O_1, O_2, \ldots, O_m\}$ oznacza podział rozłączny zbioru O taki, że $\forall_{k \in O} \exists\, !_{l \in \mu_k} k \in O_l$. Niech Θ określa zbiór wszystkich podziałów rozłącznych zbioru O spełniających powyższy warunek. Nie trudno zauważyć, że dla każdego podziału $\theta \in \Theta$ zbiory $O_1, \ldots, O_{m^p}$ są takie same, natomiast zbiory $O_{m^p+1}, O_{m^p+2} \ldots, O_m$ stanowią dowolny podział rozłączny zbioru O^t. Niech $\pi = (\pi_1, \ldots, \pi_m)$ będzie zestawem permutacji podziału rozłącznego, gdzie $\pi_l = (\pi_l(1), \ldots, \pi_l(o_l))$, $o_l = |O_l|$, jest permutacją zbioru O_l, określającą kolejność wykonania operacji z tego zbioru na maszynie l. Wtedy zbiór wszystkich zestawów permutacji podziału rozłącznego można oznaczyć przez

$$\Pi^0 = \{(\pi_1, \ldots, \pi_m) \quad : \quad (\pi_l - \text{permutacja zbioru } O_l, l \in M), \; \{O_1, \ldots, O_m\} \in \Theta\}. \tag{5.1}$$

Dla uproszczenia, w dalszej części rozdziału zestaw permutacji podziału rozłącznego $\pi \in \Pi^0$ będzie nazywany permutacją bądź po prostu rozwiązaniem problemu. Permutacja π reprezentująca uszeregowanie dopuszczalne będzie nazywana permutacją dopuszczalną (rozwiązaniem dopuszczalnym), zaś zbiór wszystkich permutacji dopuszczalnych będzie oznaczany symbolem $\Pi \subset \Pi^0$.

Dla dowolnej permutacji π ze zbioru Π^0 można skonstruować skierowany graf $G(\pi) = (O, E(\pi))$ ze zbiorem obciążonych wierzchołków O i zbiorem łuków $E(\pi) = E^T \cup E^K(\pi)$. Zbiór E^T dany jest równaniem (3.6), zaś zbiór $E^K(\pi)$ dany jest równaniem (2.11). Każdy wierzchołek $i \in O$ grafu przyjmuje obciążenie p_i i reprezentuje operację i. Zbiór nieobciążonych łuków technologicznych E^T określa relacje kolejnościowe pomiędzy operacjami w poszczególnych zadaniach. Zbiór łuków kolejnościowych $E^K(\pi)$ określa kolejność wykonania operacji na poszczególnych maszynach, gdzie obciążenie łuku $(i, j) \in E^K(\pi)$ przyjmuje wartość $s(i, j)$.

Graf $G(\pi)$ jest acykliczny wtedy i tylko wtedy, gdy permutacja π jest permutacją dopuszczalną. Przez r_i^π oraz q_i^π będzie oznaczana długość najdłuższej ścieżki odpowiednio dochodzącej i wychodzącej z wierzchołka $i \in O$ (bez jego obciążenia p_i) w acyklicznym grafie $G(\pi)$. Przyjmuje się też, że $r_0^\pi = q_0^\pi = 0$. Nie trudno zauważyć, że najwcześniejszy możliwy moment rozpoczęcia S_i operacji $i \in O$ jest równy wielkości r_i^π. Długość ścieżki krytycznej $C_{\max}(\pi)$ w grafie $G(\pi)$ jest zatem równa wartości przyjętej funkcji kryterialnej, $C_{\max}(\pi) = \max_{i \in O} C_i = \max_{i \in O}(r_i^\pi + p_i)$. Wobec tego problem sprowadza się do odnalezienia takiej permutacji π^*,

że $C_{\max}(\pi^*) = \min_{\pi\in\Pi} C_{\max}(\pi)$. Warto zauważyć, że wielkości r_i^π oraz q_i^π mogą być wyznaczone rekurencyjnie przy użyciu równań (3.7), (3.8).

5.2 Własności grafu i permutacji częściowej

Własności grafu i permutacji częściowej dla przypadku szczególnego badanego problemu, tj. w sytuacji, w której $|\mu_i| = 1$, $i \in O^t$, zostały szczegółowo omówione w sekcji 3.3. Poniżej pokazuje się, że analogiczne własności są również prawdziwe w sytuacji bardziej ogólnej, w której $|\mu_i| \geqslant 1$, $i \in O^t$. Podobnie jak dyskusja prowadzona w sekcji 3.3, poniższe rozważania mają duże znaczenie przy projektowaniu algorytmów konstrukcyjnych typu wstaw. Rozważania te pełnią też kluczową rolę w opisie metody wyznaczania najlepszego rozwiązania sąsiedniego w sąsiedztwie prezentowanym w sekcji 5.3.

Niech $\theta = \{O_1, \ldots, O_m\} \in \Theta$ będzie podziałem rozłącznym zbioru O. Dla dowolnego podziału θ przyjmuje się definicje zbioru operacji uszeregowanych $U \subseteq O$, permutacji częściowej, zbioru wszystkich permutacji częściowych $\Pi^{0,U}$, grafu częściowego $G(\beta)$, $\beta \in \Pi^{0,U}$ oraz zbioru wszystkich dopuszczalnych permutacji częściowych Π^U z sekcji 3.3. W grafie $G(\beta)$ przez b_i^T (a_i^T) oraz $b_i^K(\beta)$ $(a_i^K(\beta))$ będzie oznaczony odpowiednio bezpośredni poprzednik (następnik) technologiczny oraz bezpośredni poprzednik (następnik) kolejnościowy wierzchołka $i \in O$. Jeśli dany wierzchołek nie posiada któregoś z poprzedników lub następników, to przyjmuje się odpowiednio $b_i^T = a_i^T = b_i^K(\beta) = a_i^K(\beta) = 0$. Zbiór ZB_i oraz ZA_i będzie oznaczać odpowiednio zbiór wszystkich poprzedników i następników wierzchołka $i \in O$. Niech wartość d_i^β będzie miarą długości najdłuższej ścieżki przechodzącej przez wierzchołek $i \in O$ w grafie $G(\beta)$, $\beta \in \Pi^{0,U}$, dla ustalonego podziału $\theta \in \Theta$ i zbioru U. W przypadku acyklicznego grafu $G(\beta)$ przyjmuje się, że wielkość ta jest równa długości najdłuższej ścieżki przechodzącej przez wierzchołek i, danej równaniem (3.42). Miara długości d_i^β jest wykorzystywana we wspomnianych już wyżej algorytmach typu wstaw do oszacowania jakości próbnego wstawienia danej operacji i. Dokładniej, niech

$$Z(\beta, i) = \bigcup_{l\in\mu_i} Z(\beta, l, i), \tag{5.2}$$

gdzie

$$\begin{aligned} &Z(\beta, l, i) = \{(\beta_1, \beta_2, \ldots, \beta_{l-1}, \omega, \beta_{l+1}, \ldots, \beta_m) : \\ &\omega = (\beta_l(1), \ldots, \beta_l(z-1), i, \beta_l(z), \ldots, \beta_l(|\beta_l|)), 1 \leqslant z \leqslant |\beta_l| + 1\}, \end{aligned} \tag{5.3}$$

określa zbiór permutacji częściowych, które zostały utworzone z permutacji częściowej β poprzez wstawienie nie uszeregowanej operacji $i \in O \setminus U$

na pozycje $z = 1, 2, \ldots, |\beta_l| + 1$ w poszczególnych permutacjach β_l, $l \in \mu_i$. W algorytmach konstrukcyjnych typu wstaw należy odnaleźć najlepszą permutację δ w zbiorze $Z(\beta, i)$, tzn. taką, że $C_{\max}(\delta) = \min_{\alpha \in Z(\beta,i)} C_{\max}(\alpha)$, gdzie $C_{\max}(\alpha)$ jest długością najdłuższej ścieżki w grafie częściowym $G(\alpha)$. Wiąże się to z koniecznością wyznaczenia wartości d_i^α dla każdej permutacji częściowej $\alpha \in Z(\beta, i)$, dla której graf $G(\alpha)$ jest acykliczny. Podobnie jak w przypadku, w którym $|\mu_i| = 1$, procedura ta jest bardzo pracochłonna, ponieważ wyznaczenie jednej wartości d_i^α lub stwierdzenie, że graf $G(\alpha)$ jest cykliczny wymaga aż $O(n)$ czasu. Korzystając jednak z poniższych własności można to zrobić w czasie znacznie krótszym.

Własność 5.1 *Dany jest podział rozłączny $\theta \in \Theta$ zbioru O, zbiór operacji $U \subset O$, permutacja częściowa $\beta \in \Pi^U$, nieuszeregowana operacja $i \in O \setminus U$ oraz zbiór $Z(\beta, i)$. Jeżeli graf $G(\alpha)$, $\alpha \in Z(\beta, i)$, jest acykliczny, to długość najdłuższej ścieżki przechodzącej przez wierzchołek i w tym grafie, a tym samym wartość d_i^α, można wyznaczyć przy użyciu odpowiedniego z równań (3.44), (3.45)*

Ze względu na podobieństwo do dowodu analogicznej własności dla klasycznego problemu gniazdowego opisywanego m.in. w pracy [36], dowód powyższej własności będzie pominięty. Należy zauważyć, że dzięki równaniom (3.44), (3.45) możliwe są dwie rzeczy. Po pierwsze, czas wyznaczenia najdłuższej ścieżki przechodzącej przez wierzchołek $i \in O$ w acyklicznym grafie $G(\alpha)$, $\alpha \in Z(\beta, i)$ (a tym samym wielkości d_i^α), redukuje się z $O(n)$ do $O(1)$. Po drugie, wielkość d_i^α można wyznaczyć dla każdego (nie koniecznie acyklicznego) grafu $G(\alpha)$, $\alpha \in Z(\beta, i)$, również w czasie $O(1)$. Z powyższej własności wynika też, że permutacja $\delta \in Z(\beta, i)$ taka, że graf $G(\delta)$ jest acykliczny oraz $d_i^\delta = d_{\min}$, gdzie wielkość $d_{\min}$ dana jest równaniem (3.47), jest najlepszą permutacją w zbiorze $Z(\beta, i)$. Pozostaje jeszcze kwestia znalezienia efektywnej metody określania czy graf $G(\alpha)$, $\alpha \in Z(\beta, i)$, jest acykliczny. Umożliwiają to poniższe dwie własności, w których wykorzystuje się następujący zbiór permutacji

$$ZD(\beta, i) = \bigcup_{l \in \mu_i} ZD(\beta, l, i), \tag{5.4}$$

gdzie

$$ZD(\beta, l, i) = \{\alpha \in Z(\beta, l, i) : d_i^\alpha = d_{\min}\}. \tag{5.5}$$

Własność 5.2 *Dany jest podział rozłączny $\theta \in \Theta$ zbioru O, zbiór operacji $U \subset O$, permutacja częściowa $\beta \in \Pi^U$, nieuszeregowana operacja produkcyjna $i \in O^p \setminus U$, zbiór $Z(\beta, i)$ oraz zbiór $ZD(\beta, i)$ dany równaniem (5.4). Wtedy dla każdej permutacji $\alpha \in ZD(\beta, i)$ graf $G(\alpha)$ jest acykliczny.*

W celu przeprowadzenia dowodu powyższej własności wystarczy zauważyć, że zbiór $ZD(\beta, i)$ jest sumą zbiorów $ZD(\beta, l, i)$, $l \in \mu_i$. Dla każdego niepustego zbioru $ZD(\beta, l, i)$ i ustalonej maszyny $l \in \mu_i$ dowód przeprowadza się podobnie do dowodu analogicznej własności dla klasycznego problemu gniazdowego, prezentowanego m.in. w pracach [36, 65]. Ze względu na to podobieństwo, dowód powyższej własności będzie pominięty.

Własność 5.3 *Dany jest podział rozłączny $\theta \in \Theta$ zbioru O, zbiór operacji $U \subset O$, permutacja częściowa $\beta \in \Pi^U$, nieuszeregowana operacja transportowa $i \in O^t \setminus U$ zbiór $Z(\beta, i)$ oraz zbiór $ZD(\beta, i) = \bigcup_{l \in \mu_i} ZD(\beta, l, i)$. Każdy niepusty zbiór $ZD(\beta, l, i)$, $l \in \mu_i$, zawiera przynajmniej jedną taką permutację α, że graf $G(\alpha)$ jest acykliczny.*

W celu przeprowadzenia dowodu powyższej własności wystarczy zauważyć, że zbiór $ZD(\beta, i)$ jest sumą zbiorów $ZD(\beta, l, i)$, $l \in \mu_i$. Dla każdej ustalonej maszyny $l \in \mu_i$, zbioru $Z(\beta, l, i)$ i niepustego zbioru $ZD(\beta, l, i)$ dowód przeprowadza się analogicznie do dowodu własności 3.11. Ze względu na to podobieństwo, dowód powyższej własności będzie pominięty.

Siła powyższych własności polega na tym, że w celu wyznaczenia najlepszej maszyny i pozycji dla wstawianej operacji i, tzn. wyznaczenia najlepszej permutacji w zbiorze $Z(\beta, i)$, nie potrzeba sprawdzać acykliczności wszystkich grafów $G(\alpha)$, $\alpha \in Z(\beta, i)$. W przypadku operacji produkcyjnych nie trzeba robić tego wcale, zaś dla operacji transportowych $i \in O^t \setminus U$, sprawdzenie wystarczy ograniczyć tylko do $\min_{l \in \mu_i}\{\max\{|ZD(\beta, l, i)|, 1\}\}$ grafów; gdy istnieje takie $l \in \mu_i$, że $|ZD(\beta, l, i)| = 1$ sprawdzenie acykliczności grafu $G(\alpha)$, $\alpha \in ZD(\beta, l, i)$, również nie jest potrzebne.

5.2.1 Efektywna metoda wyznaczania najlepszej maszyny i pozycji

Mając na uwadze powyższe własności oraz postępując podobnie jak w monografii [65] (dla problemu gniazdowego z przezbrojeniami o dowolnym czasie trwania) poniżej zostanie naszkicowana metoda wyznaczania najlepszej pozycji dla wstawianej operacji $i \in O^t \setminus U$. Bez straty ogólności, dalsze rozważania mogą być ograniczone do ustalonego podziału rozłącznego $\theta \in \Theta$, zbioru operacji uszeregowanych $U \subset O$, permutacji częściowej $\beta \in \Pi^U$, acyklicznego grafu $G(\beta)$, nie uszeregowanej operacji transportowej $i \in O^t \setminus U$ i zbioru maszyn μ_i. Niech $er_{l,z} = r^{\beta}_x$, $eq_{l,z} = q^{\beta}_x$, $ep_{l,z} = p_x$ oznacza odpowiednio długość najdłuższej ścieżki dochodzącej i wychodzącej z wierzchołka $x = \beta_l(z)$, $1 \leqslant z \leqslant |\beta_l|$, $l \in \mu_i$, oraz jego obciążenie w grafie $G(\beta)$. Przyjmuje się, że $er_{l,0} = ep_{l,0} = ep_{l,|\beta_l|+1} = eq_{l,|\beta_l|+1} = 0$. Wprowadza się też nieco bardziej „wygodną" definicję miary długości dla

poniższych rozważań. Niech dla każdego $l \in \mu_i$

$$\begin{aligned} ed_{l,z} &= d_i^{\alpha^{l,z}} = \max\{r_i^{\beta}, er_{l,z-1} + ep_{l,z-1} + s(y,i)\} + p_i \\ &+ \max\{q_i^{\beta}, eq_{l,z} + ep_{l,z} + s(i,x)\}, \end{aligned} \tag{5.6}$$

będzie miarą długości najdłuższej ścieżki przechodzącej przez wierzchołek i w grafie $G(\alpha^{l,z})$, $\alpha^{l,z} \in \{\alpha \in Z(\beta,i) : \alpha_l(z) = i\}$, $1 \leqslant z \leqslant |\beta_l| + 1$, gdzie $y = \beta_l(z-1)$, $x = \beta_l(z)$. Metoda wyznaczenia najlepszej maszyny i pozycji dla wstawianej operacji i sprowadza się do procedury składającej się z czterech kroków, przedstawionej na rys. 5.1. Wyznaczona maszyna l^* i pozycja z^* jest najlepszą maszyną i pozycją dla wstawianej operacji i, zaś odpowiadające jej rozwiązanie α^{l^*,z^*} jest najlepszym rozwiązaniem w zbiorze $Z(\beta,i)$. Wartość $C_{\max}(\alpha^{l^*,z^*})$ można wyznaczyć korzystając z równania (3.46). Projektując procedurę wykorzystano fakt, że każdy graf $G(\alpha^{l,z})$ taki, że $\alpha^{l,z} \in \{\alpha \in Z(\beta,i) : \alpha_l(z) = i\}$, $e_l \leqslant z \leqslant f_l$, $e_l \in ZE_l$, $f_l \in ZF_l$, $l \in \mu_i$, gdzie zbiory ZE_l, ZF_l dane są równaniami

$$ZE_l = \max\{1 \leqslant j \leqslant |\beta_l| : \text{istnieje ścieżka z } \beta_l(j) \text{ do } i \text{ w grafie } G(\beta)\}, \tag{5.7}$$

$$ZF_l = \min\{1 \leqslant j \leqslant |\beta_l| : \text{istnieje ścieżka z } i \text{ do } \beta_l(j) \text{ w grafie } G(\beta)\}, \tag{5.8}$$

jest grafem acyklicznym; jeżeli któryś ze zbiorów ZE_l, ZF_l jest pusty, to przyjmuje się odpowiednio $e_l = 0$, $f_l = |\beta_l| + 1$. Powyższy fakt wykazuje się na jednym z etapów dowodu własności 5.3.

Krok 1 i 2 procedury wymaga $O(n)$ czasu, kroki 3, 4 wymagają $O(|\mu_i| \cdot |\beta_l|)$ czasu, więc złożoność obliczeniowa procedury wynosi $O(n)$. Podobnie jak w przypadku analogicznej procedury przedstawionej na rys. 3.6, poniżej wykazuje się, że nie jest konieczne wyznaczanie pozycji $e_l \in ZE_l$, $f_l \in ZF_l$, $l \in \mu_i$ (zatem i zbiorów $ZB_j(\beta)$ $ZA_k(\beta)$), co w omawianym przypadku wiąże się ze znacznym zmniejszeniem praktycznej złożoności obliczeniowej procedury.

W dalszych rozważaniach będą wykorzystane wielkości

$$d_{l,\min} = \min_{1 \leqslant z \leqslant |\beta_l|+1} d_{l,z}, \qquad l \in \mu_i, \tag{5.9}$$

wielkość

$$d_{\min} = \min_{l \in \mu_i} d_{l,\min}, \tag{5.10}$$

zbiór maszyn

$$ZM = \{l \in \mu_i : d_{l,\min} = d_{\min}\} \tag{5.11}$$

oraz nierówności

$$ed_{l,z} \geqslant ed_{l,e_l+1}, \qquad l \in \mu_i, z = 1, \ldots, e_l, \tag{5.12}$$

Procedura rozpoczyna działanie z ustaloną permutacją częściową β i nie uszeregowaną operacją transportową i. Procedura zwraca najlepszą maszynę l^* i pozycję z^* dla wstawianej operacji i.

1. wyznacz zbiory $ZB_j(\beta)$, $ZA_k(\beta)$ dla $j = b_i^T$, $k = a_i^T$.
2. Dla każdej maszyny $l \in \mu_i$ wyznacz wartości $er_{l,z}$, $eq_{l,z}$, $1 \leqslant z \leqslant |\beta_l| + 1$ oraz r_i^β, q_i^β.
3. Dla każdej maszyny $l \in \mu_i$ wyznacz pozycje $e_l \in ZE_l$, $f_l \in ZF_l$, gdzie zbiory ZE_l, ZF_l dane są odpowiednimi równaniami (5.7), (5.8).
4. Oblicz wartości $ed_{l,z}$, $l \in \mu_i$, $e_l < z \leqslant f_l$ i wśród nich znajdź wartość najmniejszą ed_{l^*,z^*}. Zwróć maszynę l^*, pozycję z^* i STOP.

Rysunek 5.1: Metoda wyznaczania najlepszej maszyny i pozycji dla wstawianej operacji i

$$ed_{l,z} \geqslant ed_{l,f_l}, \qquad l \in \mu_i, z = f_l + 1, \ldots, |\beta_l| + 1, \tag{5.13}$$

których prawdziwość jest konsekwencją prawdziwości własności 5.3. Istotną rolę będą też pełniły pozycje

$$a_l = \min\{1 \leqslant z \leqslant |\beta_l| + 1 : ed_{l,z} = d_{\min}\}, \qquad l \in ZM, \tag{5.14}$$

$$b_l = \max\{1 \leqslant z \leqslant |\beta_l| + 1 : ed_{l,z} = d_{\min}\}, \qquad l \in ZM. \tag{5.15}$$

Z definicji pozycji a_l, b_l i nierówności (5.12), (5.13) wynika prawdziwość nierówności

$$a_l \leqslant f_l, \qquad l \in ZM, \tag{5.16}$$

$$e_l < b_l, \qquad l \in ZM. \tag{5.17}$$

Prawdziwa jest także następująca własność.

Własność 5.4 *Dany jest podział rozłączny $\theta \in \Theta$ zbioru O, zbiór operacji uszeregowanych $U \subset O$, permutacja częściowa $\beta \in \Pi^U$, acykliczny graf $G(\beta)$, nieuszeregowana operacja transportowa $i \in O^t \setminus U$ i zbiór maszyn μ_i. Niech*

$$c_l = \max\{1 \leqslant z \leqslant |\beta_l| + 1 : er_{l,z-1} + ep_{l,z-1} \leqslant r_i^\beta\}, \qquad l \in \mu_i. \tag{5.18}$$

Wtedy $e_l < c_l \leqslant f_l$, czyli graf $G(\alpha^{l,c_l})$ jest acykliczny.

Procedura rozpoczyna działanie z ustaloną dopuszczalną permutacją częściową β, nie uszeregowaną operacją transportową i i zbiorem maszyn μ. Procedura zwraca najlepszą maszynę l^* i pozycję z^* dla wstawianej operacji i.

1. wyznacz wartości r_i^β, q_i^β oraz wartości $er_{l,z}$, $eq_{l,z}$, $1 \leqslant z \leqslant |\beta_l|$, $l \in \mu$.
2. wyznacz wartości $ed_{l,z}$, $ed_{l,\min}$, $1 \leqslant z \leqslant |\beta_l|+1$, $l \in \mu_i$, wyznacz wielkość $d_{\min}$ i zbiór ZM.
3. Dla dowolnej maszyny $l^* \in ZM$ wyznacz pozycje a_{l^*}, b_{l^*}, c_{l^*}, dane odpowiednimi równaniami (5.14), (5.15), (5.18) oraz określ pozycję z^* wykorzystując równanie (5.19). Zwróć maszynę l^*, pozycję z^* i STOP.

Rysunek 5.2: Schemat procedury $PWNP2$

Dowód przeprowadza się analogicznie do dowodu własności 3.12 i może być pominięty. Z powyższych rozważań oraz nierówności (5.12), (5.13) wynika, że prawdziwe jest następujące twierdzenie, które stanowi podstawę przy formułowaniu ostatecznej postaci procedury wyznaczającej najlepsze rozwiązanie w zbiorze $Z(\beta, i)$.

Twierdzenie 5.1 *Dany jest podział rozłączny $\theta \in \Theta$ zbioru O, zbiór operacji $U \subset O$, permutacja częściowa $\beta \in \Pi^U$, acykliczny graf $G(\beta)$, nieuszeregowana operacja transportowa $i \in O^t \setminus U$ i dowolna maszyna $l^* \in ZM$. Niech*

$$z^* = \begin{cases} \min\{c_{l^*} \leqslant z \leqslant b_{l^*} : ed_{l^*,z} = d_{\min}\}, & a_{l^*} < c_{l^*} < b_{l^*}, \\ b_{l^*}, & b_{l^*} \leqslant c_{l^*}, \\ a_{l^*}, & c_{l^*} \leqslant a_{l^*}. \end{cases} \tag{5.19}$$

Wtedy graf $G(\alpha^{l^*,z^*})$, $\alpha^{l^*,z^*} \in \{\alpha \in Z(\beta,i) : \alpha_{l^*}(l^*,z^*) = i\}$, jest acykliczny oraz $ed_{l^*,z^*} = d_{\min}$.

Dowód powyższego twierdzenia jest podobny do dowodu twierdzenia 3.2 i może być pominięty. Procedura przyspieszonego wyznaczania najlepszej maszyny i pozycji ($PWNP2$), wykorzystująca powyższe twierdzenia, przedstawiona jest na rysunku 5.2. W celu wyznaczenia najlepszej maszyny i pozycji, procedurę należy zastosować do permutacji β, operacji i oraz zbioru maszyn μ_i. Krok 1 procedury wykonywany jest w czasie $O(n)$, kroki 2, 3 wymagają odpowiednio $O(\sum_{l\in\mu_i}\{|\beta_l|+1\})$ oraz $O(|\beta_{l^*}|+1)$ czasu. Wyznaczona maszyna l^* i pozycja z^* jest najlepszą maszyną i pozycją dla

wstawianej operacji i, zaś odpowiadające jej rozwiązanie α^{l^*,z^*} jest najlepszym rozwiązaniem w zbiorze $Z(\beta, i)$. Wartość $C_{\max}(\alpha^{l^*,z^*})$ można wyznaczyć korzystając z równania (3.46).

5.3 Zbiór ruchów i sąsiedztwo

Jednym z podstawowych problemów przy opracowaniu algorytmów lokalnych poszukiwań jest dobór odpowiedniego sąsiedztwa. Należy uwzględnić, że każda operacja transportowa $i \in O^t$ może być wykonana na dowolnej maszynie transportowej ze zbioru μ_i. Dobrym pomysłem wydaje się rozszerzenie jednego ze zbiorów ruchów omawianych w rozdziale 3 (generowanego w oparciu o ruchy typu zamień sąsiednie operacje) o ruchy typu wstaw. Ważne jest jednak, aby ruchy te prowadziły do uszeregowań dopuszczalnych. Niestety fakt występowania niezerowych czasów przejazdów pustych (interpretowanych jako czasy przezbrojeń) sprawia, że „klasyczne" własności [36, 65, 70], dotyczące dopuszczalności uszeregowania, nie są teraz prawdziwe. Pojawia się zatem problem konstrukcji efektywnej metody selekcji rozwiązań dopuszczalnych, jak również wyboru najlepszego rozwiązania z powstałego sąsiedztwa. W tej sekcji przedstawia się zbiór ruchów, sąsiedztwo oraz metodę wyznaczania najlepszego rozwiązania w tym sąsiedztwie, której implementacja w zadowalający sposób rozwiązuje wspomniany problem.

W celu określenia zbioru ruchów konieczne jest bardziej precyzyjne zdefiniowanie ścieżki krytycznej. Ścieżka krytyczna (wybrana arbitralnie) w acyklicznym grafie $G(\pi)$, $\pi \in \Pi$, będzie przedstawiana jako ciąg wierzchołków $u = (u_1, u_2, \ldots, u_{lu})$, $u_i \in O$, $1 \leqslant i \leqslant lu$, gdzie lu jest liczbą wierzchołków ścieżki. Ścieżkę u w jednoznaczny sposób można podzielić na lb rozłącznych ciągów $u = (B_1, B_2, \ldots, B_{lb})$, gdzie $B_h = (\pi_{l_h}(e_h), \pi_{l_h}(e_h + 1), \ldots, \pi_{l_h}(f_h))$, $1 \leqslant e_h \leqslant f_h \leqslant o_{l_h}$, $1 \leqslant h \leqslant lb$, jest h-tym blokiem operacji na ścieżce krytycznej u. Każdy blok B_h jest ciągiem:

1. zawierającym operacje wykonywane na tej samej maszynie (oznaczonej przez $l_h \in M$) oraz
2. operacje w bloku B_h są wykonywane na maszynie innej, niż operacje w bloku B_{h-1}, tj. $l_{h-1} \neq l_h$, $h = 2, \ldots, lb$.

Blok B_h zbudowany z operacji produkcyjnych i transportowych, tzn. taki, że odpowiednio $l_h \in M^p$ i $l_h \in M^t$, będzie nazywany odpowiednio blokiem operacji produkcyjnych i blokiem operacji transportowych. Z własności grafu wynika, że pomiędzy każdą parą bloków operacji produkcyjnych znajduje się jeden i tylko jeden blok operacji transportowych; pierwszym i ostatnim blokiem na ścieżce krytycznej jest blok operacji produkcyjnych.

W dalszych rozważaniach również będzie użyteczny zbiór

$$ZV(\pi) = \bigcup_{h=1}^{lb} ZV_h(\pi), \tag{5.20}$$

gdzie

$$ZV_h(\pi) = \begin{cases} \{\pi_{l_h}(e_h), \pi_{l_h}(f_h)\}, & e_h < f_h, l_h \in M^t \\ \emptyset, & \text{w przeciwnym przypadku} \end{cases} \tag{5.21}$$

Dalej, niech m_i^π będzie maszyną wybraną do wykonania operacji $i \in O$ w permutacji π (tzn. $m_i^\pi \in \{l \in M : i \in O_l\}$). Niech $\mu_i' = \mu_i \setminus \{m_i^\pi\}$ będzie zbiorem maszyn, na których można wykonać operację i, powstałym po usunięciu maszyny m_i^π ze zbioru μ_i. Ruch typu wstaw można wtedy oznaczyć symbolem $iv_{i,l,z} = (i, \pi_l(z))$, gdzie $i \in O^t$, $l \in \mu_i'$, $1 \leqslant z \leqslant o_l + 1$. Zastosowanie ruchu $iv_{i,l,z}$ do permutacji π można podzielić na dwa etapy:

(*etap 1*) usunięcie operacji $i \in O$ z permutacji π_k, $k = m_i^\pi$, nastepnie
(*etap 2*) wstawienie operacji $i \in O$ na pozycję z w permutacji π_l, $l \in \mu_i'$.

Precyzyjnie, *etap 2* ruchu polega na położeniu $\pi_l(j+1) := \pi_l(j)$, $j = o_l, o_l - 1, \ldots, z$, następnie $\pi_l(z) := i$. Permutacja powstała w wyniku wykonania ruchu $v = iv_{i,l,z}$ będzie oznaczana symbolem π^v. Uwzględniając eliminacyjne własności ścieżki krytycznej i bloków operacji proponuje się zbiór ruchów $IV^1(\pi) = \bigcup_{i \in ZV(\pi)} IV_i^1(\pi)$, gdzie

$$IV_i^1(\pi) = \bigcup_{l \in \mu_i'} \{iv_{i,l,z} : 1 \leqslant z \leqslant o_l + 1\}. \tag{5.22}$$

jest zbiorem zawierającym wszystkie możliwe ruchy polegające na wstawieniu operacji i na maszyny ze zbioru μ_i'. Tak zdefiniowany zbiór może być dodany do większości zbiorów ruchów (na przykład zbiorów $AV^i(\pi)$, $i \in \{0, \ldots, 3\}$, omawianych w sekcji 3.2), w których operacje przenosi się jedynie w obrębie pojedynczej maszyny. Sąsiedztwem $IN^1(\pi) = \{\pi^v : v \in IV^1(\pi)\}$ rozwiązania π będzie nazywany zbiór permutacji powstały po zastosowaniu do permutacji π wszystkich ruchów ze zbioru $IV^1(\pi)$.

5.3.1 Efektywna metoda przeglądu sąsiedztwa

Zastosowanie dowolnego ruchu v ze zbioru $IV^1(\pi)$ do permutacji $\pi \in \Pi$ nie gwarantuje dopuszczalności permutacji π^v. Poniżej przedstawiona jest

metoda, dzięki której możliwe jest wyznaczenie najlepszego (w sensie wartości funkcji celu) rozwiązania dopuszczalnego w zbiorze $IN^1(\pi)$ bez konieczności przeglądu całego zbioru.

Bez straty ogólności, dalsze rozważania mogą być ograniczone do ustalonego podziału rozłącznego $\theta \in \Theta$, ustalonej permutacji $\pi \in \Pi$, ustalonej operacji transportowej $i \in ZV(\pi)$, zbioru ruchów $IV_i^1(\pi)$ i sąsiedztwa $IN_i^1(\pi) = \{\pi^v : v \in IV_i^1(\pi)\}$. Przez $G(\beta)$ będzie oznaczany graf uzyskany z grafu $G(\pi)$ poprzez usunięcie operacji i z permutacji π_k, $k = m_i^\pi$ lub inaczej mówiąc, po wykonaniu *etapu 1* ruchu $v \in IV_i^1(\pi)$. Wtedy, oczywiście, zbiór łuków kolejnościowych przyjmuje postać

$$E^K(\beta) = (E^K(\pi) \setminus \{(b_i^K(\pi), i), (i, a_i^K(\pi))\}) \cup (b_i^K(\pi), a_i^K(\pi)). \tag{5.23}$$

W tym momencie należy zauważyć, że permutacja β jest permutacją częściową, $\beta \in \Pi^U$, gdzie $U = O \setminus \{i\}$ jest zbiorem operacji uszeregowanych, zaś graf $G(\beta)$ jest grafem częściowym. Nie trudno też zauważyć, że

$$IN_i^1(\pi) = \bigcup_{l \in \mu_i'} Z(\beta, l, i), \tag{5.24}$$

gdzie zbiór $Z(\beta, l, i)$ dany jest równaniem (5.3). Najlepsze rozwiązanie dopuszczalne w sąsiedztwie $IN_i^1(\pi)$ może być zatem odnalezione przy użyciu procedury przyspieszonego wyznaczania najlepszej maszyny i pozycji $PWNP2$, przedstawionej na rys. 5.2. Procedurę należy zastosować do permutacji częściowej β, nie uszeregowanej operacji transportowej i oraz zbioru maszyn μ_i'. Procedura zwraca najlepszą maszynę l^* i pozycję z^*, na którą powinna być wstawiona szeregowana operacja i. Wykonanie ruchu $v = iv_{i,l^*,z^*}$ prowadzi zatem do najlepszego rozwiązania π^v w sąsiedztwie $IN_i^1(\pi)$.

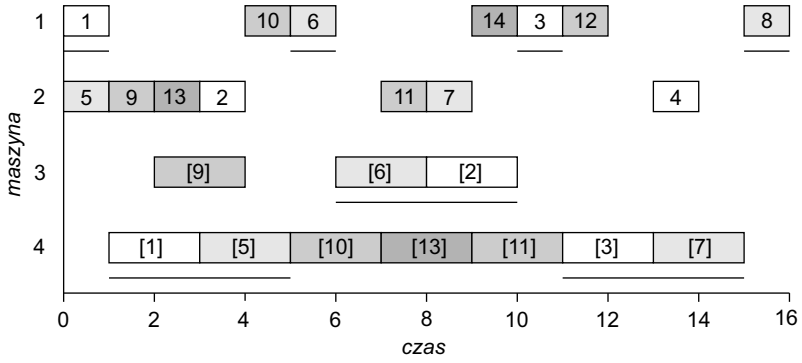
Wyznaczenie najlepszego rozwiązania w zbiorze $IN_i^1(\pi)$ przy użyciu wspomnianej procedury, ze względu na złożoność obliczeniową kroku 1, wymaga $O(n)$ czasu. Należy jednak zauważyć, że wyznaczenie wszystkich wielkości r_i^β, q_i^β, $i \in ZV(\pi)$, oraz wszystkich wartości $er_{l,z}$, $qr_{l,z}$, $1 \leqslant z \leqslant |\beta_l|$, $l \in \{\bigcup_{i \in ZV(\pi)} \mu_i'\}$, również może być wykonane w czasie $O(n)$. Zatem krok 1 wspomnianej procedury wystarczy wykonać tylko raz, niezależnie od mocy zbioru $ZV(\pi)$. Krok 2 i 3 procedury musi być powtórzony dla każdej operacji $i \in ZV(\pi)$, co wymaga odpowiednio $O(g)$, $g = |ZV(\pi)| \cdot \sum_{l \in \mu_i'}(|\beta_l| + 1)$, oraz $O(|ZV(\pi)| \cdot (|\beta_{l^*}| + 1))$ czasu. Zatem, wyznaczenie najlepszego rozwiązania w zbiorze $IN^1(\pi)$ może być wykonane w czasie $O(\max\{n, g\})$.

Poniżej zamieszczono przykład zastosowania procedury z rys. 5.2 do wyznaczenia najlepszego rozwiązania w sąsiedztwie $IN^1(\pi)$ dla przykładowej instancji problemu i przykładowej permutacji π.

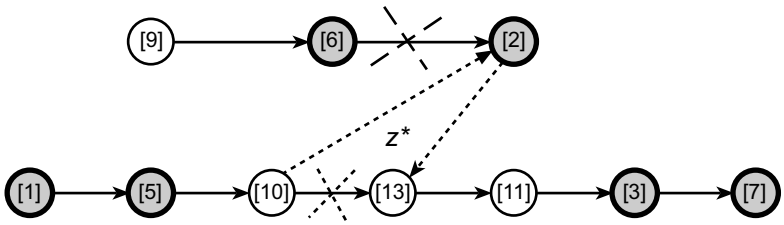
Przykład 5.1 Dany jest system produkcyjny ze zbiorem $m^p = 2$ maszyn produkcyjnych $M^p = \{1, 2\}$ i zbiorem $m^t = 2$ maszyn transportowych $M^t = \{3, 4\}$. Zakłada się, że czas przejazdu bez załadunku każdego wózka $v \in M^t$ pomiędzy maszynami produkcyjnymi wynosi $e(1, 2) = e(2, 1) = 2$. W systemie należy wykonać $r = 4$ zadania ze zbioru $J = \{1, 2, 3, 4\}$. Zadanie 1 składa się z sekwencji operacji $1, [1], 2, [2], 3, [3], 4$ na zadanie 2 składa się sekwencja operacji $5, [5], 6, [6], 7, [7], 8$, zadanie trzecie stanowią operacje $9, [9], 10, [10], 11, [11], 12$, zaś zadanie ostatnie składa się z operacji $13, [13], 14$. Zbiór operacji produkcyjnych i transportowych jest zatem równy odpowiednio $O^p = \{1, 2, \ldots, 14\}$ i $O^t = \{[1], [2], [3], [5], [6], [7], [9], [10], [11], [13]\}$. Zakłada się, że $\mu_1 = \mu_3 = \mu_6 = \mu_8 = \mu_{10} = \mu_{12} = \mu_{14} = \{1\}$, $\mu_2 = \mu_4 = \mu_5 = \mu_7 = \mu_9 = \mu_{11} = \mu_{13} = \{2\}$ oraz $\mu_i = \{3, 4\}$, $i \in O^t$. Przyjmuje się, że czasy wykonania poszczególnych operacji produkcyjnych wynoszą $p_i = 1$, $i \in O^p$. Czasy wykonania operacji transportowych wynoszą $p_i = t_k(1, 2) = t_k(2, 1) = 2$, $k \in J$, $i \in O^t$.

Dany jest podział $\theta = \{O_1, \ldots, O_4\}$ zbioru operacji $O = O^p \cup O^t$ taki, że $O_1 = \{1, 3, 6, 8, 10, 12, 14\}$, $O_2 = \{2, 4, 5, 7, 9, 11, 13\}$, $O_3 = \{[2], [6], [9]\}$, $O_4 = \{[1], [3], [5], [7], [10], [11], [13]\}$ ($o_1 = 7$, $o_2 = 7$, $o_3 = 3$, $o_4 = 7$). Dana jest permutacja dopuszczalna $\pi = (\pi_1, \ldots, \pi_4)$, gdzie $\pi_1 = (1, 10, 6, 14, 3, 12, 8)$, $\pi_2 = (5, 9, 13, 2, 11, 7, 4)$, $\pi_3 = ([9], [6], [2])$, $\pi_4 = ([1], [5], [10], [13], [11], [3], [7])$, określająca kolejność wykonania operacji ze zbioru O oraz acykliczny graf $G(\pi)$. Uszeregowanie reprezentowane przez permutację π zostało przedstawione na rys. 5.3. Na rysunku 5.4 został przedstawiony fragment grafu $G(\pi)$. Łuki należące do tego grafu zostały zaznaczone linią ciągłą. Jedna ze ścieżek krytycznych w grafie $G(\pi)$ przechodzi przez wierzchołki $u = (1, [1], [5], 6, [6], [2], 3, [3], [7], 8)$. Na rysunkach 5.3, 5.4 operacje należące do ścieżki krytycznej zostały wyróżnione. Ścieżka u generuje $lb = 7$ bloków operacji, $u = (B_1, B_2, \ldots, B_7)$, gdzie $B_1 = (1)$ ($l_1 = 1$, $e_1 = 1$, $f_1 = 1$), $B_2 = ([1], [5])$ ($l_2 = 4$, $e_2 = 1$, $f_2 = 2$), $B_3 = (6)$ ($l_3 = 1$, $e_3 = 3$, $f_3 = 3$), $B_4 = ([6], [2])$ ($l_4 = 3$, $e_4 = 2$, $f_4 = 3$), $B_5 = (3)$ ($l_5 = 1$, $e_5 = 5$, $f_5 = 5$), $B_6 = ([3], [7])$ ($l_6 = 4$, $e_6 = 6$, $f_6 = 7$), $B_7 = (8)$ ($l_7 = 1$, $e_7 = 7$, $f_7 = 7$).

Zgodnie z definicją (5.20), w zbiorze $ZV(\pi)$ znajdują się operacje $ZV(\pi) = \{[1], [5], [6], [2], [3], [7]\}$. Poniżej będzie omówiona efektywna metoda przeglądu sąsiedztwa $IN^1(\pi)$ na przykładzie operacji transportowej [2] i sąsiedztwa $IN^1_{[2]}(\pi)$. Ponieważ operacja [2] w permutacji π jest uszeregowana na maszynie $m^\pi_{[2]} = 3$, zbiór $\mu'_{[2]}$ przyjmuje postać $\mu'_{[2]} = \mu_{[2]} \setminus \{3\} = \{4\}$. Zbiór ruchów $IV^1_{[2]}(\pi)$ przybiera zatem postać $IV^1_{[2]}(\pi) = \{iv_{[2],4,z} : 1 \leqslant z \leqslant o_4\}$. Jak już wcześniej zauważono, każdy ruch $v \in IV^1_{[2]}(\pi)$ można podzielić na dwa etapy; etap pierwszy polega na usu-



Rysunek 5.3: Wykres Gantta uszeregowania reprezentowanego przez permutację π z przykładu 5.1



Rysunek 5.4: Modyfikacje grafu $G(\pi)$ dla permutacji π z przykładu 5.1

nięciu operacji [2] z permutacji π_3. W wyniku jego wykonania powstaje permutacja częściowa β oraz odpowiadający jej graf częściowy $G(\beta)$. Graf $G(\beta)$ można skonstruować na podstawie grafu $G(\pi)$ poprzez usunięcie łuku $([6],[2])$. Fragment grafu $G(\beta)$ został przedstawiony na rys. 5.4; łuk $([6],[2])$ został skreślony linią przerywaną. Ponieważ zachodzi równość $IN^1_{[2]}(\pi) = Z(\beta, 4, [2])$, w celu wyznaczenia najlepszego sąsiada w analizowanym sąsiedztwie można wykorzystać procedurę $PWNP2$ przedstawioną na rys. 5.2. Procedurę trzeba zastosować do permutacji częściowej β, ope-

Tabela 5.1: Wielkości $er_{4,z}$, $eq_{4,z}$, $ed_{4,z}$ wyznaczone dla grafu $G(\beta)$ z przykładu 5.1

z	0	1	2	3	4	5	6	7	8
$er_{4,z}$	0	1	3	5	7	9	11	13	-
$eq_{4,z}$	-	13	11	9	7	5	3	1	0
$ed_{4,z}$	-	25	25	21	21	22	21	21	25

racji [2] i zbioru maszyn $\mu'_{[2]}$. W kroku 1 procedury wyznaczane są wielkości $r^{\beta}_{[2]} = 8$, $q^{\beta}_{[2]} = 6$ oraz wielkości $er_{4,z}$, $eq_{4,z}$, $1 \leqslant z \leqslant |\beta_4|$, $|\beta_4| = 7$. Wartości te zostały zaprezentowane w tabeli 5.1. W kroku 2 wyznaczane są wartości $ed_{4,z}$ (również przedstawione w tabeli 5.1), $1 \leqslant z \leqslant |\beta_4| + 1$, oraz wartość $d_{\min} = ed_{4,\min} = ed_{4,3} = ed_{4,4} = ed_{4,6} = ed_{4,7} = 21$. W kroku 3 dla dowolnej maszyny $l^* \in ZM = \{4\}$ wyznaczana jest pozycja $a_{l^*} = 3$, $b_{l^*} = 7$, $c_{l^*} = 4$ i ostatecznie pozycja $z^* = 4$. Ruch $v = iv_{[2],4,4}$ prowadzi do najlepszego rozwiązania sąsiedniego δ w sąsiedztwie $IN^1_{[2]}(\pi)$. Graf $G(\delta)$ powstaje z grafu $G(\beta)$ poprzez usunięcie łuku $([10],[13])$ i dodanie łuków $([10],[2])$, $([2],[13])$. Na rysunku 5.4 łuki odpowiednio nie istniejące bądź istniejące w grafie $G(\delta)$ zostały skreślone bądź wyrysowane linią punktowaną.

5.4 Algorytmy konstrukcyjne

Poniżej, podobnie jak w rozdziale 4, prezentowane są dwa typy algorytmów konstrukcyjnych: algorytmy priorytetowe oraz algorytmy typu wstaw. Poniższe algorytmy charakteryzują się dużym podobieństwem do tych prezentowanych w rozdziale 4, w szczególności wykorzystują wiele podobnych, nawet identycznych definicji i sformułowań. Różnice są jednak na tyle istotne, że algorytmy te wymagają osobnego omówienia.

Idea postępowania w trakcie projektowania algorytmów konstrukcyjnych dla badanego zagadnienia jest podobna jak w przypadku problemów gniazdowych z maszynami równoległymi [36]; wybór operacji wiąże się z koniecznością wyboru maszyny, na której operacja będzie uszeregowana. W przypadku algorytmów priorytetowych, ze względu na sposób wyboru owej maszyny, można wyszczególnić trzy grupy:

1. dwufazowe algorytmy priorytetowe,
2. jednofazowe algorytmy maszynowo-operacyjne oraz
3. jednofazowe algorytmy operacyjno-maszynowe.

W pierwszej fazie algorytmów dwufazowych najpierw dokonuje się wyboru maszyn dla poszczególnych operacji (sprowadzając w ten sposób problem do problemu bez maszyn równoległych), po czym szereguje się operacje na maszynach wybranych w fazie pierwszej. W algorytmach należących do drugiej i trzeciej grupy wyboru maszyny i operacji wykonuje się jednocześnie. W każdej iteracji algorytmu maszynowo-operacyjnego najpierw wybiera się maszynę stosując maszynową regułę priorytetową. Następnie, ze zbioru operacji, które można uszeregować na tej maszynie, stosując operacyjną regułę priorytetową wybiera się operację do uszeregowania. W przypadku algorytmów należących do trzeciej grupy zasada postępowania jest odwrotna;

stosując operacyjno-maszynową regułę priorytetową wybiera się operację, potem maszynę, na której wybrana operacja będzie uszeregowana. Cechą wspólną wszystkich algorytmów priorytetowych jest to, że wybrana operacja zawsze jest szeregowana za ostatnią dotychczas uszeregowaną operacją w danej permutacji.

Strategię postępowania dla problemów gniazdowych z maszynami równoległymi można również stosować w trakcie projektowania algorytmów konstrukcyjnych typu wstaw. Szeregowana operacja jest wtedy próbnie wstawiana nie na jedną, lecz wszystkie maszyny, na których może być uszeregowana. Spośród wszystkich permutacji częściowych wygenerowanych w ten sposób wybierana jest jedna, generująca najlepszą wartość funkcji celu. Permutacja ta staje się permutacją bieżącą w następnej iteracji algorytmu.

5.4.1 Algorytmy priorytetowe

Poniżej przedstawia się krótki opis algorytmów priorytetowych, nazwanych algorytmami $APM(RM, RO)$, należących do grupy jednofazowych algorytmów maszynowo-operacyjnych; RM i RO oznacza tu odpowiednio maszynową i operacyjną regułę priorytetową.

Algorytmy priorytetowe wykonują n iteracji. W każdej iteracji dla danej permutacji częściowej $\beta \in \Pi^U$, gdzie $U \subset O$ jest zbiorem operacji uszeregowanych, wyznacza się zbiór operacji gotowych do uszeregowania OG, dany równaniem (4.1). Następnie, wyznacza się moment czasowy

$$\Delta = \min_{j \in OG} r(j), \tag{5.25}$$

gdzie

$$r(j) = \max\{\max_{i \in ZB_j(\pi)}(S_i + p_i), \min_{l \in \mu_j}\{t_l + s(\beta_l(|\beta_l|), j)\}\}. \tag{5.26}$$

Wielkość t_l jest momentem zakończenia wykonywania ostatniej operacji uszeregowanej na maszynie $l \in M$. Następny z kolei wyznaczany jest zbiór

$$M' = \{l \in M : \exists_{j \in OG}\ l \in \mu_j, r(j) = \Delta\}, \tag{5.27}$$

na których mogą być wykonywane operacje ze zbioru OG i których najwcześniejszy możliwy moment rozpoczęcia jest równy Δ. Przy użyciu maszynowej reguły priorytetowej RM ze zbioru M' wybiera się jedną maszynę l, po czym określa się tzw. zbiór operacji konfliktowych

$$OK_l = \{j \in OG : l \in \mu_j, r(j) = \Delta\}. \tag{5.28}$$

Na zakończenie każdej iteracji, stosując operacyjną regułę priorytetową RO, wybiera się operację $j \in OK_l$ i kładzie się $\beta_l := \beta_l, j$, $U := U \cup \{j\}$. Jeżeli $j \in O^p$ to kładzie się $S_j := r(j)$, $t_l := S_j + p_j$, w przeciwnym przypadku należy położyć $S_j := \max\{t_l + s(\beta_l(|\beta_l|), j), r(j)\}$, $t_l := S_j + p_j$. Złożoność obliczeniowa algorytmów priorytetowych $APM(RM, RO)$ jest zależna od złożoności obliczeniowej zastosowanych reguł priorytetowych. Przy założeniu, że reguły mają złożoność $O(n)$, złożoność całego algorytmu wynosi $O(n^2)$.

Podobnie jak w przypadku problemu $J, DR|t_{jkl}, t'_{kl}|C_{\max}$ definicja wielkości Δ i zbiorów M', OK_l mogą być zastąpione definicją wielkości Δ', daną równaniem (4.6), i odpowiednimi zbiorami M'', OK'_l, gdzie

$$M'' = \{l \in M : \exists_{j \in OG}\, l \in \mu_j, r(j) + p_j = \Delta\}, \tag{5.29}$$

$$OK'_l = \{j \in OG : l \in \mu_j, r(j) < \Delta\}. \tag{5.30}$$

Również w przypadku badanego zagadnienia taka modyfikacja powoduje znaczne pogorszenie własności algorytmów.

Typowe maszynowe i operacyjne reguły priorytetowe charakteryzują się liniową złożonością obliczeniową. Dla omawianego problemu, poza regułami operacyjnymi RO, omawianymi w sekcji 4.1.1, można wymienić również takie, których wynik działania zależny jest od liczby maszyn. Na przykład, wyborowi może podlegać operacja z największą bądź najmniejszą liczbą maszyn, na których można wykonać daną operację. Wśród reguł maszynowych RM można wymienić reguły $RM1a$, $RM1b$, w myśl których wyborowi podlega maszyna z odpowiednio największą i najmniejszą ważoną liczbą dedykowanych operacji konfliktowych. Można też wymienić reguły $RM2a$, $RM2b$, polegające na wyborze maszyny odpowiednio z największą i najmniejszą ważoną liczbą dedykowanych operacji nieuszeregowanych. Reguła maszynowa RAN polega na wyborze maszyny w sposób losowy. Bardziej kompletny opis poszczególnych reguł maszynowych zamieszczony jest w pracy [36].

Zasada postępowania przy projektowaniu algorytmów należących do, wymienianych powyżej, dwóch pozostałych grup algorytmów priorytetowych (tj. jednofazowych algorytmów operacyjno-maszynowych i algorytmów dwufazowych) jest bardzo podobna jak w przypadku algorytmów $APM(RM, RO)$. Podobna jest też jakość generowanych przez nie rozwiązań. Ze względu na to podobieństwo, wspomniane algorytmy nie będą w tej pracy dalej rozważane.

1. Połóż $\beta^0 := \emptyset$. Utwórz listę ψ taką, że $\omega(\psi(i)) \geqslant \omega(\psi(i+1))$, $i = 1, \ldots, n-1$.
2. Dla każdego $k = 1, 2, \ldots, n$ wykonuj krok 3 i krok 4
3. Połóż $i := \psi(k)$, $\beta := \beta^{k-1}$.
4. Jeżeli $i \in O^p$ to wykonaj krok 5. W przeciwnym przypadku wykonaj krok 6.
5. Wyznacz wartości r_i^β, q_i^β w grafie $G(\beta)$. Określ zbiór permutacji $ZD(\beta, i)$ dany równaniem (5.4). Wybierz dowolną permutację $\delta \in ZD(\beta, i)$ i połóż $\beta^k := \delta$.
6. Określ zbiór permutacji $Z(\beta, i)$ dany równaniem (5.2). Stosując procedurę $PWNP2$ prezentowaną na rys. 5.2 do permutacji β, operacji i oraz zbioru maszyn μ_i wyznacz maszynę l^* i pozycję z^*. Wybierz permutację $\delta \in Z(\beta, i)$ taką, że $\delta_{l^*}(z^*) = i$. Połóż $\beta^k := \delta$.

Rysunek 5.5: Schemat algorytmu $INT4$

5.4.2 Algorytm typu wstaw

Poniżej prezentuje się algorytm o nazwie $INT4$, będący modyfikacją algorytmu $INT3$, prezentowanego w sekcji 4.1.2. Algorytm $INT4$ jest algorytmem dwufazowym. W pierwszej fazie tworzona jest lista operacji ψ, w której wszystkie operacje posortowane są według nierosnących wartości priorytetów $\omega(\psi(i)) \geqslant \omega(\psi(i+1))$, $i = 1, \ldots, n-1$, gdzie

$$\omega(i) = c_i \cdot p_i. \tag{5.31}$$

Z reguły przyjmuje się, że

$$c_i = \frac{1}{|\mu_i|}, \tag{5.32}$$

lecz wielkość ta może przyjmować dowolną wartość z zakresu $[1/|\mu_i|, 1]$. W fazie drugiej algorytm wykonuje n iteracji. W k-tej iteracji z listy ψ wybierana jest operacja $i = \psi(k)$ i próbnie wstawiana na każdą pozycję w każdej permutacji częściowej β_l, $l \in \mu_i$; permutacja bieżąca β jest permutacją częściową utworzoną w iteracji $k-1$. Zbiór wszystkich permutacji $Z(\beta, i)$ powstałych w ten sposób dany jest równaniem (5.2). Następnie, dla każdej permutacji $\alpha \in Z(\beta, i)$ oblicza się wartość d_i^α, daną odpowiednimi równaniami (3.44), (3.45), oraz wybiera się permutację najlepszą, tzn. taką permutację $\delta \in Z(\beta, i)$, że graf $G(\delta)$ jest acykliczny oraz $d_i^\delta = d_{\min}$, gdzie

wielkość $d_{\min}$ dana jest równaniem (3.47). W przypadku operacji produkcyjnych $i \in O^p$, w celu akceleracji obliczeń konstruuje się zbiór $ZD(\beta, i)$, dany równaniem (5.4), po czym ze zbioru tego wybiera się dowolną permutację δ. Na mocy twierdzenia 5.2 permutacja δ jest najlepszą permutacją w zbiorze $Z(\beta, i)$. W przypadku operacji transportowych, najlepszą permutację odnajduje się za pomocą procedury przyspieszonego wyznaczania najlepszej maszyny i pozycji dla wstawianej operacji $i \in O^t$, przedstawioną na rysunku 5.2. Najlepsza permutacja δ staje się permutacją bieżącą w $k+1$ iteracji algorytmu. Schemat algorytmu $INT4$ został zaprezentowany na rys. 5.5.

5.5 Algorytmy lokalnych poszukiwań

Można wyróżnić przynajmniej kilka podejść do rozwiązywania badanego problemu przy użyciu technik lokalnych poszukiwań. Pierwsze, najprostsze, polega na sprowadzeniu problemu do problemu $J, DR|t_{jkl}, t'_{kl}|C_{\max}$ poprzez skonstruowanie dowolną techniką przydziału operacji transportowych do poszczególnych wózków (skonstruowanie podziału rozłącznego zbioru operacji). Przydział może być skonstruowany losowo, może też być wynikiem działania dowolnego algorytmu konstrukcyjnego dla omawianego problemu. Drugie z podejść polega na wyposażeniu algorytmów rozwiązywania problemu $J, DR|t_{jkl}, t'_{kl}|C_{\max}$ w dodatkowe mechanizmy pozwalające na zmianę przydziału wózków do operacji transportowych. Podejście to można zrealizować na przykład poprzez dodanie do siebie zbioru ruchów typu zamień sąsiednie operacje i zbioru ruchów typu wstaw. Przykładowo, algorytmem zrealizowanym przy użyciu tego podejścia jest algorytm $TSAM_{AGV}$, przedstawiony w sekcji 5.5.1, będący modyfikacją algorytmu $TSAB_{AGV}$.

Realizacja podejścia drugiego nie zawsze jednak jest możliwa. Na przykład, w schemacie dywersyfikacji obliczeń algorytmów *i–*$TSAB$, *i–*$TSAB_{AGV}$ wykorzystywana jest miara odległości tau Kendalla (omawiana w sekcji 3.4.1) która jest poprawnie określona tylko dla zestawów permutacji tego samego podziału rozłącznego zbioru operacji. Modyfikacja algorytmów *i–*$TSAB$, *i–*$TSAB_{AGV}$ dla badanego problemu nie będzie zatem rozważana. W tym miejscu można zauważyć, że w kontekście rozwiązań badanego problemu wciąż można wykorzystać miarę geometryczną (również omawianą w sekcji 3.4.1). Fakt ten został wykorzystany przy realizacji algorytmu genetycznego $GAMX$, stanowiącego modyfikację algorytmu $GAGX$, przedstawionego w sekcji 5.5.2.

5.5.1 Algorytm poszukiwań z zabronieniami

W tej sekcji omawia się algorytm poszukiwań z zabronieniami, w którym wykorzystuje się zbiór $V(\pi) = AV^3(\pi) \cup IV^1(\pi)$, $\pi \in \Pi$ (i odpowiadające mu sąsiedztwo $N(\pi)$), będący sumą zbioru ruchów typu zamień sąsiednie operacje (opisywanego w sekcji 3.2) i zbioru ruchów typu wstaw (sekcja 5.3). Połączenie obu tych zbiorów umożliwia przemieszczanie operacji w obrębie pojedynczej maszyny, jak również w obrębie innych maszyn, na których dana operacja (transportowa) może być wykonywana. Wykorzystanie zbioru $V(\pi)$, ze względu na różny charakter przechowywanych ruchów, wiąże się z koniecznością wprowadzenia pewnych dodatkowych rozwiązań, których opis zamieszczono poniżej.

Technika poszukiwań z zabronieniami, jak już wspominano w sekcjach 2.2.1, 4.2.1, jest techniką lokalnych poszukiwań, której charakterystycznym elementem jest lista tabu. Lista tabu jest krótkoterminową pamięcią, która zabezpiecza algorytm przed powrotem do wcześniej już przeglądniętych rozwiązań. Charakter przechowywanej na liście informacji w dużym stopniu zależy od konkretnego problemu i inwencji twórczej projektanta listy. Na liście takiej często zapamiętywane są ruchy, które uzyskują status zabronionych i których nie można zastosować do żadnej permutacji. W myśl efektywnej metody przeglądu sąsiedztwa omawianej w sekcji 5.3.1, w trakcie wyznaczania najlepszego rozwiązania w sąsiedztwie $IN^1(\pi)$, $\pi \in \Pi$, nie wykonuje się wszystkich ruchów ze zbioru $IV^1(\pi)$, lecz, po pewnych zabiegach, „od razu" wyznacza się ruch prowadzący do rozwiązania najlepszego. Wiąże się to z koniecznością zaprojektowania specyficznej listy tabu. Autor tej rozprawy rozwiązał ten problem przez podział listy tabu na dwie rozłączne listy.

Pierwsza z proponowanych list tabu przechowuje tylko ruchy typu zamień sąsiednie operacje lub ruchy sztuczne $dv = (0, 0)$. Formalnie, lista ta będzie notowana jako $T^A = (T_1^A, T_2^A, \ldots, T_{max\_a}^A)$, gdzie $T_i^A = (h, j)$ lub $T_i^A = dv$, $1 \leqslant i \leqslant max\_a$, $h, j \in O$, zaś max_a jest długością listy. Korzystając z notacji podobnej jak w pracy [70], ruch odwrotny do ruchu $v = (h, j)$ będzie oznaczany przez $\overline{v} = (j, h)$. Każdorazowo, gdy do ustalonej permutacji π stosowany jest ruch $v = (h, j) \in AV^3(\pi)$, do listy T^A dodawany jest ruch $\overline{v}$. Operacja ta będzie notowana $T^A \oplus \overline{v}$ i będzie polegać na położeniu $T_i^A := T_{i+1}^A$, $i = 1, 2, \ldots, max\_a - 1$ oraz $T_{max\_a}^A := \overline{v}$. Każdy ruch znajdujący się na liście T^A otrzymuje status zabronionego i nie może być zastosowany do żadnej permutacji.

Druga lista będzie oznaczana przez $T^I = (T_1^I, T_2^I, \ldots, T_{max\_i}^I)$, gdzie $T_i^I \in O^t \cup \{do\}$, $1 \leqslant i \leqslant max\_i$, zaś max_i jest długością listy. Lista ta przechowuje operacje transportowe bądź operacje sztuczne $do = 0$. Operację

dodawania elementu $h \in O^t \cup \{do\}$ do listy T^I przebiega w sposób analogiczny jak w przypadku listy T^A i będzie notowana $T^I \oplus h$. Zbiór ruchów zabronionych w zbiorze $IV^1(\pi)$ będzie notowany jako zbiór

$$L^I = \{iv_{h,k,z} \in IV^1(\pi) : h \in T^I\}. \tag{5.33}$$

W dalszej części rozdziału zapis $T = \emptyset$, gdzie $T = T^A \cup T^I$, będzie oznaczać, że na każdej z list T^A, T^I przechowywane są odpowiednio tylko ruchy sztuczne i tylko operacje sztuczne. Zdefiniowana powyżej lista tabu T w pewnych sytuacjach może okazać się zbyt restrykcyjna i może uniemożliwiać osiągniecie relatywnie dobrych regionów przestrzeni rozwiązań. Dlatego dodatkowo dopuszcza się możliwość zastosowania ruchów ze zbioru

$$\begin{array}{rcl} ZP & = & \{v \in AV^3(\pi) \cap T^A : C_{\max}(\pi^v) < C^*\} \\ & \cup & \{v \in L^I : C_{\max}(\pi^v) < C^*\}, \end{array} \tag{5.34}$$

gdzie C^* jest najlepszą wartością funkcji celu znalezioną we wcześniejszych iteracjach omawianego algorytmu. Zbiór ZP będzie nazywany zbiorem ruchów zabronionych perspektywicznych. Ostatecznie, zbiór ruchów niezabronionych i zabronionych perspektywicznych można zapisać następująco:

$$ZR = \{AV^3(\pi) \setminus T^A\} \cup \{IV^1(\pi) \setminus L^I\} \cup ZP. \tag{5.35}$$

Główna idea zastosowania listy tabu o stałej długości polega na tym, że poszczególne ruchy są zabronione przez stałą liczbę iteracji, równą długości listy tabu. Aby idea ta była zachowana również w przypadku list T^A, T^I, do obu list dodaje się elementy równocześnie według następującej zasady: jeżeli kładzie się $T^A \oplus \overline{v}$, $v \in AV^3(\pi)$, to kładzie się też $T^I \oplus do$. Symetrycznie, jeżeli kładzie się $T^I \oplus h$, $h \in O^t$, to kładzie się też $T^A \oplus dv$.

Najważniejszym elementem badanego w pracy algorytmu jest procedura przeszukiwania sąsiedztwa (procedura $MPPS$) przedstawiona na rys. 5.6. Procedura $MPPS$ jest wynikiem modyfikacji procedury NSP, wykorzystywanej w algorytmie $TSAB$, prezentowanym w pracy [70] (algorytm ten pobieżnie został omówiony w sekcji 4.2.1). W krokach 1-3 wyznaczane są wartości funkcji celu rozwiązań w sąsiedztwie $N(\pi)$. Pewnego dodatkowego komentarza wymaga krok 3. Otóż procedura $PWNP2$ wyznacza między innymi wartości $d_{\min}$ oraz r_i^β, q_i^β, $i \in ZV(\pi)$ (zbiór $ZV(\pi)$ dany jest równaniem 5.20). Koszt wyznaczenia tych wartości jest taki sam jak koszt wyznaczenia wartości $C_{\max}(\beta)$, zatem wartość tą można wyznaczyć niejako „przy okazji". Wykorzystując własności permutacji częściowej, wartość $C_{\max}(\pi^v)$ można wtedy wyznaczyć z równania $C_{\max}(\pi^v) := \max\{C_{\max}(\beta), d_{\min}\}$.

Procedura rozpoczyna działanie z permutacją π, aktualną listą tabu $T = T^A \cup T^I$, niepustym zbiorem ruchów $V(\pi)$ i najlepszą znalezioną wartością funkcji celu C^*. Procedura zwraca ruch v', zmodyfikowaną listę tabu T' i nową permutację π'.

1. Dla każdego ruchu $v \in AV^3(\pi)$ wyznacz i zapamiętaj wartość $C_{\max}(\pi^v)$ używając równania (3.40).
2. Połóż $W := \emptyset$. Dla każdego $h \in ZV(\pi)$, gdzie zbiór $ZV(\pi)$ dany jest równaniem (5.20), powtarzaj krok 3
3. Utwórz permutację częściową β poprzez usunięcie operacji h z permutacji π. Wyznacz maszynę l_h i pozycję z_h stosując do permutacji β, operacji h i zbioru μ'_h procedurę $PWNP2$ przedstawioną na rys. 5.2. Połóż $v := iv_{h,l_h,z_h}$ oraz $W := W \cup \{v\}$. Wyznacz wielkość $C_{\max}(\pi^v)$.
4. Wyznacz zbiory ZP, ZR dane odpowiednimi równaniami (5.34), (5.35). Jeżeli $ZR \neq \emptyset$, to wybierz ruch $v' \in ZR$ taki, że $C_{\max}(\pi^{v'}) = \min_{v \in V(\pi)}\{C_{\max}(\pi^v)\}$ i idź do kroku 10.
5. Jeżeli $|V(\pi)| = 1$, to wybierz ruch $v' \in V(\pi)$ i idź do kroku 10.
6. Jeżeli $T^A \neq \emptyset$, to wyznacz pozycję $i = \min\{1 \leqslant j \leqslant max\_a : T^A_j \neq \emptyset\}$. W przeciwnym przypadku idź do kroku 8.
7. Jeżeli $i \neq max\_a$, to powtarzaj $T^A := T^A \oplus T^A_{max\_a}$ dopóki $AV^3(\pi) \setminus T^A = \emptyset$. W przeciwnym przypadku powtarzaj $T^A := T^A \oplus dv$ dopóki $AV^3(\pi) \setminus T^A = \emptyset$. Wybierz ruch $v' \in AV^3(\pi) \setminus T^A$ i idź do kroku 10.
8. Znajdź pozycję $i = \min\{1 \leqslant j \leqslant max\_i : T^I_j \neq do\}$. Połóż $h := T^I_j$ oraz połóż $v' := iv_{h,l_h,z_h}$.
9. Jeżeli $i \neq max\_i$, to powtarzaj $T^I := T^I \oplus T^I_{max\_i}$ dopóki $IV^1(\pi) \setminus L^I = \emptyset$. W przeciwnym przypadku powtarzaj $T^I := T^I \oplus do$ dopóki $IV^1(\pi) \setminus L^I = \emptyset$.
10. Jeżeli $v' \in AV^3(\pi)$, to połóż $T^A := T^A \oplus \overline{v'}$ oraz $T^I := T^I \oplus do$. W przeciwnym przypadku połóż $T^I := T^I \oplus h$ oraz $T^A := T^A \oplus dv$. Połóż $\pi' := \pi^{v'}$, $T' := T^A \cup T^I$.

Rysunek 5.6: Schemat procedury $MPPS$

W kroku 4 wyznaczany jest zbiór ZR, zawierający ruchy niezabronione oraz zabronione perspektywiczne. Jeżeli zbiór nie jest pusty, ze zbioru tego wybierany jest ruch najlepszy. W przeciwnym przypadku, w krokach 6-9 dokonuje się modyfikacji listy tabu w celu usunięcia „zabronienia" z niektórych ruchów. W kroku 10 uwzględniany jest wcześniejszy wybór ruchu $v' \in V(\pi)$ poprzez modyfikację list tabu, tworzona jest permutacja $\pi' = \pi^{v'}$, po czym procedura kończy działanie.

Najprostszy algorytm poszukiwań z zabronieniami uzyskuje się poprzez iterowanie procedury $MPPS$ przez określoną liczbę iteracji. Jednakże, autor tej rozprawy uzyskał najlepsze wyniki po osadzeniu procedury $MPPS$ w miejscu procedury NSP we wspomnianym algorytmie $TSAB$ i jednoczesnym zastąpieniu oryginalnego sąsiedztwa sąsiedztwem $N(\pi)$. W ten sposób skonstruowany algorytm będzie nazywany algorytmem $TSAM_{AGV}$. Podobnie jak oryginalny algorytm, algorytm $TSAM_{AGV}$ wymaga określenia szeregu parametrów, takich jak $maxiter$ – maksymalna liczba iteracji bez poprawy wartości C^*, $maxl$ – maksymalna długość listy dla skoków powrotnych oraz parametrów $max\delta$ i $maxc$, używanych w detektorze cykli. Oryginalny parametr $maxt$ (oznaczający długość listy tabu) został oczywiście zastąpiony przez parametry max_a i max_i, opisywane powyżej.

5.5.2 Algorytm genetyczny

Jednym z najlepszych algorytmów rozwiązywania problemu $J, DR|t_{jkl}, t'_{kl}|C_{\max}$, wśród algorytmów opisywanych w poprzednim rozdziale, okazał się algorytm $GAGX$. Przed zastosowaniem algorytmu do rozwiązywania problemu badanego w tym rozdziale, algorytm ten trzeba poddać niewielkim modyfikacjom. W wyniku wprowadzenia poniżej przedstawionych modyfikacji powstaje algorytm, który będzie nazwany algorytmem $GAMX$.

Pierwszym z komponentów algorytmu $GAGX$, który należy poddać modyfikacji, jest sam operator GX przedstawiony na rys. 4.4. Oryginalnie, operator GX został zaprojektowany do rozwiązywania problemu gniazdowego z ustalonym przydziałem wózków AGV. Ustalenie owego przydziału jest równoznaczne z narzuceniem „z góry" pewnego, konkretnego podziału rozłącznego zbioru operacji, który nie może podlegać modyfikacji. Z kolei, rozwiązania problemu gniazdowego ze swobodnym przydziałem wózków (ze względu na możliwość wyboru wózków do wykonania poszczególnych operacji) reprezentowane są poprzez zestawy permutacji różnych podziałów rozłącznych zbioru operacji. Modyfikacja operatora GX będzie zatem polegać na umożliwieniu krzyżowania chromosomów stanowiących zestawy permutacji różnych podziałów rozłącznych.

Operator rozpoczyna działanie z ustalonymi chromosomami π^1, π^2 i parametrem *maxiter*. Operator zwraca chromosom π^* będący permutacją dopuszczalną i wartość funkcji celu C^*.

1. Znajdź punkty $A, B \in \mathbb{R}^n$, odpowiadające odpowiednim permutacjom π^1, π^2. Połóż $A' := A$, $B' := B$, $\theta := \emptyset$, $\pi^* := \emptyset$, $C^* := \infty$ oraz $iter := 0$.
2. Dla każdej operacji $k \in O$ wykonaj krok 3.
3. Połóż $i := m_k^{\pi^1}$, $j := m_k^{\pi^2}$. Wylosuj liczbę r z przedziału $(0,1)$. Jeżeli $r \leqslant 0,5$, to połóż $m_k := i$. W przeciwnym przypadku połóż $m_k := j$. Połóż $\theta_l := \theta_l \cup \{k\}$, gdzie $l = m_k$.
4. Połóż $W := \theta$.
5. Utwórz listę $\gamma = (\gamma(1), \gamma(2), \ldots, \gamma(n))$ taką, że $\gamma(i) = 0$, $i = 1, 2, \ldots, n$.
6. Wylosuj liczbę r z przedziału $(0,1)$. Znajdź punkt $C = (c_1, c_2, \ldots, c_n)$ taki, że $||AB|| = ||AC|| + ||CB||$ oraz $||AC|| = r \cdot ||AB||$. Dla każdego $k = 1, 2, \ldots, n$ wykonuj krok 7.
7. Połóż $l := m_k$. Znajdź $t = \min_{i \in W_l} |c_k - i|$. Wylosuj element z ze zbioru $\{i \in W_l : |c_k - i| = t\}$. Połóż $\gamma(k) := z$ oraz $W_l := W_l \setminus \{z\}$.
8. Utwórz permutację δ taką, że $\delta^{-1} = \gamma$.
9. Połóż $(\omega, C_{\max}(\omega)) := APM(RAN, PWK(\delta))$. Jeżeli $C_{\max}(\omega) < C^*$ wtedy połóż $\pi^* := \omega$, $C^* := C_{\max}(\omega)$.
10. Znajdź punkt $A \in \mathbb{R}^n$, odpowiadający permutacji ω. Wylosuj liczbę r z przedziału $[0,1]$. Jeżeli $r \leqslant 0,5$, to połóż $B := A'$. W przeciwnym przypadku połóż $B := B'$.
11. Połóż $iter := iter + 1$. Jeżeli $iter < maxiter$ to idź do kroku 4. W przeciwnym przypadku zwróć permutację π^* i wartość funkcji celu C^* i STOP.

Rysunek 5.7: Schemat operatora MX

Rozważmy dwa podziały rozłączne $\theta^1 = \{O_1^1, \ldots, O_m^1\}$, $\theta^2 = \{O_1^2, \ldots, O_m^2\}$ zbioru operacji, $o_l^i = |O_l^i|$, $l \in M$, $\theta^i \in \Theta$, $i = 1, 2$. Niech π^1 oraz π^2 będą dowolnymi chromosomami stanowiącymi zestaw permutacji odpowiednio podziału rozłącznego θ^1 oraz θ^2. Celem modyfika-

cji operatora GX jest umożliwienie jego zastosowania do permutacji π^1, π^2. Z założenia, w wyniku wykonania jednego z kroków każdej z *maxiter* iteracji będzie formowany chromosom δ, będący zestawem permutacji podziału $\theta = \{O_1, \ldots, O_m\}$, $o_l = |O_l|$, $l \in M$, gdzie *maxiter* jest parametrem. Należy zatem znaleźć metodę konstrukcji podziału θ. Najbardziej naturalnym rozwiązaniem wydaje się być dodanie każdej operacji $i \in O$ z równym prawdopodobieństwem do zbioru O_k albo zbioru O_l, gdzie $k = m_i^{\pi^1}$, $l = m_i^{\pi^2}$. Operator MX, powstały w wyniku wprowadzenia powyższej modyfikacji, został przedstawiony na rys. 5.7. W stosunku do operatora GX zostały dodane kroki 2, 3, w których tworzony jest podział θ. Modyfikacji też uległ krok 9, w którym algorytm $AP(PWK(\delta))$ został zastąpiony algorytmem $APM(RAN, PWK(\delta))$. Zmiana ta, ze względu na znajomość podziału θ, nie jest jednak konieczna.

Drugim składnikiem algorytmu $GAGX$, który został poddany modyfikacji, jest algorytm symulowanego wyżarzania SW'. Należy jednak dodać, że z przyczyn podobnych jak w przypadku algorytmu priorytetowego – modyfikacja ta nie jest konieczna. Oryginalnie, w algorytmie SW' zostało użyte sąsiedztwo $AN^3(\pi)$, $\pi \in \Pi$. Algorytm SW'' uzyskuje się zatem poprzez zastąpienie oryginalnego sąsiedztwa sąsiedztwem $N(\pi) = AN^3(\pi) \cup IN^1(\pi)$.

Podsumowując, algorytm $GAMX$ uzyskuje się z algorytmu $GAGX$ poprzez zastąpienie operatora GX i algorytmu SW' odpowiednio operatorem MX i algorytmem SW''. Zmianie nie ulegają parametry *maxp*, *maxiter*, λ, dt, σp, σk, których ustalenie jest niezbędne do poprawnego działania algorytmu.

5.6 Wyniki badań numerycznych

Celem przeprowadzonych badań numerycznych było porównanie jakości algorytmów prezentowanych zarówno w tym rozdziale, rozdziale poprzednim jak i literaturze pod względem czasu pracy jak i jakości generowanych rozwiązań. Badania można podzielić na dwa etapy. W etapie pierwszym zostały porównane ze sobą algorytmy prezentowane zarówno w tym, jak i poprzednim rozdziale. Porównanie takie było możliwe między innymi po sprowadzeniu problemu $J, MPR|t_{jkl}, t'_{kl}|C_{\max}$ do problemu $J, DR|t_{jkl}, t'_{kl}|C_{\max}$ poprzez konstrukcję przydziału wózków do operacji transportowych. Etap pierwszy był wykonany przy pomocy odpowiednio zmodyfikowanych instancji testowych TR (prezentowanych w sekcji 4.3.1). Opis ich modyfikacji zamieszczono w sekcji 5.6.1. Drugi etap badań polegał na porównaniu algorytmów prezentowanych w tej pracy z algorytmami literaturowymi, zaprojektowanymi do rozwiązywania przypadku szczególnego badanego pro-

blemu – problemu $J,R|t_{jkl},t'_{kl}|C_{\max}$. Realizacja etapu drugiego polegała na rozwiązaniu instancji testowych prezentowanych w pracy [7] i porównaniu dostarczonych wyników z wynikami prezentowanymi w pracach [7,109]. Wspomniane instancje testowe zostały też szczegółowo opisane w sekcji 5.6.1. Należy dodać, że wszystkie algorytmy opisywane w tym rozdziale były zaimplementowane w Delphi 6 i uruchamiane na komputerze z procesorem AMD Athlon XP 2500+.

5.6.1 Instancje testowe

Pierwszy zestaw instancji testowych dla badanego problemu stanowi modyfikację 480 instancji prezentowanych w sekcji 4.3.1. Podobnie jak instancje oryginalne, instancje te są podzielone na 40 zestawów, oznaczanych przez $TM01, TM02, \ldots, TM40$, po 12 instancji w każdym zestawie. Każda instancja ma swoją unikalną nazwę w formacie $TMa/m^t/b/c/d$, gdzie a, b, c, d, m^t są zmiennymi mogącymi przyjmować te same wartości co oryginalne instancje. Nie zmienia się też sposób generowania czasów transportu i przejazdów pustych, czasów wykonania poszczególnych operacji produkcyjnych i transportowych oraz przydziału maszyn produkcyjnych do operacji produkcyjnych. Jedynym elementem, który uległ zmianie, jest sposób generowania zbioru maszyn, na których mogą być wykonane operacje transportowe. Oryginalnie, zbiór ten był jednoelementowy, tzn. określana była maszyna m_i, na której należało wykonać operację $i \in O^t$, przy użyciu równania (4.24). W przypadku zmodyfikowanych instancji, dla każdej operacji transportowej $i \in O^t$ określony jest dwuelementowy zbiór $\mu_i = \{a, a+1\}$, gdzie

$$a = \min\left\{m^t - 1, 1 + \left\lfloor \frac{(z_i - 1)\cdot(m^t - 1)}{r} \right\rfloor\right\}, \tag{5.36}$$

zaś z_i jest numerem zadania, w ramach którego wykonywana jest operacja transportowa $i \in O^t$.

Drugim użytym zestawem instancji testowych jest zestaw 82 instancji opisanych w pracy [7]. W pracy tej zaprezentowano 4 układy maszyn bazując na typowych układach takich jak pętla czy drabina[1] (ang. *ladder*). Układy te posłużyły do określenia czasów transportu i przejazdów pustych wózków AGV. Zakładano też, że liczba wózków każdorazowo wynosiła $m^t = 2$, zaś czasy transportu i przejazdów pustych są sobie równe. W pracy zaprezentowano 10 zestawów zadań wraz z ich marszrutami technologicznymi, w których zbiór zadań J zawierał od 5 do 8 elementów, zaś

[1] Układ ten jest dość specyficznym układem typu siatka

liczba operacji produkcyjnych n^p wahała się pomiędzy 13 a 21. Dodatkowo, uwzględniono czas transportu każdego z zadań ze stacji załadowczo-wyładowczej (stację tą można utożsamić z maszyną o numerze 0, w której wykonywana jest pierwsza operacja każdego zadania w czasie równym 0). W wyniku różnych kombinacji 10 zestawów zadań, 4 układów maszyn i odpowiedniego skalowania czasów wykonania poszczególnych operacji, zostały wygenerowane 82 instancje. Każda instancja posiada swoją unikalną nazwę w postaci $EXxyz$, gdzie x, y, z są zmiennymi. Zmienne $x \in \{1, \ldots, 10\}$, $y = \{1, \ldots, 4\}$ określają odpowiednio numer użytego zestawu zadań oraz układu maszyn. Zmienna $z \in \{0, 1\}$ jest współczynnikiem skalującym i występuje w nazwie nieobowiązkowo. Dla $z = 0$ i $z = 1$ przyjmuje się, że czasy wykonania operacji produkcyjnych zostały odpowiednio podwojone bądź potrojone, podczas gdy w obu przypadkach czasy transportów i przejazdów pustych zostały skrócone o połowę.

5.6.2 Wyniki badań algorytmów konstrukcyjnych

Badania numeryczne algorytmów konstrukcyjnych zostały przeprowadzone przy użyciu „najbardziej interesujących" grup instancji $TM$16-25, $TM$36-40. Celem badań numerycznych algorytmów konstrukcyjnych było wyłonienie algorytmu generującego najlepsze rozwiązania jak też porównanie zaobserwowanych tendencji z tendencjami zaobserwowanymi w przypadku analogicznych badań prowadzonych w rozdziale 4. W tym miejscu można poczynić pierwszą obserwację dotyczącą czasu pracy algorytmów. Czas pracy badanych algorytmów $APM(RM, RO)$, $INT4$ w stosunku do analogicznych algorytmów $AP(R)$, $INT3$ dla problemu $J, DR|t_{jkl}, t'_{kl}|C_{\max}$ w zasadzie nie uległ zmianie, dlatego nie będzie dalej rozważany.

Algorytmy priorytetowe $APM(RM, RO)$, opisywane w sekcji 5.4.1, dla każdej instancji z zestawu $TM$16-25, $TM$36-40 były uruchamiane z operacyjnymi regułami priorytetowymi $RO \in Z$, $Z = \{STT, LTT, SPT, LPT, RAN\}$ i maszynową regułą priorytetową $RM = RAN$. Wartości funkcji celu rozwiązań wyznaczonych dla poszczególnych instancji będą oznaczane odpowiednio przez C^{STT}, C^{LTT}, C^{SPT}, C^{LPT} i C^{RAN}. Na podstawie tych wielkości wyznaczona została też wartość minimalna

$$C^{MIN} = \min_{A \in Z}\{C^A\}. \tag{5.37}$$

Poszczególne wartości funkcji celu wyznaczone przez algorytm $INT4$ będą oznaczane symbolem C^{IN}. Na podstawie wyznaczonych wartości, korzystając z równania (4.26), została obliczona względna poprawa Φ_A^{MIN} wartości minimalnej C^{MIN} przez każdy z badanych algorytmów, gdzie $A \in Z \cup \{IN\}$.

Tabela 5.2: Średnie arytmetyczne poprawy $av\Phi_A^{MIN}$ dla zestawów $TR16$-25, $TR36$-40

TMa	$av\Phi_{STT}^{MIN}$ [%]	$av\Phi_{LTT}^{MIN}$ [%]	$av\Phi_{SPT}^{MIN}$ [%]	$av\Phi_{LPT}^{MIN}$ [%]	$av\Phi_{RAN}^{MIN}$ [%]	$av\Phi_{IN}^{MIN}$ [%]
$TR16$	-7,23	-10,78	-5,32	**-5,03**	-6,55	9,34
$TM17$	-7,58	-9,39	-4,97	-9,86	**-3,87**	12,16
$TM18$	**-1,44**	-10,77	-6,97	-9,43	-3,14	13,10
$TM19$	**-2,45**	-12,35	-6,62	-8,23	-8,96	3,89
$TM20$	-5,10	-21,03	-22,28	**-1,77**	-12,38	5,37
$TM21$	**-3,02**	-8,37	-10,56	-6,81	-4,61	10,37
$TM22$	**-1,58**	-12,17	-7,98	-14,76	-6,95	13,76
$TM23$	**-1,19**	-10,71	-10,52	-16,21	-8,40	9,11
$TM24$	-11,93	-8,41	-6,83	-16,03	**-1,39**	16,67
$TM25$	-6,00	-6,59	-10,01	-7,68	**-3,61**	16,37
$TM36$	-7,90	-7,82	-6,84	-11,83	**-5,10**	20,84
$TM37$	-5,47	-9,04	**-5,27**	-6,30	-6,60	16,26
$TM38$	-5,20	-8,82	-9,21	**-3,19**	-6,03	20,59
$TM39$	**-6,08**	-9,74	-7,45	-9,19	-8,10	16,75
$TM40$	**-2,74**	-9,56	-9,20	-6,40	-6,82	16,56
średnia	**-4,99**	-10,37	-8,67	-8,85	-6,17	13,41

Ostatecznie, dla każdego zestawu $TR16$-25, $TR36$-40 wyliczono średnie arytmetyczne poprawy $av\Phi_A^{MIN}$ i przedstawiono w tabeli 5.2. Największe wielkości $av\Phi_A^{MIN}$, $A \in Z$, dla każdego z zestawów zostały pogrubione.

Z tabeli 5.2 wynika, że najlepszymi spośród testowanych algorytmów priorytetowych są algorytmy kolejno z operacyjną regułą priorytetową STT, RAN, SPT, LPT, LTT. Kolejność ta jest identyczna jak w przypadku instancji TR. Ciekawe jest też porównanie wielkości C^{MIN} wyznaczonych dla poszczególnych instancji TM i odpowiadających im instancji TR (poprzez instancję TR odpowiadającą instancji $TMa/m^t/b/c/d$ będzie rozumiana instancja o nazwie $TRa/m^t/b/c/d$). Okazuje się, że dla poszczególnych instancji TM wielkości C^{MIN} te są z reguły gorsze, niż dla instancji TR. W instancjach TR przydział maszyn do poszczególnych operacji transportowych został dokonany arbitralnie przy użyciu równania (4.24). W przypadku instancji TM przydział był konstruowany losowo przez algorytmy priorytetowe przy użyciu maszynowej reguły priorytetowej $RM = RAN$. Można zatem wysnuć wniosek, że losowy wybór maszyny transportowej do wykonania poszczególnych transportów przynosi rezultaty ogólnie gorsze niż, wspomniany powyżej, wybór arbitralny.

Tabela 5.3: Wartości funkcji celu rozwiązań wygenerowanych przez algorytm $INT4$

TMa	$TMa/2/1/c/d$			$TMa/2/2/c/d$		
	2/2	2/5	5/5	2/2	2/5	5/5
$TM16$	1122	1177	1240	1112	1072	1131
$TM17$	851	937	1052	849	889	943
$TM18$	959	987	1076	944	983	1027
$TM19$	959	1076	1157	971	1037	1064
$TM20$	1002	1051	1109	1001	1064	1093
$TM21$	1210	1338	1428	1205	1277	1344
$TM22$	1082	1206	1379	1128	1202	1310
$TM23$	1159	1278	1397	1153	1197	1309
$TM24$	1076	1275	1411	1028	1166	1315
$TM25$	1099	1228	1366	1161	1245	1316
$TM36$	1540	2594	2950	1472	1909	2229
$TM37$	1712	2514	2904	1653	1885	2137
$TM38$	1497	2567	2892	1389	1770	2061
$TM39$	1491	2446	2749	1475	1776	2088
$TM40$	1533	2488	2842	1432	1828	2126

TMa	$TMa/4/1/c/d$			$TMa/4/2/c/d$		
	2/2	2/5	5/5	2/2	2/5	5/5
$TM16$	1122	1169	1170	1112	1069	1069
$TM17$	851	911	912	849	889	889
$TM18$	957	973	973	944	963	963
$TM19$	959	1070	1070	971	1025	1025
$TM20$	1002	1040	1040	1001	1058	1058
$TM21$	1210	1307	1307	1205	1251	1271
$TM22$	1082	1122	1141	1128	1149	1174
$TM23$	1159	1213	1213	1153	1184	1184
$TM24$	1076	1122	1153	1028	1088	1092
$TM25$	1097	1135	1149	1161	1227	1233
$TM36$	1449	1736	2029	1462	1532	1636
$TM37$	1693	1820	1979	1653	1723	1730
$TM38$	1407	1666	1899	1383	1432	1523
$TM39$	1439	1709	1917	1474	1492	1577
$TM40$	1484	1668	1855	1431	1511	1582

Jakość poszczególnych algorytmów priorytetowych można też określić na podstawie liczby instancji, dla których determinowały one wielkość C^{MIN}. I tutaj, w przeciwieństwie do instancji TR, kolejność uległa pewnej zmianie: najlepszy okazał się algorytm z regułą operacyjną STT determinując wielkość C^{MIN} dla 67 instancji, później algorytm z regułą RAN (40 instancji), dalej kolejno algorytm z regułą LPT (35 instancji) wyprzedzając algorytm z regułą SPT (26 instancji) i LTT (13 instancji).

Pomimo stosunkowo niskiej jakości rozwiązań dostarczonych przez algorytmy priorytetowe, algorytm $INT4$ nie zdołał poprawić wartości C^{MIN} dla 6 instancji. Wartości funkcji celu C^{IN} dla wszystkich 180 instancji zostały przedstawione w tabeli 5.3. Średnia arytmetyczna wszystkich wyznaczonych wartości Φ_{IN}^{MIN}, poprawa najmniejsza i największa wynosi odpowiednio 13%, −6% oraz 35%.

Wielkości C^{IN} wyznaczone dla instancji TM były mniejsze od wielkości C^{IN} wyznaczonych przez algorytm $INT3$ dla odpowiadających im instancji w zestawie TR w 139 przypadkach, zaś nieznacznie większe jedynie dla 4 instancji. Wynik ten jest dość oczywisty, ponieważ algorytm $INT4$ dokonywał próbnego wstawienia operacji transportowych na więcej niż jedną maszynę transportową. Wyznaczone w ten sposób poszczególne wielkości $d_{\min}$ były z reguły mniejsze niż analogiczne wielkości wyznaczane w poszczególnych iteracjach przez algorytm $INT3$, co przyczyniło się do wygenerowania rozwiązań o mniejszych wartościach funkcji celu.

Analizując rozwiązania dostarczone przez algorytm $INT4$ pod kątem czasów transportu, liczby wózków i typu układu maszyn produkcyjnych można poczynić obserwacje analogiczne jak w przypadku algorytmu $INT3$ i instancji TR. Wraz ze wzrostem czasów transportu i przejazdów pustych (zwiększeniem współczynników skalujących c, d) przewaga algorytmu $INT4$ nad algorytmami priorytetowymi $APM(RM, RO)$ maleje. Podobnie jest w przypadku zastąpienia układu maszyn typu siatka układem typu pętla (zmiana wartości współczynnika b z 2 na 1) oraz zmniejszenia liczby maszyn transportowych. Podobnych spostrzeżeń nie można poczynić w kontekście algorytmów priorytetowych.

5.6.3 Wyniki badań algorytmów popraw – instancje TM

Badania numeryczne opisywanych w tym rozdziale algorytmów poprawy będą rozpoczęte od badań przy użyciu zestawów instancji testowych $TM01$-$TM40$. Przeprowadzone badania mają na celu porównanie jakości wygenerowanych rozwiązań w kontekście czasów pracy badanych algorytmów. Uzyskane wyniki będą też odniesione do wyników dostarczonych przez

algorytm i–$TSAB_{AGV}$, który okazał się jednym z dwóch najlepszych algorytmów badanych w rozdziale 4. Ze względu na miarę odległości pomiędzy permutacjami użytą w schemacie dywersyfikacji obliczeń, algorytmu tego nie da się w prosty sposób zmodyfikować i zastosować do bezpośredniego rozwiązywania problemu $J, MPR|t_{jkl}, t'_{kl}|C_{\max}$. Jednakże, jak już wspominano, rozwiązanie badanego problemu możliwe jest również poprzez sprowadzenie go do problemu $J, DR|t_{jkl}, t'_{kl}|C_{\max}$. Po wygenerowaniu rozwiązania początkowego przy użyciu algorytmu konstrukcyjnego $INT4$ (co wiąże się też z konstrukcją przydziału maszyn do poszczególnych operacji transportowych) algorytm i–$TSAB_{AGV}$ został uruchomiony dla każdej instancji TM dwukrotnie z takimi samymi wartościami parametrów jak dla instancji TR. Wartość funkcji celu najlepszego rozwiązania znalezionego przez algorytm dla każdej instancji będzie oznaczana przez C^{IT}. Sumaryczny czas pracy algorytmu dla każdej instancji będzie oznaczany symbolem T^{IT}.

Algorytm $TSAM_{AGV}$ dla każdej instancji TM został uruchomiony dwukrotnie z wartościami parametrów $maxl = 4$, $max\delta = 100$, $maxc = 2$, $max\_a = 15$, $max\_i = 5$ i dodatkowym ograniczeniem czasowym, równym 10 minut. W trakcie pierwszego uruchomienia maksymalna liczba iteracji była równa $maxiter = 30000$ po starcie algorytmu i po każdej poprawie wartości funkcji celu oraz $maxiter = 10000$ po wykonaniu skoku powrotnego. W trakcie drugiego uruchomienia została przyjęta stała wartość $maxiter = 16000$. Rozwiązanie początkowe było każdorazowo wygenerowane przy użyciu algorytmu $INT4$. Przez C^{TS} będzie oznaczana wartość funkcji celu najlepszego rozwiązania znalezionego przez algorytm dla każdej badanej instancji. Sumaryczny czas pracy algorytmu dla każdej instancji będzie oznaczany symbolem T^{TS}.

Algorytm $GAMX$ został uruchomiony dla każdej instancji trzykrotnie z ograniczeniem czasowym wynoszącym odpowiednio 200, 400 i 600 sekund oraz ustalonymi wartościami parametrów $maxp = 30$, $maxiter = 100$, $\sigma p = 0,99$, $\sigma k = 0,1$, $\lambda = 0,98$, $dt = 300$. W celu uproszczenia notacji, algorytm z ograniczeniem czasowym 200, 400 i 600 sekund będzie nazywany odpowiednio algorytmem $GAMX1$, $GAMX2$ i $GAMX3$, zaś zwrócone przez niego wartości funkcji celu najlepszych rozwiązań będą oznaczane odpowiednio przez C^{MX1}, C^{MX2} oraz C^{MX3}.

Dzięki własności 3.4 również w przypadku instancji TM wiadomo, że powyższe algorytmy rozwiązały optymalnie większość instancji w zestawach TM01-15, $TM$26-35. Dlatego instancje pochodzące z tych zestawów nie będą dalej rozważane. Wśród pozostałych zestawów optymalność dostarczonych rozwiązań została udowodniona jedynie dla 4 instancji. Są to instancje $TM$23/2/1/2/2, $TM$23/2/2/2/2, $TM$23/4/1/2/2, $TM$23/4/2/2/2.

Tabela 5.4: Średnie arytmetyczne poprawy $av\Phi_A^{IN}$ i średnie czasy pracy algorytmów popraw dla zestawów $TM16$-25, $TM36$-40

TMa	$av\Phi_{IT}^{IN}$ [%]	$av\Phi_{TS}^{IN}$ [%]	$av\Phi_{MX1}^{IN}$ [%]	$av\Phi_{MX2}^{IN}$ [%]	$av\Phi_{MX3}^{IN}$ [%]	T_{av}^{IT} [s]	T_{av}^{TS} [s]
$TM16$	9,53	9,82	10,03	10,05	**10,06**	350,0	37,2
$TM17$	5,18	**6,83**	6,47	6,54	6,57	365,7	76,2
$TM18$	5,55	6,46	6,72	**6,75**	**6,75**	279,1	54,4
$TM19$	9,85	11,25	11,28	**11,45**	**11,45**	406,5	66,4
$TM20$	7,09	7,82	7,76	**7,84**	**7,84**	249,0	43,5
$TM21$	10,27	**11,38**	10,66	10,71	10,81	693,5	191,0
$TM22$	12,29	**14,02**	12,75	12,86	13,01	692,7	282,1
$TM23$	10,51	**11,75**	10,80	10,98	11,01	620,4	243,8
$TM24$	7,29	**8,17**	7,57	7,62	7,90	766,0	335,8
$TM25$	10,33	**11,36**	10,56	10,69	10,79	730,2	220,8
$TM36$	6,50	**8,52**	6,72	7,15	7,29	1098,7	704,6
$TM37$	7,40	**9,33**	7,65	7,91	8,26	1089,1	602,4
$TM38$	5,52	**7,79**	5,01	5,41	5,62	1153,3	796,8
$TM39$	7,53	**9,97**	8,34	8,47	8,63	1126,9	691,0
$TM40$	7,52	**9,92**	7,77	8,07	8,39	1174,6	748,0
średnia	8,16	**9,63**	8,67	8,83	8,96	719,7	339,6

Wygenerowane przez algorytmy popraw poszczególne wielkości C^A, $A \in Z = \{IT, TS, MX1, MX2, MX3\}$, zostały odniesione do wartości C^{IN} poprzez wyznaczenie poprawy Φ_A^{IN}, korzystając z równania (4.26). Ostatecznie, dla każdego zestawu $TM16$-25, $TM36$-40 wyliczono średnie arytmetyczne poprawy $av\Phi_A^{IN}$, $A \in Z$, jak i średnie arytmetyczne T_{av}^A czasów pracy T^A, $A \in \{IT, TS\}$, poszczególnych algorytmów. Wszystkie wyznaczone wartości zostały przedstawione w tabeli 5.4. Największe wielkości $av\Phi_A^{IN}$, $A \in Z$, zostały pogrubione. W tabeli 5.5 naniesione są wartości funkcji celu najlepszych rozwiązań znalezionych przez wszystkie algorytmy; wielkości wygenerowane przez algorytm $TSAM_{AGV}$ i algorytm $GAMX3$ zostały odpowiednio pogrubione i oznaczone kursywą.

Niewątpliwie najlepszym, spośród testowanych algorytmów popraw, okazał się algorytm $TSAM_{AGV}$. Algorytm ten generował rozwiązania średnio lepsze niż pozostałe testowane algorytmy w krótszym, bądź porównywalnym czasie. Średnia arytmetyczna wszystkich wyznaczonych popraw Φ_{TS}^{IN} jest większa odpowiednio o ponad $0,6$ i prawie $1,5$ punktu procentowego od analogicznych średnich wyznaczonych dla algorytmów $GAMX3$ oraz i–$TSAB_{AGV}$. Najmniejszą i największą poprawę Φ_{TS}^{IN}, wynoszącą $3,6\%$

Tabela 5.5: Wartości funkcji celu najlepszych rozwiązań dla instancji z zestawów $TM16$-25, $TM36$-40

TMa	$TMa/2/1/c/d$			$TMa/2/2/c/d$		
	2/2	2/5	5/5	2/2	2/5	5/5
$TM16$	***976***	***1049***	**1077**	*976*	*1017*	*1025*
$TM17$	***805***	**866**	**923**	***801***	*831*	**857**
$TM18$	*881*	*932*	**971**	*874*	*919*	*929*
$TM19$	***874***	*922*	**965**	*876*	*920*	**955**
$TM20$	*931*	***978***	**994**	***934***	***967***	*971*
$TM21$	*1082*	**1161**	**1224**	***1085***	*1160*	**1212**
$TM22$	*947*	**1078**	**1172**	*947*	**1027**	**1099**
$TM23$	***1032***	*1144*	**1213**	***1032***	**1065**	**1156**
$TM24$	*971*	**1159**	**1246**	*965*	**1074**	**1164**
$TM25$	***1001***	***1138***	**1210**	*999*	*1079*	**1146**
$TM36$	**1446**	**2479**	*2757*	***1345***	**1761**	**2013**
$TM37$	**1538**	**2383**	**2690**	*1480*	**1731**	**1933**
$TM38$	**1391**	**2445**	**2751**	1280	**1635**	**1888**
$TM39$	*1396*	**2268**	**2516**	**1279**	**1669**	**1887**
$TM40$	*1381*	**2351**	**2642**	*1294*	**1679**	**1916**

TMa	$TMa/4/1/c/d$			$TMa/4/2/c/d$		
	2/2	2/5	5/5	2/2	2/5	5/5
$TM16$	***976***	*1040*	***1040***	*976*	***1017***	***1017***
$TM17$	***805***	***848***	***848***	***801***	***830***	***830***
$TM18$	*877*	***927***	***927***	*874*	*913*	*913*
$TM19$	***874***	***918***	***918***	***876***	***920***	***920***
$TM20$	***931***	***975***	***975***	***934***	***967***	***967***
$TM21$	*1082*	***1124***	*1126*	*1080*	*1123*	***1124***
$TM22$	***953***	*988*	**989**	***947***	*976*	*978*
$TM23$	***1032***	*1037*	**1048**	***1032***	*1035*	*1038*
$TM24$	*961*	*1022*	**1021**	*965*	*1007*	*1018*
$TM25$	*1001*	*1041*	*1049*	***998***	*1040*	*1038*
$TM36$	*1347*	**1562**	**1720**	***1331***	***1407***	**1456**
$TM37$	*1496*	**1655**	**1741**	*1457*	*1552*	**1561**
$TM38$	*1305*	**1498**	**1641**	1254	**1337**	**1381**
$TM39$	**1308**	**1494**	**1596**	*1274*	*1373*	**1405**
$TM40$	**1293**	**1471**	**1605**	***1268***	*1362*	**1398**

i $16,7\%$, zanotowano odpowiednio dla instancji $TM24/2/2/2/2$ oraz instancji $TR39/4/1/5/5$. Przewaga algorytmu $TSAM_{AGV}$ nad algorytmem $GAMX3$ maleje w przypadku instancji o małym rozmiarze. Średnie poprawy $av\Phi_{TS}^{IN}$ mniejsze od wartości $av\Phi_{MX3}^{IN}$ zanotowano dla zestawów $TM16$ oraz $TM18$-20. Wspomniane spostrzeżenie można również poczynić na podstawie analizy danych z tabeli 5.5. Z tabeli tej wynika też, że algorytm $GAMX3$ generuje rozwiązania z reguły gorsze dla instancji, gdzie czasy wykonania operacji transportowych są stosunkowo długie. Fakt ten jest szczególnie widoczny w przypadku instancji $TMa/2/1/5/5$, $TMa/2/2/5/5$, $a = 16, \ldots, 25, 36, \ldots, 40$. Ogólnie, algorytm $TSAM_{AGV}$ wygenerował najlepsze rozwiązanie dla 116 instancji, zaś algorytm $GAGX$ dla 110 spośród 180 badanych instancji. Wielkości C^{TS} i C^{GX3} były sobie równe tylko 55 razy. To oznacza, że pomimo wyższości tego pierwszego, algorytm $GAGX$ stanowi dobre uzupełnienie algorytmu $TSAM_{AGV}$.

Charakterystyczny jest fakt, że żadna z wielkości przedstawionych w tabeli 5.5 nie została wygenerowana przez algorytm i–$TSAB_{AGV}$. Pomimo dodatkowych mechanizmów dywersyfikacji obliczeń i faktu, że wartości C^{IT} w przypadku instancji TM i odpowiadających im instancji TR są porównywalne, algorytm ten dostarczył najgorszych rozwiązań spośród wszystkich badanych w tym rozdziale algorytmów popraw. Jest to oczywistą konsekwencją braku mechanizmów przemieszczania operacji pomiędzy poszczególnymi maszynami transportowymi.

5.6.4 Wyniki badań algorytmów popraw – instancje EX

Poniżej prezentowane są wnioski wynikające z porównania algorytmów przedstawionych w tej pracy z algorytmami literaturowymi, zaprojektowanymi do rozwiązywania przypadku szczególnego rozważanego problemu – problemu $J, R|t_{jkl}, t'_{kl}|C_{\max}$. Badania będą przeprowadzone przy użyciu 82 instancji testowych z pracy [7], nazywanych dalej instancjami EX. Wspomniane instancje zostały w skrócie opisane w sekcji 5.6.1.

W pracy autorstwa Bilge'a i Ulusoy'a [7] problem $J, R|t_{jkl}, t'_{kl}|C_{\max}$ rozwiązany został w sposób przybliżony poprzez równoległe rozwiązanie dwóch subproblemów; problemu szeregowania zadań i problemu szeregowania wózków. W pracy został przedstawiony algorytm TW (od ang. *time window*), którego jednym z podstawowych elementów był algorytm heurystyczny, generujący w każdej iteracji uszeregowanie operacji na maszynach produkcyjnych. Następnie, dla danego uszeregowania stosowano procedurę STW (od ang. *sliding time window*) konstruującą przydział oraz kolejność wykonania transportów przez wózki AGV. Dla każdej instancji EX różne warianty procedury były uruchamiane trzykrotnie. W pracy nie ma jednak informacji

zarówno na temat czasu pracy, jak i komputera, za pomocą którego wykonano badania numeryczne. W pracy autorstwa Ulusoy'a, Sivrikaya-şerifoğlu i Bilge'a [109] instancje EX zostały rozwiązane przy użyciu algorytmu genetycznego. W pracy prezentuje się specyficzną reprezentację rozwiązania, w której chromosom reprezentuje zarówno kolejność wykonania operacji na poszczególnych maszynach produkcyjnych, jak również przydział i kolejność wykonania poszczególnych transportów przez wózki AGV. Idea ta jest jednak zupełnie inna od przedstawianej w tej pracy. Mianowicie, na każdy gen w chromosomie składa się litera określająca operację oraz numer wózka AGV przypisanego do wykonania transportu w kierunku maszyny, na której należy wykonać reprezentowaną operację. Dla każdej instancji EX algorytm genetyczny został łącznie uruchomiony 20 razy i poprawił większość rozwiązań dostarczonych przez algorytm TW. W cytowanej pracy ponownie jednak brakuje informacji zarówno na temat czasu pracy algorytmu, jak i komputera, za pomocą którego wykonano obliczenia. W dalszej części tej pracy algorytmy zaprojektowane przez cytowanych autorów będą nazywane algorytmami BU, zaś wartości funkcji celu najlepszych rozwiązań dostarczonych przez te algorytmy dla każdej instancji EX będą oznaczane symbolem C^{BU}.

W cytowanej już pracy [109] przedstawiono też metodę wyznaczania dolnych ograniczeń wartości funkcji celu dla poszczególnych instancji badanego tam problemu $J, R|t_{jkl}, t'_{kl}|C_{\max}$. Jest to bardzo prosta metoda bazująca na dolnych ograniczeniach generowanych przez maszyny (ang. *machine-based bound*). Tym niemniej, dzięki metodzie możliwe było udowodnienie optymalności wielu, spośród wszystkich rozważanych instancji EX.

Badania numeryczne algorytmów i–$TSAB_{AGV}$, $TSAM_{AGV}$ i $GAMX$ dla instancji EX zostały przeprowadzone w sposób analogiczny jak w przypadku instancji TM. Algorytmy zostały uruchomione z identyczną liczbą powtórzeń oraz identycznymi wartościami poszczególnych parametrów. Wyjątkiem są tu niektóre parametry algorytmu $GAMX$, które przyjęły wartości $\sigma p = 0,9$, $\lambda = 0,9$, $dt = 500$. W identyczny sposób będą też oznaczane wartości funkcji celu najlepszych rozwiązań C^A, $A \in Z = \{IT, TS, GX1, GX2, GX3\}$ oraz sumaryczne czasy pracy T^{TS}, T^{IT} poszczególnych algorytmów. Nie ulega też zmianie sposób generowania rozwiązań i populacji początkowych. Poza wielkościami C^A, $A \in Z$, poniżej będzie również rozważana wielkość C^{IN}, oznaczająca wartość funkcji celu rozwiązań dostarczonych dla poszczególnych instancji EX przez algorytm $INT4$, oraz wielkość

$$C^{BST} = \min_{A \in Z \cup \{IN\}} \{C^A\}. \tag{5.38}$$

Dla każdej instancji EX, przy użyciu równania (4.26), wyznaczona została poprawa Φ_A^{BU} wielkości C^{BU} przez poszczególne algorytmy, gdzie $A \in Z \cup \{IN, BST\}$. Wielkości C^{BU} oraz ich poprawy zostały zaprezentowane w tabelach 5.6, 5.7. Wartości C^{BU}, dla których została udowodniona optymalność, zostały pogrubione.

Jako pierwsze zostaną omówione instancje $EXxyz$ (przedstawione w tabeli 5.6), dla których współczynnik skalujący z przyjmuje wartość $z = 0$ lub $z = 1$. W przypadku omawianej grupy instancji najlepszym, spośród algorytmów prezentowanych w tej pracy, okazał się algorytm $TSAM_{AGV}$. Dla każdej instancji z omawianej grupy wartość Φ_{TS}^{BU} jest równa wartości Φ_{BST}^{BU}. Poprawę najmniejszą, wynoszącą $-7,75\%$, zanotowano dla instancji $EX810$. Średnia arytmetyczna wszystkich wyznaczonych popraw Φ_{TS}^{BU} wynosi $-0,75\%$. Nieco gorsze okazały się algorytmy $GAMX$1-3 oraz i–$TSAB_{AGV}$, dla których średnia arytmetyczna wszystkich popraw $\Phi_{GX1}^{BU}, \ldots, \Phi_{GX3}^{BU}$, Φ_{IT}^{BU} wynosi odpowiednio $-0,92\%, \ldots, -0,92\%$ oraz $-1,03\%$.

Na podstawie analizy tabeli 5.6 można poczynić kilka ważnych spostrzeżeń. Po pierwsze, żaden z badanych algorytmów nie zdołał poprawić żadnej z wartości C^{BU}. Charakterystyczny jest też fakt, że w przypadku aż 36, spośród 42 instancji, wartości C^{BST} i C^{BU} są sobie równe (o czym, oczywiście, świadczy równość $\Phi_{BST}^{BU} = 0$). Powyższe spostrzeżenia można tłumaczyć następująco. Dzięki technice wyznaczania dolnych ograniczeń z pracy [109] możliwe było udowodnienie optymalności rozwiązań dla 17, spośród 42 instancji. Niemożność poprawy wielkości C^{BU} dla tych instancji jest oczywista. Należy jednak pamiętać, że wspomniana technika wyznaczania dolnych ograniczeń charakteryzuje się dużą prostotą, przez co optymalność wielu pozostałych, potencjalnie optymalnych rozwiązań, mogła pozostać nie zauważona.

Pozostaje jeszcze kwestia 6 instancji, dla których wartość Φ_{BST}^{BU} była ujemna. W tym przypadku zaskakująca jest powtarzalność wartości funkcji celu wygenerowanych rozwiązań. Na przykład, wartości Φ_{GX1}^{BU}, Φ_{GX2}^{BU}, Φ_{GX3}^{BU}, Φ_{IT}^{BU} były różne (mniejsze) od wartości Φ_{TS}^{BU} tylko w przypadku jednej instancji. Powyższe fakty można tłumaczyć następująco. Rozważane instancje charakteryzują się zarówno niewielkim rozmiarem jak i niewielkim stosunkiem czasów transportu do czasów wykonania operacji produkcyjnych. Jak już wiadomo, instancje takie „nie sprzyjają" powstawaniu bloków operacji transportowych na ścieżkach krytycznych grafów konstruowanych dla poszczególnych rozwiązań tych instancji. Niewielka liczba i długość bloków operacji transportowych może być przyczyną nadmiernego zmniejszenia się liczebności sąsiedztwa $AN^3(\pi)$, $\pi \in \Pi$, które, jak wiado-

Tabela 5.6: Arytmetyczne poprawy Φ_A^{BU} dla instancji EX

Nazwa	C^{BU}	Φ_{IN}^{BU} [%]	Φ_{IT}^{BU} [%]	Φ_{TS}^{BU} [%]	Φ_{MX1}^{BU} [%]	Φ_{MX2}^{BU} [%]	Φ_{MX3}^{BU} [%]	Φ_{BST}^{BU} [%]
$EX110$	**126**	0,00	0,00	0,00	0,00	0,00	0,00	0,00
$EX210$	**148**	-9,46	0,00	0,00	0,00	0,00	0,00	0,00
$EX310$	148	-1,35	-1,35	-1,35	-1,35	-1,35	-1,35	-1,35
$EX410$	119	-2,52	-2,52	0,00	0,00	0,00	0,00	0,00
$EX510$	**102**	0,00	0,00	0,00	0,00	0,00	0,00	0,00
$EX610$	186	-8,60	0,00	0,00	0,00	0,00	0,00	0,00
$EX710$	**137**	-6,57	0,00	0,00	0,00	0,00	0,00	0,00
$EX810$	**271**	-7,75	-7,75	-7,75	-7,75	-7,75	-7,75	-7,75
$EX910$	176	-5,11	-1,70	0,00	0,00	0,00	0,00	0,00
$EX1010$	236	-4,66	-1,27	-0,85	-2,54	-2,54	-2,54	-0,85
$EX120$	**123**	0,00	0,00	0,00	0,00	0,00	0,00	0,00
$EX220$	**143**	-11,19	0,00	0,00	0,00	0,00	0,00	0,00
$EX320$	145	0,00	0,00	0,00	0,00	0,00	0,00	0,00
$EX420$	114	-1,75	-1,75	0,00	-1,75	-1,75	-1,75	0,00
$EX520$	100	0,00	0,00	0,00	0,00	0,00	0,00	0,00
$EX620$	181	-8,84	0,00	0,00	0,00	0,00	0,00	0,00
$EX720$	**136**	-5,15	0,00	0,00	0,00	0,00	0,00	0,00
$EX820$	**268**	-7,09	-7,09	-7,09	-7,09	-7,09	-7,09	-7,09
$EX920$	173	-5,20	0,00	0,00	0,00	0,00	0,00	0,00
$EX1020$	236	-2,12	0,00	0,00	-1,27	-1,27	-1,27	0,00
$EX130$	**122**	0,00	0,00	0,00	0,00	0,00	0,00	0,00
$EX230$	**146**	-9,59	0,00	0,00	0,00	0,00	0,00	0,00
$EX330$	146	0,00	0,00	0,00	0,00	0,00	0,00	0,00
$EX430$	114	-0,88	-0,88	0,00	-1,75	-1,75	-1,75	0,00
$EX530$	99	0,00	0,00	0,00	0,00	0,00	0,00	0,00
$EX630$	182	-8,79	0,00	0,00	0,00	0,00	0,00	0,00
$EX730$	**137**	-5,11	0,00	0,00	0,00	0,00	0,00	0,00
$EX830$	270	-6,67	-6,67	-6,67	-6,67	-6,67	-6,67	-6,67
$EX930$	174	-6,32	0,00	0,00	0,00	0,00	0,00	0,00
$EX1030$	237	-2,53	0,00	0,00	0,00	0,00	0,00	0,00
$EX140$	**124**	0,00	0,00	0,00	0,00	0,00	0,00	0,00
$EX241$	**217**	-13,82	0,00	0,00	0,00	0,00	0,00	0,00
$EX340$	151	0,00	0,00	0,00	0,00	0,00	0,00	0,00
$EX341$	221	0,00	0,00	0,00	0,00	0,00	0,00	0,00
$EX441$	172	-2,33	-1,16	0,00	0,00	0,00	0,00	0,00
$EX541$	**148**	0,00	0,00	0,00	0,00	0,00	0,00	0,00
$EX640$	184	-8,70	0,00	0,00	0,00	0,00	0,00	0,00
$EX740$	**137**	-8,03	0,00	0,00	0,00	0,00	0,00	0,00
$EX741$	**203**	-5,42	0,00	0,00	0,00	0,00	0,00	0,00
$EX840$	273	-7,33	-7,33	-7,33	-7,33	-7,33	-7,33	-7,33
$EX940$	175	-8,57	-3,43	0,00	0,00	0,00	0,00	0,00
$EX1040$	240	-3,75	-0,42	0,00	-0,42	-0,42	-0,42	0,00
średnia		-4,39	-1,03	-0,75	-0,92	-0,92	-0,92	-0,75

Tabela 5.7: Arytmetyczne poprawy Φ_A^{BU} dla instancji EX, c.d.

Nazwa	C^{BU}	Φ_{IN}^{BU} [%]	Φ_{IT}^{BU} [%]	Φ_{TS}^{BU} [%]	Φ_{MX1}^{BU} [%]	Φ_{MX2}^{BU} [%]	Φ_{MX3}^{BU} [%]	Φ_{BST}^{BU} [%]
$EX11$	96	-14,58	0,00	0,00	0,00	0,00	0,00	0,00
$EX21$	104	-9,62	-2,88	1,92	-2,88	-2,88	-2,88	1,92
$EX31$	105	-11,43	-5,71	5,71	0,95	2,86	5,71	5,71
$EX41$	116	-8,62	-4,31	3,45	3,45	3,45	3,45	3,45
$EX51$	87	-18,39	-8,05	0,00	0,00	0,00	0,00	0,00
$EX61$	120	-8,33	-4,17	1,67	0,00	0,00	0,00	1,67
$EX71$	118	-17,80	-2,54	3,39	4,24	5,08	5,08	5,08
$EX81$	152	-5,92	-5,92	-5,92	-5,92	-5,92	-5,92	-5,92
$EX91$	117	-7,69	-7,69	0,85	0,85	0,85	0,85	0,85
$EX101$	150	-7,33	-1,33	2,00	2,00	2,00	2,00	2,00
$EX12$	82	-4,88	-4,88	0,00	0,00	0,00	0,00	0,00
$EX22$	**76**	-5,26	-5,26	0,00	0,00	0,00	0,00	0,00
$EX32$	85	-1,18	-1,18	0,00	0,00	0,00	0,00	0,00
$EX42$	88	-13,64	-6,82	1,14	-1,14	-1,14	-1,14	1,14
$EX52$	69	-7,25	-2,90	0,00	0,00	0,00	0,00	0,00
$EX62$	98	-10,20	-5,10	0,00	0,00	0,00	0,00	0,00
$EX72$	85	-10,59	3,53	4,71	4,71	5,88	5,88	5,88
$EX82$	142	-6,34	-6,34	-6,34	-6,34	-6,34	-6,34	-6,34
$EX92$	102	-3,92	-3,92	0,00	0,00	0,00	0,00	0,00
$EX102$	137	-7,30	-2,92	1,46	-0,73	0,00	0,00	1,46
$EX13$	84	-2,38	-2,38	0,00	-2,38	-2,38	-2,38	0,00
$EX23$	86	-8,14	-6,98	0,00	0,00	0,00	0,00	0,00
$EX33$	86	-4,65	0,00	0,00	0,00	0,00	0,00	0,00
$EX43$	91	-6,59	-6,59	2,20	1,10	1,10	1,10	2,20
$EX53$	75	-1,33	-1,33	1,33	1,33	1,33	1,33	1,33
$EX63$	104	-7,69	-1,92	0,96	0,96	0,96	0,96	0,96
$EX73$	88	-12,50	-1,14	5,68	1,14	5,68	5,68	5,68
$EX83$	143	-6,99	-6,99	-6,99	-6,99	-6,99	-6,99	-6,99
$EX93$	105	-4,76	-4,76	0,00	0,00	0,00	0,00	0,00
$EX103$	143	-7,69	-1,40	4,20	2,10	2,10	2,10	4,20
$EX14$	103	-8,74	-4,85	0,00	-0,97	-0,97	-0,97	0,00
$EX24$	113	-11,50	-2,65	4,42	-2,65	-2,65	-2,65	4,42
$EX34$	113	-7,08	-7,08	0,88	-5,31	-5,31	-5,31	0,88
$EX44$	126	-18,25	-9,52	3,97	-6,35	-6,35	-0,79	3,97
$EX54$	97	-10,31	-8,25	1,03	1,03	1,03	1,03	1,03
$EX64$	120	-24,17	-8,33	0,00	-7,50	-7,50	-7,50	0,00
$EX74$	128	-23,44	-6,25	0,00	-9,38	-6,25	-6,25	0,00
$EX84$	163	-6,13	-3,68	0,00	0,00	0,00	0,00	0,00
$EX94$	123	-14,63	-4,07	1,63	1,63	1,63	1,63	1,63
$EX104$	164	-12,20	-5,49	3,66	-1,22	-1,22	-1,22	3,66
średnia		-9,49	-4,30	0,93	-0,86	-0,55	-0,34	1,00

mo, jest wykorzystywane w każdym z badanych algorytmów. Podobnie jest w przypadku sąsiedztwa $IN^1(\pi)$, które wykorzystane jest w algorytmach $TSAM_{AGV}$ i $GAMX$. Niedostateczny rozmiar sąsiedztwa najprawdopodobniej uniemożliwił przeprowadzenie dostatecznej eksploracji przestrzeni rozwiązań analizowanych instancji problemu. W przypadku algorytmu $TSAM_{AGV}$, którego czas pracy zależny jest od liczebności sąsiedztwa, fakt ten jest szczególnie widoczny; sumaryczny czas pracy T^{TS} dla każdej z omawianych instancji nie przekracza dwóch sekund. Sytuacja wygląda podobnie w przypadku algorytmu i–$TSAB_{AGV}$, którego sumaryczny czas pracy dla każdej instancji nie przekracza pół minuty. Na podstawie powyższych spostrzeżeń można mówić o pewnej wadzie użytego zbioru ruchów i sąsiedztwa.

Poniżej zostały omówione rezultaty badań numerycznych uzyskane dla pozostałych instancji EX, prezentowanych w tabeli 5.7. Ogólne tendencje są bardzo podobne do tych, wynikających z analizy tabeli 5.6; najlepszym spośród badanych algorytmów okazał się algorytm $TSAM_{AGV}$. Średnia arytmetyczna wszystkich wyznaczonych popraw Φ_{TS}^{BU} jest tym razem dodatnia i wynosi $0,93\%$. Najmniejszą i największą poprawę Φ_{TS}^{BU}, wynoszącą odpowiednio $-6,99\%$ i $5,71\%$, zanotowano odpowiednio dla instancji $EX83$ oraz instancji $EX31$. Algorytm $TSAM_{AGV}$ poprawił 21, spośród 40 wartości C^{BU}, zaś rozwiązania gorsze zwrócił jedynie dla 3 rozważanych instancji. Nieco gorsze od algorytmu $TSAM_{AGV}$ okazały się algorytmy $GAMX1$-3, dla których średnia arytmetyczna wszystkich popraw $\Phi_{GX1}^{BU}, \ldots, \Phi_{GX3}^{BU}$ wynosi odpowiednio $-0,34\%$, $-0,55\%$ oraz $-0,86\%$. Wartość funkcji celu C^{GX3} była mniejsza od wielkości C^{BU} dla 13 instancji i mniejsza od wielkości C^{TS} dla dwóch instancji. Algorytmem generującym wyniki najniższej jakości jest algorytm i–$TSAB_{AGV}$. Średnia arytmetyczna wszystkich popraw Φ_{IT}^{BU} wynosi $-4,30\%$. Algorytm ten poprawił wartość C^{BU} jedynie dla instancji $EX72$, jednakże wyznaczona wielkość pozostała większa od wielkości C^{TS}. Ogółem, testowane algorytmy poprawiły najlepsze wyniki literaturowe w przypadku 21 instancji, z czego w 19 przypadkach najlepsze rozwiązanie było wygenerowane przez algorytm $TSAM_{AGV}$. Wygenerowane rozwiązania były gorsze niż rozwiązania dostarczone przez algorytmy BU jedynie dla 3 instancji. Średnia arytmetyczna popraw Φ_{BST}^{BU} wynosi $1,0\%$.

W przypadku instancji przedstawionych w tabeli 5.7 nie można stwierdzić występowania tendencji podobnych do tych zaobserwowanych w tabeli 5.6; poszczególne poprawy z reguły mają różne wartości, są zarówno ujemne jak i dodatnie. Stosunkowo niewiele popraw przyjmuje wartość zero. Wydłużyły się również czasy pracy poszczególnych algorytmów. Sumaryczny czas pracy algorytmu $TSAM_{AGV}$ dla poszczególnych instancji wahał się

od 0 do 19 sekund, zaś czas pracy T^{IT} zmieniał się w zakresie od 0,1 do 66 sekund. Powyższe fakty mają związek z wydłużeniem się czasów transportu, czasów przejazdów pustych i skróceniem się czasów wykonania operacji produkcyjnych w poszczególnych instancjach w stosunku do instancji z tabeli 5.6.

Podsumowując, po raz kolejny najlepszym z badanych algorytmów okazał się algorytm wykorzystujący technikę poszukiwań z zabronieniami. W przypadku instancji EX przewaga algorytmu $TSAM_{AGV}$ nad pozostałymi wyrysowała się szczególnie wyraźnie.

5.7 Wnioski i uwagi

W tym rozdziale rozważono uogólniony problem gniazdowy, w którym uwzględniono czasy transportu zadań pomiędzy poszczególnymi maszynami produkcyjnymi, przy czym przydział wózków AGV do poszczególnych operacji transportowych nie był dany a priori. Założono również, że wózki pracują w szczególnie praktycznym trybie „zabierz i zostaw". Za kryterium optymalizacji przyjęto moment zakończenia wykonywania wszystkich zadań. W początkowych sekcjach rozdziału przedstawia się modele matematyczne oraz własności problemu. Dalej przedstawia się zbiór ruchów oraz sąsiedztwo, wykorzystane później w konstrukcji algorytmów rozwiązywania problemu. Na zakończenie zaprezentowano wyniki badań numerycznych prezentowanych algorytmów.

Badany problem charakteryzuje się dużym podobieństwem do problemu rozpatrywanego w rozdziale trzecim. Jedyna, aczkolwiek niezwykle istotna różnica polega na tym, że przydział maszyn jest dany a priori jedynie dla operacji produkcyjnych. Konstrukcja przydziału wózków AGV do poszczególnych operacji transportowych stanowi zatem dodatkowy element decyzyjny. Ze względu na podobieństwo problemów, w rozdziale tym udało się uogólnić wiele rezultatów uzyskanych w rozdziale trzecim. Dotyczy to głównie własności permutacji i grafu częściowego, które następnie zostały wykorzystane w efektywnej metodzie przeglądu sąsiedztwa generowanego w oparciu o zbiór ruchów typu wstaw.

Prezentowane wyniki badań numerycznych pozwoliły wnioskować na temat wyższości niektórych metod lokalnych poszukiwań nad innymi, również literaturowymi, w odniesieniu do badanego zagadnienia. Badania numeryczne ujawniły jednak pewne wady wspomnianych metod. Mowa jest tutaj o wadzie użytego sąsiedztwa i zbioru ruchów, ujawnionej przy okazji badań przeprowadzonych przy użyciu instancji EX (sekcja 5.6.4).

Eliminacja wspomnianej wady sąsiedztwa stanowi jedną z możliwych dróg dalszego rozwoju algorytmów rozwiązywania badanego problemu. Kolejna możliwa droga rozwoju polega na zdefiniowaniu i przebadaniu własności nowych miar odległości pomiędzy rozwiązaniami problemu będącymi zestawami permutacji dwóch różnych podziałów rozłącznych zbioru operacji. Miary takie mogą się przyczynić do powstania bardziej efektywnych schematów dywersyfikacji obliczeń jak również dalszego rozwoju badań dotyczących krajobrazu przestrzeni rozwiązań.

Podobnie jak w rozdziale 4 wszystkie badane w tym rozdziale instancje testowe zostały rozwiązane przy użyciu algorytmu poszukiwań losowych, który będzie nazywany algorytmem RS. Algorytm ten powstał poprzez iterowanie algorytmu $APM(RAN, RAN)$ z ograniczeniem czasowym, wynoszącym 10 minut. Algorytm RS we wspomnianym czasie jest w stanie przejrzeć od około 200 tysięcy do ponad 18 milionów rozwiązań, w zależności od rozmiaru problemu. Należy się spodziewać, że rozwiązania uzyskane w ten sposób będą jakościowo znacznie gorsze od rozwiązań wygenerowanych przez bardziej zaawansowane algorytmy popraw. Przypuszczenie to sprawdziło się w przypadku instancji TM. Dla tych instancji rozwiązanie minimalnie gorsze od rozwiązania wygenerowanego przez algorytm RS zanotowano jedynie w przypadku instancji $TM20/4/2/2/5$ oraz $TR20/4/2/5/5$ i algorytmu $i\text{–}TSAB_{AGV}$. Nieco inaczej wygląda sytuacja w przypadku instancji EX. W przypadku pierwszej grupy instancji, prezentowanej w tabeli 5.6, algorytm RS wygenerował rozwiązania o porównywalnej jakości do najlepszych z badanych algorytmów (dotyczy to również instancji, dla których udowodniono optymalność rozwiązań). Fakt ten Potwierdza tezę o „łatwości" owych instancji. W przypadku pozostałych instancji EX algorytm dostarcza rozwiązań, które są z reguły lepsze jedynie w stosunku do rozwiązań zwróconych przez algorytm $i\text{–}TSAB_{AGV}$.

Rozdział 6

Problem przepływowy z jednym wózkiem AGV

Najczęściej spotykanymi układami maszyn w systemach wykorzystujących wózki AGV są układy typu linia, pętla oraz siatka. System z liniowym układem maszyn charakteryzuje się prostotą, lecz nie wykorzystuje w pełni potencjału wózków AGV. W przypadku pracy bezawaryjnej, najbardziej wydajnymi wydają się być systemy cechowane układem maszyn typu siatka. Poważny problem stanowią tu jednak konflikty transportowe, których częste powstawanie prowadzi do blokad i, w konsekwencji, spadku ogólnej wydajności systemu. Jednym ze sposobów zapobiegania konfliktom transportowym jest dekompozycja systemu typu siatka w zbiór niezależnych, ale połączonych ze sobą, systemów typu pętla. Systemy tego typu łączą w sobie prostotę układów typu linia i zalety układów typu siatka, co czyni je szczególnie interesującymi dla praktyków i badaczy [12,13,39,40]. Z reguły, w systemie typu pętla znajduje się stacja załadowcza, stacja wyładowcza, jeden lub więcej wózków AGV i określona liczba maszyn produkcyjnych. Aby uniknąć konfliktów transportowych, narzuca się jednokierunkowy przepływ zadań, zaś na wózkach wymusza się cykliczny, jednokierunkowy tryb pracy. W poszczególnych cyklach pracy każdego wózka do systemu wprowadzane jest jedno zadanie przez stację załadowczą, jedno zadanie opuszcza system przez stację wyładowczą, zaś wszystkie pozostałe zadania są transportowane w kierunku kolejnych maszyn w pętli. Systemy takie mają zatem strukturę przepływową permutacyjną, zaś optymalizacja harmonogramu produkcji jest tu związana z koniecznością rozwiązania permutacyjnego problemu przepływowego.

W literaturze opisuje się wiele metod rozwiązywania permutacyjnego problemu przepływowego. Wśród najlepszych algorytmów „klasycznych”

wymienia się algorytm TSAB, wykorzystujący technikę poszukiwań z zabronieniami i opisany w pracy [71]. W trakcie swojej pracy żaden ze wspomnianych algorytmów nie uwzględnia jednak czasów transportu. To sprawia, że systemy produkcyjne, w których czasy transportu mają nietrywialny wpływ na pracę całego systemu, pracując według harmonogramów „klasycznych", wciąż pracują stosunkowo mało efektywnie. Jedną z szans na poprawę jakości pracy takich systemów upatruje się w próbie zaadaptowania algorytmów klasycznych do rozwiązywania permutacyjnego problemu przepływowego z transportem, co czyni się w niniejszym rozdziale.

W tym rozdziale rozważa się system produkcyjny o strukturze przepływowej (permutacyjnej), w którym maszyny zorganizowane są w układ typu pętla. Do transportu zadań pomiędzy maszynami wykorzystuje się jeden wózek AGV, pracujący w trybie jednokierunkowym. Rozważany problem polega na określeniu takiej kolejności wykonywania zadań przez maszyny, przy jednoczesnym uwzględnieniu pracy wózka AGV, by moment zakończenia wykonywania procesu technologicznego przyjął wartość minimalną. W rozszerzonej notacji Grahama z prac [43, 50] problem ten zalicza się do grupy problemów $FP, R1|t_{kl}, t'_{kl}|C_{\max}$. Dla sformułowanego problemu prezentuje się oryginalnie zaprojektowany model matematyczny oraz wprowadza się jego reprezentację permutacyjno-grafową. Po zbadaniu własności problemu proponuje się algorytm jego rozwiązywania. Powstały algorytm poddaje się badaniom numerycznym przy użyciu specjalnie dobranych instancji testowych.

6.1 Model matematyczny

Problem można sformułować następująco. Dany jest elastyczny system produkcyjny, w którym znajduje się zbiór $m^p \geqslant 2$ maszyn produkcyjnych $M^p = \{1, \ldots, m^p\}$. Każda z maszyn $2, \ldots, m^p - 1$ posiada jeden bufor wejściowy i jeden wyjściowy, zaś maszynę 1 i maszynę m^p utożsamia się odpowiednio ze stacją załadowczą i wyładowczą o nielimitowanych pojemnościach buforów. Maszyny ze zbioru M^p zorganizowane są w układ typu pętla, tj. rozmieszczone są wzdłuż zamkniętej drogi w ten sposób, że maszyna l, $1 < l \leqslant m^p$, poprzedzona jest maszyną $l - 1$. Dodatkowo, stacja załadowcza (maszyna 1) znajduje się w bezpośredniej bliskości stacji wyładowczej (maszyny m^p) a pomiędzy nimi znajduje się parking dla wózków AGV. Układ tego typu przedstawiony jest na rysunku 2.5b. Z założenia, wózki znajdujące się w systemie mogą poruszać się tylko w jednym kierunku, zgodnym z kierunkiem przepływu zadań. Zbiór wózków AGV (utożsamianych z maszynami transportowymi) określa się przez

$M^t = \{m^p + 1, m^p + 2, \ldots, m^p + m^t\}$, gdzie m^t jest liczbą wózków. W tym rozdziale analizuje się szczególny, jednakże równie praktyczny przypadek, w którym zbiór wózków jest jednoelementowy, tj. $m^t = 1$. Czas przejazdu wózka $v \in M^t$ pomiędzy maszyną l i maszyną $l+1$, $1 \leqslant l < m^p$, będzie oznaczany przez $t(l, l+1) > 0$. Podobnie, przez $t(m^p, 1) > 0$ będzie oznaczany czas przejazdu wózka pomiędzy stacją wyładowczą i załadowczą. Jednokierunkowy tryb pracy wózka i sposób rozmieszczenia maszyn jednoznacznie określa czasy przejazdu pomiędzy dowolną parą maszyn produkcyjnych $l, k \in M^p$ i dany jest równaniem

$$t(l,k) = \begin{cases} \sum_{i=l}^{k-1} t(i, i+1), & l < k, \\ \sum_{i=1}^{m^p-1} t(i, i+1) + t(m^p, 1) - \sum_{i=k}^{l-1} t(i, i+1), & l > k \\ 0, & l = k. \end{cases} \tag{6.1}$$

W systemie należy wykonać r zadań ze zbioru $J = \{1, 2, \ldots, r\}$, polegających na zrealizowaniu naprzemiennej sekwencji operacji produkcyjnych i transportowych. Detal wykonywany w ramach zadania $j \in J$ trafia do systemu poprzez stację załadowczą, gdzie wykonywana jest operacja załadowcza, w ramach której detal przytwierdzany jest do palety. Następnie, paleta z detalem transportowana jest przez wózek v w kierunku maszyn $2, \ldots, m^p$. W stacji wyładowczej wykonywana jest ostatnia, operacja wyładowcza, gdzie detal demontowany jest z palety i opuszcza system. Bardziej precyzyjnie, każde zadanie $j \in J$ podzielone jest na $m = 2m^p - 1$ operacji (m^p operacji produkcyjnych i $m^p - 1$ operacji transportowych). Operację produkcyjną l zadania j, notowaną jako para (l, j), należy wykonać na maszynie $l \in M^p$ w czasie $p_{l,j} > 0$. Operacja ta polega na:

1. pobraniu palety z detalem z bufora wejściowego maszyny l,
2. obróbce detalu na maszynie przez $p_{l,j}$ jednostek czasu oraz
3. wyładunku palety do bufora wyjściowego maszyny l.

Zbiór wszystkich operacji produkcyjnych będzie oznaczany przez $O^p = \{(l, j) : l \in M^p, j \in J\}$. Nietrudno zauważyć, że moc zbioru O^p wynosi $n^p = |O^p| = m^p \cdot r$.

Pomiędzy każdą parą operacji produkcyjnych (l, j), $(l+1, j)$, $1 \leqslant l < m^p$, $j \in J$, wykonywana jest operacja transportowa, notowana jako para $(m^p + l, j)$, polegająca na:

1. pobraniu przez wózek v palety z bufora wyjściowego maszyny l,
2. przewiezieniu jej pomiędzy maszynami $l, l+1 \in M^p$ w czasie $p_{m^p+l,j} = t(l, l+1)$ oraz
3. pozostawieniu jej w buforze wejściowym maszyny $l+1$.

Zakłada się, że czas pobrania i pozostawienia palety przez wózek jest pomijalnie mały. Założenie to nie jest ograniczające, bowiem jeśli nie jest ono spełnione dla rozpatrywanego systemu, odpowiednie czasy pobrania/pozostawienia mogą być dodane do czasu wykonania operacji transportowych i nie wpływają na prawdziwość dalszych rozważań. Zbiór wszystkich operacji transportowych, wykonywanych przez wózek v, będzie oznaczany przez $O^t = \{(m^p + l, j) : 1 \leqslant l < m^p, j \in J\}$. Moc zbioru O^t wynosi $n^t = |O^t| = (m^p - 1) \cdot r$. Dla kompletności oznaczeń, definiuje się zbiór wszystkich operacji $O = O^p \cup O^t = N \times J$, zawierający $n = |O| = n^p + n^t = m \cdot r$ elementów, gdzie $N = \{1, 2, \ldots, m\}$.

Przyjęta struktura systemu oraz technika przepływu zadań wymuszają jednakową kolejność przejścia zadań przez każde ze stanowisk (system ma strukturę przepływową permutacyjną). Dalej, niech permutacja $\pi = (\pi(1), \pi(2), \ldots, \pi(r))$ (nazywana również rozwiązaniem problemu), oznacza kolejność wykonywania zadań ze zbioru J przez każdą z maszyn produkcyjnych, zaś niech Π będzie zbiorem wszystkich permutacji. Dla permutacji $\pi \in \Pi$ przyjmuje się arbitralną kolejność wykonywania operacji transportowych przez wózek, generowaną na bazie π w sposób opisany poniżej (szczegółowa dyskusja na temat polityki pracy wózka została przeprowadzona w sekcji 6.1.3). Wózek v wykonuje $lc = m^p + r - 2$ cykli, gdzie cykl i pracy wózka, $1 \leqslant i \leqslant lc$, przebiega następująco. Jeśli $i \leqslant r$, to wózek czeka przy maszynie 1 na zakończenie się operacji załadowczej $(1, \pi(i))$, po czym wykonuje operację transportową $(m^p + 1, \pi(i))$ w czasie $p_{m^p+1,\pi(i)} = t(1, 2)$. Jeżeli ostatnie z zadań do wykonania zostało już wprowadzone do systemu, tzn. jeśli $i > r$, wózek wykonuje przejazd pusty (przejazd bez załadunku) w kierunku maszyny 2 (w czasie $t(1, 2)$). Jeśli $1 \leqslant i - 1 \leqslant r$, to wózek oczekuje na zakończenie się operacji $(2, \pi(i - 1))$, po czym wykonuje operację transportową $(m^p + 2, \pi(i - 1))$. W przeciwnym razie wózek wykonuje przejazd pusty w kierunku maszyny 3. Proces powtarza się aż do momentu przyjazdu wózka do maszyny m^p. Wykonanie przejazdu pustego pomiędzy stacją wyładowczą i załadowczą, w czasie $t(m^p, 1)$, kończy cykl.

Dane są operacje transportowe $x' = (m^p + l, i)$, $x'' = (m^p + k, j)$, $x', x'' \in O^t$. Zgodnie z powyższym opisem, operacja x' (operacja x'') polegają na przetransportowaniu palety z detalem wykonywanym w ramach zadania i (zadania j) pomiędzy maszyną l i maszyną $l+1$ (maszyną k i maszyną $k + 1$). Jeżeli operacje x', x'' wykonywane są kolejno, zaś $l + 1 \neq k$ (tzn. jeżeli operacje x', x'' wykonywane są w dwóch kolejnych cyklach c, $c + 1$, $1 \leqslant c < lc$, pracy wózka, operacja x' jest ostatnią wykonywaną operacją transportową w cyklu c, zaś x'' pierwszą operacją cyklu $c + 1$), to wózek wykonuje przejazd pusty pomiędzy maszynami $l + 1$, k w czasie

$t(l+1,k)$. Przejazd pusty może być utożsamiany z przezbrojeniem maszyny transportowej, wykonywanym pomiędzy operacjami x', x'' w czasie

$$s(x',x'') = t(l+1,k). \tag{6.2}$$

Na system narzuca się dodatkowe ograniczenia polegające na tym, że:

1. w każdej chwili czasu może być wykonywana co najwyżej jedna operacja każdego z zadań,
2. każda maszyna (produkcyjna i transportowa) w każdej chwili może wykonywać co najwyżej jedną operację,
3. rozpoczęta operacja nie może być przerwana.

Uszeregowanie definiuje się jako zestaw czasów rozpoczęcia wykonywania $S_{l,j}$, $l \in M$, $j \in J$, poszczególnych operacji. Uszeregowanie jest dopuszczalne, gdy spełnia wszystkie ograniczenia technologiczne narzucone na system, czego wyrazem jest prawdziwość poniższych nierówności

$$S_{1,\pi(1)} \geqslant 0, \tag{6.3}$$

$$C_{l,\pi(j)} \leqslant S_{m^p+l,\pi(j)} \wedge C_{m^p+l,\pi(i)} \leqslant S_{l+1,\pi(i)}, \qquad 1 \leqslant l < m^p, i \in J, \tag{6.4}$$

$$C_{l,\pi(j)} \leqslant S_{l,\pi(j+1)}, \qquad l \in M^p, 1 \leqslant j < r, \tag{6.5}$$

$$C_{l,\pi(j)} \leqslant S_{l+1,\pi(j-1)}, \qquad m^p < l < m, 1 < j \leqslant r, \tag{6.6}$$

$$C_{a(j),\pi(b(j))} + s((a(j),\pi(b(j))),(1,\pi(j))) \leqslant S_{1,\pi(j)}, \qquad 1 < j \leqslant r, \tag{6.7}$$

$$C_{c(l),\pi(d(l))} + s((c(l),\pi(d(l))),(l,\pi(r))) \leqslant S_{l,\pi(r)}, \qquad 1 < l < m^p, \tag{6.8}$$

gdzie $C_{l,j} = S_{l,j} + p_{l,j}$ jest momentem zakończenia wykonywania operacji $(l,j) \in O$ oraz

$$a(j) = \min\{m, m^p + j - 1\}, \tag{6.9}$$

$$b(j) = \max\{1, j - m^p + 1\}, \tag{6.10}$$

$$c(l) = \min\{m, m^p + r + l - 2\}, \tag{6.11}$$

$$d(l) = \max\{1, r - m^p + l\}. \tag{6.12}$$

Nierówność (6.3) nie wymaga komentarza. Nierówności (6.4), (6.5) wynikają odpowiednio z relacji kolejnościowych pomiędzy operacjami w zadaniach i operacjami na maszynach. Nierówność (6.6) jest konsekwencją arbitralnie przyjętej kolejności wykonania operacji transportowych przez wózek, zaś nierówności (6.7), (6.8) są konsekwencją wykonywanych przejazdów pustych. W tym miejscu warto zauważyć, że dopuszczalność uszeregowania nie zależy od permutacji zbioru zadań, tzn. dla każdej permutacji $\pi \in \Pi$ możliwa jest konstrukcja uszeregowania dopuszczalnego. Ostatecznie, problem polega na odnalezieniu takiego uszeregowania dopuszczalnego (takiej permutacji $\pi \in \Pi$), by moment zakończenia wykonywania procesu technologicznego, równy $C_{m^p,\pi(r)}$, przyjął wartość minimalną. Problem jest silnie NP-trudny.

6.1.1 Model permutacyjno-grafowy

Poniżej wprowadza się model permutacyjno-grafowy problemu jako „wygodne" narzędzie umożliwiające zarówno szybkie wyznaczenie uszeregowania reprezentowanego przez daną permutację, wyznaczenie wartości funkcji celu, jak i umożliwiające analizę własności problemu. Niech $G(\pi) = (O, E)$ będzie skierowanym grafem reprezentującym dowolną permutację $\pi \in \Pi$, ze zbiorem wierzchołków O i zbiorem łuków $E = E^T \cup E^K \cup E^V \cup E^S$, gdzie

$$E^T = \bigcup_{j \in J} \bigcup_{l=1}^{m^p-1} \{((l,j),(m^p+l,j)),((m^p+l,j),(l+1,j))\}, \tag{6.13}$$

$$E^K = \bigcup_{l \in M^p} \bigcup_{j=1}^{r-1} \{((l,j),(l,j+1))\}, \tag{6.14}$$

$$E^V = \bigcup_{j=2}^{r} \bigcup_{l=1}^{m^p-2} \{((m^p+l,j),(m^p+l+1,j-1))\}. \tag{6.15}$$

Zbiór E^S stanowi sumę zbiorów $E^S = E^{S1} \cup E^{S2}$, gdzie

$$E^{S1} = \bigcup_{j=2}^{r} \{((a(j),b(j)),(m^p+1,j))\}, \tag{6.16}$$

$$E^{S2} = \bigcup_{l=2}^{m^p-1} \{(c(l),d(l)),(m^p+l,r)\}, \tag{6.17}$$

zaś wielkości $a(j)$, $b(j)$, $c(l)$, $d(l)$ dane są odpowiednimi równaniami (6.9)-(6.12). Każdy wierzchołek $(l,j) \in O$ reprezentuje operację $(l,\pi(j))$ i przyjmuje obciążenie $p_{l,\pi(j)}$. Łuki ze zbiorów E^T, E^K, E^V mają zerowe obciążenie. Reprezentują odpowiednio kolejność wykonania operacji w zadaniu, kolejność wykonania operacji produkcyjnych przez maszyny produkcyjne i kolejność wykonania operacji transportowych przez wózek AGV. Każdy łuk $(i,j) \in E^S$ ma obciążenie $s(i,j)$. Łuki te reprezentują przejazdy wózka bez załadunku. Definicja zbioru łuków E gwarantuje acykliczność grafu dla każdej permutacji $\pi \in \Pi$. Niech $r^\pi_{l,j}$, $q^\pi_{l,j}$ oznacza najdłuższą ścieżkę odpowiednio dochodzącą i wychodzącą z wierzchołka $(l,j) \in O$ bez jego obciążenia $p_{l,\pi(j)}$ w grafie $G(\pi)$. Nie trudno zauważyć, że wartość $r^\pi_{l,j}$ jest równa najwcześniejszemu możliwemu momentowi rozpoczęcia $S_{l,\pi(j)}$ operacji $(l,\pi(j))$. Zatem, rozważany tutaj problem sprowadza się do odnalezienia takiej permutacji $\pi \in \Pi$, że ścieżka krytyczna w grafie $G(\pi)$, równa $C_{\max}(\pi) = r^\pi_{m^p,r} + p_{m^p,\pi(r)}$, przyjmuje wartość minimalną.

6.1.2 Przykład ilustracyjny

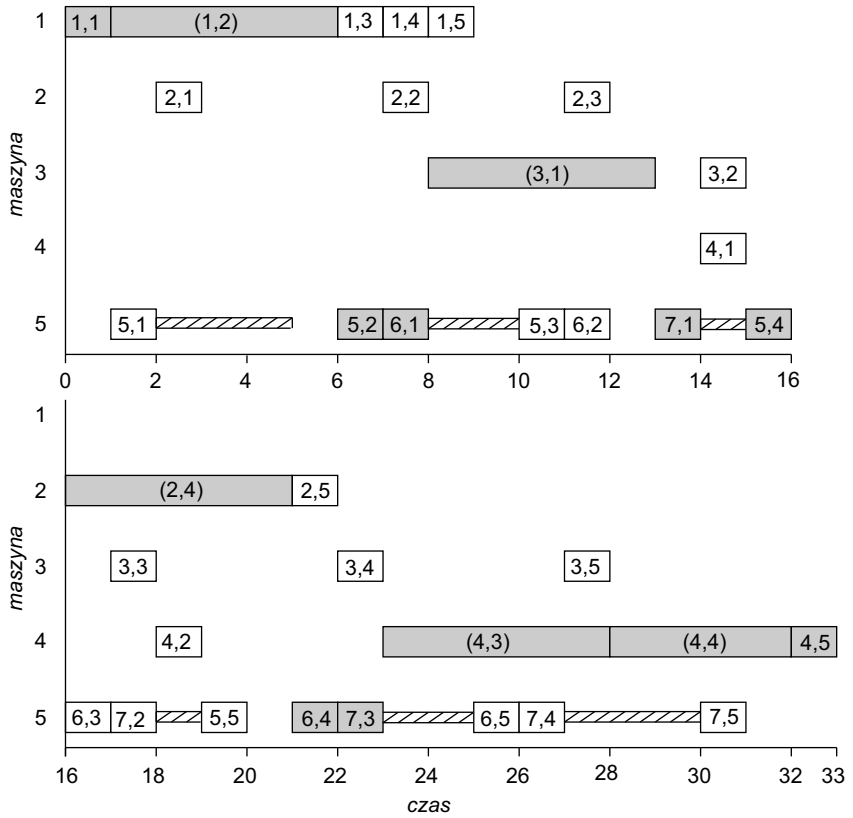
Poniżej znajduje się szczegółowa analiza przykładowej instancji rozpatrywanego problemu.

Przykład 6.1 Dane są $m^p = 4$ ($M^p = \{1, 2, 3, 4\}$) maszyny produkcyjne zorganizowane w układ typu pętla oraz $m^t = 1$ wózek AGV ($M^t = \{5\}$). Zakłada się jednostkowy czas przejazdu wózka $v \in M^t$ pomiędzy następującymi po sobie maszynami produkcyjnymi, tj. przyjmuje się $t(l, l+1) = 1$, $1 \leqslant l < m^p$ oraz $t(m^p, 1) = 1$. W systemie należy wykonać $r = 5$ zadań ($J = \{1, 2, \ldots, 5\}$). Każde zadanie $j \in J$ składa się z naprzemiennej sekwencji $m^p = 4$ operacji produkcyjnych i $m^p - 1$ operacji transportowych, które należy wykonać w kolejności $(1, j), (5, j), (2, j), (6, j), (3, j), (7, j), (4, j)$. Czasy wykonania wszystkich $n^p = 20$ operacji produkcyjnych zostały podane w tabeli 6.1. Czasy wykonania wszystkich $n^t = 15$ operacji transportowych wynikają z czasów przejazdu wózka v pomiędzy maszynami produkcyjnymi i wynoszą $p_{l,j} = 1$, $(l, j) \in O^t$.

Niech $\pi = (1, 2, \ldots, 5)$ będzie permutacją zbioru zadań J. Uszeregowanie zgodne z permutacją π przedstawione jest na rysunku 6.1. Moment zakończenia wykonywania procesu technologicznego wynosi $C_{m^p,\pi(r)} = C_{4,5} = 33$. Operacje determinujące wartość funkcji kryterialnej zostały wyróżnione. Graf $G(\pi)$ został przedstawiony na rysunku 6.2. Numerację wierzchołków podano z lewej, obciążenia w wierzchołkach. Łuki bez podanego obciążenia mają wagę zero. Zbiory E^T, E^K określają odpowiednio kolejność wykonania operacji w zadaniach, kolejność wykonania operacji przez maszyny produkcyjne i nie wymagają komentarza. Łuki należące do tych zbiorów oznaczone są linią ciągłą. Zbiory łuków E^V, E^S określają kolejność wykonywania operacji transportowych przez wózek. Łuki należące do tych zbiorów oznaczone są za pomocą linii przerywanej. Łuki ze zbioru E^V określają kolejność wykonania operacji w poszczególnych cyklach pracy wózka. Po każdej sekwencji łuków ze zbioru E^V znajduje się jeden obciążony łuk ze zbioru E^S, reprezentujący czasy przejazdów bez załadunku. W zbiorach E^V, E^S znajduje się odpowiednio 8 i 6 elementów.

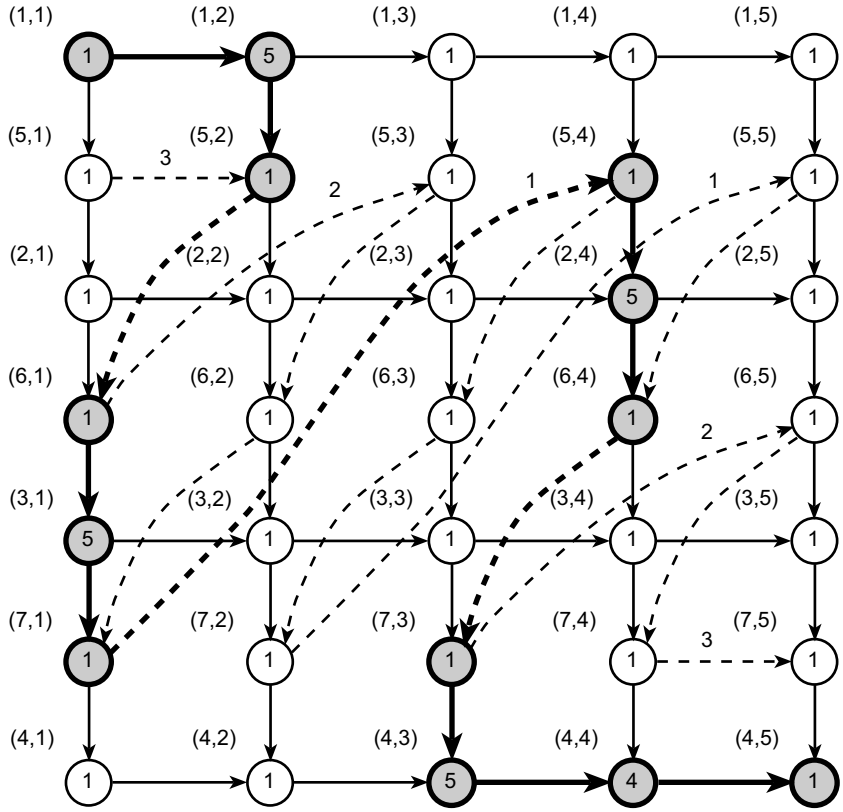
Tabela 6.1: Czasy wykonania operacji produkcyjnych

j	1	2	3	4	5
$p_{1,j}$	1	5	1	1	1
$p_{2,j}$	1	1	1	5	1
$p_{3,j}$	5	1	1	1	1
$p_{4,j}$	1	5	1	1	1
$p_{5,j}$	1	1	5	4	1



Rysunek 6.1: Wykres Gantta uszeregowania zgodnego z permutacją π z przykładu 6.1

Wózek wykonuje $lc = 4 + 5 - 2 = 7$ cykli. W pierwszym cyklu pracy, wózek wykonuje tylko jedną operację transportową $(5, 1)$. Polega na przetransportowaniu palety z detalem wykonywanym w ramach zadania 1 pomiędzy maszynami produkcyjnymi 1 i 2. Następnie, wózek wykonuje przejazd pusty pomiędzy maszynami 2 i 1 w czasie $s((5, 1), (5, 2)) = t(2, 1) = 3$ i rozpoczyna wykonanie operacji transportowej $(5, 2)$. Przejazd pusty jest reprezentowany w grafie $G(\pi)$ przez obciążony łuk $((5, 1), (5, 2)) \in E^S$. W drugim cyklu pracy wózka, po wykonaniu operacji $(5, 2)$ wykonywana jest operacja $(6, 1)$. Kolejność tą określa łuk $((5, 2), (6, 1)) \in E^V$. Po zakończeniu operacji $(6, 1)$ i przed rozpoczęciem operacji $(5, 3)$ następuje kolejny, kończący drugi cykl pracy wózka, przejazd bez załadunku. Przejazd ten reprezentowany jest przez łuk $((6, 1), (5, 3)) \in E^S$ o obciążeniu $s((6, 1), (5, 3)) = t(3, 1) = 2$. W trzecim cyklu pracy wózka wykonywane są kolejno operacje $(5, 3)$, $(6, 2)$, $(7, 1)$. Kolejność ta odnotowana jest przez



Rysunek 6.2: Graf $G(\pi)$ dla permutacji π z przykładu 6.1

łuki $((5,3),(6,2)),((6,2),(7,1)) \in E^V$. Po zakończeniu trzeciego cyklu, wózek wykonuje kolejne cykle aż do momentu wykonania ostatniej operacji transportowej $(7,5)$ i powrotu na parking dla wózków AGV.

Graf $G(\pi)$ skonstruowany jest w ten sposób, że każdy z wierzchołków posiada co najwyżej dwa bezpośrednie poprzedniki i następniki, dzięki czemu stosunkowo łatwo można wyznaczyć poszczególne wartości $r^{\pi}_{l,j}$, $q^{\pi}_{l,j}$ oraz długość ścieżki krytycznej. Długość ścieżki krytycznej w grafie (równa wartości funkcji kryterialnej) wynosi $C_{\max}(\pi) = 33$. Na rysunku 6.2 łuki i wierzchołki należące do ścieżki krytycznej zostały wyróżnione.

Dana jest permutacja $\delta = (1,2,5,4,3)$, powstała z permutacji π po zamianie miejscami elementów $\pi(3)$, $\pi(5)$. Aby przejść do grafu $G(\delta)$ z grafu $G(\pi)$ wystarczy jedynie zmienić obciążenie części wierzchołków. Obciążenie wierzchołków $(l,3)$, $(l,5)$, $1 \leqslant l \leqslant 7$, w grafie $G(\delta)$ wyniesie odpowiednio $p_{l,\pi(5)}$ i $p_{l,\pi(3)}$. Przy poszczególnych modyfikacjach grafu nie ma konieczności dokonywania zmian w zbiorze łuków E.

6.1.3 Polityka pracy wózka

Pomimo stosunkowo prostych założeń dotyczących pracy systemu, permutacja $\pi \in \Pi$ nie określa jednoznacznie kolejności wykonania operacji transportowych przez wózek. Przyjęta w sekcji 6.1 arbitralna polityka kolejności realizacji operacji transportowych jest wynikiem sposobu programowania komputera pokładowego wózka AGV i sposobem koordynacji realizacji harmonogramu przez system produkcyjny. W przyjętym modelu, program w komputerze pokładowym wózka jest bardzo prosty:

1. Jedź i zatrzymaj się przy maszynie produkcyjnej.
2. Jeśli wózek transportuje paletę i maszyna produkcyjna nie jest stacją załadowczą, to czekaj na opróżnienie się bufora wejściowego maszyny i wyładuj paletę do bufora.
3. Jeśli maszyna produkcyjna nie jest stacją wyładowczą i bufor wyjściowy jest pełny, to zabierz paletę z bufora wyjściowego maszyny.
4. Idź do kroku 1.

Jeśli maszyny również rozpoczynają serię automatycznych czynności po zapełnieniu się ich bufora wejściowego, to realizacja zaplanowanego harmonogramu rozpoczyna się automatycznie po dostarczeniu zadań do stacji załadowczej w ustalonej kolejności. W takim systemie ingerencja centralnego komputera, koordynującego na bieżąco pracę maszyn i wózka jest w zasadzie nie potrzebna.

Alternatywnym rozwiązaniem jest użycie kolejności wykonania operacji transportowych jako dodatkowej zmiennej sterującej. Wymaga zastosowania komputera na bieżąco komunikującego się i „informującego" wózek o dalszych czynnościach, które należy wykonać. Takie rozwiązanie jest droższe, bardziej skomplikowane i raczej charakterystyczne dla systemów modelowanych za pomocą gniazdowych problemów szeregowania. Różnice w efektach zastosowania odmiennych polityk ruchu wózka można prześledzić na krótkim przykładzie.

Przykład 6.2 Dany jest zbiór $r = 2$ zadań oraz system zawierający $m^p = 3$ stanowiska (załadowcze, produkcyjne, wyładowcze) oraz jeden wózek AGV. Czasy wykonywania zadania pierwszego na stanowiskach $1, 2, 3$ wynoszą odpowiednio $(1, 1, 4)$. Czasy wykonywania zadania drugiego na stanowiskach $1, 2, 3$ wynoszą odpowiednio $(1, 1, 1)$. Zakłada się jednostkowe czasy przejazdu wózka pomiędzy każdą parą sąsiednich maszyn. Przyjmując kolejność wprowadzania zadań do systemu $\pi = (1, 2)$ można rozważyć dwie skrajnie odmienne strategie ruchu wózka AGV:

1. wózek realizuje wszystkie operacje transportowe zadania pierwszego czekając każdorazowo przy stanowisku na zakończenie obróbki detalu wykonywanego w ramach zadania, po czym obsługuje w identyczny sposób transport zadania drugiego,
2. wózek realizuje operacje transportowe w sposób przeplatany, raz zadania pierwszego, raz drugiego, zgodnie z koncepcją opisywaną w sekcji 6.1.

Termin zakończenia wykonywania obu zadań według intuicyjnie irracjonalnej strategii 1 wynosi 9, podczas gdy według strategii 2 wynosi 11. Można podać inny przykład, w którym relacja ta jest odwrotna. Stąd wniosek, że kolejność obsługi operacji transportowych przez wózek zależy od danych przykładu i może być optymalizowana podobnie jak sekwencja wejściowa zadań do systemu. Jak wspomniano, pociąga to za sobą zarówno wzrost skomplikowania modelu matematycznego, algorytmu optymalizacji i realizacji sterowania wózkiem. Dlatego strategia 1 pracy wózka nie będzie rozważana.

6.2 Eliminacyjne własności bloków zadań

Poniżej znajduje się szczegółowa analiza własności ścieżki krytycznej i bloków zadań w grafie $G(\pi)$, $\pi \in \Pi$. W celu dokonania analizy wykorzystano podejście zbliżone do podejścia prezentowanego w pracy [65] dla permutacyjnego problemu przepływowego z ograniczoną liczbą palet oraz problemu z ograniczoną pojemnością buforów. Ścieżkę krytyczną w grafie $G(\pi)$ można opisać za pomocą ciągu wierzchołków $\omega = (\omega_1, \omega_2, \ldots, \omega_{ls})$, $\omega_t = (l[t], i[t]) \in O$, $1 \leqslant t \leqslant ls$, gdzie ls jest liczbą wierzchołków ścieżki. Ścieżka rozpoczyna się w wierzchołku $\omega_1 = (l[1], i[1]) = (1, 1)$ i kończy się w wierzchołku $\omega_{ls} = (l[ls], i[ls]) = (m^p, r)$. Ścieżka ta przechodzi naprzemiennie przez pewną liczbę łuków ze zbioru E^K (nazywanych dalej łukami poziomymi), po czym przechodzi przez przynajmniej jeden łuk ze zbioru E^T (łuk pionowy), by później przejść znów przez pewną liczbę łuków poziomych lub łuków ze zbioru $E^V \cup E^S$ (nazywanych dalej łukami skośnymi). Na ścieżce krytycznej można zatem wydzielić segmenty poziome, pionowe i skośne. Przez segment poziomy będzie rozumiany podciąg wierzchołków $\omega_c, \omega_{c+1}, \ldots, \omega_d$, reprezentowany przez parę (c, d), $c < d$, taki, że

1. każda kolejna para wierzchołków podciągu jest połączona łukiem poziomym,
2. liczba c oraz d są odpowiednio najmniejszą i największą liczbą, dla których zachodzi własność pierwsza.

W sposób analogiczny definiuje się segment pionowy oraz skośny. Z definicji łuków grafu wynika, że segmenty poziome zbudowane są wyłącznie z operacji produkcyjnych, segmenty skośne wyłącznie z operacji transportowych, zaś segmenty pionowe zawsze tworzone są przez operacje tego samego zadania.

Niech lista par $(c_1, d_1), \ldots, (c_{lb}, d_{lb})$ określa kolejne segmenty poziome i skośne na ścieżce ω, gdzie lb jest ich liczbą. Dla powyższych ciągów spełnione są następujące nierówności: $1 \leqslant c_1 < d_1 < c_2 < d_2 < \cdots < c_{lb} < d_{lb} \leqslant ls$. Spełnione są też równości $i[c_1] = 1$, $i[d_{lb}] = r$ oraz $i[d_{j-1}] = i[c_j]$, $1 < j \leqslant lb$. Jednakże, w przeciwieństwie do nierówności opisywanych w pracy [65], nierówność $i[c_j] < i[d_j]$, $1 \leqslant j \leqslant lb$, nie zawsze jest spełniona. Oznacza to, że to samo zadanie może współtworzyć więcej niż jeden segment poziomy lub skośny. Powyższe spostrzeżenie wymaga dość specyficznej definicji bloków zadań. W celu zdefiniowania bloków zadań i wykazania ich eliminacyjnych własności konieczne jest wprowadzenie pewnych dodatkowych oznaczeń. Niech zbiór $Z' = \{z_0, z_1, \ldots, z_{lb}\}$ będzie zbiorem takim, że $z_0 = i[c_1] = 1$, $z_h = i[d_h] = i[c_{h+1}]$, $1 \leqslant h < lb$, $z_{lb} = i[d_{lb}] = r$. Ze zbioru Z' należy usunąć elementy powtarzające się, tj. tworzy się zbiór $Z = Z' \setminus \{z_j : \exists_{0 \leqslant k < j} z_k = z_j, 0 < j \leqslant lb\}$, którego moc wynosi $lz+1 = |Z|$ elementów. Niech ciąg $u = (u_0, u_1, \ldots, u_{lz})$ stanowi permutację elementów zbioru Z taką, że $u_{h-1} < u_h$, $1 \leqslant h \leqslant lz$. Sekwencja zadań $B_h = (\pi(u_{h-1}), \pi(u_{h-1}+1), \ldots, \pi(u_h))$ będzie nazywana h-tym blokiem zadań w grafie $G(\pi)$. Zbiorem pozycji niewrażliwych bloku B_h, $1 \leqslant h \leqslant lz$, będzie nazywany zbiór

$$U_h = \begin{cases} \{u_{h-1}+1, \ldots, u_h - 1\}, & u_{h-1}+1 \leqslant u_h - 1, \\ \emptyset, & \text{w przeciwnym przypadku.} \end{cases} \tag{6.18}$$

Zbiór U_h jest jednocześnie zbiorem wewnętrznych pozycji bloku B_h. Poniższe twierdzenie pokazuje, że zmiana kolejności zadań należących do bloku B_h, $1 \leqslant h \leqslant lz$, znajdujących się na pozycjach ze zbioru U_h, nie wiąże się ze skróceniem długości $C_{\max}(\pi)$ ścieżki krytycznej ω w grafie $G(\pi)$.

Twierdzenie 6.1 *Niech* $\omega = (\omega_1, \omega_2, \ldots, \omega_{ls})$, $w_t = (l[t], i[t]) \in O$, $1 \leqslant t \leqslant ls$, *będzie ścieżką krytyczną w grafie* $G(\pi)$, $\pi \in \Pi$, *która generuje bloki* B_h, $1 \leqslant h \leqslant lz$. *Dla każdej permutacji* $\beta \in W(\pi)$ *zachodzi nierówność* $C_{\max}(\beta) \geqslant C_{\max}(\pi)$, *gdzie*

$$\begin{aligned} W(\pi) \;=\; & \{\beta \in \Pi : (\beta(j) = \pi(j), j \in J \setminus \textstyle\bigcup_{h=1}^{lz} U_h) \wedge \\ & (\textstyle\bigcup_{j \in U_h} \{\beta(j)\} = \bigcup_{j \in U_h} \{\pi(j)\}, 1 \leqslant h \leqslant lz)\}. \end{aligned} \tag{6.19}$$

Dowód. Niech

$$H_l = \{1 \leqslant j \leqslant r : \exists_{1 \leqslant t \leqslant ls}\, \omega_t = (l, j)\}, \qquad 1 \leqslant l \leqslant m. \tag{6.20}$$

Wtedy zbiór $\bigcup_{l\in N}\bigcup_{j\in H_l}(l,j)$ jest zbiorem wszystkich wierzchołków, przez które przechodzi ścieżka krytyczna w grafie $G(\pi)$. Niech

$$P_l = \bigcup_{h=1}^{lz} \{j \in U_h : \exists_{1\leqslant t\leqslant ls}\ \omega_t = (l,j)\}, \qquad 1 \leqslant l \leqslant m. \tag{6.21}$$

Zbiór $\bigcup_{l\in N}\bigcup_{j\in P_l}(l,j)$ jest wtedy zbiorem wszystkich wierzchołków znajdujących się na pozycjach niewrażliwych w grafie $G(\pi)$, przez które przechodzi ścieżka krytyczna. Przez

$$\Delta = \sum_{j=2}^{ls} s((l[j-1],i[j-1]),(l[j],i[j])) \tag{6.22}$$

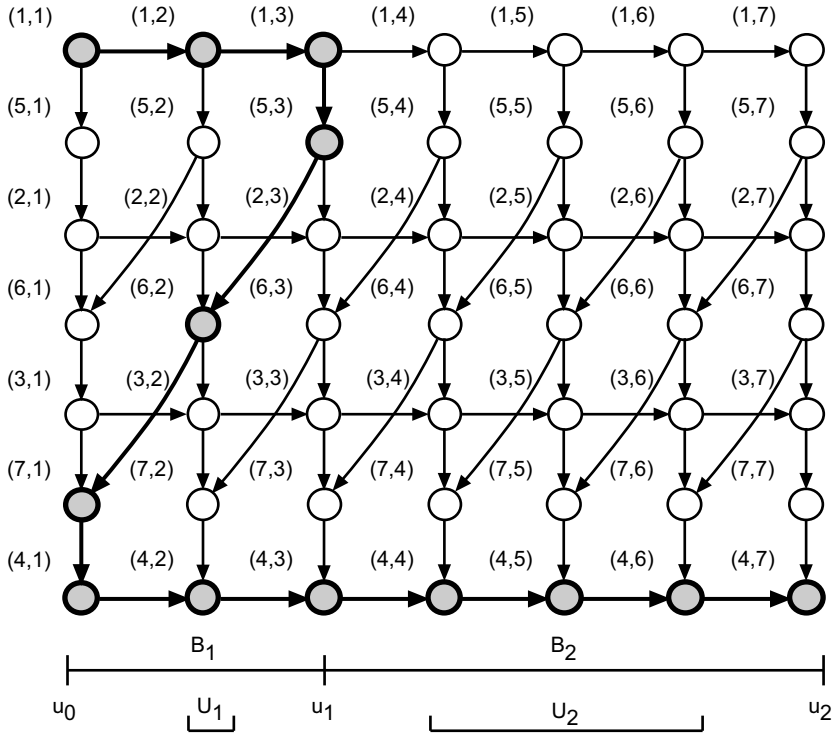
będzie oznaczana wartość wynikającą z obciążenia łuków pomiędzy każdą parą wierzchołków ω_{j-1}, ω_j, $1 < j \leqslant ls$, należących do ścieżki krytycznej ω. Struktura grafu (tzn. zbiór wierzchołków, zbiór i obciążenie łuków) jest niezależna od reprezentowanej permutacji. Oznacza to, że ścieżka krytyczna ω w grafie $G(\pi)$ jest pewną ścieżką, niekoniecznie krytyczną, w grafie $G(\beta)$, $\beta \in W(\pi)$, której długość będzie oznaczana przez d^β. Wtedy zachodzi

$$\begin{aligned} C_{\max}(\pi) &= d^\pi = \textstyle\sum_{l\in N}\sum_{j\in H_l} p_{l,\pi(j)} + \Delta \\ &= \textstyle\sum_{l\in N}(\sum_{j\in P_l} p_{l,\pi(j)} + \sum_{j\in H_l\setminus P_l} p_{l,\pi(j)}) + \Delta \\ &= \textstyle\sum_{l\in N}(\sum_{j\in P_l} p_{l,\beta(j)} + \sum_{j\in H_l\setminus P_l} p_{l,\beta(j)}) + \Delta \\ &= \textstyle\sum_{l\in N}\sum_{j\in H_l} p_{l,\beta(j)} + \Delta \\ &= d^\beta \leqslant C_{\max}(\beta). \end{aligned} \tag{6.23}$$

Pierwsza równość jest oczywista, druga wynika z definicji zbioru H_l i wielkości Δ, danych odpowiednimi równaniami (6.20), (6.22). Trzecia równość wynika z definicji zbioru pozycji niewrażliwych U_h oraz zbioru P_l, danych odpowiednimi równaniami (6.18), (6.21). Czwarta równość wynika z definicji zbioru $W(\pi)$, danej równaniem (6.19). Pozostałe równości i nierówność są oczywiste ■

Prawdziwość powyższego twierdzenia oznacza, że w algorytmach rozwiązywania badanego problemu można bezpośrednio stosować niektóre zbiory ruchów i sąsiedztwa znane dla klasycznego permutacyjnego problemu przepływowego. Praktyczne znaczenie twierdzenia 6.1 zostało też zilustrowane przy pomocy poniższego przykładu.

Przykład 6.3 Dany jest zbiór $m^p + 1 = 5$ maszyn $M = \{1,2,\ldots,5\}$ i zbiór $r = 7$ zadań $J = \{1,2,\ldots,7\}$. Dana jest pewna permutacja zbioru zadań $\pi = (1,5,2,6,3,7,4)$ oraz graf $G(\pi) = (O,E)$, przedstawiony na



Rysunek 6.3: Graf $G(\pi)$ dla permutacji π z przykładu 6.3

rysunku 6.3. Numerację wierzchołków grafu podano z lewej, wierzchołki należące do ścieżki krytycznej zostały pogrubione. Ze względu na przejrzystość, łuki ze zbioru E^S nie zostały naniesione. Graf $G(\pi)$ generuje ścieżkę krytyczną $\omega = ((1,1),\ (1,2),\ (1,3),\ (5,3),\ (6,2),\ (7,1),\ (4,1),\ldots,\ (4,7))$, przechodzącą przez $ls = 13$ wierzchołków. Na ścieżce ω można wyszczególnić dwa segmenty poziome, przechodzące przez wierzchołki $(\omega_1,\ \omega_2,\ \omega_3) = ((1,1),\ (1,2),\ (1,3))$, $(\omega_7,\ldots,\ \omega_{13}) = ((4,1),\ldots,\ (4,7))$ i jeden segment skośny, przechodzący przez wierzchołki $(\omega_4,\ \omega_5,\ \omega_6) = ((5,3),\ (6,2),\ (7,1))$. Zatem, dane są $lb = 3$ segmenty określone przez listę par $(c_1,d_1),\ldots,(c_3,d_3) = (1,3),(4,6),(7,13)$, $i[c_1] = i[1] = 1$, $i[d_1] = i[3] = i[c_2] = i[4] = 3$, $i[d_2] = i[6] = i[c_3] = i[7] = 1$, $i[d_3] = i[13] = 7$. Zgodnie z definicją, zbiory Z' i Z zawierają elementy $Z' = \{1,3,1,7\}$ i $Z = \{1,3,7\}$. W konsekwencji, ciąg $u = (1,3,7)$ generuje $lz = |Z| - 1 = 2$ bloki zadań, $B_1 = (\pi(1),\ldots,\pi(3)) = (1,5,2)$, $B_2 = (\pi(3),\ldots,\pi(7)) = (2,6,3,7,4)$, oraz dwa niepuste zbiory pozycji niewrażliwych $U_1 = \{2\}$, $U_2 = \{4,5,6\}$. Można określić również zbiory H_l, P_l, $1 \leqslant l \leqslant m$, $m = 2m^p - 1 = 7$, gdzie $H_1 = \{1,2,3\}$,

$H_2 = H_3 = \emptyset$, $H_4 = \{1, \ldots, 7\}$, $H_5 = \{3\}$, $H_6 = \{2\}$, $H_7 = \{1\}$ oraz $P_1 = \{2\}$, $P_4 = \{2, 4, 5, 6\}$, $P_6 = \{2\}$, $P_2 = P_3 = P_5 = P_7 = \emptyset$.

Zgodnie z równaniem (6.19), permutacja $\beta = (1, 5, 2, 3, 7, 6, 4)$ należy do zbioru $W(\pi)$, ponieważ różni się od permutacji π tylko kolejnością elementów na pozycjach 4, 5, 6, które należą do zbioru U_2. Na podstawie analizy zbiorów $P_1, \ldots, P_7$ można wywnioskować, że w grafie $G(\pi)$ na pozycjach 4, 5, 6 znajdują się tylko trzy wierzchołki $(4, 4), \ldots, (4, 6)$ należące do ścieżki ω (ponieważ $\{4, 5, 6\} \subset P_4$ oraz $\{4, 5, 6\} \nsubseteq P_l$ dla $1 \leqslant l \leqslant m$, $l \neq 4$), których sumaryczne obciążenie wynosi

$$\sum_{j \in P_4} p_{4,\beta(j)} = \sum_{j \in P_4} p_{4,\pi(j)}, \tag{6.24}$$

więc $d^\beta = d^\pi$.

6.3 Algorytmy rozwiązywania problemu

Dzięki wprowadzonym definicjom, rozwiązanie badanego w tym rozdziale problemu możliwe jest przy użyciu algorytmów stosowanych do rozwiązywania klasycznego permutacyjnego problemu przepływowego. Za najlepszy algorytm konstrukcyjny dla klasycznego permutacyjnego problemu przepływowego uważa się algorytm NEH, którego nazwa pochodzi od pierwszych liter nazwisk autorów. Algorytm ten został zaprezentowany po raz pierwszy w pracy [63]. Przyjmując za definicję funkcji kryterialnej długość $C_{\max}(\pi)$ ścieżki krytycznej w grafie $G(\pi)$, $\pi \in \Pi$, problem z transportem można rozwiązać w sposób przybliżony stosując algorytm konstrukcyjny NEH. Dla rozróżnienia, algorytm NEH dla problemu z transportem będzie nazywany algorytmem NEH_{AGV}.

Działanie algorytmu NEH (algorytmu NEH_{AGV}) na przykładzie omawianego problemu z transportem można w skrócie opisać następująco. W pierwszym kroku algorytmu tworzona jest permutacja $\pi = \emptyset$ oraz permutacja $\psi = (\psi(1), \psi(2), \ldots, \psi(r))$ zbioru zadań, na której zadania są uszeregowane zgodnie z nierosnącymi sumarycznymi czasami wykonania operacji w zadaniu, tj. dla każdego $1 \leqslant i < j \leqslant r$ spełniony jest warunek $\omega(\psi(i)) \geqslant \omega(\psi(j))$, gdzie

$$\omega(j) = \sum_{l=1}^{m} p_{l,j}, \qquad j \in J. \tag{6.25}$$

Następnie, algorytm wykonuje r iteracji. W k-tej iteracji z listy ψ wybierana jest operacja $i = \psi(k)$ i jest próbnie wstawiana na wszystkie $k + 1$

pozycji w permutacji π. Bardziej precyzyjnie, tworzone są permutacje próbne $\beta^1, \beta^2, \ldots, \beta^{k+1}$, gdzie

$$\beta^z = (\pi(1), \ldots, \pi(z-1), i, \pi(z), \ldots, \pi(k)). \tag{6.26}$$

Dla każdej powstałej w ten sposób permutacji próbnej β^z wyznaczana jest wartość funkcji celu $C_{\max}(\beta^z)$, $1 \leqslant z \leqslant k+1$. W ostatnim kroku iteracji przyjmuje się $\pi := \delta$, gdzie δ jest najlepszą, spośród permutacji próbnych, utworzonych w danej iteracji, tzn. $\delta \in \{\alpha : C_{\max}(\alpha) = \min_{1 \leqslant z \leqslant k+1} C_{\max}(\beta^z)\}$.

Wyznaczenie wartości $C_{\max}(\beta)$ dla badanego problemu przepływowego wymaga $O(n)$ czasu. Najbardziej czasochłonnym elementem iteracji k, $1 \leqslant k \leqslant r$, algorytmu NEH_{AGV} jest wyznaczenie poszczególnych wielkości $C_{\max}(\beta^z)$, $1 \leqslant z \leqslant k+1$ i wymaga $O(r \cdot n)$ czasu. Złożoność obliczeniowa algorytmu wynosi zatem $O(r^2 \cdot n)$. W przypadku problemu klasycznego znane są metody redukcji złożoności obliczeniowej algorytmu NEH. W przypadku badanego problemu bezpośrednie zastosowanie metod klasycznych nie jest możliwe, głównie za sprawą łuków skośnych, występujących w grafie $G(\pi)$ i nie występujących w grafie „klasycznym". Co gorsza, inne własności pozwalające na jakąkolwiek redukcję czasu obliczeń algorytmu NEH_{AGV} dotychczas również nie zostały wykryte.

Jednym z najlepszych algorytmów popraw dla klasycznego permutacyjnego problemu przepływowego jest algorytm $TSAB$, zaprezentowany w pracy [71]. Przyjmując za definicję funkcji kryterialnej długość $C_{\max}(\pi)$ ścieżki krytycznej w grafie $G(\pi)$, $\pi \in \Pi$, po przyjęciu definicji ciągu $u = (u_0, u_1, \ldots, u_{lz})$ i bloków zadań B_h, $1 \leqslant h \leqslant lz$, takich jak w sekcji 6.2, można przyjąć za definicję ruchu, zbioru ruchów, struktury sąsiedztwa i kryterium aspiracji odpowiednie definicje prezentowane w pracy [71]. W konsekwencji, do przybliżonego rozwiązania problemu z transportem można bezpośrednio zastosować algorytm $TSAB$, który, dla rozróżnienia, będzie nazywany algorytmem $TSAB_{AGV}$.

Krótki opis działania algorytmu $TSAB$ (algorytmu $TSAB_{AGV}$) na przykładzie badanego problemu zamieszczony jest poniżej. W opisie wykorzystano oznaczenia i definicje z cytowanej pracy. Pierwszymi elementami wymagającymi zdefiniowania są ruch i zbiór ruchów. Niech $v = (a, b)$ będzie ruchem typu wstaw takim, że po zastosowaniu tego ruchu do permutacji $\pi \in \Pi$, powstaje permutacja sąsiednia $\pi^v = (\pi(1), \ldots, \pi(a-1), \pi(a+1), \ldots, \pi(b), \pi(a), \pi(b+1), \ldots, \pi(r))$, jeżeli $a < b$, oraz permutacja $\pi^v = (\pi(1), \ldots, \pi(b-1), \pi(a), \pi(b+1), \ldots, \pi(b-1), \pi(b+1), \ldots, \pi(r))$, jeżeli $a > b$. Niech $l_L(j)$ oraz $l_R(j)$ będzie odpowiednio najmniejszym i największym indeksem bloku zawierającego zadanie $\pi(j)$ dla ustalonej

permutacji π. Niech

$$\Delta_l = \begin{cases} \lfloor (u_l - u_{l-1}) \cdot \epsilon \rfloor, & l = 1, \ldots, lz, \\ 0, & l = 0 \vee l = lz + 1, \end{cases} \tag{6.27}$$

gdzie ϵ jest parametrem. Wtedy zbiór ruchów można zapisać w postaci

$$V(\pi, \epsilon) = \bigcup_{j=1}^{r-1} ZR_j(\pi, \epsilon) \cup \bigcup_{j=2}^{r} ZL_j(\pi, \epsilon), \tag{6.28}$$

gdzie

$$ZR_j(\pi, \epsilon) = \{(j, t) : u_{l_R(j)} \leqslant t \leqslant u_{l_R(j)} + \Delta_{l_R(j)+1}\}, \tag{6.29}$$

$$\begin{aligned} ZL_j(\pi, \epsilon) = \{(j, t) : u_{l_L(j)-1} - \Delta_{l_L(j)-1} \\ \leqslant t \leqslant u_{l_L(j)-1} - \omega(u_{l_L(j)} - u_{l_L(j)-1})\}, \end{aligned} \tag{6.30}$$

oraz $\omega(x) = 0$ dla $x > 1$, $\omega(x) = 1$ w przeciwnym przypadku.

Kolejnym elementem, niezbędnym w opisie algorytmu, jest lista tabu. Niech $T = (T_1, \ldots, T_{maxt})$ będzie listą tabu o stałej długości, określanej parametrem $maxt$, gdzie $T_j = (g, h)$ jest parą zadań. Niech $v = (a, b)$ będzie ruchem zastosowanym do ustalonej permutacji π. Ruch dodaje się do listy kładąc $T_j := T_{j+1}$, $j = 1, \ldots, maxt-1$, następnie $T_{maxt} := (\pi(a), \pi(a+1))$, jeżeli $a < b$ lub $T_{maxt} := (\pi(a-1), \pi(a))$, jeżeli $a > b$. Operacja dodawania ruchu v do listy będzie notowana jako $T \oplus v$. Ruch $v = (a, b)$, $a < b$, nie może być zastosowany do permutacji β (ruch jest zabroniony), jeżeli przynajmniej jedna para $(\beta(j), \beta(a))$, $j = a+1, \ldots, b$ znajduje się na liście T. Symetrycznie, ruch $v = (a, b)$, $a > b$, nie może być zastosowany do permutacji β, jeśli na liście T znajduje się przynajmniej jedna para $(\beta(a), \beta(j))$, $j = b, \ldots, a-1$.

Wartość funkcji aspiracji F zależy od aktualnej iteracji t i przybiera postać $F(C) = \min\{F_1(C), F_0(C)\}$, gdzie, dla danej wartości funkcji celu C, $F_1(C) = \min\{C_{\max}(\pi^{(i-1)}) : C_{\max}(\pi^{(i)}) = C, i = 2, \ldots, t\}$ oraz $F_0(C) = \min\{C_{\max}(\pi^{(i+1)}) : C_{\max}(\pi^{(i)}) = C, i = 1, \ldots, t-1\}$.

Kluczowym elementem strategii przeszukiwania sąsiedztwa jest tzw. zbiór reprezentantów $V'(\pi, \epsilon) \subseteq V(\pi, \epsilon)$. Dla $\epsilon = 0$ przyjmuje się, że $V'(\pi, \epsilon) = V(\pi, \epsilon)$. W przeciwnym przypadku konstrukcja zbioru przebiega następująco. Każdy zbiór $R \in \{ZR_1(\pi, \epsilon), ZL2(\pi, \epsilon), \ldots, ZR_{r-1}(\pi, \epsilon), ZL_r(\pi, \epsilon)\}$ jest przeszukiwany oddzielnie w poszukiwaniu pojedynczego ruchu reprezentatywnego $v(R)$, prowadzącego do najlepszego rozwiązania w tym zbiorze, tzn. takiego, że

$$C_{\max}(\pi^{v(R)}) = \min_{v \in R} C_{\max}(\pi^v). \tag{6.31}$$

Procedura rozpoczyna działanie z permutacją π, nie pustym zbiorem $V'(\pi,\epsilon)$ i listą tabu T. Procedura zwraca ruch v', zmodyfikowaną listę tabu T', nową permutację π' i wartość funkcji celu C'.

1. Utwórz zbiór $X = \{v \in V'(\pi,\epsilon) : v$ nie jest zabroniony$\}$ oraz zbiór $Y = \{v \in V'(\pi,\epsilon) : v$ jest zabroniony, $C_{\max}(\pi^v) < F(C_{\max}(\pi))\}$
2. Jeżeli $X \cup Y$, wybierz ruch $v' \in X \cup Y$ taki, że $C_{\max}(\pi^{v'}) = \min_{v \in X \cup Y} C_{\max}(\pi^v)$, połóż $\pi' := \pi^{v'}$, $T' := T \oplus v'$, $C' := C_{\max}(\pi')$ i STOP.
3. Połóż $T := T \oplus (0,0)$, wyznacz ponownie zbiory X, Y i idź do kroku 2.

Rysunek 6.4: Schemat procedury NSP

Algorytm rozpoczyna działanie z permutacją π i ustalonymi wartościami parametrów $maxt$, $maxiter$, $maxret$ i ϵ. Algorytm zwraca najlepsze znalezione rozwiązanie π^* oraz wartość C^*.

1. Połóż $\pi^* := \pi$, $C := C_{\max}(\pi)$, $C^* := C$, $iter := 0$, $ret := 0$, $T := \emptyset$ oraz $save := true$. Połóż $F(x) := \infty$ dla każdego x należącego do zbioru liczb naturalnych.
2. Połóż $iter := iter+1$, $ret := ret+1$. Określ zbiór $V(\pi,\epsilon)$.
3. Znajdź ruch v', permutację π', wielkość $C' := C_{\max}(\pi')$ oraz listę tabu T', stosując procedurę NSP. Połóż $F(C) := \min\{F(C), C'\}$, $F(C') := \min\{F(C'), C\}$. Jeśli $save = true$, połóż $V^* := V'(\pi,\epsilon) \setminus \{v'\}$, $T^* := T$ oraz $save := false$. Połóż $\pi := \pi'$, $T := T'$ i $C := C'$.
4. Jeśli $C < C^*$, połóż $\pi^* := \pi$, $C^* := C$, $ret := 0$, $save := true$ i idź do kroku 2.
5. Jeśli $ret < maxert$, to idź do kroku 2.
6. Jeśli $V^* \neq \emptyset$ oraz $iter < maxiter$, to połóż $\pi := \pi^*$, $V' := V^*$, $C := C^*$, $T := T^*$, $save := true$, $ret := 1$ i idź do kroku 3. W przeciwnym przypadku zwróć π^*, wartość C^* i STOP.

Rysunek 6.5: Schemat algorytmu $TSAB$

Zbiorem reprezentantów będzie nazywany zbiór

$$V'(\pi,\epsilon) = \bigcup_{j=1}^{r-1} v(ZR_j(\pi,\epsilon)) \cup \bigcup_{j=2}^{r} v(ZL_j(\pi,\epsilon)). \tag{6.32}$$

Procedura przeszukiwania sąsiedztwa (procedura NSP) przedstawiona jest na rys. 6.4, zaś algorytm $TSAB$ ($TSAB_{AGV}$) zamieszczono na rys. 6.5. Algorytm rozpoczyna działanie z dowolną permutacją $\pi \in \Pi$, która może być wygenerowana przez np. algorytm NEH_{AGV}. W krokach 2-5 algorytmu iterowana jest procedura NSP; w każdej iteracji generowany jest zbiór ruchów i zbiór reprezentantów, wyznaczany jest ruch v' oraz rozwiązanie sąsiednie π'. Jeżeli rozwiązanie π' jest lepsze od najlepszego znalezionego rozwiązania π^*, rozwiązanie to jest zapamiętywane w kroku 4. W kroku 6, jeśli $Z^* \neq \emptyset$ i nie jest przekroczona maksymalna liczba iteracji $maxiter$, wykonywany jest skok powrotny; rozwiązaniem bieżącym staje się rozwiązanie najlepsze π^*. Należy pamiętać, że po każdorazowym poprawieniu najlepszej znalezionej wartości funkcji celu C^*, zbiór ruchów Z^* „zubażany" jest o ruch v' stosowany do permutacji π^*. Zatem, w kroku 6 sytuacja, w której $Z^* = \emptyset$, stanowi drugi warunek zakończenia pracy algorytmu.

Najbardziej czasochłonnym elementem algorytmu jest przegląd sąsiedztwa, wykonywany w kroku 2 procedury NSP. Wyznaczenie wielkości $C_{\max}(\pi)$, $\pi \in \Pi$, zajmuje $O(n)$ czasu. Zbiór reprezentantów zawiera maksymalnie $2(r-1)$ elementów, zatem złożoność obliczeniowa procedury NSP i tym samym poszczególnych iteracji algorytmu wynosi $O(r \cdot n)$. W pracy [71] zostały przedstawione dwie efektywne metody przeglądu sąsiedztwa, pozwalające na zmniejszenie złożoności obliczeniowej poszczególnych iteracji algorytmu $TSAB$. Jednakże, dla badanego problemu z transportem, własności pozwalające na redukcję złożoności obliczeniowej poszczególnych iteracji algorytmu $TSAB_{AGV}$ nie zostały dotychczas wykryte.

6.4 Wyniki badań numerycznych

Ze względu na brak literaturowych instancji testowych, autor tej rozprawy zdecydował się na zaprojektowanie własnych instancji. Instancje te zostały wygenerowane na bazie instancji Taillarda [104], zaproponowanych dla klasycznego problemu przepływowego. Zestaw Taillarda składa się ze 120 instancji podzielonych na 12 grup. W każdej grupie, określonej przez pary $r \times m^p = 20 \times 5$, 20×10, 20×20, 50×5, 50×10, 50×20, 100×5, 100×10, 100×20, 200×10, 200×20, 500×20, znajduje się po 10 instancji. Poprzez przyjęcie układu maszyn typu pętla, założenie różnych odległości pomiędzy maszynami i uwzględnienie obecności wózka AGV, na bazie instancji Taillarda zostało wygenerowane 140 instancji dla problemu przepływowego z transportem, podzielonych na 14 grup po 10 instancji w każdej grupie. Każda instancja ma swoją unikalną nazwę w postaci $TFr/m^p/x/k$, gdzie r, m^p, x, k są zmiennymi. Zmienne $r \in \{20, 50, 100, 200\}$, $m^p \in \{5, 10\}$

i zmienna $k \in \{1, \ldots, 10$ określają odpowiednio grupę oraz numer instancji Taillarda, na bazie której została wygenerowana instancja dla problemu z transportem. Bardziej precyzyjnie, dla każdej instancji $TFr/m^p/x/k$ przyjmuje się liczbę maszyn produkcyjnych, liczbę zadań i operacji produkcyjnych oraz czasy ich wykonania takie same jak dla k-tej instancji Taillarda w grupie określanej przez parametry $r \times m^p$ (ze względu na dodatkową zmienną x, grupa instancji dla problemu z transportem będzie określana przez parametry r, m^p, x). Zmienna $x \in \{1, 2\}$, określa czasy przejazdu wózka pomiędzy maszynami produkcyjnymi, tj. dla $x = 1$ przyjmuje się $t(l, l+1) = t(m^p, 1) = 5$, $1 \leqslant l < m^p$. W przypadku każdej instancji, dla której $x = 2$, czasy transportu $t(l, l+1)$, $t(m^p, 1)$, $1 \leqslant l < m^p$, losowane są z równym prawdopodobieństwem ze zbioru $\{2, \ldots, 7\}$.

W rozważaniach celowo zostały pominięte instancje, dla których $m^p = 20$, ponieważ w ich przypadku sumaryczny czas transportu w pojedynczym cyklu pracy wózka znacznie przekracza czas wykonania każdej operacji produkcyjnej; wózek pracuje wtedy nieprzerwanie przez cały okres trwania procesu technologicznego. Optymalizacja pracy systemu w takim przypadku nie ma sensu, ponieważ każda permutacja zbioru zadań jest permutacją optymalną.

Wszystkie instancje zostały w sposób przybliżony rozwiązane przez algorytmy NEH_{AGV} i $TSAB_{AGV}$, zaimplementowane w Delphi 6 i uruchamiane na komputerze z procesorem AMD Athlon XP 2500+ (1833 MHz). Permutacja wygenerowana przez algorytm NEH_{AGV} była wykorzystana jako permutacja początkowa dla algorytmu $TSAB_{AGV}$. Algorytm $TSAB_{AGV}$ dla każdej instancji został uruchomiony jednokrotnie z doświadczalnie dobranymi parametrami $\epsilon = 0$, $maxt = 8$, $maxiter = \infty$, $maxret = 1000/500$ ($maxret = 1000$, jeżeli algorytm poprawił wartość funkcji celu i $maxret = 500$, jeśli nastąpił skok powrotny). Przyjęto również dodatkowe kryterium stopu – limit czasu pracy algorytmu, równy 10 minut. Wartości funkcji celu permutacji dostarczonych przez algorytm NEH_{AGV} i $TSAB_{AGV}$ będą oznaczane odpowiednio przez C^{NH} i C^{TS}. Wartości C^{NH} i C^{TS} dla każdej instancji TF zostały przedstawione odpowiednio w tabeli 6.2 i tabeli 6.3. Wyznaczone zostały również średnie czasy pracy odpowiednich algorytmów dla każdej grupy instancji, które będą oznaczane przez T_{av}^{NH} i T_{av}^{TS}. Czasy te zostały przedstawione w tabeli 6.4.

Ze względu na brak literaturowych algorytmów rozwiązywania problemu prezentowanego w tym rozdziale, autor postanowił porównać jakość permutacji (w sensie wartości funkcji celu) generowanych przez algorytmy NEH_{AGV} i $TSAB_{AGV}$ z permutacjami wygenerowanymi przez algorytm $TSAB$. Bardziej precyzyjnie, dla każdej instancji Taillarda (z wyjątkiem

Tabela 6.2: Wartości funkcji celu dostarczone przez algorytm NEH_{AGV} dla poszczególnych instancji $TFr/m^p/x/k$

k	$TFr/m^p/x$						
	20/5/1	20/10/1	50/5/1	50/10/1	100/5/1	100/10/1	200/10/1
1	1359	2006	2956	4267	5894	7886	15150
2	1432	2088	3062	4131	5727	7773	15102
3	1249	1960	2926	4159	5673	7866	15074
4	1402	1907	2977	4250	5493	8057	15083
5	1346	1949	3060	4218	5701	7902	14974
6	1328	1940	3128	4234	5509	7679	14979
7	1363	1963	2952	4238	5769	7827	15281
8	1344	2048	3069	4194	5713	7731	15153
9	1391	2040	2928	4199	5991	8006	15075
10	1240	2021	3091	4217	5896	7926	15128

k	$TFr/m^p/x$						
	20/5/2	20/10/2	50/5/2	50/10/2	100/5/2	100/10/2	200/10/2
1	1360	2018	2957	4315	5895	7951	15258
2	1433	2119	3063	4159	5728	7875	15262
3	1250	1986	2927	4216	5674	7941	15182
4	1403	1930	2978	4274	5494	8146	15231
5	1347	1972	3061	4246	5702	7912	15218
6	1329	1963	3129	4292	5510	7770	15059
7	1364	1976	2953	4329	5770	7961	15312
8	1345	2063	3070	4200	5714	7848	15313
9	1392	2080	2929	4217	5992	8077	15195
10	1241	2054	3092	4225	5897	7886	15269

instancji, dla których $m^p = 20$) algorytm $TSAB$ został uruchomiony na komputerze z procesorem AMD Athlon XP 2500+ z parametrami $\epsilon = 0$, $maxt = 8$ oraz $maxiter = \infty$. Wartość parametru $maxret$ zależna była od liczby operacji, tj. dla $r \cdot m^p \leqslant 400$, $400 \leqslant r \cdot m^p \leqslant 1000$ oraz $r \cdot m^p > 1000$ przyjęto odpowiednio $maxret = 16000/8000$, $4000/2000$ oraz $1000/500$. Permutacja π wygenerowana przez algorytm $TSAB$ dla k-tej instancji Taillarda w grupie $r \times m^p$ była traktowana jak rozwiązanie instancji $TFr/m^p/x/k$ i wykorzystana do wyznaczenia wartości funkcji celu. Wartość ta będzie oznaczana przez C^{TC}. Wyznaczony został również średni czas pracy algorytmu dla poszczególnych grup instancji Taillarda i został oznaczony przez T_{av}^{TC}. Czasy te zostały przedstawione w tabeli 6.4.

Dla każdej instancji $TFr/m^p/x/k$ wyznaczono poprawę względną

$$\Phi_A^{TC} = 100\% \cdot \frac{C^{TC} - C^A}{C^{TC}} \tag{6.33}$$

wartości C^{TC} przez algorytmy NEH_{AGV} i $TSAB_{AGV}$, gdzie $A \in \{NH, TS\}$. Dla każdej grupy instancji została wyznaczona średnia poprawa względna $av\Phi_A^{TC}$. Średnie poprawy względne $av\Phi_A^{TC}$ zostały przedstawione w tabeli 6.4.

Tabela 6.3: Wartości funkcji celu dostarczone przez algorytm $TSAB_{AGV}$ dla poszczególnych instancji $TFr/m^p/x/k$

k	$TFr/m^p/x$						
	20/5/1	20/10/1	50/5/1	50/10/1	100/5/1	100/10/1	200/10/1
1	1317	1934	2928	4039	5776	7733	15005
2	1408	1988	2999	3921	5685	7600	14949
3	1186	1880	2815	3929	5612	7636	14941
4	1349	1820	2883	4070	5398	7833	14928
5	1291	1865	3000	4012	5632	7619	14879
6	1295	1815	3060	4060	5422	7431	14835
7	1336	1844	2819	4016	5678	7588	15080
8	1272	1943	2911	4003	5645	7621	15048
9	1310	1920	2732	3975	5905	7759	14961
10	1188	1931	2960	4005	5817	7692	14965

k	$TFr/m^p/x$						
	20/5/2	20/10/2	50/5/2	50/10/2	100/5/2	100/10/2	200/10/2
1	1318	1957	2929	4063	5777	7782	15089
2	1409	2010	3000	3994	5686	7670	15092
3	1191	1895	2816	3997	5613	7634	15108
4	1350	1844	2879	4063	5399	7899	15062
5	1292	1892	3001	4072	5633	7725	15127
6	1297	1839	3061	4043	5423	7544	14908
7	1337	1865	2834	4105	5679	7771	15137
8	1270	1967	2918	4041	5646	7654	15257
9	1311	1938	2733	4062	5906	7865	15051
10	1189	1950	2961	4099	5818	7749	15129

Algorytm $TSAB$ dostarcza permutacji wysokiej jakości dla klasycznego permutacyjnego problemu przepływowego. Pomimo to, po potraktowaniu ich jak rozwiązań problemu z transportem, permutacje te okazały się

Tabela 6.4: Średnie względne poprawy i średnie czasy pracy algorytmów dla grup instancji $TFr/m^p/x/k$

Grupa $r/m^p/x$	$av\Phi_{NH}^{TC}$ [%]	$av\Phi_{TS}^{TC}$ [%]	T_{av}^{TC} [s]	T_{av}^{NH} [s]	T_{av}^{TS} [s]
20/5/1	2,67	6,33	48,58	0,00	13,01
20/10/1	2,85	7,64	86,47	0,01	27,13
50/5/1	7,70	10,93	164,21	0,02	312,06
50/10/1	3,07	7,85	86,07	0,05	558,97
100/5/1	9,15	10,41	96,32	0,17	600,01
100/10/1	5,18	7,76	191,62	0,36	600,01
200/10/1	6,33	7,20	148,35	2,82	600,06
20/5/2	2,67	6,31	48,58	0,00	14,38
20/10/2	2,64	7,49	86,47	0,01	32,14
50/5/2	7,70	10,88	164,21	0,02	361,24
50/10/2	3,07	7,48	86,07	0,05	581,95
100/5/2	9,15	10,41	96,32	0,17	600,00
100/10/2	4,99	7,47	191,62	0,36	600,01
200/10/2	6,14	6,97	148,35	2,81	600,05
średnia	5,24	8,23	117,37	0,49	392,93

znacznie gorsze nawet od permutacji generowanych przez algorytm konstrukcyjny NEH_{AGV}. Algorytm ten poprawił wartość C^{TC} dla 136, spośród 140 badanych instancji, generując rozwiązania średnio lepsze o $5,2\%$. Nie trudno również zauważyć, że permutacje dostarczone przez algorytm NEH_{AGV} zostały wygenerowane w czasie znacznie krótszym od czasu pracy algorytmu $TSAB$. Czas pracy tego algorytmu dla każdej instancji nie przekroczył trzech sekund, podczas gdy średni czas pracy algorytmu $TSAB$ wynosił około dwóch minut. Minimalna poprawa względna Φ_{NEH}^{TC} została zanotowana dla instancji $TF20/5/1/9$ i wyniosła $-3,8\%$. Poprawa maksymalna, wynosząca $11,2\%$, została zanotowana dla dwóch instancji, $TF100/5/1/2$ oraz $TF100/5/2/2$. Algorytm $TSAB_{AGV}$ poprawił wszystkie wartości C^{TC}, C^{NH}, dostarczone przez algorytm $TSAB$ i NEH_{AGV}. Minimalna i maksymalna poprawa względna Φ_{TS}^{TC} wyniosła odpowiednio $2,2\%$ oraz $12,8\%$ i została wyznaczona odpowiednio dla instancji $TF20/5/1/9$ oraz $TF50/5/1/5$.

Na podstawie tabeli 6.4 można poczynić kilka dodatkowych obserwacji. Otóż wielkość $av\Phi_{NEH}^{TC}$ zmienia się wprost proporcjonalnie do liczby zadań r przy stałej liczbie maszyn produkcyjnych m^p. Tej samej tenden-

cji nie można zaobserwować w przypadku średnich popraw względnych $av\Phi_{TS}^{TC}$. Dla zestawów z $m^p = 5$ maszynami (z wyjątkiem zestawów 20/5/1, 20/5/2), wielkość ta oscylowała pomiędzy 10% i 11%. Podobnie, dla zestawów z $m^p = 10$ maszynami, poprawa ta wyniosła około 7%. Na podstawie powyższych obserwacji można wysnuć dwa wnioski. Po pierwsze, Jakość rozwiązań generowanych przez algorytm NEH_{AGV} rośnie wraz z liczbą zadań. Wniosek ten jest dość oczywisty, ponieważ większa liczba zadań oznacza wybór rozwiązania spośród większej liczby rozwiązań próbnych, konstruowanych w każdej iteracji algorytmu. Po drugie, jakość rozwiązań generowanych przez algorytm $TSAB_{AGV}$ w niewielkim tylko stopniu zależy od rozwiązania początkowego, jest natomiast zależna od liczby maszyn produkcyjnych.

6.5 Wnioski i uwagi

W niniejszym rozdziale rozważano elastyczny system produkcyjny o strukturze przepływowej permutacyjnej, w którym maszyny zorganizowane zostały w układ typu pętla. Do transportu zadań pomiędzy maszynami wykorzystano jeden jednokierunkowy wózek AGV. Za kryterium optymalizacji przyjęto moment zakończenia wykonywania procesu technologicznego. Dla problemu skonstruowano model matematyczny oraz wprowadzono reprezentację permutacyjno-grafową, popartą wyczerpującym przykładem obliczeniowym. Niewątpliwymi zaletami zdefiniowanego grafu jest jego acykliczność oraz łatwość modyfikacji, towarzysząca modyfikacjom reprezentowanych permutacji. Wprowadzony graf umożliwia również stosunkowo szybkie i „wygodne" wyznaczenie wartości funkcji kryterialnej. Kolejną zaletą jest możliwość podziału ścieżki krytycznej w grafie na segmenty i bloki zadań o podobnych własnościach jak w przypadku klasycznym. Dzięki temu, w algorytmach rozwiązywania problemu można bezpośrednio stosować niektóre zbiory ruchów i sąsiedztwa, znane dla klasycznego permutacyjnego problemu przepływowego. Największym mankamentem jest niemożność bezpośredniego zastosowania bądź modyfikacji własności „klasycznych", pozwalających na akceleracją obliczeń związanych z przeglądem sąsiedztwa. Dalej, w rozdziale rozważano algorytm NEH oraz $TSAB$, oryginalnie zaprojektowane do rozwiązywania klasycznego permutacyjnego problemu przepływowego. Okazuje się, że po stosunkowo niewielkich modyfikacjach, algorytmy te są również bardzo efektywne w kontekście rozważanego problemu, czego wyrazem są wyniki przeprowadzonych badań numerycznych.

Wprowadzony model matematyczny problemu może stanowić bazę do dalszych uogólnień, prowadzących do coraz lepszych opisów rzeczywistości.

W rzeczywistych systemach produkcyjnych o opisywanej strukturze może być wykorzystywane więcej, niż jeden wózek AGV. Wózki korzystają z tej samej drogi, co wiąże się z koniecznością wprowadzenia dodatkowych mechanizmów zapobiegania konfliktom transportowym – należy zachować odpowiednią odległość pomiędzy wózkami. W praktyce, jedno z rozwiązań tego problemu polega dodatkowym na podziale przestrzeni pomiędzy poszczególnymi maszynami produkcyjnymi na strefy. Wymaga się, by w trakcie trwania procesu produkcyjnego wózki zawsze znajdowały się w różnych strefach. W modelu matematycznym wspomnianego systemu produkcyjnego powinny być zatem uwzględnione dwie rzeczy. Po pierwsze, przydział wózków do operacji transportowych nie jest dany. Należy zatem uwzględnić zarówno możliwość tworzenia jak i modyfikacji takiego przydziału. Po drugie, harmonogram pracy wózków, skonstruowany poprzez ustalenie przydziału i kolejności wykonania operacji transportowych, powinien być skonstruowany tak, by respektować zasadę dotyczącą pobytu wózków w różnych strefach.

Rozdział 7

Wnioski końcowe

W poprzednich rozdziałach pracy zostały zaproponowane modele matematyczne problemów przepływowych i gniazdowych z uwzględnieniem czasów transportu. W modelach tych uwzględniono ograniczenia związane ze skończoną liczbą środków transportu. Uwzględniono też cały szereg dodatkowych faktów i ograniczeń, dzięki którym opisane modele bardziej precyzyjnie odzwierciedlają pracę rzeczywistych systemów produkcyjnych. W szczególności, uwzględniono praktyczny tryb pracy wózków – tryb „zabierz i zostaw". Każdorazowo rozpatrywano maszynowo-zależne czasy transportu i przejazdów bez załadunku. W przypadku problemów gniazdowych rozpatrywano dodatkowo zadaniowo-zależne czasy transportu. Dzięki odpowiedniemu doborowi parametrów określających poszczególne instancje problemów możliwe jest zamodelowanie szerokiej grupy rzeczywistych systemów produkcyjnych. Powyższe spostrzeżenia pozwalają wnioskować o prawdziwości pierwszej z tez, wysnutych na początku niniejszej rozprawy.

W pracy został wychwycony i udowodniony szereg specyficznych własności, wykazujących zarówno podobieństwa jak i różnice pomiędzy badanymi problemami a problemami klasycznymi. W przypadku problemów gniazdowych, znaczne różnice pojawiają się w kontekście własności sąsiedztw oraz niektórych technik ich przeglądu. W przypadku problemu przepływowego z transportem, dzięki wykrytym własnościom, możliwe jest w zasadzie bezpośrednie stosowanie „klasycznych" sąsiedztw i technik jego rozwiązywania. Wyjątkiem tu są klasyczne techniki efektywnego przeglądu sąsiedztwa, których nie można zastosować w rozważanym przypadku. Nie udało się też w to miejsce zaproponować innych efektywnych technik, co w rozważanym problemie przepływowym czyni przegląd sąsiedztwa stosunkowo czasochłonnym.

Dalej, w pracy został przedstawiony szereg algorytmów konstruujących bądź poprawiających poszczególne rozwiązania badanych problemów. Dzięki implementacji wcześniej zaproponowanych własności, we wspomnianych algorytmach możliwe było zastosowanie wielu rozwiązań, znanych dla klasycznych problemów przepływowych i gniazdowych. Również zastosowane techniki przeglądu wspomnianych sąsiedztw, dzięki odpowiednim własnościom, przyczyniły się do znacznego zwiększenia wydajności większości algorytmów. Powyższe spostrzeżenia potwierdzają prawdziwość drugiej i trzeciej tezy przedstawionej w pierwszym rozdziale tej pracy.

W trakcie implementacji algorytmów wykorzystano wiele różnych technik, które w literaturze uchodzą za najlepsze w odniesieniu do problemów klasycznych. Wśród nich można wyróżnić technikę symulowanego wyżarzania, technikę poszukiwań z zabronieniami, technikę poszukiwania rozproszonego, czy podejście genetyczne. Zaprojektowane algorytmy zostały poddane badaniom numerycznym, polegającym na przybliżonym rozwiązaniu szeregu instancji testowych zarówno zaprojektowanych przez autora rozprawy, jak i tych, proponowanych w literaturze. Wyniki badań posłużyły do określenia relacji czasowych i jakościowych pomiędzy badanymi algorytmami oraz, w konsekwencji, wyłonienia algorytmów najlepszych. Jakość najlepszych algorytmów została potwierdzona przy użyciu instancji testowych znanych z literatury – większość najlepszych rozwiązań literaturowych została poprawiona. Część rezultatów prezentowanych w tej rozprawie została opublikowana w czasopismach krajowych [75–78, 94, 96–98, 108] i międzynarodowych [95, 99].

Można wyszczególnić kilka możliwych nurtów dalszego rozwoju zagadnień związanych z problemami przepływowymi i gniazdowymi opisanymi w tej pracy. Pierwszy z nurtów dotyczy opracowania i zbadania własności nowych miar odległości pomiędzy poszczególnymi rozwiązaniami omawianych problemów (dotyczy to sytuacji, w której przydział środków transportu do poszczególnych operacji transportowych nie jest z góry określony i stanowi dodatkową zmienną decyzyjną). Miary takie z pewnością mogą się przyczynić do dalszego rozwoju badań dotyczących krajobrazu przestrzeni rozwiązań badanych problemów. W konsekwencji będzie możliwe, między innymi, opracowanie lepszych schematów dywersyfikacji obliczeń w algorytmach poszukiwania rozproszonego, bądź lepszych quasi-operatorów krzyżowania i mutacji dla algorytmów genetycznych.

Sąsiedztwa stosowane w poszczególnych algorytmach, poza niewątpliwymi zaletami, posiadają też pewne wady, które zostały ujawnione na etapie badań numerycznych przy użyciu niektórych instancji testowych. Otóż liczba rozwiązań sąsiednich w tych sąsiedztwach w znacznym stopniu zależy

od rozmiaru rozwiązywanej instancji. W przypadku instancji o niewielkim oraz istotnym znaczeniu transportów liczba ta jest odpowiednio zbyt mała i zbyt duża, co negatywnie wpływa na proces eksploracji przestrzeni rozwiązań. Drugi nurt badań może zatem mieć na celu opracowanie nowych, lepszych sąsiedztw pozbawionych wad swoich poprzedników.

Trzeci możliwy nurt badań dotyczy dalszego rozwoju modeli matematycznych systemów przepływowych z transportem. Model zaproponowany w tej pracy może stanowić bazę do dalszych uogólnień, polegających na m.in. uwzględnieniu większej liczby środków transportu. Wiąże się to z koniecznością uwzględnienia konstrukcji i modyfikacji przydziału wózków do operacji transportowych i potrzebą eliminacji ryzyka powstawania konfliktów transportowych już na etapie konstrukcji poszczególnych uszeregowań. W dalszych pracach warto też skupić się na wykryciu własności pozwalających na przyspieszenie przeglądu zaproponowanego sąsiedztwa poszczególnych rozwiązań problemu.

Literatura

[1] AARTS E.H.I., van LAARHOVEN P.J.M., LENSTRA J.K., ULDER N.I.J., *A computational study of local search algorithms for job shop scheduling*, ORSA Journal on Computing, 1994, 6, 118-125.

[2] AARTS E.H.I., LENSTRA J.K., *Local search in Combinatorial Optimization*, John Wiley and Sons Ltd, Chichester, England, 1997.

[3] BALAS E.,LENSTRA J.K., VAZACOPOULOS A., *The one-machine problem with delayed precedence constraints and its use in job shop scheduling*, Management Science, 1995, 41, 1, 94-109.

[4] BARNES J.W., CHAMBERS J.B., *Solving the job shop scheduling problem with tabu search*, IIE Transactions, 1995, 27, 257-263.

[5] BANASZAK Z.A., SKOLUD B., ZAREMBA M.B., *Computer-aided prototyping of production flows for a virtual enterprise*, Journal of Intelligent Manufacturing, 2003, 14, 83-106.

[6] BANASZAK Z.A., TANG X.Q., WANG S.C., ZAREMBA M.B., *Logistics models in flexible manufacturing*, Computers in Industry, 2000, 43, 237-248.

[7] BILGE Ü., ULUSOY G., *A time window approach to simultaneous scheduling of machines and material handling system in an FMS*, Operations Research, 1995, 43, 1058-1070.

[8] BŁAŻEWICZ J., *Problemy optymalizacji kombinatorycznej – złożoność obliczeniowa, algorytmy aproksymacyjne*, PWN, Warszawa – Łódź, 1986.

[9] BŁAŻEWICZ J., *Złożoność obliczeniowa problemów kombinatorycznych*, WNT, Warszawa, 1988.

[10] BŁAŻEWICZ J., DOMSCHKE W., PESH E., *The job shop scheduling problem: Conventional and new solution techniques*, European Journal of Operational Research, 1996, 93, 1-33.

[11] BOESE K., KAHNG A.,MUDDU S., *A new adaptive multi-start technique for combinatorial global optimizations*, Operations Research Letters, 1994, 16, 101-113.

[12] BOZER Y.A., SRINIVASAN M.M., *Tandem configurations for automated guided vehicle systems and the analysis of single vehicle loops*, IIE Transactions, 1991, 23, 72-82.

[13] BOZER Y.A., SRINIVASAN M.M., *Tandem AGV systems: A partitioning algorithm and performance comparison with conventional AGV systems*, European Journal of Operational Research, 1992, 63, 173-191.

[14] BRUCKER P., *Scheduling Algorithms*, Springer-Verlag, Berlin-Heidelberg, 1995.

[15] BRUCKER P. JURISH B., SIEVIERS B., *A fast branch and bound algorithm for the job shop problem*, Discrete Applied Mathematics, 1994, 49, 107-127.

[16] BRUCKER P., KNUST S., *Lower bounds for scheduling a single robot in a job-shop environment*, Annals of Operations Research, 2002, 115, 147-172.

[17] CHENG R., GEN M., TSUJIMURA Y., *A tutorial survey of job-shop scheduling problems using genetic algorithms, part I: representation*, International Journal of Computers and Industrial Engineering, 1996, 30, 983–997.

[18] CHENG R., GEN M., TSUJIMURA Y., *A tutorial survey of job-shop scheduling problems using genetic algorithms, part II: hybrid genetic search strategies*, International Journal of Computers and Industrial Engineering, 1999, 36, 343–364.

[19] CHEUNG W., ZHOU H., *Using algorithms and heuristics for job shop scheduling with sequence-dependent setup times*, Annals of Operations Research, 2001, 107, 65-81.

[20] CORMEN T. H., LEISERSON C. E., RIVEST R. L., *Wprowadzenie do algorytmów*, WNT, Warszawa, 1997.

[21] DAVIS L., *Job shop scheduling with genetic algorithms*, Proceedings of the First International Conference on Genetic Algorithms and their Applications, 1985, 136-140.

[22] DELL'AMICO M., *Shop problems with two machines and time lags*, Operations Research, 1996, 44, 777-787.

[23] DELL'AMICO M., TRUBIAN M., *Applying tabu search to the job shop scheduling problem*, Annals of Operations Research, 1993, 41, 231-252.

[24] DELLA CROCE F., TADEI R., VOLTA G., *A genetic algorithm for the job shop problem*, Computers and Operations Research, 1995, 22, 15-24.

[25] FRANCIS R.L., McGINNIS L.F. Jr., WHITE J.A., *Facility layout and location: an analytical approach*, Prentice-Hall, Englewood Cliffs, NJ, 1992.

[26] GANESHARAJAH T., HALL N.G., SRISKANDARAJAH C., *Design and operational issues in AGV-served manufacturing systems*, Annals of Operations Research, 1998, 76, 109-154.

[27] GLOVER F., *Heuristics for integer programming using surrogate constraints* Decision Sciences, 1977, 8, 156-166.

[28] GLOVER F., *Tabu search. Part I*, ORSA Journal of Computing, 1989, 1, 190-206.

[29] GLOVER F., *Tabu search. Part II*, ORSA Journal of Computing, 1990, 2, 4-32.

[30] GLOVER F., LAGUNA M., *Tabu search*, Kluwer Academic Publishers, Massachusetts, USA, 1997.

[31] GOH K.S., LIM A., RODRIGUES B., *Sexual selection for genetic algorithms*, Artifical Intelligence Review, 2003, 19, 123-152.

[32] GOLDBERG D., *Algorytmy genetyczne i ich zastosowania*, WNT, Warszawa, 1998.

[33] GOLDEN B.L., ASSAD A., *Perspectives on vehicle routing: exciting new developments*, Operations Research, 1986, 34, 803-810.

[34] GRABOWSKI J., *Uogólnione zagadnienia optymalizacji kolejności operacji w dyskretnych zagadnieniach produkcyjnych*, Prace naukowe Instytutu Cybernetyki Technicznej Politechniki Wrocławskiej, seria Monografie, 1979.

[35] GRABOWSKI J., NOWICKI E., SMUTNICKI C., *Algorytm blokowy szeregowania operacji w systemie gniazdowym*, Przegląd Statystyczny, 1988, 35, 67-80.

[36] GRABOWSKI J., NOWICKI E., SMUTNICKI C., *Metoda blokowa w zagadnieniach szeregowania zadań* Akademicka Oficyna Wydawnicza EXIT, Warszawa, 2003.

[37] GRAHAM R.L., LAWLER E.L., LENSTRA J.K., RINNOOY KAN A.H.G., *Optimization and approximation in deterministic sequencing and scheduling*, Annals of Discrete Matematics, 1979, 5, 287-326.

[38] HALL N.G., KAMOUN H., SRISKANDARAJAH C., *Scheduling in robotic cells: classification, two and three machine cells*, Operations Research, 1997, 45, 421-439.

[39] HALL N.G., SRISKANDARAJAH C., GANESHARAJAH T., *Operational decisions in AGV-served flowshop loops: scheduling*, Annals of Operations Research, 2001, 107, 161-188.

[40] HALL N.G., SRISKANDARAJAH C., GANESHARAJAH T., *Operational decisions in AGV-served flowshop loops: fleet sizing and decomposition*, Annals of Operations Research, 2001, 107, 189-209.

[41] HENNIG A., *Praktische job-shop scheduling-probleme*, Ph. d. Thesis, Friedrich-Schiller-Universität, Jena, 2002.

[42] HOLLAND J.H., *Adaptation in natural and artificial systems*, University of Michigan Press, Ann Arbor, 1975.

[43] HURINK J., KNUST S., *Makespan minimization for flow-shop problems with transportation times and a single robot*, Discrete Applied Mathematics, 2001, 112, 199-216.

[44] HURINK J., KNUST S., *A tabu search algorithm for scheduling a single robot in a job-shop environment*, Discrete Applied Mathematics, 2002, 119, 181-203.

[45] HURINK J., KNUST S., *Tabu search algorithms for job-shop problems with a single transport robot*, European Journal of Operational Research, 2005, 162, 99-111.

[46] JAIN A.S., MEERAN S., *Deterministic job-shop scheduling: Past, present and future*, European Journal of Operational Researchm, 1999, 113, 390-434.

[47] JAIN A.S., RANGASWAMY B., MEERAN S., *New and "stronger" job-shop neighbourhoods: A focus on the method of Nowicki and Smutnicki (1996)*, Journal of Heuristics, 2000, 6, 457-480.

[48] JOHNSON S.M., *Optimal two- and three-stage production schedules with setup times included*, Naval Res. Logist. Quart., 1954, 1, 61-68.

[49] KIRKPATRICK S., GELATT C.D., VECCHI M.P., *Optimization by simulated annealing*, Science, 1983, 220, 671-680.

[50] KNUST S., *Shop-scheduling problems with transportation*, Ph.D. thesis, Fachbereich Mathematik/Informatik, Universität Osnabrück, 1999.

[51] KNUTH D.E., *The art of computer programming*, Addison Wesley, Longman, 1997.

[52] KUBIAK W., SETHI S., SRISKANDARAJAH C., *An affcient algorithm for a job shop problem*, Working paper, Faculty of Business Administration, Memorial University of New foundland, 1993.

[53] KUSIAK A., *Material handling in flexible manufacturing systems*, Material Flow, 1985, 2, 79-95.

[54] VAN LAARHOVEN P.J.M., AARTS E.H.L., LENSTRA J.K., *Job-shop scheduling by simulated annealing*, Operations Research, 1992, 40, 113-125.

[55] LAWRENCE S., *Resource constrained project scheduling: an experimental investigation of heuristic scheduling techniques*, Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, Pennsylvania, 1984.

[56] LIM J.K., KIM K.H., LEE J.H., *A dispatching method for automated guided vehicles by using a bidding concept*, OR Spectrum, 2003, 25, 25-44.

[57] LIPSKI W., *Kombinatoryka dla programistów*, WNT, Warszawa, 1982.

[58] MANLY B.F.J., *Randomization and Monte Carlo Methods in biology*, Chapman and Hall, Londyn, 1991.

[59] MATTFELD D.C., BIERWITH C., KOPFER H., *A search space analysis of the job shop scheduling problem*, Annals of Operations Research, 1999, 86, 441-453.

[60] MATSUO H., SUH C.J., SULLIVAN R.S., *A controlled search simulated annealing method for the general jobshop scheduling problem*, Working paper 03-04-88, University of Texas, Austin, 1988.

[61] MITTEN L.G., *Sequencing n jobs on two machines with arbitrary time lags*, Management Science, 1958, 5, 293-298.

[62] MUTH J.F., THOMPSON G.L., *Industrial scheduling*, Prentice Hall, Englewood Cliffs, New Jersey, 1963.

[63] NAWAZ M., ENSCORE Jr. E.E., HAM I., *A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem*, OMEGA International Journal of Management Science, 1983, 11, 91-95.

[64] NEMHAUSER G. L., RINNOOY KAN A. H. G., TODD M. J., *Handbooks in operations research and management science*, Optimalization vol.1, North-Holland 1989.

[65] NOWICKI E., *Metoda tabu w problemach szeregowania zadań produkcyjnych*, Oficyna Wydawnicza Politechniki Wrocławskiej, Monografie, 27, Wrocław, 1999.

[66] NOWICKI E., *Algorytm tabu dla problemu gniazdowego z czasami transportu*, Zeszyty Naukowe Politechniki Śląskiej, seria Automatyka, 2002, 134, 327-337.

[67] NOWICKI E., MAKUCHOWSKI M., *Krajobraz przestrzeni rozwiązań problemu gniazdowego*, Automatyka, 2001, tom 5 z. 1/2, 449-456.

[68] NOWICKI E., MAKUCHOWSKI M., *Metoda wstawień w klasycznych problemach szeregowania. Cz. 1. Problem przepływowy*, w: Komputerowo Zintegrowane Zarządzanie pod red. R. Knosali, tom II, WNT, Warszawa, 2001, 113-122.

[69] NOWICKI E., MAKUCHOWSKI M., *Metoda wstawień w klasycznych problemach szeregowania. Cz. 2. Problem gniazdowy*, w: Komputerowo Zintegrowane Zarządzanie pod red. R. Knosali, tom II, WNT, Warszawa, 2001, 123-132.

[70] NOWICKI E., SMUTNICKI C., *A fast taboo search algorithm for the job shop problem*, Management Science, 1996, 42, 797-813.

[71] NOWICKI E., SMUTNICKI C., *A fast tabu search algorithm for the permutation flow-shop problem*, European Journal of Operational Research, 1996, 91, 160-175.

[72] NOWICKI E., SMUTNICKI C., *An advanced TS algorithm for the job shop problem*, Journal of Scheduling, 2005, 8, 145-159.

[73] NOWICKI E., SMUTNICKI C., *Some new ideas in TS for job-shop scheduling*, w: Metaheuristic Optimization via Memory and Evolution. Tabu Search and Scatter Search pod red. C. Rego i B. Alidaee, Kluwer Academic Publishers, 2005.

[74] NOWICKI E., SMUTNICKI C., *Some aspectts of scatter search in the flow shop problem*, European Journal of Operational Research, 2006, 169, 654-666.

[75] NOWICKI E., TYŃSKI A., *Algorytmy konstrukcyjne dla problemu gniazdowego z czasami transportu*, w: Komputerowo Zintegrowane Zarządzanie pod red. R. Knosali, tom II, WNT, Warszawa, 2003, 140-149.

[76] NOWICKI E., TYŃSKI A., *Analiza sąsiedztwa w problemie gniazdowym z ograniczoną liczbą wózków AGV*, w: Automatyzacja procesów dyskretnych pod red. R. Gessinga, T. Szkodnego, WNT, Warszawa, 2004, 141-149.

[77] NOWICKI E., TYŃSKI A., *Własności przestrzeni rozwiązań problemu gniazdowego z czasami transportu*, w: Komputerowo zintegrowane zarządzanie pod red. R. Knosali. Tom II, WNT, Warszawa, 2004, 192-199.

[78] NOWICKI E., TYŃSKI A., *Problem gniazdowy z transportem*, w: Badania operacyjne i systemowe wobec wyzwań XXI wieku pod red. Z. Bubnickiego, O. Hryniewicza, J. Węglarza, Akademicka Oficyna Wydawnicza EXIT, Warszawa, 2004, 201-211.

[79] OGBU F.F., SHMITH D.K., *The application of the simulated annealing algorithm to the solution of the* $n\,|m|\,C_{\max}$ *flow shop problem*, Computer and Operations Research, 1990, 17, 243-253.

[80] OSMAN I.H., POTTS C.N., *Simulated annealing for permutation flow shop scheduling*, OMEGA International Journal of Management Science, 1989, 17, 551-557.

[81] PAULLI J., *A hierarchical approach for the FMS scheduling problem*, European Journal of Operational Research, 86, 1995, 32-42.

[82] PEZZELLA F., MERELLI E., *A tabu search method guided by shifting bottleneck for the job-shop scheduling problem*, European Journal of Operational Research, 2000, 120, 297-310.

[83] RAYWALD-SMITH V.J., REBAINE D. *Open shop scheduling with delays*, Theoretical Informatics and Applications, 1992, 26, 439-448.

[84] REEVES C.R., *A genetic algorithm for flowshop sequencing*, Computers and Operations Research, 1995, 22, 5–13.

[85] REEVES C.R., *Landscapes, operators and heuristic search*, Technical Report, School of MIS Conventry University, 1998.

[86] REEVES C., YAMADA T., *Genetic algorithms, path relinking and the flowshop sequencing problem*, Evolutionary Computation, 1995, 80, 397-403.

[87] RO I., KIM J., *Multi-criteria operational control rules in flexible manufacturing systems*, International Journal of Production Research, 1990, 28, 47-63.

[88] SAWIK T., *Optymalizacja dyskretna w elastycznych systemach produkcyjnych*, WNT, Warszawa, 1992.

[89] SAWIK T., *Planowanie i sterowanie produkcji w elastycznych systemach montażowych*, WNT, Warszawa, 1996.

[90] SEO Y., EGBELU P.J., *Integrated manufacturing planning for an AGV-based FMS*, International Journal of Production Economics, 1999, 60, 473-478.

[91] SETHI S.P., SRISKANDARAJAH C., SORGER G., BŁAŻEWICZ J., KUBIAK W., *Sequencing of parts and robot moves in a robotic cell*, International Journal of Flexible Manufacturing Systems, 1992, 4, 331-358.

[92] SINRIECH D., TANCHOCO J.M.A., *Solution methods for the mathematical models of single-loop AGV systems*, International Journal of Production Research, 1993, 31, 705-725.

[93] SMUTNICKI C., *Algorytmy szeregowania*, Akademicka Oficyna Wydawnicza EXIT, Warszawa, 2002.

[94] SMUTNICKI C., TYŃSKI A., *Algorytm genetyczny dla problemu gniazdowego z czasami transportu*, w: Systemy Sterowania. Zbiór prac pod red. W. Greblickiego, Cz. Smutnickiego, Warszawa, WKŁ, 2005, 393-404.

[95] SMUTNICKI C., TYŃSKI A., *Job shop scheduling by GA. A new crossover operator*, w: Proceedings of International Scientific Annual Conference Operations Research 2005, Bremen, Springer Verlag, 2005 (w druku).

[96] SMUTNICKI C., TYŃSKI A., *Modelowanie przepływu zadań w elastycznym systemie produkcyjnym z wózkami AGV*, Automatyka, 2005, 9, 223-232.

[97] SMUTNICKI C., TYŃSKI A., *Nowy operator krzyżowania dla problemu gniazdowego z kryterium addytywnym*, w: Komputerowo Zintegrowane Zarządzanie. Zbiór prac pod red. R. Knosali. Tom II. Warszawa, WNT, 2005, 416-425.

[98] SMUTNICKI C., TYŃSKI A., *Problem gniazdowy z transportem i ograniczoną liczbą niededykowanych wózków AGV*, Automatyka, 2005, 9, 233-244.

[99] SMUTNICKI C., TYŃSKI A., *Simultaneous machine scheduling and AGV route planning in the FMS*, Proceedings of the 11th IEEE International Conference on Methods and Models in Automation and Robotics, 2005, 1131-1136.

[100] SOTSKOV Y.N., STRUSEVICH Y.A., *NP-hardnes of shop-scheduling problem with three jobs*, Discrete Applied Mathenatics 1993.

[101] SRISKANDARAJAH C., HALL N.G.,KAMOUN H., *Scheduling large robotic cells without buffers*, Annals of Operations Research 1998, 76, 287-321.

[102] SURI R., DESIRAJU R., *Performance analysis of flexible manufacturing systems with a single discrete material-handling device*, The International Journal of Flexible Manufacturing Systems, 1997, 9, 223-249.

[103] SYSŁO M. M., NARSINGH D., KOWALSKI J. S., *Algorytmy optymalizacji dyskretnej*, PWN, Warszawa 1995.

[104] TAILLARD E., *Benchmarks for basic scheduling problems*, European Journal of Operational Research, 64, 1993, 278-285.

[105] TAILLARD E., *Parallel taboo search techniques for the job shop scheduling problem*, ORSA Journal on Computing, 1994, 6, 108-117.

[106] TAILLARD E., *Some efficient heuristic methods for flow shop sequencing*, European Journal of Operational Research, 1990, 47, 65-74.

[107] TANCHOCO J.M.A., EGBELU P.J., TAGHABONI F., *Determination of the total number of vehicles in an AGV-based material transport system*, Material Flow, 1987, 4, 33-51.

[108] TYŃSKI A., *Zastosowanie techniki poszukiwań z zabronieniami w rozwiązywaniu problemu przepływowego z transportem* w: Komputerowo Zintegrowane Zarządzanie. Zbiór prac pod red. R. Knosali. Tom II. Warszawa, WNT, 2006, 595-604.

[109] ULUSOY G., SIVRIKAYA-ŞERIFOĞLU F., BILGE Ü., *A genetic algorithm approach to the simultaneous scheduling of machines and automated guided vehicles* Computers and Operations Research, 1997, 24, 4, 335-351.

[110] VAESSENS R.J.M., AARTS E.H.I., LENSTRA J.K., *Job shop scheduling by local search*, INFORMS Journal on Computing, 1996, 8, 302-317.

[111] VALLS V., PEREZ M.A., QUINTANILLA M.S., *A tabu search approach to machine scheduling*, European Journal of Operational Research, 1998, 106, 277-300.

[112] WANG H., WU K., *Hybrid genetic algorithm for optimization problems with permutation property*, Computers & Operations Research, 2004, 31, 2453-2471.

[113] WERNER F., WINKLER A., *Insertion techniques for the heuristic solution of the job shop problem*, Discrete Applied Mathematics, 1995, 58, 191-211.

[114] WIDMER M., HERTZ A., *A new heuristic method for the flow shop sequencing problem*, European Journal of Operational Research, 1989, 41, 186-193.

Spis tabel

3.1 Przydział maszyn, czasy wykonania oraz długości najdłuższych ścieżek dla poszczególnych operacji z przykładu 3.1 . 35
3.2 Maszyny dedykowane, czasy wykonywania i długości najdłuższych ścieżek 55
3.3 Czasy przezbrojeń pomiędzy operacjami 55
3.4 Współczynniki korelacji i inne parametry dla poszczególnych instancji 64
3.5 Odległości i błędy względne dopuszczalnych permutacji losowych 66

4.1 Średnie arytmetyczne poprawy $av\Phi_A^{MIN}$ dla zestawów $TR$16-25, $TR$36-40 91
4.2 Wartości funkcji celu rozwiązań wygenerowanych przez algorytm $INT3$ dla zestawów $TR$16-25, $TR$36-40 92
4.3 Średnie arytmetyczne poprawy $av\Phi_A^{IN}$ oraz średnie czasy pracy algorytmów popraw dla zestawów $TR$16-25, $TR$36-40 96
4.4 Średnie arytmetyczne poprawy $av\Phi_A^{IN}$ oraz średnie czasy pracy algorytmów popraw dla zestawów $TR$16-25, $TR$36-40, c.d. 97
4.5 Wartości funkcji celu najlepszych rozwiązań wygenerowanych przez wszystkie algorytmy dla zestawów $TR$16-25, $TR$36-40 98
4.6 Czasy pracy algorytmów popraw dla instancji HK 101
4.7 Wielkości C^{HK} oraz arytmetyczne poprawy Φ_A^{HK} dla instancji HK 102

5.1 Wielkości $er_{4,z}$, $eq_{4,z}$, $ed_{4,z}$ wyznaczone dla grafu $G(\beta)$ z przykładu 5.1 121
5.2 Średnie arytmetyczne poprawy $av\Phi_A^{MIN}$ dla zestawów $TR$16-25, $TR$36-40 135

5.3 Wartości funkcji celu rozwiązań wygenerowanych przez algorytm $INT4$ 136
5.4 Średnie arytmetyczne poprawy $av\Phi_A^{IN}$ i średnie czasy pracy algorytmów popraw dla zestawów $TM$16-25, $TM$36-40 . . . 139
5.5 Wartości funkcji celu najlepszych rozwiązań dla instancji z zestawów $TM$16-25, $TM$36-40 140
5.6 Arytmetyczne poprawy Φ_A^{BU} dla instancji EX 144
5.7 Arytmetyczne poprawy Φ_A^{BU} dla instancji EX, c.d. 145

6.1 Czasy wykonania operacji produkcyjnych 155
6.2 Wartości funkcji celu dostarczone przez algorytm NEH_{AGV} dla poszczególnych instancji $TFr/m^p/x/k$ 169
6.3 Wartości funkcji celu dostarczone przez algorytm $TSAB_{AGV}$ dla poszczególnych instancji $TFr/m^p/x/k$ 170
6.4 Średnie względne poprawy i średnie czasy pracy algorytmów dla grup instancji $TFr/m^p/x/k$ 171

Spis rysunków

2.1 Schemat algorytmu tabu search 16
2.2 Schemat algorytmu symulowanego wyżarzania 17
2.3 Schemat algorytmu genetycznego 18
2.4 Schemat algorytmu poszukiwania rozproszonego 19
2.5 Najbardziej typowe układy maszyn: (a) linia, (b) pętla, (c) siatka 26

3.1 Wykres Gantta uszeregowania reprezentowanego przez permutację π z przykładu 3.1 35
3.2 Graf $G(\pi)$ dla permutacji π z przykładu 3.1 36
3.3 Wykres Gantta uszeregowania reprezentowanego przez permutację β z przykładu 3.2 44
3.4 Modyfikacje grafu $G(\beta)$ 53
3.5 Rozmieszczenie maszyn i dróg pomiędzy maszynami 56
3.6 Metoda wyznaczania najlepszej pozycji dla wstawianej operacji i 57
3.7 Schemat procedury $PWNP1$ 59
3.8 Wykres par $(H(\pi^i, \pi^0), \Delta C^i)$ oraz $(H_{av}(\pi^i), \Delta C^i)$ dla instancji testowej nr 27 65

4.1 Schemat algorytmu $INT3$ 73
4.2 Schemat procedury PPS 76
4.3 Schemat algorytmu SW 80
4.4 Schemat operatora GX 84
4.5 Schemat algorytmu $GADS$ 86
4.6 Schemat algorytmu $GAGX$ 87

5.1 Metoda wyznaczania najlepszej maszyny i pozycji dla wstawianej operacji i 115
5.2 Schemat procedury $PWNP2$ 116

5.3 Wykres Gantta uszeregowania reprezentowanego przez permutację π z przykładu 5.1 121
5.4 Modyfikacje grafu $G(\pi)$ dla permutacji π z przykładu 5.1 . 121
5.5 Schemat algorytmu $INT4$ 125
5.6 Schemat procedury $MPPS$ 129
5.7 Schemat operatora MX 131

6.1 Wykres Gantta uszeregowania zgodnego z permutacją π z przykładu 6.1 156
6.2 Graf $G(\pi)$ dla permutacji π z przykładu 6.1 157
6.3 Graf $G(\pi)$ dla permutacji π z przykładu 6.3 162
6.4 Schemat procedury NSP 166
6.5 Schemat algorytmu $TSAB$ 166